Lec 4

- Model Flynn's Classical Taxonomy:
  - Single Instruction, Single Data (SISD):
    - A serial (non-parallel) computer
    - the most common type of computer
    - only one data stream is being used as input during any one clock cycle
    - older generation mainframes, minicomputers and workstations; most modern day PCs
  - Single Instruction, Multiple Data (SIMD):
    - Single instruction: All processing units execute the same instruction at any given clock cycle
    - Multiple data: Each processing unit can operate on a different data element
    - Most modern computers, particularly those with graphics processor units (GPUs)
  - Multiple Instruction, Single Data (MISD):
    - A single data stream is fed into multiple processing units.
    - Each processing unit operates on the data independently via independent instruction streams.
    - Multiple cryptography algorithms
  - Multiple Instruction, Multiple Data (MIMD):
    - the most common type of parallel computer. Most modern computers fall into this category.
    - Multiple Instruction: every processor may be executing a different instruction stream.
    - Multiple Data: every processor may be working with a different data stream.
    - most current supercomputers, networked parallel computer clusters and "grids", multi-processor SMP computers, multi-core PCs.
    - many MIMD architectures also include SIMD execution sub-components
- what are the SIMD drawbacks?
  - cannot be applied to all algorithms
  - consumes quite a bit of power and occupies a great deal of space on the microprocessor chip
  - requires significantly more human interaction than traditional C or C++ programming
  - very architecture specific and require recoding to extend the application(s) that is using it across platforms
- Logical organization:
  - Control structure -> ways to express parallel tasks
  - Communication models -> mechanism for specifying interaction between tasks
    - Accessing shared data -> Platforms that provide a shared data space are called shared-address-space machines or multiprocessors
    - Exchanging messages-> Platforms that support messaging are called message passing platforms or multi-computers
- Shared address space is a programming abstraction.

- Shared memory is a physical machine attribute
- Accessing Shared data:
  - Part (or all) of the memory is accessible to all processors.
  - Processors interact by modifying data objects stored in this shared-address space.
  - Changes in a memory location effected by one processor are visible to all other processors (**global address space**).
- Shared memory machines can divided into two main classes **based upon memory access times**:
  - Uniform Memory Access (UMA)
    - ✓ Most commonly represented today by Symmetric Multiprocessors (SMPs).
    - ✓ Identical processors.
    - ✓ Each processor has equal access and access times to memory
    - ✓ Sometimes called **Cache Coherent UMA (CC-UMA)** if one processor updates a location in shared memory, all the other processors know about the update
  - Non- Uniform Memory Access (distributed) (NUMA)
    - ✓ Often made by physically linking two or more Symmetric Multiprocessors SMPs.
    - ✓ One SMP can directly access memory of another SMP.
    - ✓ Not all processors have equal access time to all memories.
    - ✓ Memory access across link is slower.
- Distributed memory systems require a communication network to connect inter-processor memory.
- Exchanging messages:
  - Processors have their own local memory, so:
    - ✓ Each one operates independently.
    - ✓ Changes it makes to its local memory have no effect on the memory of other processors.
    - ✓ the concept of cache coherency does not apply.
  - These platforms comprise of a set of processors and their own (exclusive) memory.
  - Instances of such a view come naturally from clustered workstations and **non-shared-address-space** multi-computers
  - These platforms are programmed using (variants of) send and receive primitives.
  - Principal functions send(), receive(), each processor has unique ID
  - Synchronization between tasks is the programmer's responsibility
  - Each processor P (with its own local cache C) is connected to exclusive local memory, i.e. no other CPU has direct access to it.
  - Each node comprises at least one network interface (NI) that mediates the connection to a communication network.
  - On each CPU runs a serial process that can communicate with other processes on other CPUs by means of the network
- Compare between shared address space and message passing platforms

Shared Address Space Platforms	Message Passing
can easily emulate message passing. The reverse is more difficult to do	requires little hardware support, other than a network

Shared Memory	Distributed Memory
Lack of scalability between memory and CPUs. Adding more CPUs can increases traffic on the shared memory and CPU path	Memory is scalable with number of processor, Increase the number of processors and the size of memory increases proportionally.
Data sharing between tasks is both fast and uniform due to the proximity of memory to CPUs	Each processor can rapidly access its own memory without interference and without overhead incurred with trying to maintain cache coherency.
Expensive	Cost effectiveness
Programmer responsibility for synchronization	Programmer responsible for many details associated with data communication between processors.
Global address space provides a user-friendly programming prespective to memory.	Difficult to map existing data structures, based on global memory, to this memory organization

Lec 5&6

- Natural extension of serial model (Random access machines RAM) → PRAM
- Processors share a common clock but may execute different instructions in each cycle.
- The PRAM was designed so that the user could design and analyze parallel algorithms without concern for communication
- Resolving Data Access Conflicts:
  - Exclusive-read, exclusive-write (EREW) PRAM.
  - Concurrent-read, exclusive-write (CREW) PRAM.
  - Exclusive-read, concurrent-write (ERCW) PRAM.
  - Concurrent-read, concurrent-write (CRCW) PRAM.
- Policies used to resolve Data Access Conflicts:
  - Priority CW: Processor are organized into a predefined priority list. processor with the highest priority succeeds and rest of processor fails.
  - Common CW: all processors attempting a simultaneously write to a given memory location will write the identical value.
  - Arbitrary CW: This model assumes that if multiple processors try to write simultaneously to a given memory location, then one of them, arbitrarily, will succeed.
  - Sum: Write the sum (or any associative operator) of all data items.
- PRAM Model:
  - Processors are connected to memory through a set of switches
  - These switches determined memory word being accessed by each processor
  - To ensure such connectivity, the total number of switches must be (m*P)
  - For a reasonable memory size, constructing a switching network of this complexity is very expensive
  - are impossible to realize in practice
- Interconnection Networks:
  - Set of links (physical media) -> set of wires or fibers for carrying information
  - Switches ->
- Interconnection networks can be classified as static or dynamic.

- Static networks (direct):
  - Consist of point-to-point communication links among processing nodes.
  - Complete network (clique), Linear, star, ring, tree, fat tree, hypercube, 2D&3D mesh
- Dynamic networks (indirect):
  - Are built using switches and communication links.
  - Communication links are connected to one another dynamically by the switches to establish paths among processing nodes and memory banks.
  - Bus-based, switch-based (crossbar, multi stage)
- Network Topologies:
  - These topologies tradeoff performance for cost.
  - Commercial machines often implement hybrids of multiple topologies for reasons of packaging, cost, and available components.
- Diameter: The maximum distance between any two processing nodes in the network. (number of hops through which a message in transferred on its way from one point to another)
- Bisection Width: The minimum number of wires you must cut to divide the network into two equal parts.
- Connectivity: the multiplicity of paths between any two processing nodes
- Cost: The number of links or switches besides the length of wires are factors in to the cost.
- Completely Connected:
  - Each processor is connected to every other processor.
  - While the performance scales very well, the hardware complexity is not realizable for large values of p.
  - Completely connected networks are static counterparts of crossbar.
- Star:
  - Every node is connected only to a common node at the center.
  - Distance between any pair of nodes is O(1).
  - The central node becomes a bottleneck.
  - In this sense, star connected networks are **static counterparts of buses**.
- linear -> Each node has two neighbors, one to its left and one to its right.
- Ring (1D) -> It is linear but the nodes at either end are connected.
- 2D & 3D mesh:
  - Has nodes with 4 neighbors, to the north, south, east, and west.
  - Good match for discrete simulation and matrix operations
  - Easy to manufacture and extend
- Hypercubes:
  - A special case of a d-dimensional mesh (d = log2 p, where p is the total number of nodes)
  - Each node has log p neighbors.
  - The distance between two nodes is given by the number of bit positions at which the two nodes differ.
  - costly/difficult to manufacture for high n, not so popular nowadays
- Tree:
  - The distance between any two nodes is no more than 2log p.
  - Links higher up the tree potentially carry more traffic than those at the lower levels.
  - a variant called a fat-tree, fattens the links as we go up the tree.
  - Trees can be laid out in 2D with no wire crossings. This is an attractive property of trees.

- tree suffers from a communication bottleneck at higher levels of the tree (specially if right part of tree try sending to left part) how to solve it ?
  - ✓ Solution a FAT tree increased number of communication links and switching nodes closer to root

- Fat Tree Network:
  - there was n path (n -> number of level-current level +1) between any two pairs of nodes
- Complete network (clique), Star network, Linear array, Ring, Tree -> none of them is very practical

Network	Diameter	Bisection Width	Arc Connectivity	Cost (No. of links)
Completely-connected	1	$p^2/4$	$p-1$	$p(p-1)/2$
Star	2	1	1	$p-1$
Complete binary tree	$2\log((p+1)/2)$	1	1	$p-1$
Linear array	$p-1$	1	1	$p-1$
2-D mesh, no wraparound	$2(\sqrt{p}-1)$	$\sqrt{p}$	2	$2(p-\sqrt{p})$
2-D wraparound mesh	$2\lfloor\sqrt{p}/2\rfloor$	$2\sqrt{p}$	4	$2p$
Hypercube	$\log p$	$p/2$	$\log p$	$(p\log p)/2$

- Dynamic Interconnection Networks:
  - Bus based network
  - Crossbar network
  - Multistage networks
- Bus based networks:
  - Some of the simplest and earliest parallel machines used buses.
  - All processors access a common bus for exchanging data.
  - The distance between any two nodes is O(1) in a bus.
  - the bandwidth of the shared bus is a major bottleneck (scalable in terms of cost and non scalable in terms of performance).
- Crossbar networks:
  - Simple way to connect P processor to b memory banks
  - Total number of switching nodes required to implement such network is O(pb).
  - As the number of processing nodes becomes large, this switch complexity is difficult to realize at high data rates.
  - crossbar networks are not very scalable in terms of cost.
- Multistage networks:
  - Crossbars have excellent performance scalability but poor cost scalability.
  - Buses have excellent cost scalability, but poor performance scalability.
  - Multistage interconnects strike a compromise between these extremes.
  - It is more scalable than the bus in terms of performance and more scalable than the crossbar in terms of cost
- Omega network
  - One of the most commonly used multistage interconnects
  - This network consists of log p stages, where p is the no of processor and no of memory.
  - Every stage consists of an interconnection pattern connect p input & p output
- Number of stages = $\log_2 p$

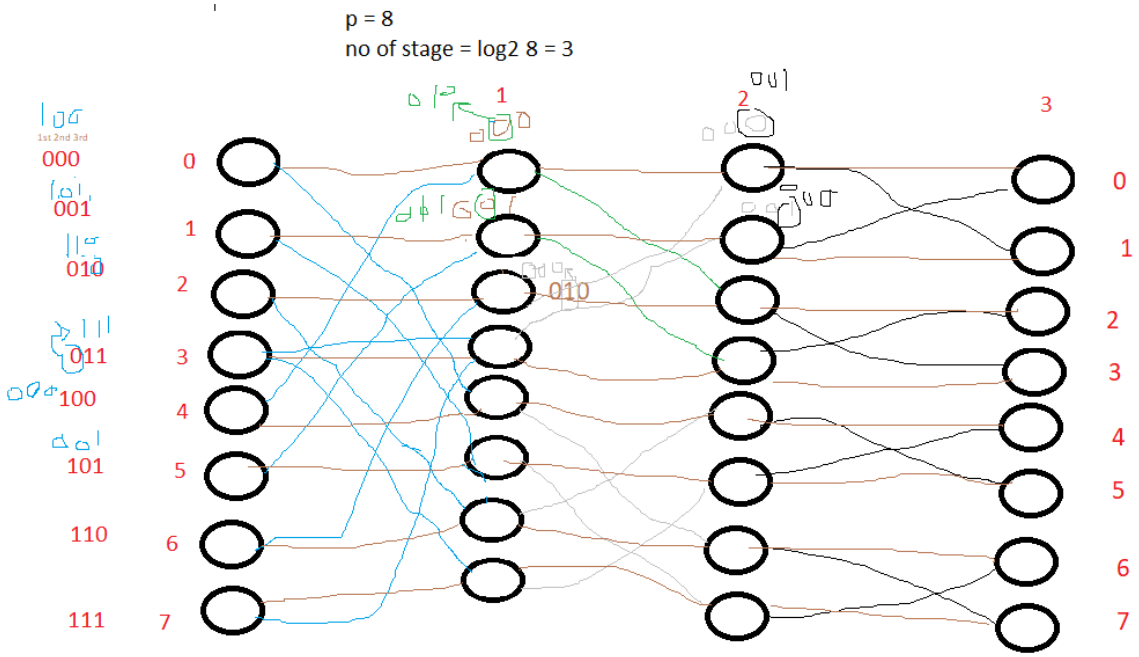
- Number of switches at each stage = p/2
- Total Number of Switches =(P/2) * ($\log_2$ P)

$$j = \begin{cases} 2i, & 0 \le i \le p/2 - 1 \\ 2i + 1 - p, & p/2 \le i \le p - 1 \end{cases}$$

- Simple routing algorithm is done by comparing the bit-level representation of source and destination addresses.
- pass-through connection -> inputs are sent straight through to the outputs
- cross-over connection –> the inputs to the switching node are crossed over and then sent out

Network	Diameter	Bisection Width	Arc Connectivity	Cost (No. of links)
Crossbar	1	p	1	p^2
Omega Network	$\log p$	$p/2$	2	$p/2$

- Butterfly network :



1st 2nd 3rd

ال processor رقم 0 -> 0 0 0 لو هشوف انا في اي stage وهقلب الرقم اللي بيعبرعنه يعني لو 0 هيبقى 1 والعكس 000 <-
100 ولو في stage 2 000 -> 010 ولو في stage 3 000 -> 001 وهكذا