Classifications of (parallel) computer architectures

Intensive Computation

Annalisa Massini
2018/2019

References

- *Advanced Computer Architecture and Parallel Processing*

  H. El-Rewini, M. Abd-El-Barr, John Wiley and Sons, 2005

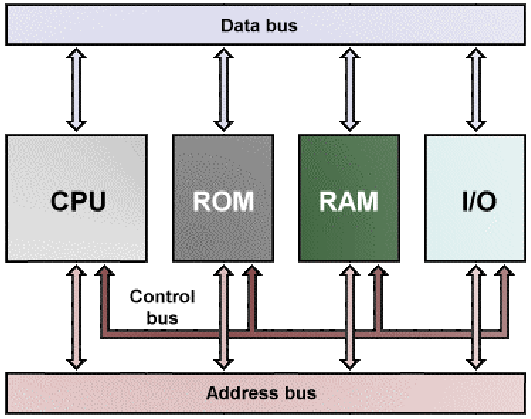
- *Parallel computing for real-time signal processing and control* – **Ch. 2 *Parallel Architectures***

  M. O. Tokhi, M. A. Hossain, M. H. Shaheed, Springer, 2003

INTRODUCTION

Motivations to Parallel Architectures

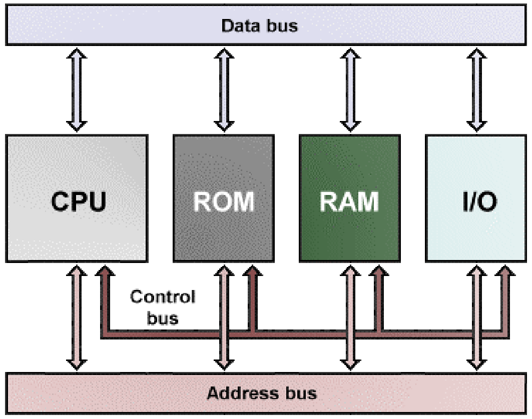
- Starting from the knowledge of the conventional computer architecture, it is important
  - to acquire an **understanding** and **appreciation** of a computer system
  - to learn to harness **parallelism to sustain performance improvements**



Motivations to Parallel Architectures

In fact the design of *parallel algorithms* and the study *of strategies for problem decomposition* **are sustained by**

- A deep **knowledge of the computer architecture**,
- A careful **use of parallelism** and
- The **performance analysis**



Motivations to Parallel Architectures

- **Parallel computer architecture** forms an important thread in the evolution of computer architecture

- It has its roots in the beginnings of computing, and exploits *advancement* over what the *base technology* can provide

- Parallel computer designs have demonstrated a rich *diversity of structure*, usually motivated by specific higher level parallel programming models

- The **speed** with which computer can process information has been **increasing exponentially** over the time

Motivations to Parallel Architectures

Role of a computer architect:

- To design and engineer the various levels of a computer system to maximize *performance* and *programmability* within limits of *technology* and *cost*

Parallelism:

- Provides an *interesting perspective* from which to understand computer architecture
- Provides *alternative to faster clock* for performance
- Applies at *all levels of system design*
- Is *increasingly central* in information processing

Motivations to Parallel Architectures

- **A *parallel computer* is a collection of processing elements** that cooperate to solve large problems fast

- This simple definition raises *many questions*

Motivations to Parallel Architectures

- **A *parallel computer* is a collection of processing elements** that cooperate to solve large problems fast

- This simple definition raises *many questions*

- **Resource Allocation**
  - how large a collection?
  - how powerful are the elements?
  - how much memory?

Motivations to Parallel Architectures

- **A *parallel computer* is a collection of processing elements** that cooperate to solve large problems fast
- This simple definition raises *many questions*
- **Data access, Communication and Synchronization**
  - how do the elements cooperate and communicate?
  - how are data transmitted between processors?
  - what are the abstractions and primitives for cooperation?

Motivations to Parallel Architectures

- **A *parallel computer* is a collection of processing elements** that cooperate to solve large problems fast
- This simple definition raises *many questions*
- **Performance and Scalability**
  - how does it all translate into performance?
  - how does it scale?

Motivations to Parallel Architectures

- To understand parallel architectures it is important to examine*:*
  - the principles of computer design at the processor level
  - the design issues present for each of the system components
    - memory systems
    - processors
    - networks
  - the relationships between these components
  - the division of responsibilities between hardware and software

Motivations to Parallel Architectures

- Parallel machines have been built at various scales since the earliest days of computing, but the approach is more viable today than ever before

- In fact
  - Whatever the performance of a single processor at a given time → **higher performance** can be achieved by utilizing many processors
  - But today the basic **processor** building block is **better suited** to the job

- How much additional performance is gained and at what additional cost *depends on a number of factors*

CLASSIFICATION

Parallel Architectures

- ***Parallel processors*** are computer systems consisting of
  - multiple **processing units**
  - connected via some **interconnection network**
  - plus the **software** needed to make the processing units work together

- There are two major factors used to categorize such systems:
  - the **processing units** themselves
  - the **interconnection network** that ties them together

Parallel Architectures

- A **vast number parallel architecture** types have been devised
- Various types of parallel architecture have **overlapping characteristics to different extents**

- It is ***not easy*** to develop a simple **classification** system for parallel architectures

Parallel Architectures

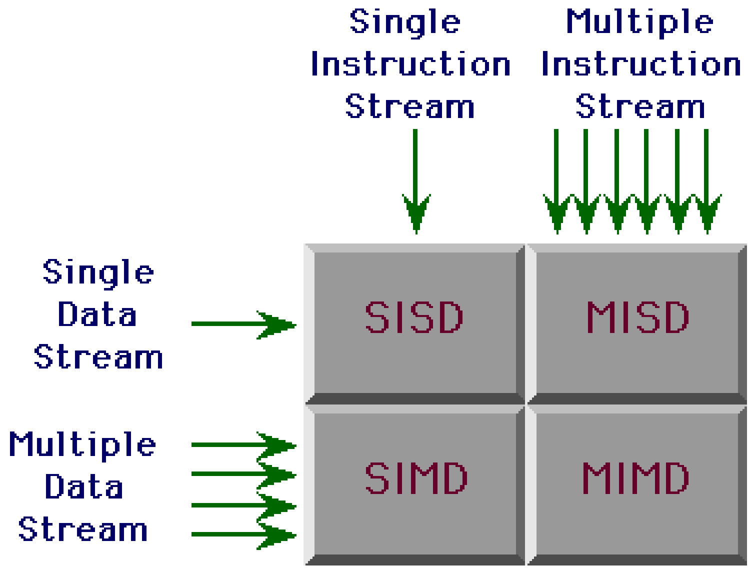
- Parallel architecture can be distinguished under the following broad categories:
  - **Flynn’s classification**
  - Classification based on **memory** arrangement
  - Classification based on **interconnections** among PEs and memory modules
  - Classification based on characteristic nature of **PEs**

Flynn’s classification

- Flynn’s classification is based on the notion of a **stream of information**
  - 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
- Either of the instruction or data streams can be single or multiple

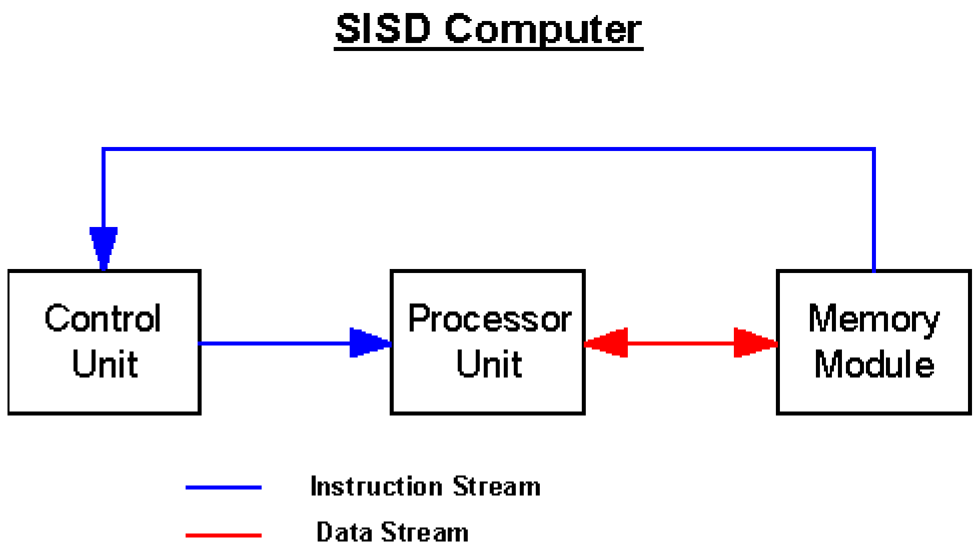
Flynn’s classification

- **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**)



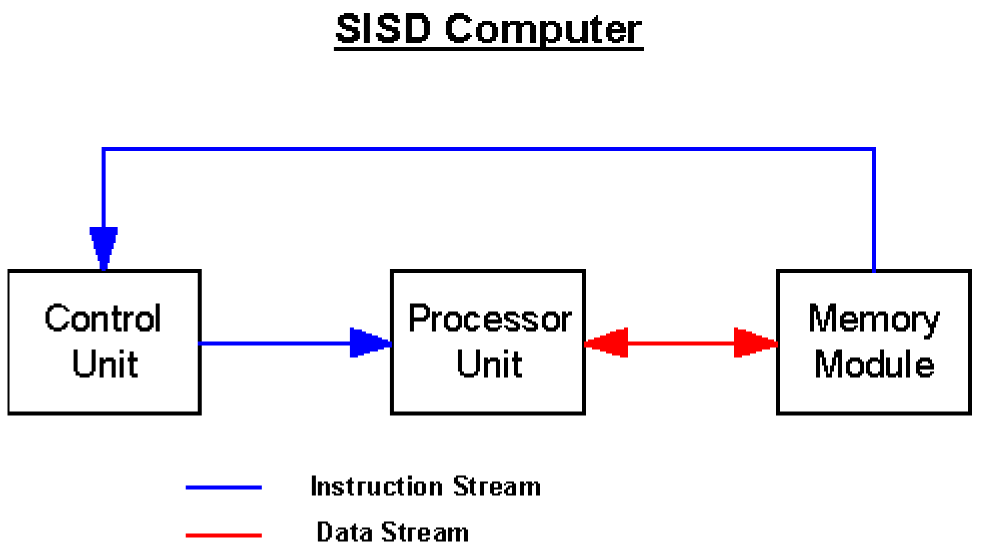
Single Instruction, Single Data Stream - SISD

- **Single** processor
- **Single** instruction stream
- Data stored in **single** memory



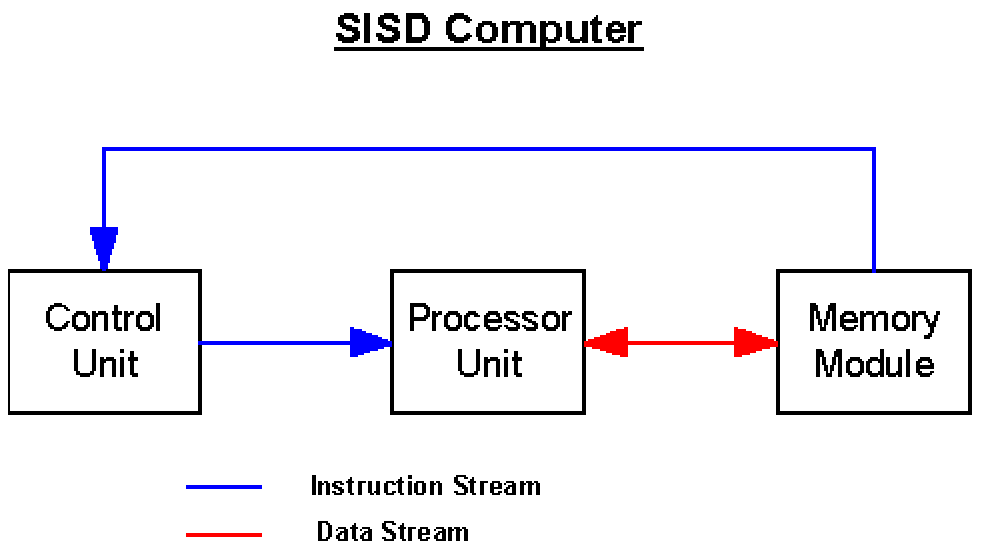
Single Instruction, Single Data Stream - SISD

- During program execution
  - the *PE fetches instructions and data* from the main memory
  - *processes the data* as per the instructions and
  - *sends the results to the main memory* after processing has been completed



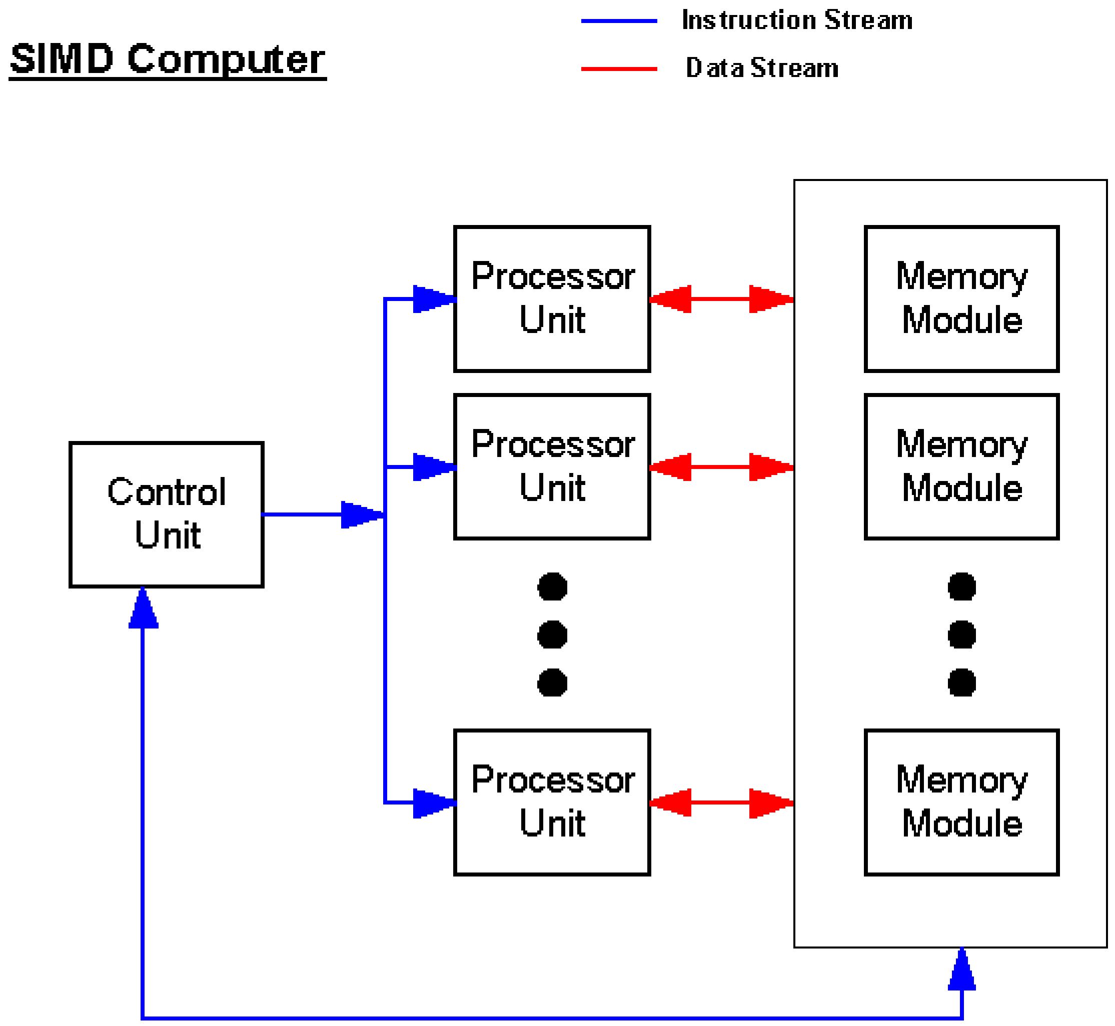
Single Instruction, Single Data Stream - SISD

- A **single** processor executes a **single** instruction at a time operating on data stored in a **single** memory
  - The Von Neumann computer (uniprocessor) falls under this category
  - The majority of contemporary CPUs is multicore → a single core can be considered a SISD machine



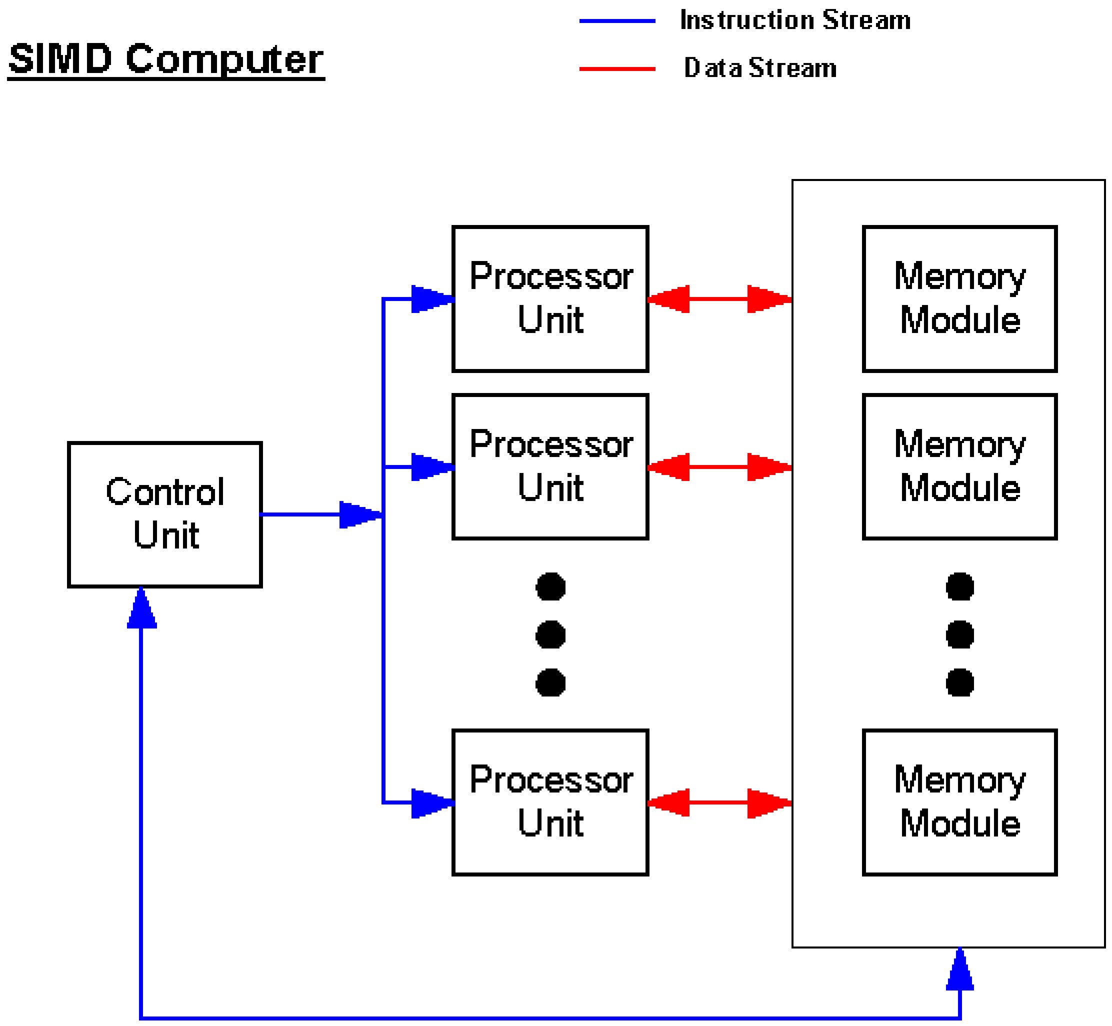
Single Instruction, Multiple Data Stream - SIMD

- A ***single machine instruction*** controls the simultaneous execution of a number of processing elements on a lockstep basis



Single Instruction, Multiple Data Stream - SIMD

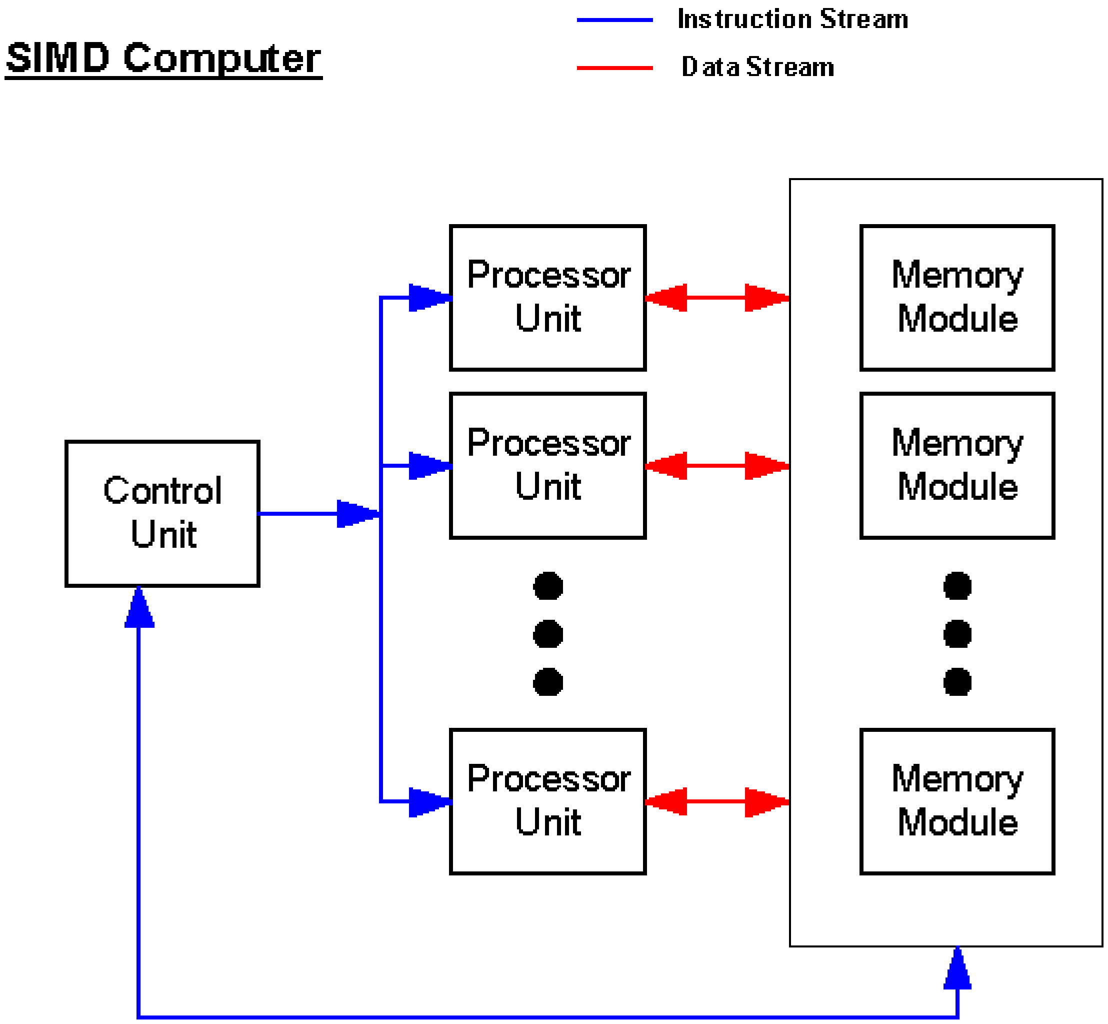
- Each **processing element** has an **associated data memory**
- ***Each instruction*** is executed on a ***different set of data*** by the ***different processors***



Single Instruction, Multiple Data Stream - SIMD

- **Vector processors** were the first SIMD machines
- **GPUs** follow this design at the level of Streaming multiprocessor

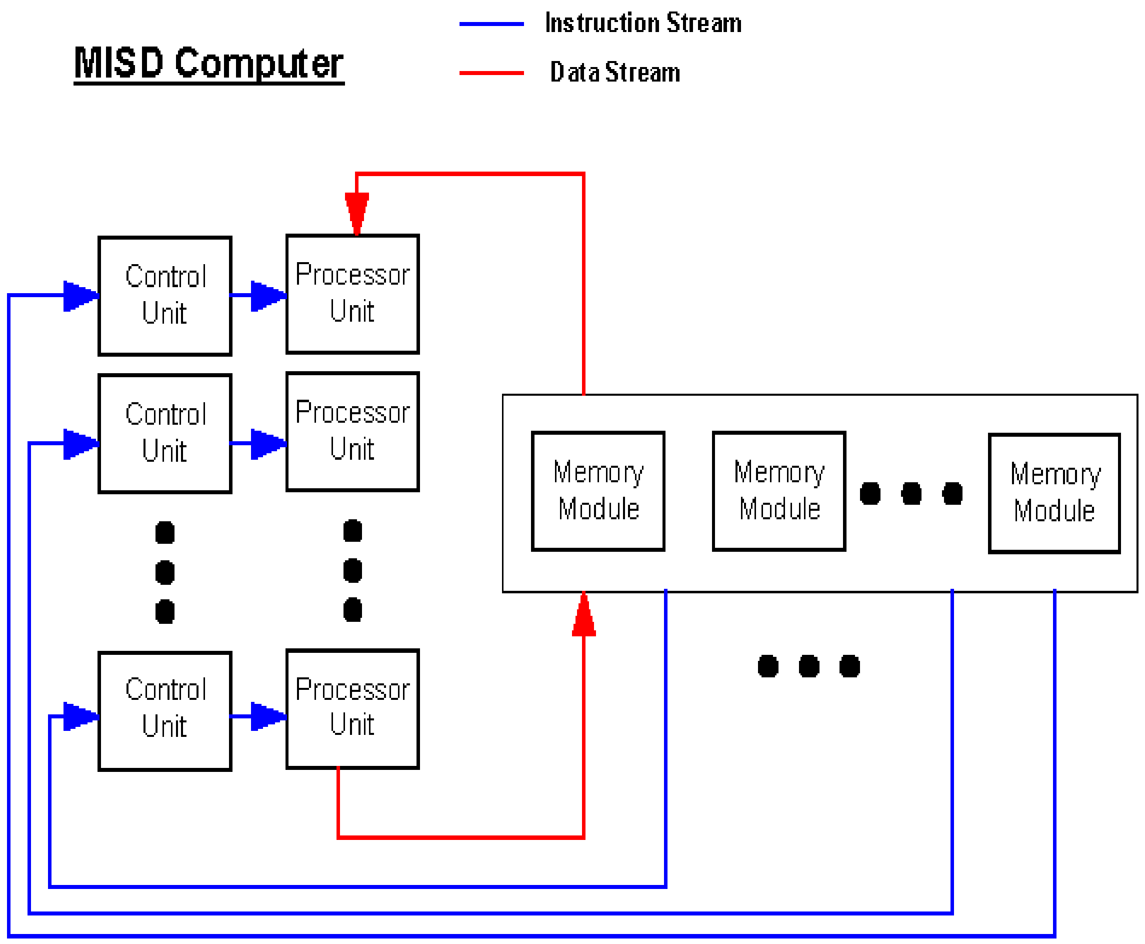
- Applications:
  - Image processing
  - Matrix manipulations
  - Sorting



Multiple Instruction, Single Data Stream - MISD

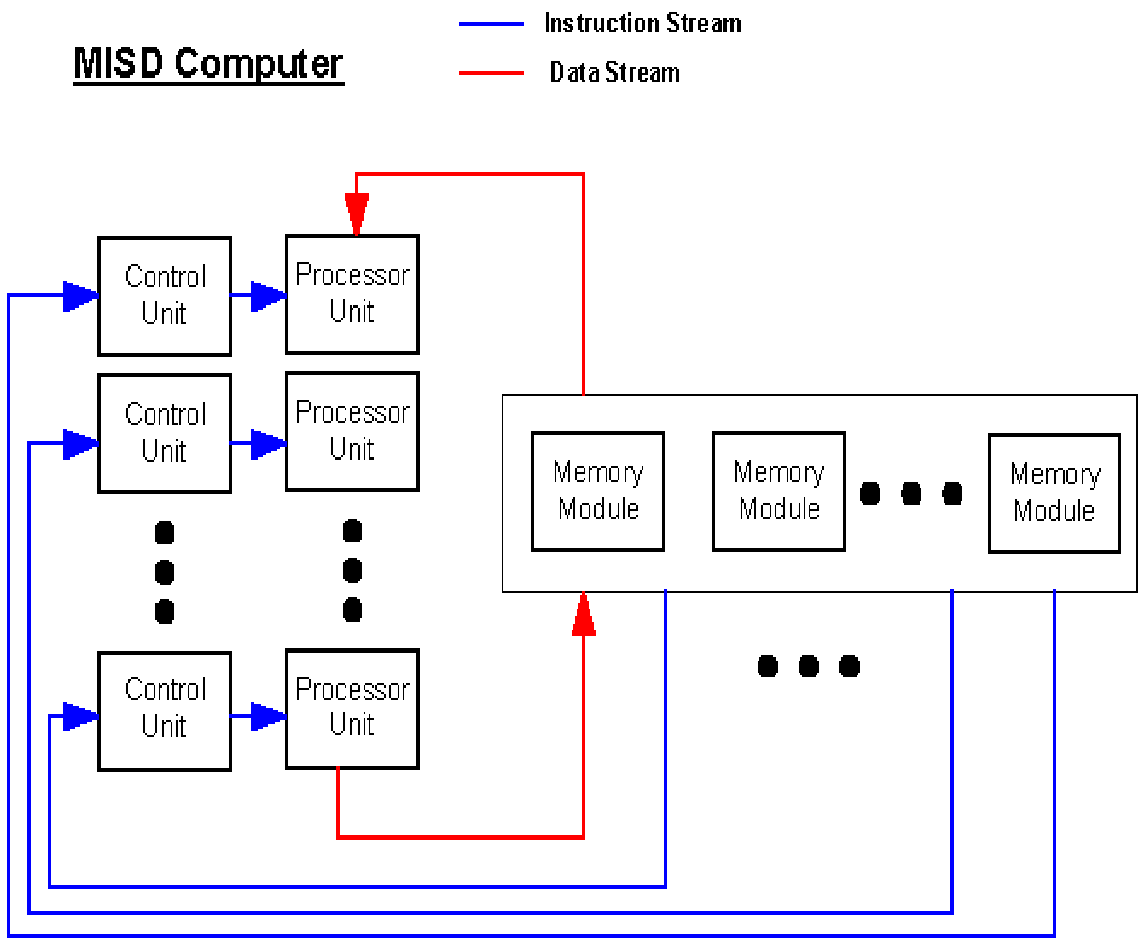
- A ***sequence of data*** is transmitted to a ***set of processors***, each of which *executes a* ***different*** *instruction* sequence

- This structure is not *commercially* implemented



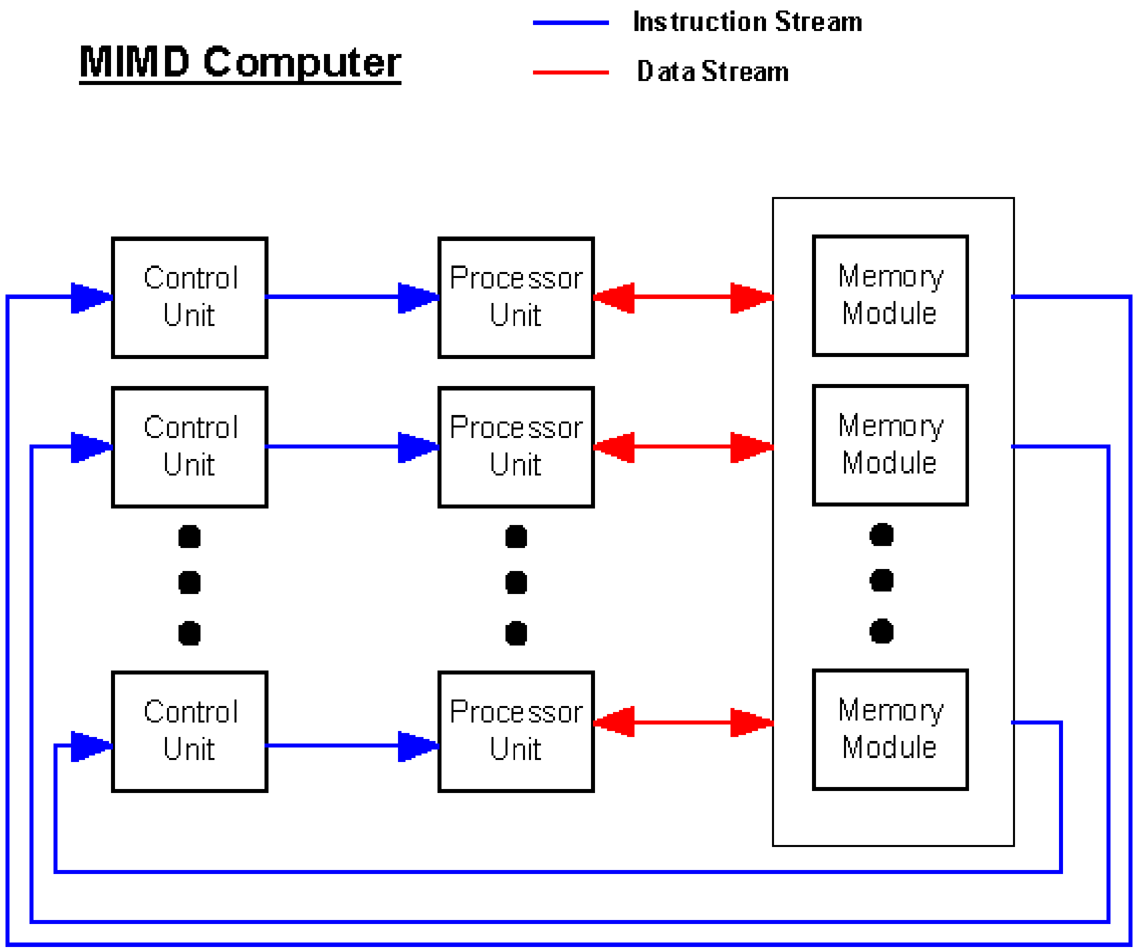
Multiple Instruction, Single Data Stream - MISD

- MISD computers can be useful in applications of a specialized nature:
  - robot vision
  - when *fault tolerance* is required (military or aerospace application) data can be processed by multiple machines and decisions can be made on a majority principle



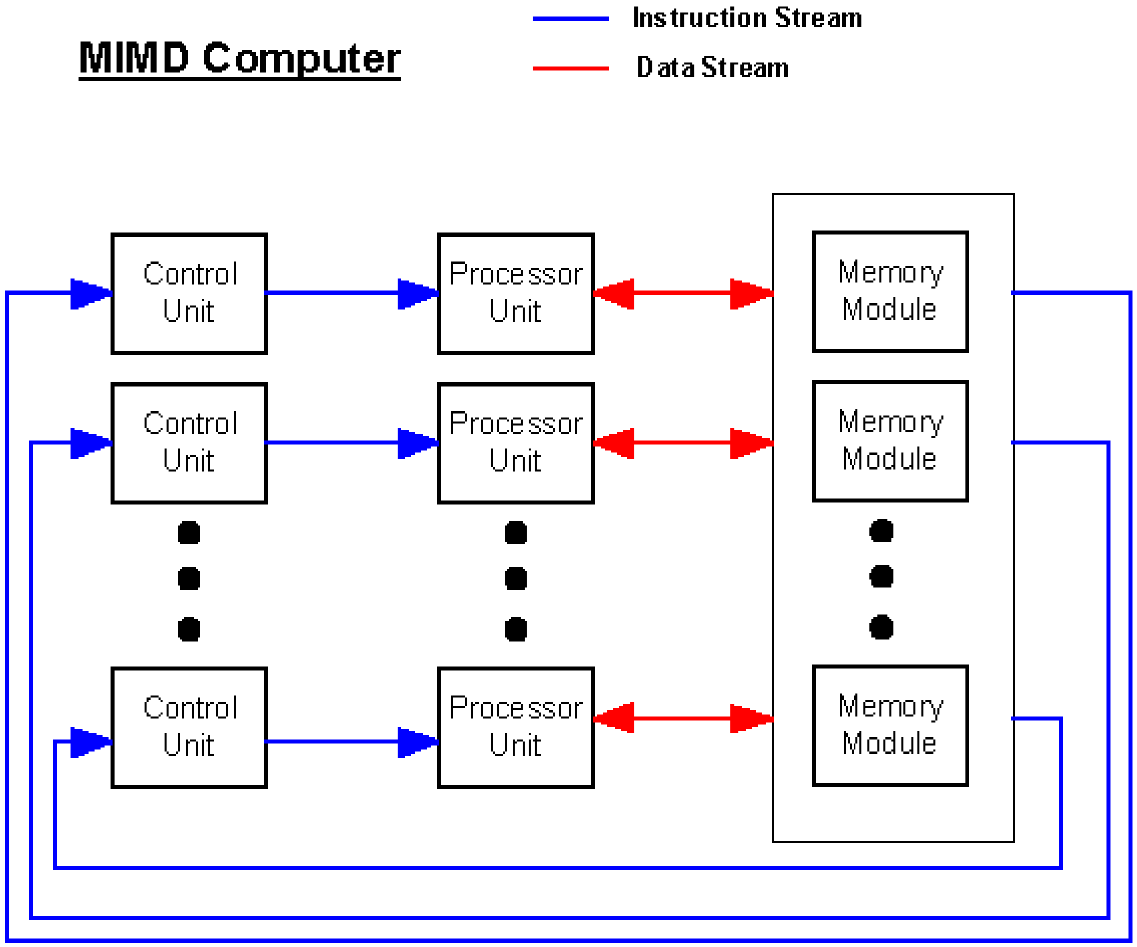
Multiple Instruction, Multiple Data Stream- MIMD

- A **set of processors** simultaneously execute **different instruction sequences** on **different data sets**
- This architecture is the most common and widely used form of parallel architectures



Multiple Instruction, Multiple Data Stream- MIMD

- General purpose processors
- Each processor can process all instructions necessary
- **Further classified** by method of processor communication



Flynn’s classification

- **Advantages of Flynn**
  - Universally accepted
  - Compact Notation
  - Easy to classify a system

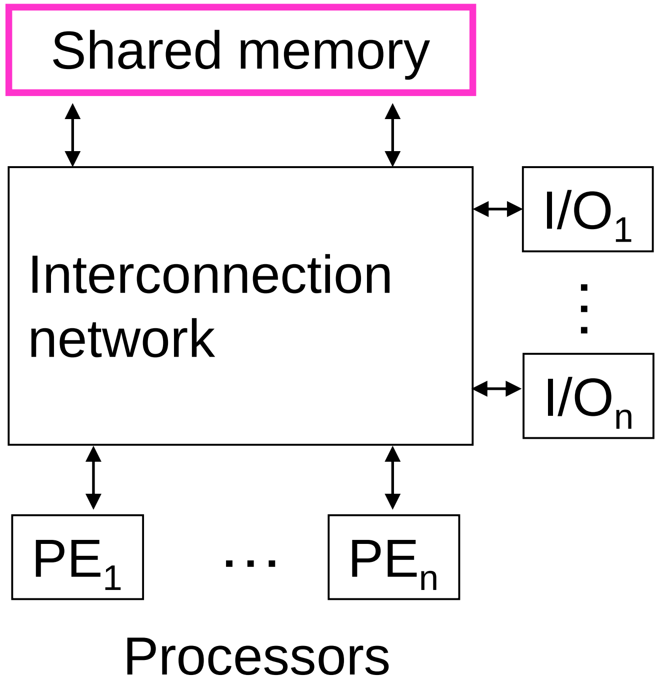
- **Disadvantages of Flynn**
  - Very coarse-grain differentiation among machine systems
  - Comparison of different systems is limited
  - Interconnections, I/O, memory not considered in the scheme

Classification based on memory arrangement

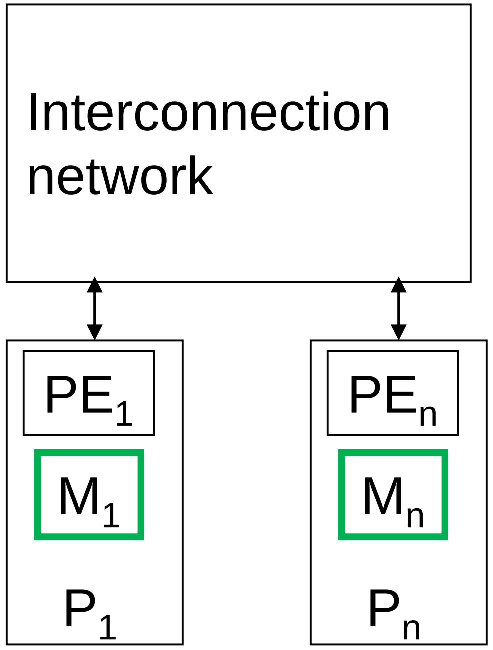
- Parallel architectures can be classified into two major categories in terms of **memory arrangement**:
  - *Shared memory*
  - Message passing or *distributed memory*

- This classification constitutes a subdivision of MIMD parallel architecture and are also known as:
  - **Shared memory** architecture → **tightly coupled** architecture
  - **Distributed memory** architecture → **loosely coupled** architecture

Classification based on memory arrangement



Shared memory - multiprocessors



Distributed memory
message passing - multicomputers

Shared Memory Multiprocessor

- Multiple processors share a **common memory unit** comprising a single or several memory modules

- All the **processors** have **equal access to the memory modules**
- The **memory modules** are seen as
  - a single address space by all the processors

- The memory modules
  - store data
  - establish communication among the processors via some bus arrangement

Shared Memory Multiprocessor

- **Communication** is established through memory access instructions
  - processors exchange messages between one another by one processor *writing data into the shared memory* and another *reading that data from the memory*

- The executable programming codes are stored in the memory for each processor to execute
- The data related to each program is also stored in this memory
- Each program can gain access to all data sets present in the memory if necessary

Shared Memory Multiprocessor

- There is **no direct processor-to-processor communication** involved in the programming process
- **Communication** is handled mainly ***via the shared memory*** modules
- Access to these memory modules can easily be controlled through appropriate programming mechanisms
- However, this architecture suffers from a **bottleneck** problem when a **number of processors** endeavour to access the global **memory** at the **same time**
- This **limits the scalability** of the system

Shared Memory Multiprocessor

- Shared memory multiprocessors can be of two types:
  - **uniform** memory access (UMA) architecture
  - **non-uniform** memory access (NUMA) architecture

- In the case of UMA architectures, the memory access time to the different parts of the memory are almost the same
- UMA architectures are also called symmetric multiprocessors

Shared Memory Multiprocessor

- An **UMA** architecture comprises two or more **processors with identical characteristics**
- The processors:
  - share the same main memory and I/O facilities
  - are interconnected by some form of bus-based interconnection scheme
- The memory access time is approximately the same for all processors
- Processors perform the same functions under control of an operating system, which provides interaction between processors and their programs at the job, task, file and data element levels

Shared Memory Multiprocessor

- In the case of **NUMA** architectures the **memory access time** of processors **differs** depending on which region of the main memory is accessed

- A subclass of NUMA system is cache coherent NUMA (CC-NUMA) where cache coherence is maintained among the caches of various processors
- The main advantage of a CC-NUMA system is that it can deliver effective performance at higher levels of parallelism than UMA architecture

Message Passing Multicomputer

- In a **distributed memory architecture** each unit is a complete computer building block including the processor, memory and I/O system

- A processor can access the memory, which is directly attached to it
- Communication among the processors is established in the form of I/O operations through message signals and bus networks

Message Passing Multicomputer

- **Example**
  - If a processor needs data from another processor
  - It sends a signal to that processor through an interconnected bus network demanding the required data
  - The remote processor then responds accordingly
- Certainly, access to local memory is faster than access to remote processors
- Most importantly, the further the physical distance to the remote processor, the longer it will take to access the remote data
- This architecture suffers from the drawback of requiring direct communication from processor to processor

Message Passing Multicomputer

- The **speed performance** of distributed memory architecture largely depends upon **how the processors are connected** to each other
- It is impractical to connect each processor to the remaining processors through independent cables → it can work for a very low number of processors but becomes impossible as the number of processors in the system increases
- The **most common solution** is to use **specialized bus** networks to connect all the processors in the system in order that each processor can communicate with any other processor attached to the system

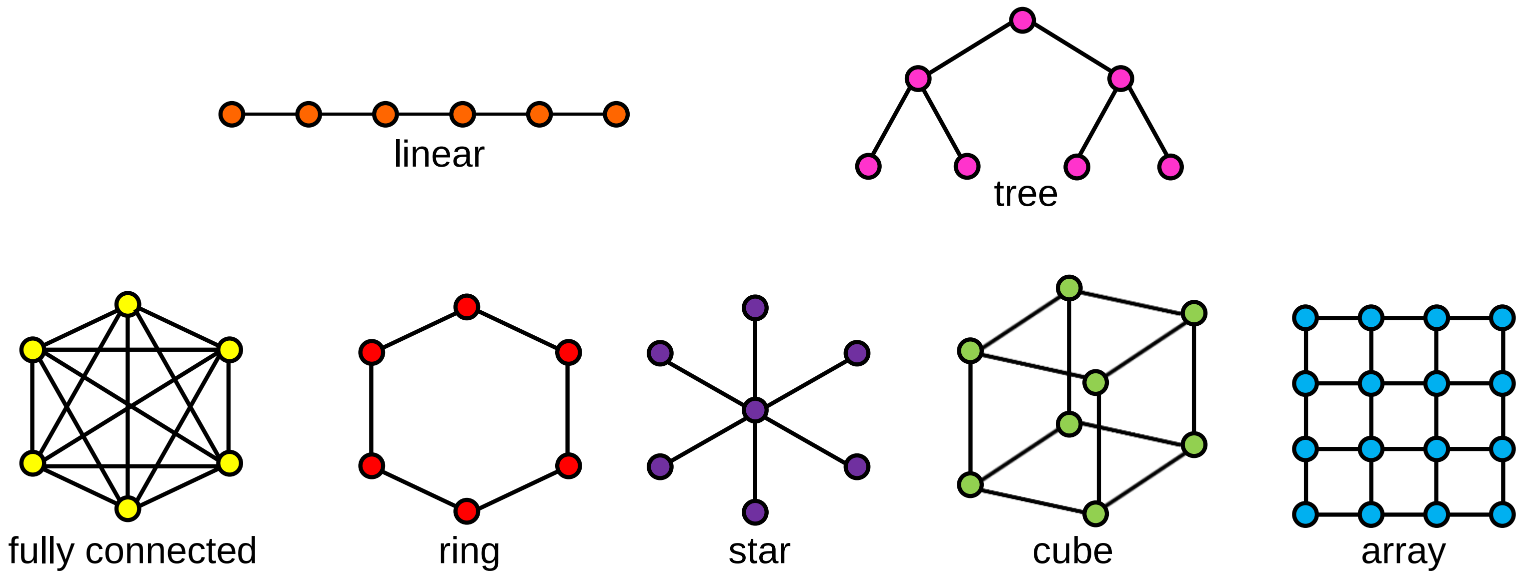
Classification based on type of interconnections

- This classification is quite specific to **MIMD architectures** as they, generally, comprises multiple PEs and memory modules

- The various **interconnecting communication networks** used for establishing communication schemes among the PEs of a parallel architecture include: **linear, shared single bus, shared multiple bus, crossbar, ring, mesh, star, tree, hypercube and complete graph**

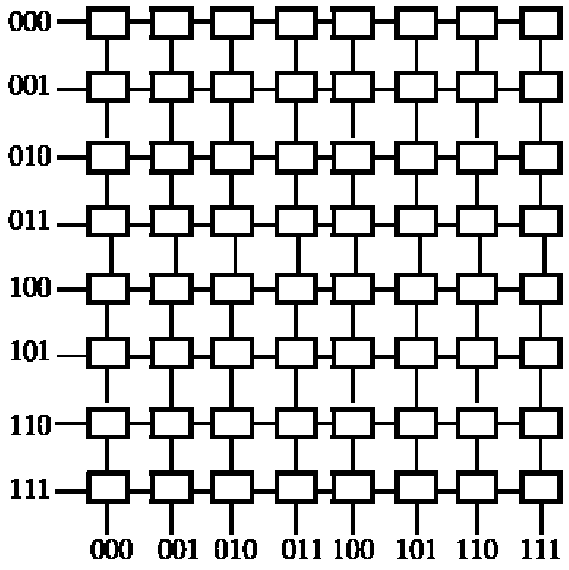
Classification based on type of interconnections

- Among these interconnecting networks:
  - *linear, mesh, ring, star, tree, hypercube and complete graph* are **static** connection structures



Classification based on type of interconnections

- Among these interconnecting networks:
  - *shared single bus, shared multiple bus and crossbar* are **dynamic** interconnection structures as they are **reconfigurable** under system control



Classification based on characteristic of PEs

- Parallel architectures are also classified in terms of the nature of the PEs comprising them
- An architecture may consist of either only **one type of PE** or various types of Pes
- The different types of processors that are commonly used to form parallel architectures are:
  - CISC Processors
  - RISC Processors
  - Vector Processors and DSP (Digital Signal Processor)
  - Homogeneous and Heterogeneous Parallel Architectures