Advanced Parallel Architecture

Annalisa Massini - 2016/2017

References

- *Advanced Computer Architecture and Parallel Processing*

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

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

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

- Culler Singh – Ch. 10
- Hennessy Patterson - Appendix F

Multiprocessors

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**

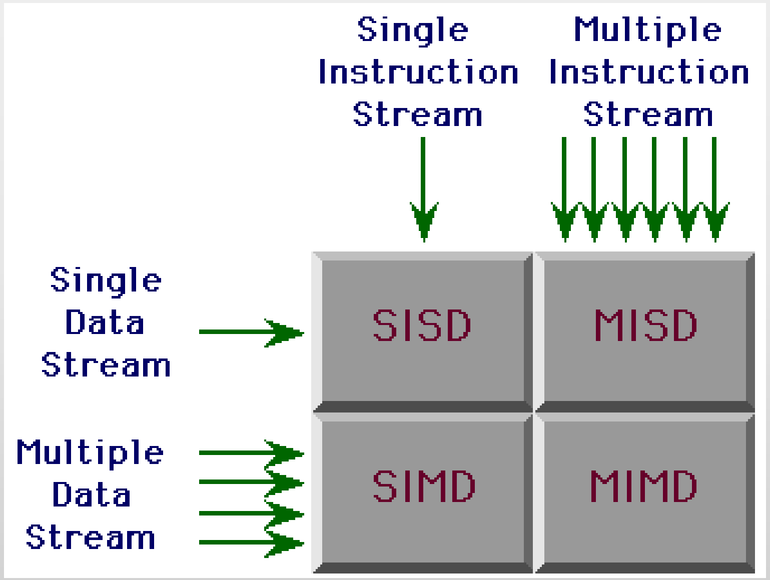
Flynn’s classification

- 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

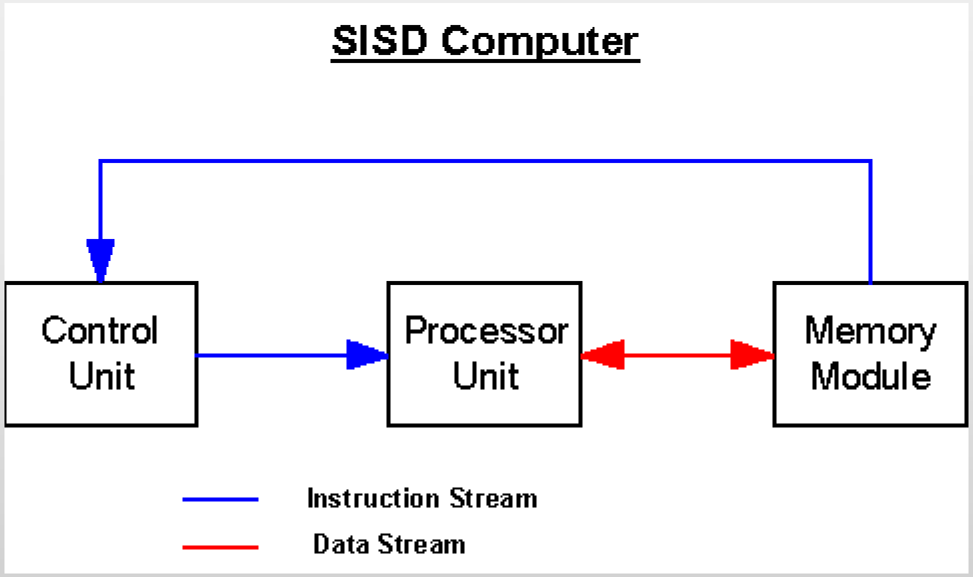
Flynn's classification

- 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)



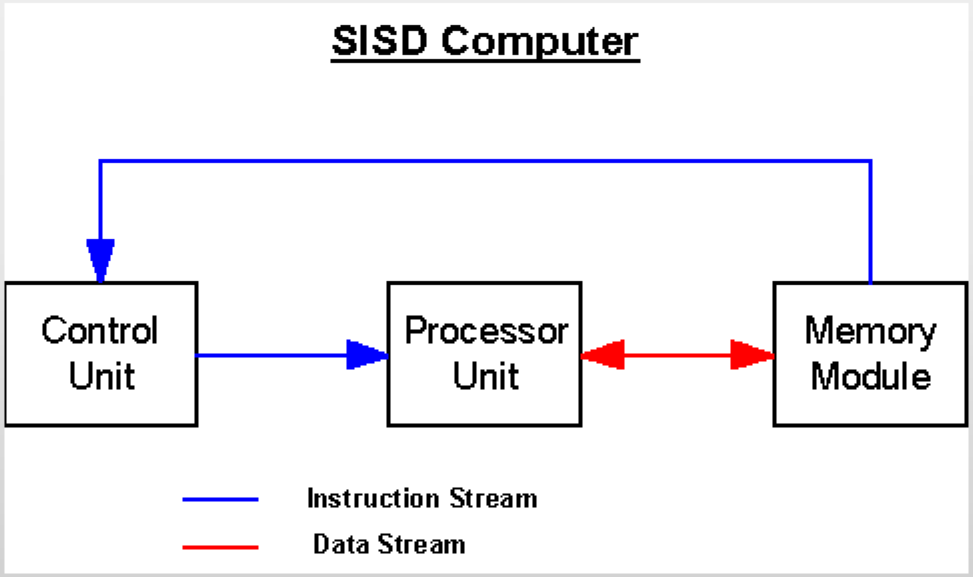
Single Instruction, Single Data Stream - SISD

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



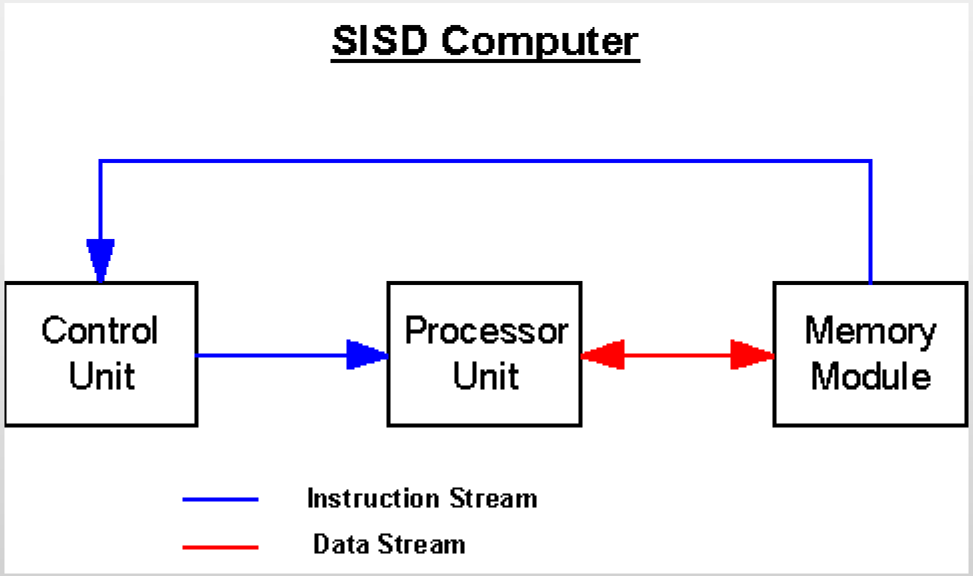
Single Instruction, Single Data Stream - SISD

- 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



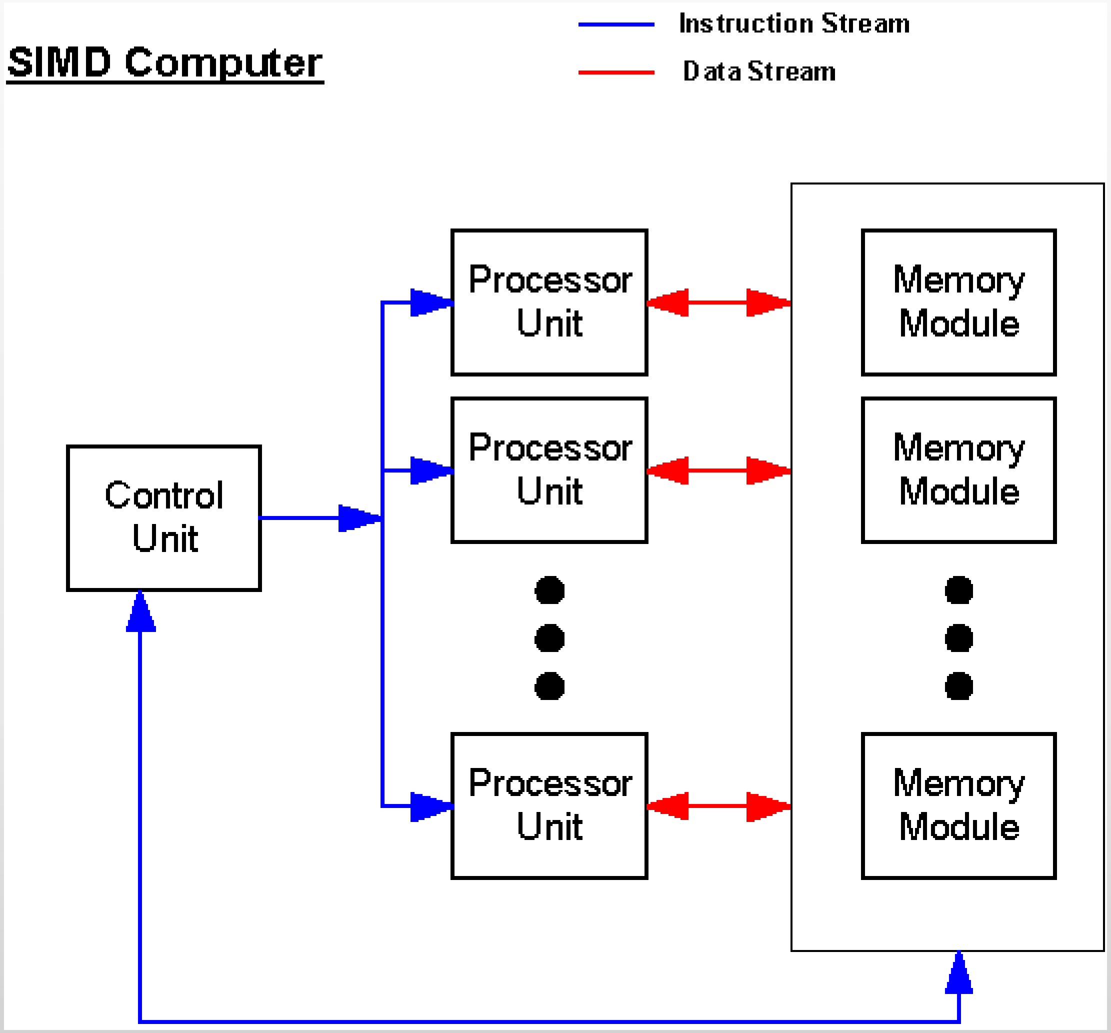
Single Instruction, Single Data Stream - SISD

- 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



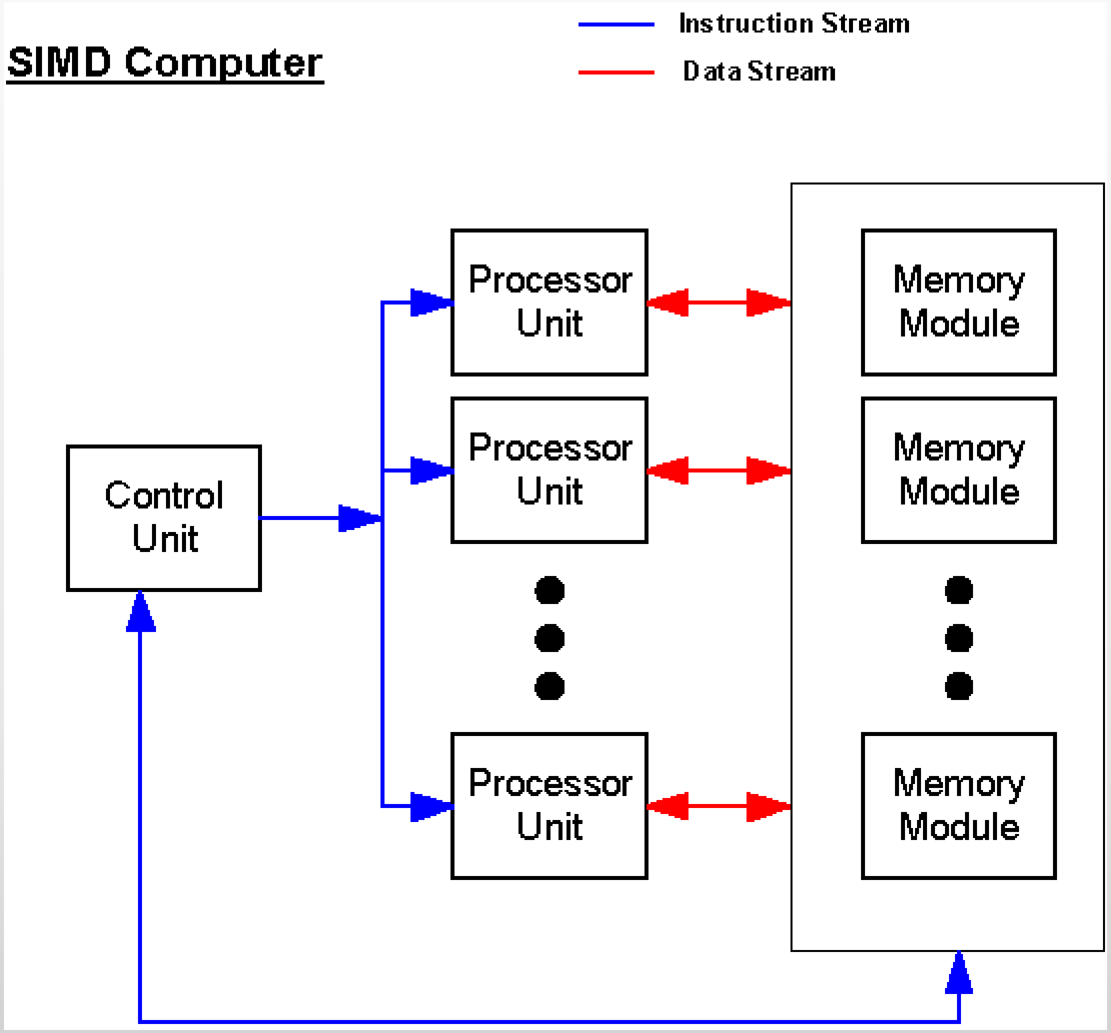
Single Instruction, Multiple Data Stream - SIMD

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



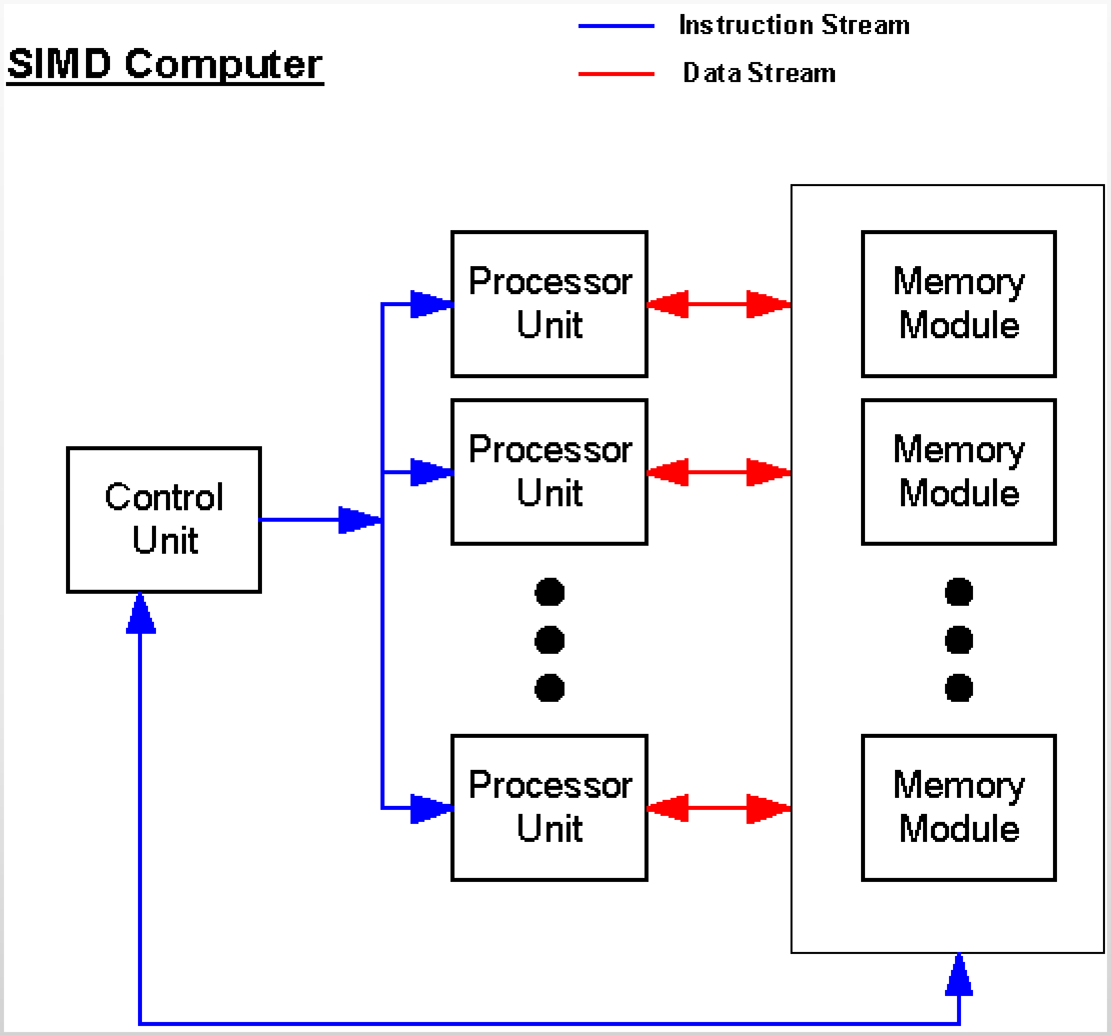
Single Instruction, Multiple Data Stream - SIMD

- 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

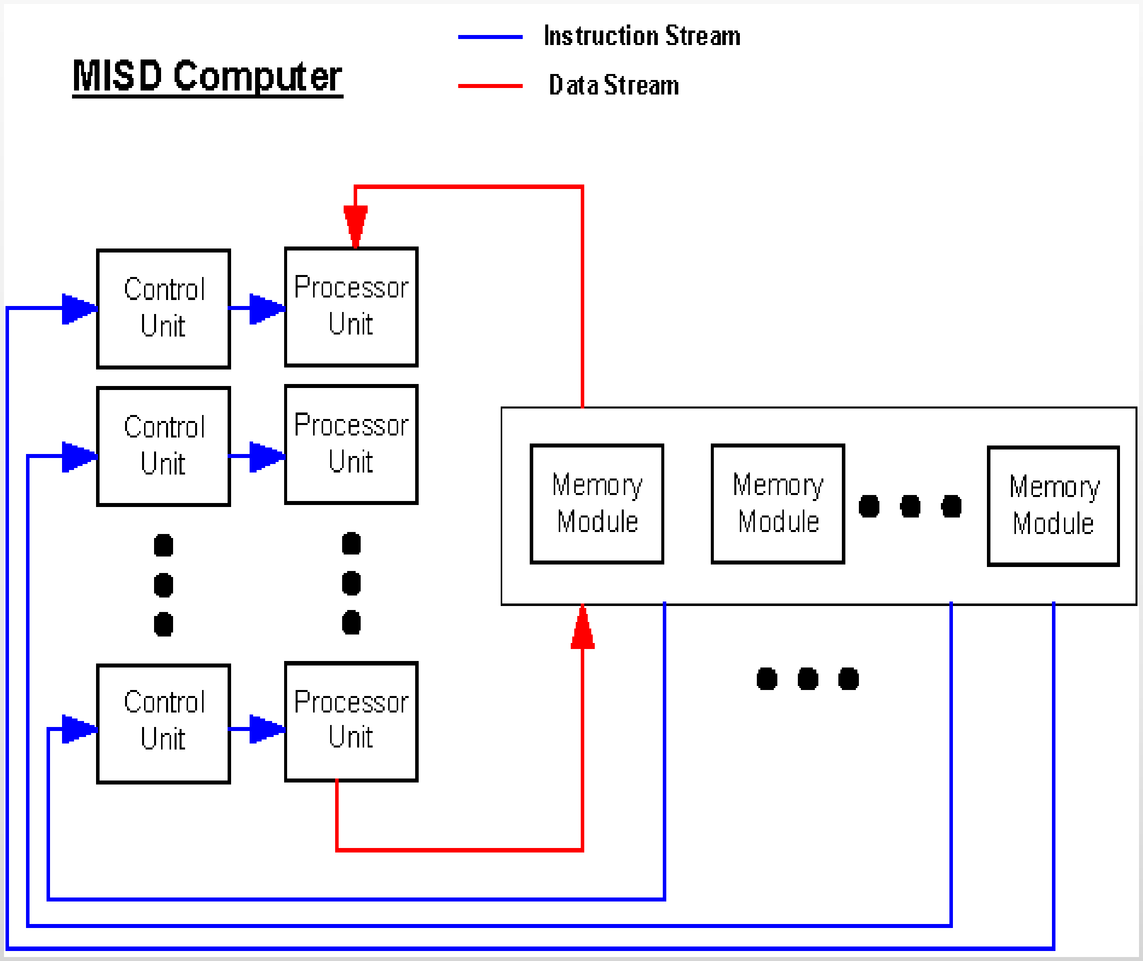
- **Vector processors** were the first SIMD machines
- **GPUs** follow this design at the level of Streaming multiprocessor
- Applications:
  - Image processing
  - Matrix manipulations
  - Sorting



Multiple Instruction, Single Data Stream - MISD

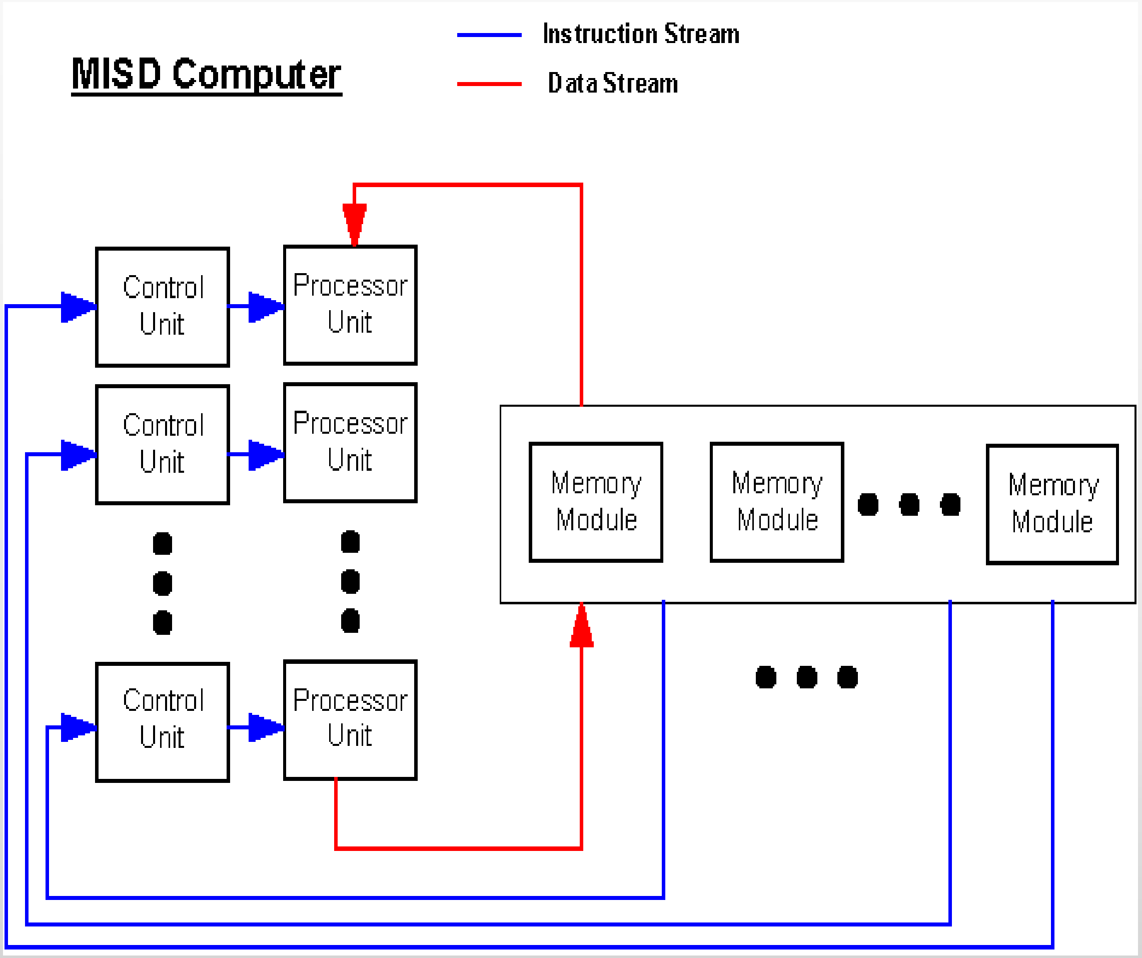
- A ***sequence of data*** is transmitted to a ***set of processors***, each of which *executes a* ***different*** *instruction* sequence

- This structure is not *commercially* implemented



Multiple Instruction, Single Data Stream - MISD

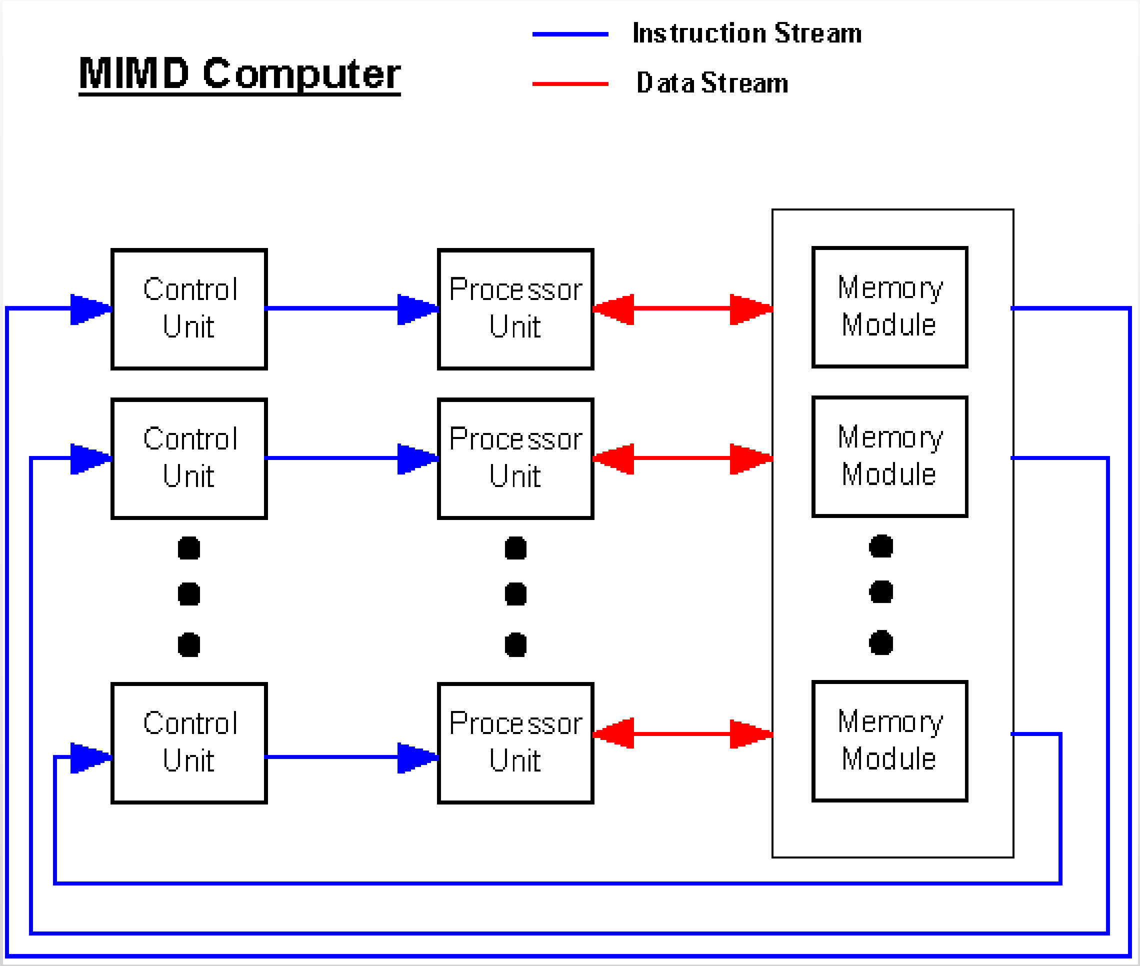
- MISD computers can be useful in applications of a specialized nature:
  - robot vision
  - when *fault tolerance* is required (military or aerospace application) data can be processed by multiple machines and decisions can be made on a majority principle



Multiple Instruction, Multiple Data Stream- MIMD

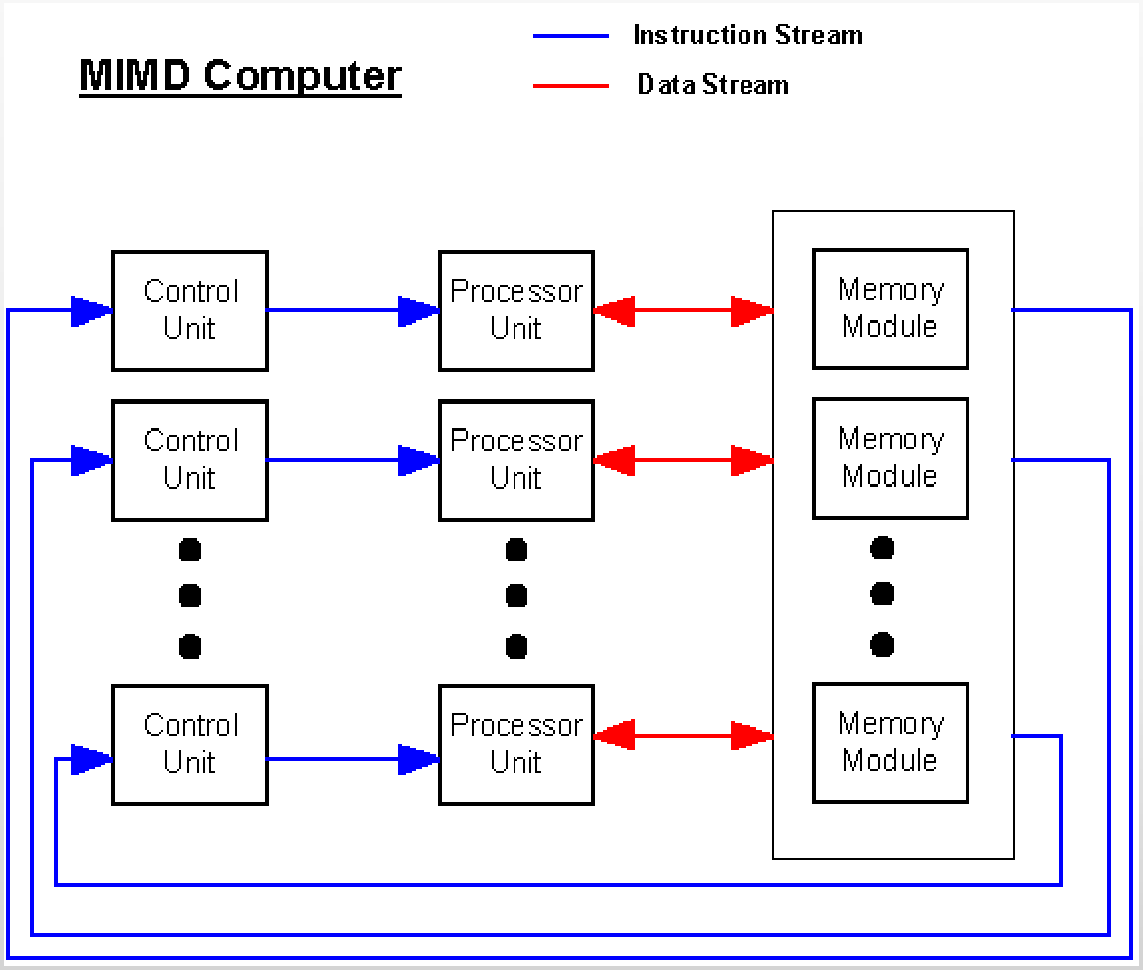
- A **set of processors** simultaneously execute **different instruction sequences** on **different data sets**

- 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



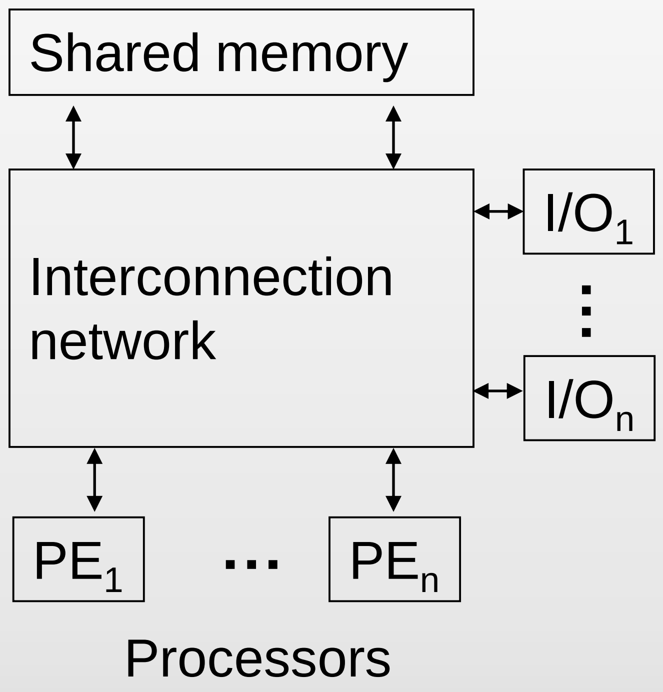
Flynn’s classification

- 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

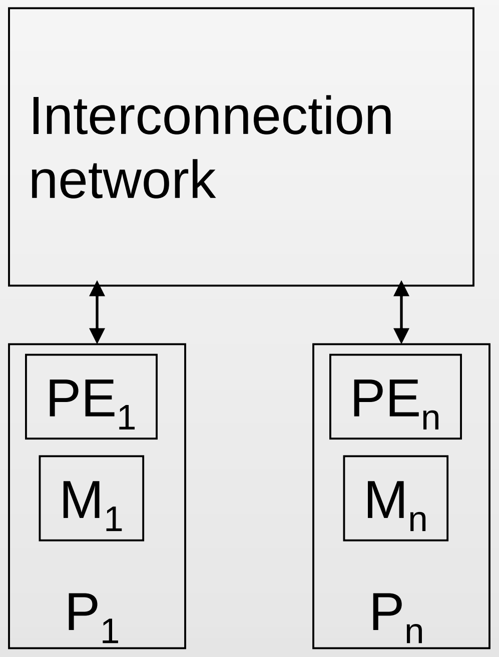
Classification based on memory arrangement

- 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



Message passing – multicomputers (distributed memory)

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 as well as serve to 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

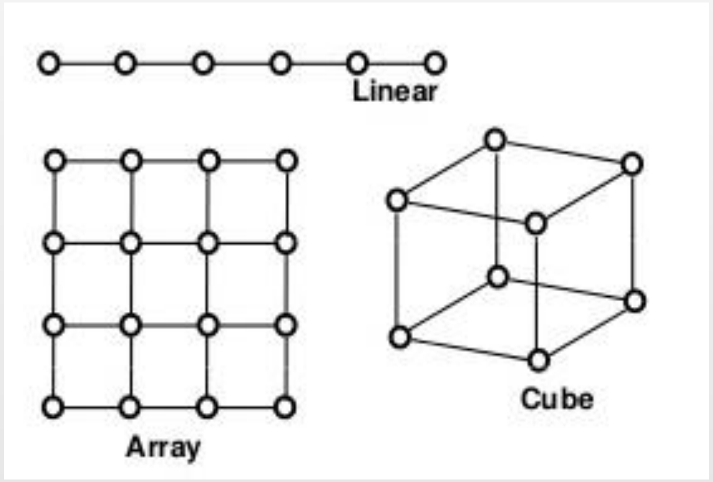
- The **speed performance** of distributed memory architecture largely depends upon **how the processors are connected** to each other
- It is impractical to connect each processor to the remaining processors through independent cables → it can work for a very low number of processors but becomes impossible 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

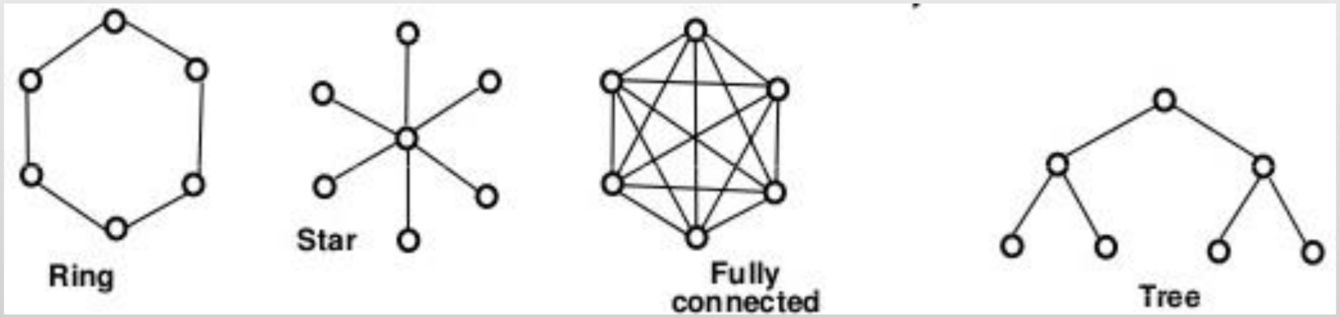
Classification based on type of interconnections

- 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

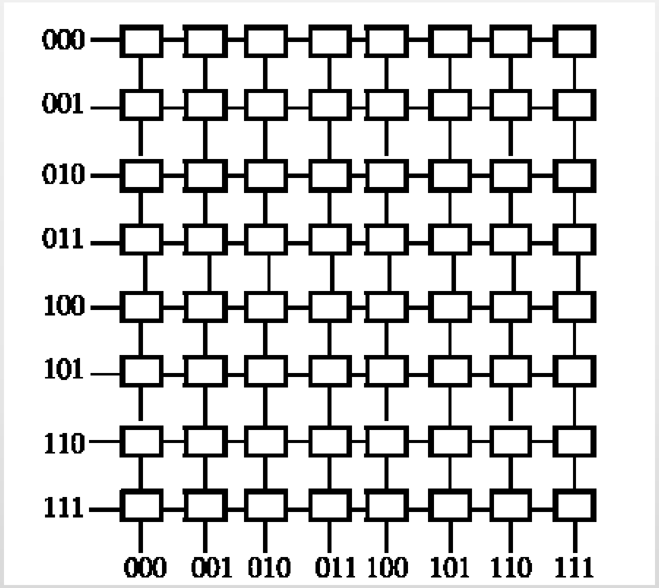
- Among these interconnecting networks:
  - *linear, mesh, ring, star, tree, hypercube and complete graph* are **static** connection structures





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

Interconnection Networks

Criteria for classification

- Multiprocessors interconnection networks (INs) can be classified based on a number of criteria:
  - Mode of Operation (Synchronous vs. Asynchronous)
  - Control Strategy (Centralized vs. Decentralized)
  - Switching Techniques (Packet switching vs. Circuit switching)
  - Topology (Static Vs. Dynamic)

Mode of operation

- According to the mode of operation, INs are classified as ***synchronous*** versus ***asynchronous***
- In **synchronous** mode of operation:
  - a **single global clock** is used by all components in the system such that the whole system is operating in a lock–step manner
- **Asynchronous** mode of operation:
  - Does not require a global clock
  - Handshaking signals are used instead in order to coordinate the operation of asynchronous systems
- While synchronous systems tend to be slower compared to asynchronous systems, they are race and hazard-free

Control strategy

- According to the control strategy, INs can be classified as ***centralized*** versus ***decentralized***
- In **centralized** control systems:
  - a single central control unit is used to oversee and control the operation of the components of the system
- In **decentralized** control:
  - the control function is distributed among different components in the system
- The function and reliability of the central control unit can become the bottleneck in a centralized control system. While the *crossbar* is a centralized system, the *multistage interconnection networks* are decentralized

Switching techniques

- Interconnection networks can be classified according to the switching mechanism as ***circuit*** versus ***packet switching networks***

- In the **circuit switching mechanism**:
  - A complete path has to be established prior to the start of communication between a source and a destination
  - The established path will remain in existence during the whole communication period

Switching techniques

- Interconnection networks can be classified according to the switching mechanism as ***circuit*** versus ***packet switching networks***

- In a **packet switching mechanism**:
  - communication between a source and destination takes place via messages that are divided into smaller entities, called packets
  - On their way to the destination, packets can be sent from a node to another in a store-and-forward manner until they reach their destination

Topology

- An interconnection network topology is a mapping function from the set of processors and memories onto the same set of processors and memories
- In other words, the topology describes how to connect processors and memories to other processors and memories
- For example:
  - A **fully connected topology** is a mapping in which each processor is connected to all other processors in the computer
  - A **ring** topology is a mapping that connects processor k to its neighbors, processors (k - 1) and (k + 1)

Topology

- In general, interconnection networks can be classified as ***static*** versus ***dynamic*** networks
- In static networks:
  - direct fixed links are established among nodes to form a fixed network
- In dynamic networks:
  - connections are established as needed
- Switching elements are used to establish connections among inputs and outputs
- Depending on the switch settings, different interconnections can be established

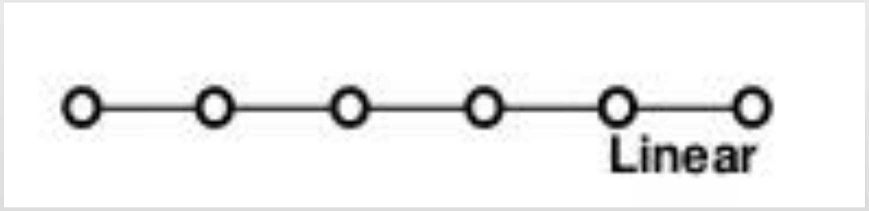
Static Networks

- Connections in a static network are fixed links
- Static networks can be further classified according to their interconnection pattern as:
  - one-dimension (1D)
  - two-dimension (2D)
  - hypercube (HC)

Static Networks

Linear Network

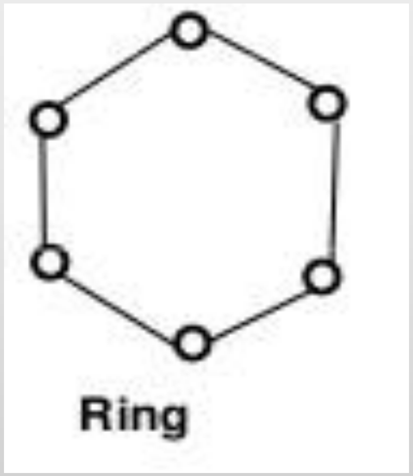
- Every node, except the nodes at the two ends, in this configuration is directly connected to two other nodes
- To connect *n nodes in this configuration n− 1 buses are required and the maximum* internodes distance is *n− 1*



Static Networks

Ring Interconnection Network

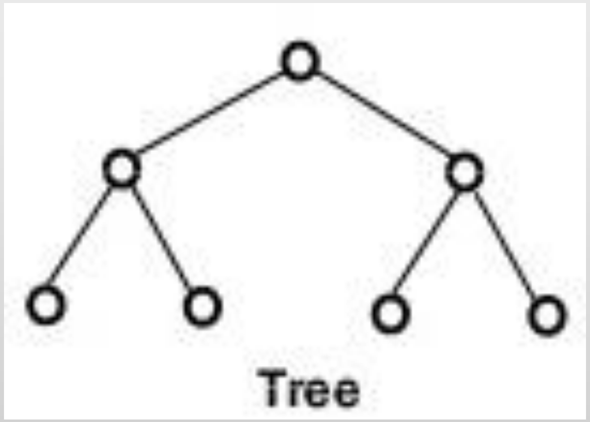
- *n* buses are required to connect *n* nodes
- *the maximum* internodes distance is *n / 2*
- Several commercial machines have been designed using ring networks (e.g. Hewlett-Packard’s Exemplar V2600 Kendal Square Research’s KSR-2)



Static Networks

Tree Interconnection Network

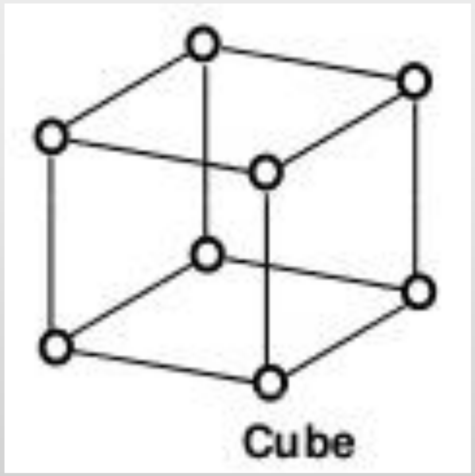
- In the tree structure (*n -level tree)* any intermediate node acts as a medium to establish communication between its parents and children
- Communication can be established between any two nodes in the structure
- the root node can be the bottleneck



Static Networks

Hypercube Interconnection Network

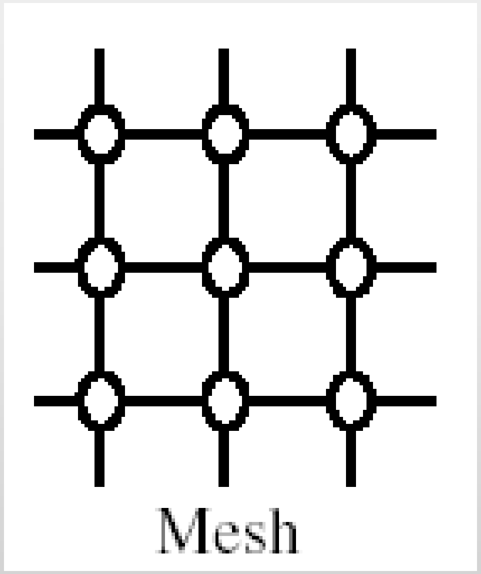
- An *n -dimensional hypercube can connect* 2^n *nodes*
- The **nodes** are labelled using **binary addresses** (addresses of the two neighbouring nodes differ by one bit position)
- Many commercial multiprocessors (especially NUMA multiprocessors) have used hypercube interconnections



Static Networks

Mesh and Torus Interconnection Network

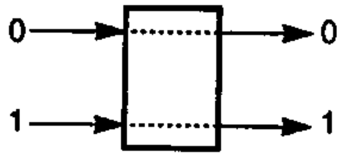
- Mesh is used to connect large numbers of nodes
- It is an alternative to hypercube in large multiprocessors
- To formulate a mesh structure comprising *n nodes 2(n – n0.5) buses are required and the* maximum internodes distance is 2($\sqrt{n}$ *– 1)*
- A Torus is obtained by using wraparound connections between the nodes at opposite edges



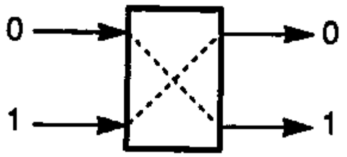
Dynamic Networks

- Connections in a dynamic network are established on the fly as needed
- Dynamic networks can be classified based on interconnection scheme as **bus-based** or **switch-based**
- **Bus-based** networks can further be classified as single bus or multiple buses
- **Switch-based** can be classified according to the structure of the interconnection network:
  - single-stage
  - multistage
  - crossbar networks

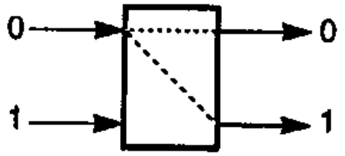
2 × 2 Switches



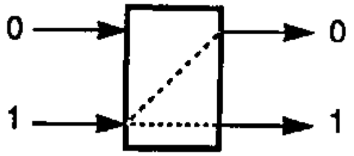
(a) Straight



(b) Crossover



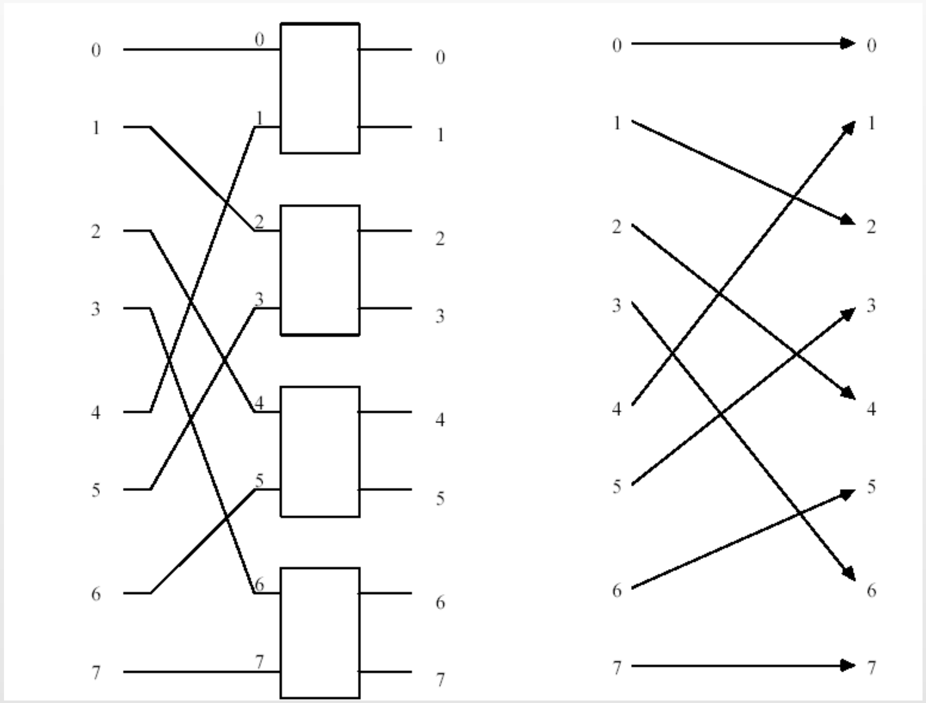
(c) Upper broadcast



(d) Lower broadcast

Single-stage networks

- Single stage **Shuffle-Exchange** IN (left)
- Perfect **shuffle** mapping function (right)
- Perfect shuffle operation: cyclic shift 1 place left, eg 101 --> 011
- Exchange operation: invert least significant bit, e.g. 101 --> 100



Multistage Interconnection Networks

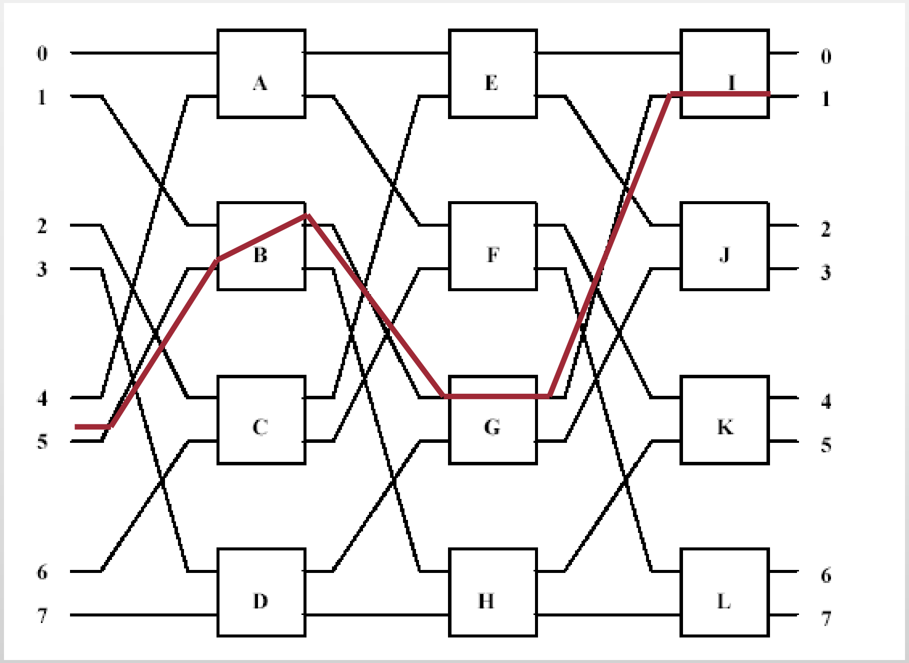
- The capability of single stage networks is limited
- If we **cascade** enough of them together, they form a **Multistage Interconnection Network** (MIN)
- Switches can perform their own routing or can be controlled by a central router

Multistage Interconnection Networks

- ***Nonblocking***
  - A network is called strictly nonblocking if it can connect any idle input to any idle output regardless of what other connections are currently in process
- ***Rearrangeable nonblocking***
  - In this case a network should be able to establish all possible connections between inputs and outputs by rearranging its existing connections
- ***Blocking interconnection***
  - A network is said to be blocking if it can perform many, but not all, possible connections between terminals.
  - Example: log N stage networks such as Omega, Baseline, Butterfly, …

Omega networks

- A MIN using 2 × 2 switches and a perfect shuffle interconnect pattern between the stages
- There is one unique path from each input to each output
- No redundant paths → no fault tolerance, blocking

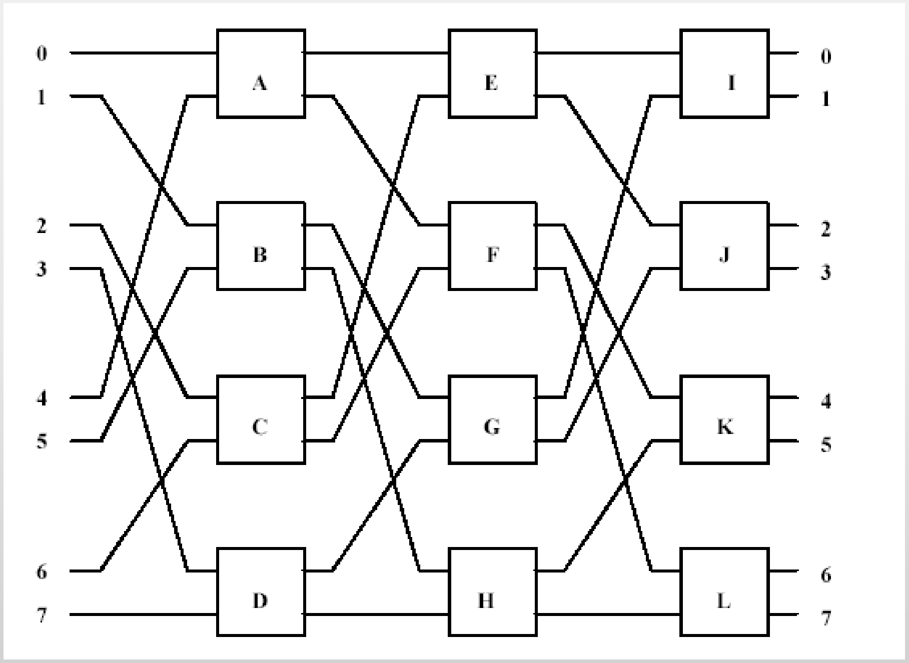


Example:

- Connect input 101 to output 001
- Use the bits of the destination address, 001, for dynamically selecting a path
- Routing:
  - 0 means use upper output
  - 1 means use lower output

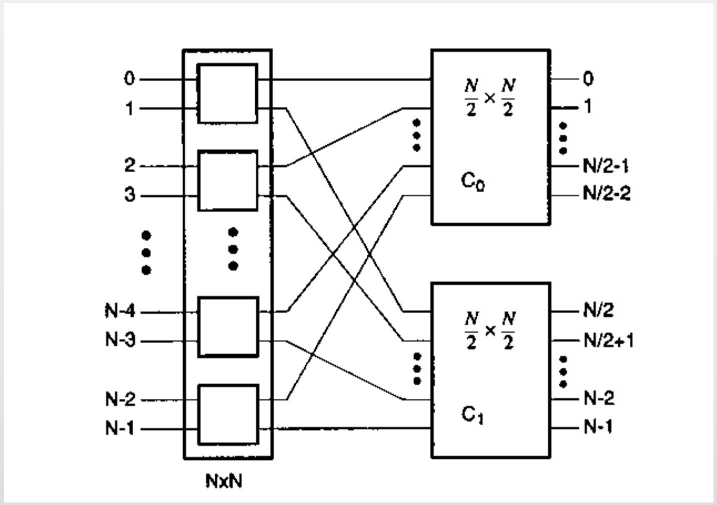
Omega networks

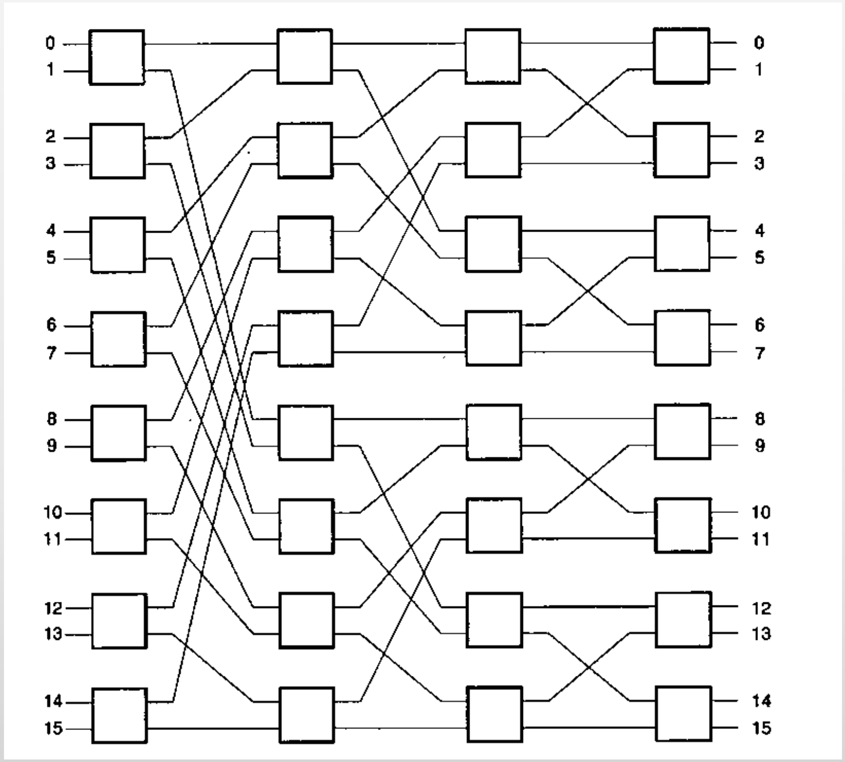
- $\log_2 N$ stages of 2×2 switches
- N/2 switches per stage
- $S=(N/2)\log_2(N)$ switches
- Number of permutations in an Omega network 2^S



Baseline networks

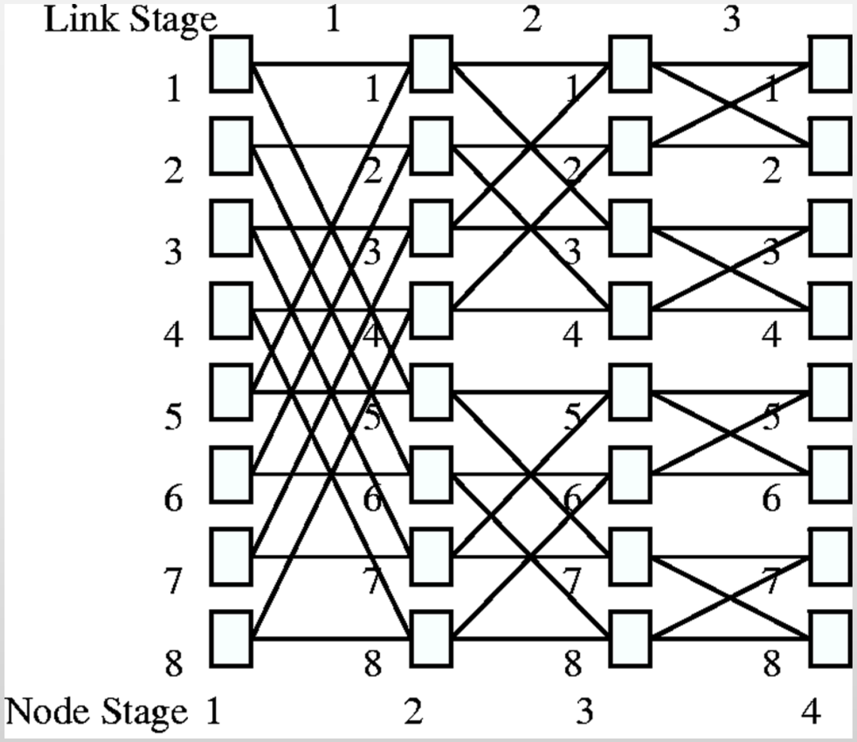
- The network can be generated recursively
- The first stage N × N, the second (N/2) × (N/2)

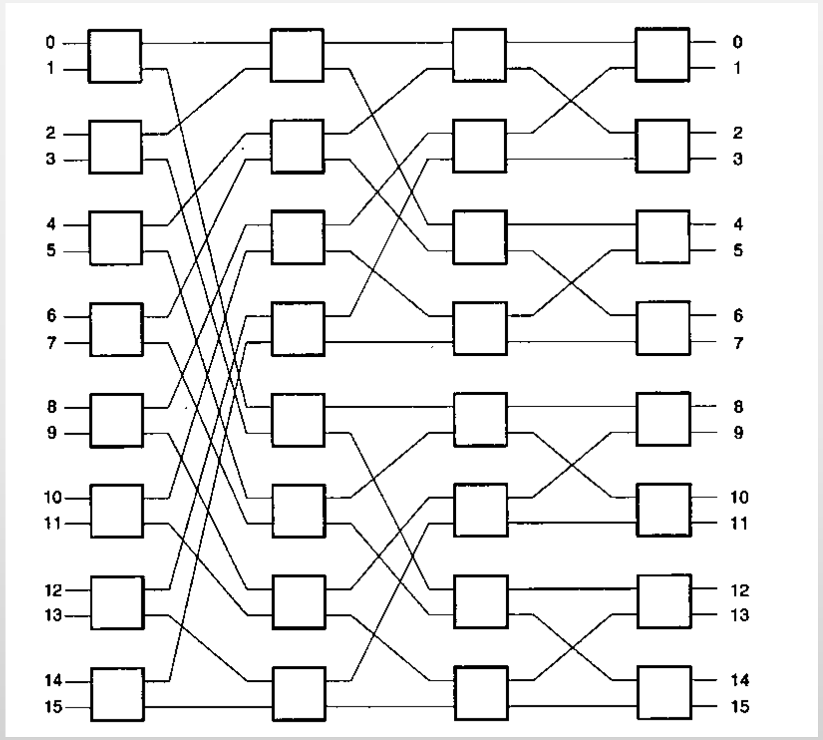

0
1
2
3
N-4
N-3
N-2
N-1
NxN
$\frac{N}{2} \times \frac{N}{2}$
C_0
$\frac{N}{2} \times \frac{N}{2}$
C_1
0
1
N/2-1
N/2-2
N/2
N/2+1
N-2
N-1



Topological equivalence

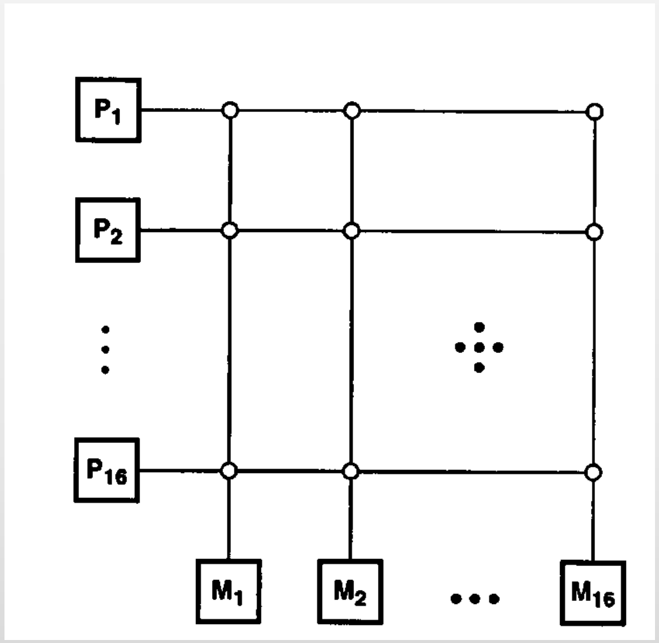
- Networks are topologically equivalent if one network can be easily reproduced from the other networks by simply rearranging nodes at each stage





Crossbar Network

- Each junction is a switching component – connecting the row to the column
- Can only have one connection in each column



Crossbar Network

- The major advantage of the crossbar switch is its **speed**
- In one clock, a connection can be made between source and destination
- Blocking if the destination is in use
- Because of its complexity, the **cost** of the crossbar switch can become the dominant factor for a large multiprocessor system
- Crossbars can be used to implement the $a{\times}b$ switches used in MIN’s
- In this case each crossbar is small so costs are kept down

Network properties

- *Node degree d* - the number of edges incident on a node
  - In degree
  - Out degree
- *Diameter* D of a network is the maximum shortest path between any two nodes
- The network is *symmetric* if it looks the same from any node
- The network is *scalable* if it expandable with scalable performance when the machine resources are increased