Processor Organizations :-

* Mesh
* Pyramid
* Shuffle - exchange network
* Butterfly network
* Hypercube (Cube Connected)
* Cube connected Cycles

Processor Arrays:-

* SIMD Model
* Mesh Connected SIMD Model
* Cube Connected SIMD Model

Multiprocessors :-

* Tightly coupled multiprocessors
* Loosely coupled multiprocessors
* Multicomputers

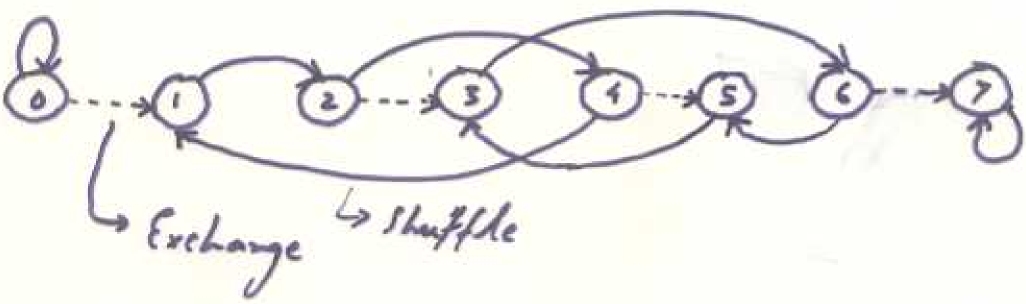
Pyramid

- A pyramid network of size p is a complete 4-ary rooted tree of height $\log_4 p$ augmented with additional inter-processor links so that the processors in every tree level from a two dimensional mesh network.
- A pyramid of size p has at its base a 2-D mesh network containing $p = k^2$ processors.
- The total number of processors in a pyramid of size p is $\frac{4}{3}p - \frac{1}{3}$.

- The levels of the pyramid are numbered in ascending order. The base has level number 0, and the apex of the pyramid has level number $\log_4 p$.

Shuffle Exchange Network :-

- A shuffle exchange network consists of $n = 2^k$ nodes, numbered 0, 1, -- n-1 and two kinds of connections - Shuffle & Exchange.
- Exchange connections link pairs of nodes whose numbers differ in their least significant bit.
- The perfect shuffle connections link node i with node $2i$ modulo $n-1$, with the exception that node $n-1$ is connected to itself.



Butterfly :-

- Butterfly network consists of $(k+1)2^k$ nodes divided into $k+1$ rows or ranks, containing $n = 2^k$ nodes each.
- The ranks are labelled 0 thro' k, although the ranks 0 & k are sometimes identified, giving each nodes four connections to other nodes.

DESIGNING PARALLEL ALGORITHMS

- There are atleast three ways to design a parallel algorithm :-
  * Detect & exploit any inherent parallelism in an existing sequential algorithm
  * Invent a new parallel algorithm
  * Adapt another parallel algorithm that solves a similar problem

Generalities :-

* Insight has an important role
  - To solve a particular problem, a well known sequential algorithm may be chosen as the starting point
  - If the sequential algorithm is not particularly parallelizable, then some external knowledge may be applied to break it up.
  - Let us consider a problem of finding sum of n integers where $n > 0$

```
SUMMATION (SISD)
begin
      sum ← a0
      for i ← 1 to n-1 do
            sum ← sum + ai
      endfor
end
```

For, $n = 4$, The summation is $[(a_0 + a_1) + a_2] + a_3$ → Sequential form

but, $(a_0 + a_1) + (a_2 + a_3)$ → Parallel form

* Communication costs Must be considered
  - It is a mistake to ignore communication costs in determining the complexity of a parallel algorithm.
  - Sometimes the communication complexity is higher than the computational complexity.
* The algorithm must fit the architecture
  - The performance of an algorithm can be different on different architecture.

Algorithms for Processor Arrays :-

Assume that $n = 2^m = l^2$, where m & l are positive integers and the n values to be added.
Denoted $A = (a_0, a_1, \ldots a_{n-1})$ are distributed one value per processing element.

Let, $p \rightarrow$ Number of processors

$P_i \rightarrow$ Denotes the processing elements

Parallelism is indicated by using for all loop.
The for all statement activates a set of processors that executes the statements before a matching end for.

$\Leftarrow$ denotes communication of data item from an adjacent processors local memory into the active processors local memory.

SIMD-CC (Cube Connected) Model :-

```
SUMMATION (SIMD-CC)
begin
  for i ← log n - 1 down to 0 do
     d ← 2^i
     for all P_j, where 0 ≤ j < d do
        t_j ⇐ a_{j+d}
        a_j ← a_j + t_j
     end for
  end for
end
```

The algorithm adds $n = 2^m$ values. Processor P_i possesses local variables a_i and t_i for all i such that $0 \leq i \leq n-1$.

When the algo. begins execution, the a_i's contain the values to be added. At its termination a_0 contains the sum.

Complexity → $\Theta(\log n)$

SIMD-PS (Perfect Shuffle) Model :-

```
SUMMATION (SIMD-PS)
begin
  for i ← 1 to log n do
     for all P_j, where 0 ≤ j < n do
        shuffle (a_j)
          b_j ← a_j
        exchange (b_j)
          a_j ← a_j + b_j
     end for
  end for
end
```

Complexity - $\Theta(\log n)$

SIMD-MC² (Mesh Connected with dimension 2) Model:-

```
SUMMATION (SIMD-MC²)
begin
  for i ← l-1 down to 1 do
    for all P_{j,i} where 1 ≤ j ≤ l do {Column i active}
        t_{j,i} ⇐ a_{j,i+1}
        a_{j,i} ← a_{j,i} + t_{j,i}
    end for
  end for
  for i ← l-1 downto 1 do
    for all P_{i,1} do { Only a single processing element active}
        t_{i,1} ⇐ a_{i+1,1}
        a_{i,1} ← a_{i,1} + t_{i,1}
    end for
  end for
end
```

Complexity $\Theta(\sqrt{n})$ as it requires $2(l-1) = 2\sqrt{n} - 2$ Constant time iterations.

Developing Algorithms for MIMD Computers :-

- MIMD Computers enable the asynchronous execution of multiple instructions streams. Thus the designer of algorithms for MIMD Computers has more flexibility.

Categorization of MIMD Algorithms :-

* Pipelined Algorithms
* Partitioned Algorithms
* Relaxed Algorithms

Pipelined Algorithms :-

- A pipelined algorithm is an ordered set of segments in which the output of each segment is the input of its successor.
- The input to the algorithm serves as the input to the first segment, the output of the last segment is the output of the algorithm.
- Some authors use the term "Macropipelining" to refer to this kind of algorithm.
- A "Systolic algorithm" is a special kind of pipelined algorithm. Three attributes distinguish systolic algos.
  - The flow of data is rhythmic & regular.
  - Data can flow in more than one direction
  - The computations performed at each segment are essentially identical.

Partitioned Algorithms :-

- Partitioning is the sharing of a computation. A problem is divided into subproblems that are solved by individual processors.
- The solutions to the subproblems are then combined to form the problem solution.
- This pooling of solutions implies synchronization of processors.
- For this reason, Partitioned algorithms are also called "Synchronized Algorithms."
- Partitioned algorithms can be divided into two categories
  * Prescheduled algorithms - Each process is allocated its share of computation at the compile time.
  * Self scheduled Algorithms - Each process is assigned it's work at run time.

Relaxed Algorithms :-

- An algorithm that works without process synchronization is said to be relaxed.
- Also known as Asynchronous algorithms.

eg: summation (Tightly Coupled Multiprocessor)

```
begin
    global_sum ← 0
    for all P_i, where 0 ≤ i ≤ p-1 do
        local_sum ← 0
        for j ← i to n step p do
            local_sum ← local_sum + a_j
        end for
        lock (global_sum)
            global_sum ← global_sum + local_sum
        unlock (global_sum)
    end for
end
```

Process Communication & Synchronization on MIMD Models:-

Expressing Concurrency:-

- Fork and Join
  - The fork statement is similar to a procedure call in the sense that it enables the commencement of a particular routine.
  - However unlike a procedure call, the calling process continues execution.
  - Hence, a single process is forked into two processes.
  - The invoking process can synchronize with termination of forked process by executing the join statement.

- Cobegin & Coend :-
  - In contrast to the unstructured fork & join statements, Cobegin & Coend are a structured way of indicating a set of statements that can be executed in parallel.

  eg: Cobegin $S_1 \parallel S_2 \parallel \ldots \parallel S_n$ Coend.

- Process Declarations :-
  - It is common for a concurrent program to be made up of a number of sequential procedures operating concurrently.

Synchronization :-

- Mutual Exclusion
- Busy waiting
- Semaphores
- Monitors

Low Level Synchronization :.

- Message Passing

Task Scheduling in MIMD Computers :-

- Deterministic Modelling _ Gantt Charts
- Non deterministic Modelling. - Random Variables.