Lecture 29-30: October 27, 2021
Computer Architecture and Organization-II
Biplab K Sikdar

Parallel Computers -II

0.3.1.1 EREW shared memory SIMD computers

EREW stands for exclusive read and exclusive write to the same memory location.

No two PEs are allowed to simultaneously read or write to same memory location.

Property of an EREW SIMD computer is summarized in Figure 7.

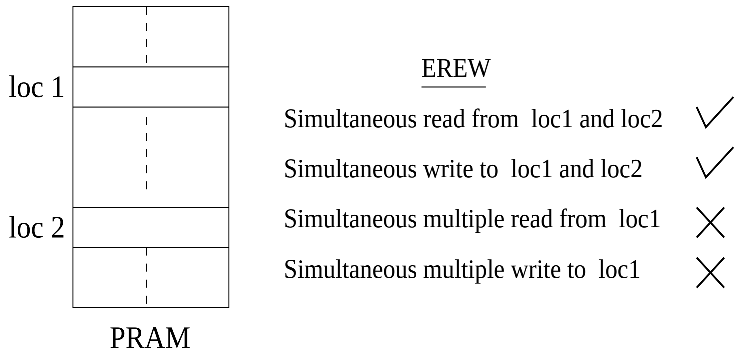

loc 1
loc 2
PRAM
EREW
Simultaneous read from loc1 and loc2 ✓
Simultaneous write to loc1 and loc2 ✓
Simultaneous multiple read from loc1 ✗
Simultaneous multiple write to loc1 ✗

Figure 7: EREW PRAM properties

Performance of EREW model:

Let consider searching of an item x in a table of n elements.

It can be done by EREW SIMD computer of N$\leq n$ PEs. Solution requires

1. Broadcasting of x to PEs so that x is available to them.
2. Comparison of x with the elements of table.
3. Storing of the outcome of search by each PE.

1. Broadcasting of x to PEs

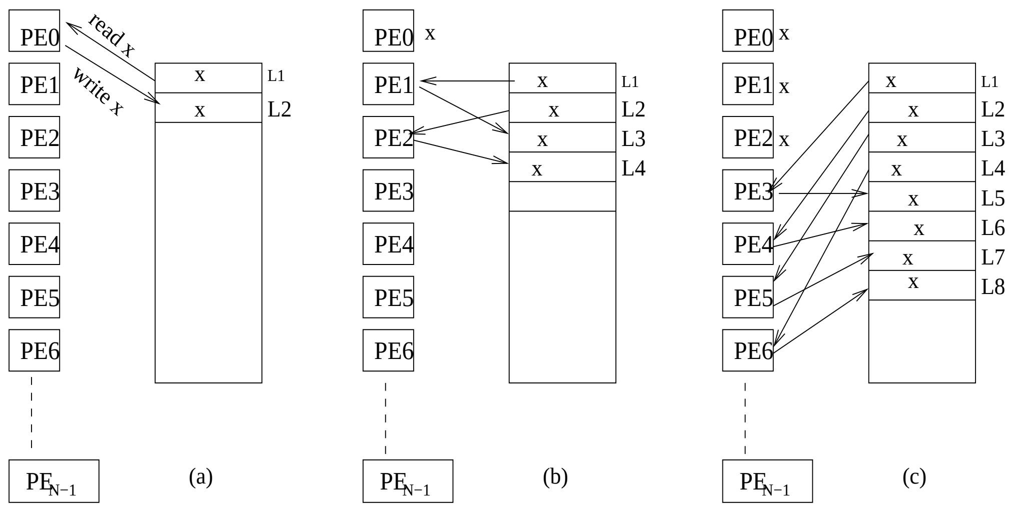


Figure 8: Broadcast

In an EREW computer, broadcasting can be realized as described next.

Let x is in memory location L_1 (Figure 8(a)).

In time step 1, x is read by PE_0 from L_1 and then PE_0 writes it to L_2 (Figure 8(a)).

In next time step, PE_1 and PE_2 both simultaneously read x from L_1 and L_2 and they write x to L_3 and L_4 (Figure 8(b)).

Now, x is available in 4 memory locations.

Another four PEs (say, PE_3 - PE_6) can have x in next time step (Figure 8(c)).

Such a process can ensure that in $\log N$ time steps all N PEs can get x.

2. Comparison of x with the elements of table

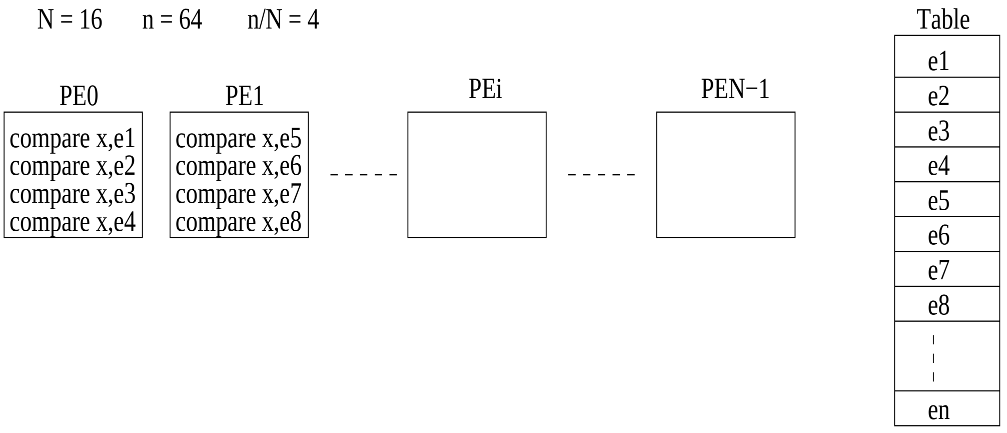


Figure 9: Comparison

The number of PEs (N) is less or equal to number of elements (n) to be searched.

Therefore, each processor compares x with $\frac{n}{N}$ elements.

Comparisons within a PE are sequential (Figure 9).

Ccomarisons in PE$_i$ and PE$_j$, $\forall_{i,j}$ are in parallel.

A comparison requires O(1) time.
Therefore, comparisons of x with $\frac{n}{N}$ table elements in PE require O($\frac{n}{N}$) time.

As PEs perform comparisons in parallel, total comparison time for n elements is O($\frac{n}{N}$) time.

3. Storing of the outcome of search by each PE

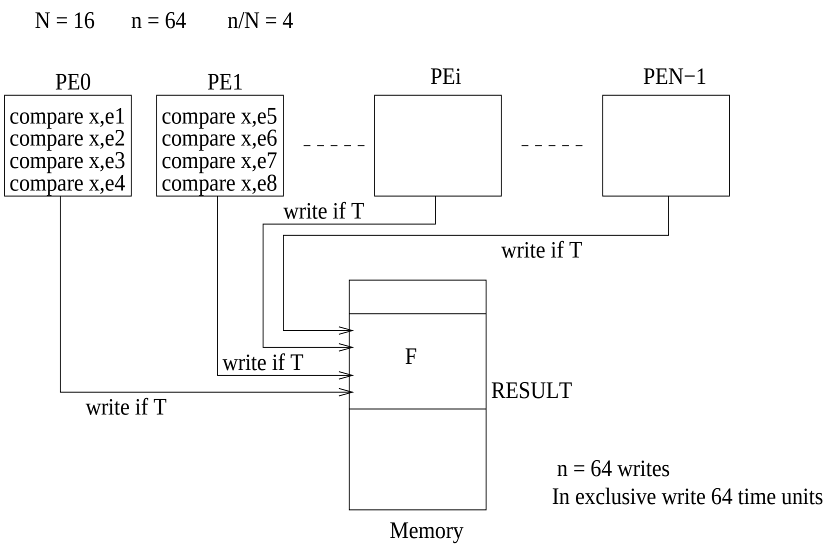


Figure 10: Exclusive write

Each PE stores outcome of $\frac{n}{N}$ comparisons (x with $\frac{n}{N}$ elements).

Location RESULT, initialized as F, is set to T if PE finds a match,
otherwise PE skips write to RESULT (Figure 10).

Time taken to write T to location RESULT is O(n) in an EREW m/c.

In worst case, each of N PEs may find search results positive for all
$\frac{n}{N}$ comparisons done by it and need to write 'T' to RESULT.

As write to same location RESULT is exclusive in EREW,
it requires $\frac{n}{N}$ sequential writes by a PE.

That is, in total n writes by all the PEs in O(n) time.

That is, searching time for x, in an EREW computer with N PEs, is

$$O(\log N) + O(\tfrac{n}{N}) + O(n) = O(n).$$

When number of PEs is same as table size (N=n), then searching for item x requires

$$O(\log n) + O(1) + O(n) = O(n).$$

0.3.1.2 CREW shared memory SIMD computers

CREW is the concurrent read exclusive write machine.

Here, multiple processors are allowed to read from same memory location but right to write to a memory location is exclusive (Figure 11).

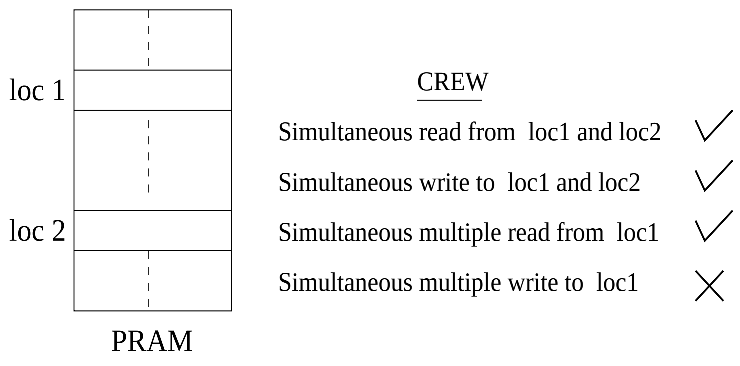

loc 1
loc 2
PRAM
CREW
Simultaneous read from loc1 and loc2 ✓
Simultaneous write to loc1 and loc2 ✓
Simultaneous multiple read from loc1 ✓
Simultaneous multiple write to loc1 ✗

Figure 11: CREW PRAM properties

Searching of x from a table of n elements, in CREW m/c with N PEs

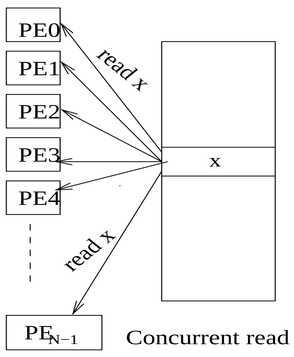


Figure 12: Concurrent read

All PEs can get x in O(1) time in an CREW computer (Figure 12).

As in EREW, each PE compares x with $\frac{n}{N}$ elements (Figure 13).

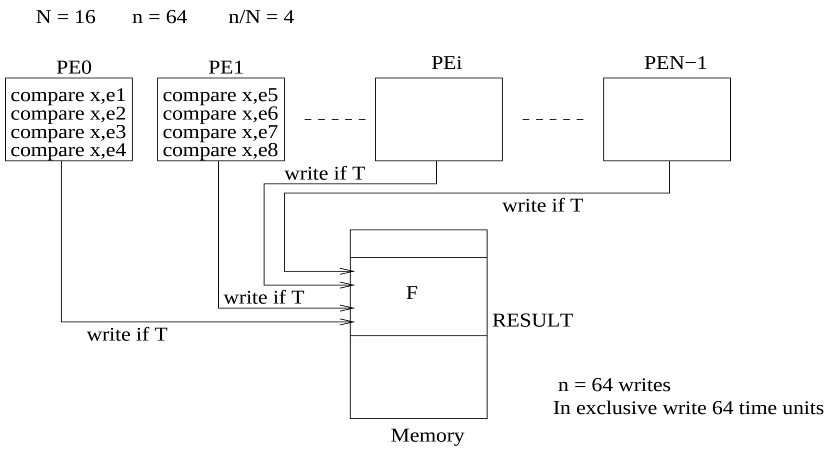


Figure 13: Comparison and exclusive write

Comparisons within a PE are sequential. Comarisons require O($\frac{n}{N}$) time.

Write to same location to store outcome (RESULT) is exclusive as in EREW.

In total n writes by all PEs in O(n) time (Figure 13).

That is, searching time in CREW with N processing elements, is

$$O(1) + O(\frac{n}{N}) + O(n) = O(n).$$

0.3.1.3 ERCW shared memory SIMD computers

ERCW is the exclusive read concurrent write machine.

In ERCW, multiple PEs are simultaneously allowed to write to same location but read access from a location is exclusive (Figure 14).

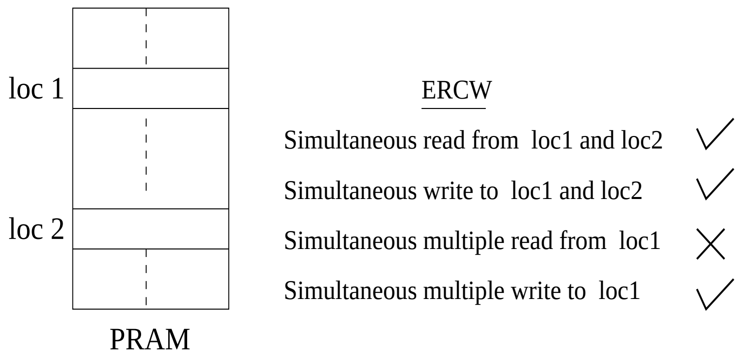

loc 1
loc 2
PRAM
ERCW
Simultaneous read from loc1 and loc2 ✓
Simultaneous write to loc1 and loc2 ✓
Simultaneous multiple read from loc1 ✗
Simultaneous multiple write to loc1 ✓

Figure 14: ERCW PRAM properties

Searching of x from a table of n elements, in ERCW m/c with N PEs

All the N PEs of ERCW get x in O(logN) time as in EREW computer.

Comparison of x with n table elements require O($\frac{n}{N}$) time.

In worst case, each PE finds match (x and element) and try to store T to RESULT.

In ERCW, simultaneous write to RESULT is allowed.

Assume - after a comparison PEs may send write request -that is,

N such write requests can be completed in O(1) time (Figure 15).

For $\frac{n}{N}$ instance of comparison in each PE, writes to RESULT require O($\frac{n}{N}$) time.

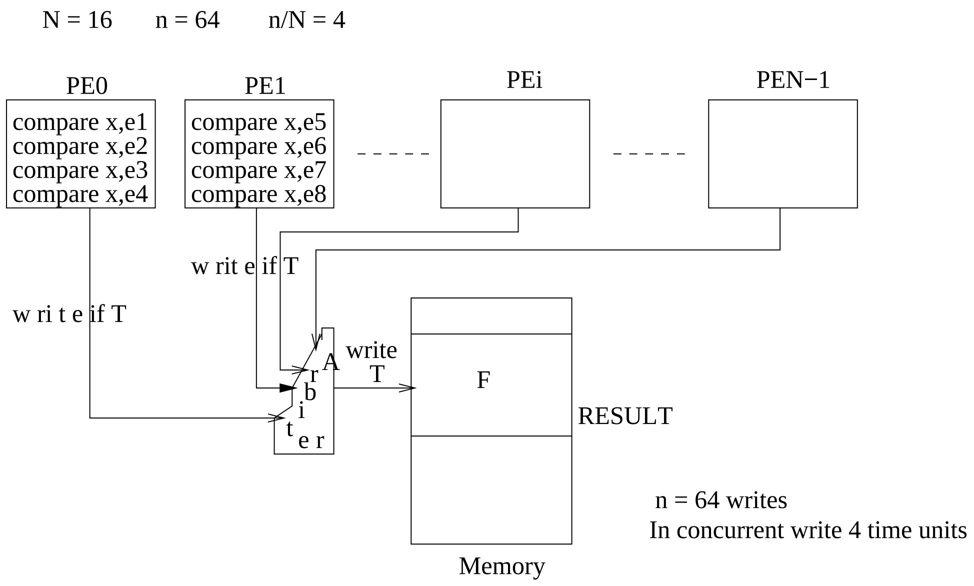


Figure 15: Comparison and concurrent write

That is, total time for searching, in an ERCW computer with N PEs, is

$$\text{O}(\log \text{N}) + O(\tfrac{n}{N}) + \text{O}(\tfrac{n}{N}) = \text{O}(\tfrac{n}{N}).$$

Concurrent write can be implemented with an arbiter

when all writes are same (here write T) or take sum of all and then write sum.

0.3.1.4 CRCW shared memory SIMD computers

CRCW stands for concurrent read/concurrent write to same location (Figure 16).

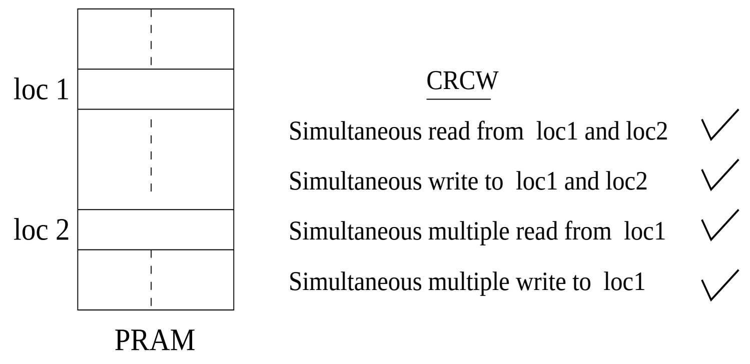

loc 1
loc 2
PRAM
CRCW
Simultaneous read from loc1 and loc2 ✓
Simultaneous write to loc1 and loc2 ✓
Simultaneous multiple read from loc1 ✓
Simultaneous multiple write to loc1 ✓

Figure 16: CRCW PRAM properties

Searching of x from a table of n elements, in CRCW m/c with N PEs

Concurrent read allows broadcasting of x to all PEs in O(1) time.

Comparison of x with n elements and storing of outcome of comparisons require O($\frac{n}{N}$) time as in ERCW computer.

That is, total time for searching of x in a table of n elements, in CRCW is

$$O(1) + O(\tfrac{n}{N}) + O(\tfrac{n}{N}) = O(\tfrac{n}{N}).$$

0.3.2 Feasibility of SM model of SIMD computers

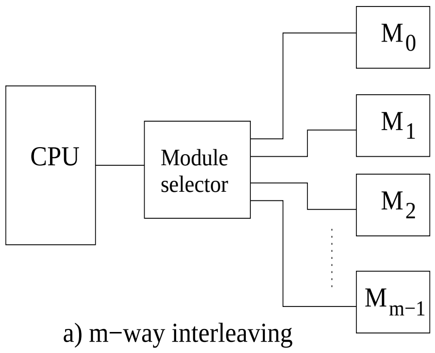
SM SIMD machine is not a realistic design.

Minimum requirement is - memory subsystem supports

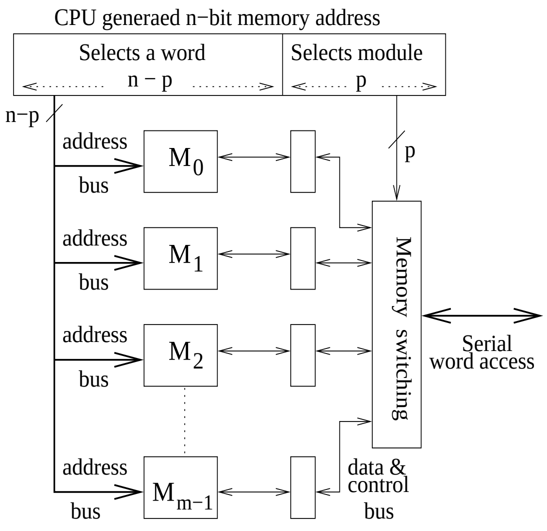
simultaneous read as well as simultaneous write to different locations.

It leads to an EREW SIMD computer.

Interleaved memory provides a platform that is close to EREW concept.



a) m−way interleaving



b) Memory module addressing

Memory address (n)	Module (p)	Word (n−p)
0000..........01 00	0	1
0000..........01 01	1	1
0000..........01 10	2	1
0000..........01 11	3	1
0000..........10 00	0	2
0000..........10 01	1	2

c) Address mapping (m=4)

Figure 17: Memory interleaving

Follow Figure 17.

SIMD computer of N PEs can simultaneously access N different memory locations,

where locations fall into N different memory modules of interleaved memory.

Realization of concurrent read from same memory location, may not be realizable.

Multi-port memory can be a solution - but it is expensive and not scalable.

Approximations of concurrent write to a memory location are done.

An arbiter is set to control concurrent write.

Semantics of concurrent write in ERCW/CRCW PRAMs are shown in Figure 18.

1. When all requests target writing of the same value, write any one.
2. Pick one write operation at random.
3. When PEs are assigned priority, allow a write with highest priority.
4. Write sum of all values attempting to be written.

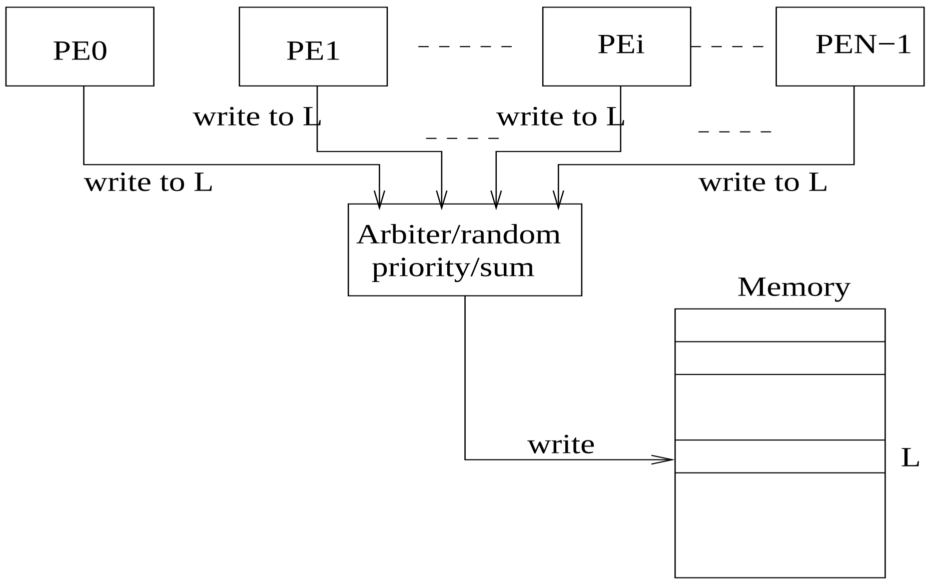


Figure 18: Concurrent write schemes

This signifies - in reality, we design an EREW SIMD or a weaker machine than that.

0.3.3 Interconnection network SIMD computers

Here, each processor or PE has its own local memory.

Data transfer between two PEs is realized directly via interconnection network.

Commonly used basic interconnection topologies in SIMD architecture are

linear array, 2-d array or mesh, binary tree, cube, hypercube, shuffle network etc.

0.3.3.1 Linear array

In linear array or one-dimensional mesh, each PE is connected to its neighbor.

Link between two PEs can either be unidirectional or bidirectional (Figure 19).

A PE/processor P_i when receives a data, accepts the data destined for it.
If data is not destined for P_i, it relays data to its next neighbor P_{i+1}.

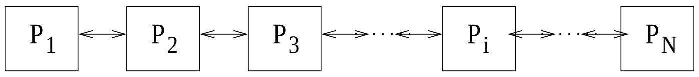


Figure 19: Linear array

Major advantages of linear array are -

1. It is simple to realize.
2. It is scalable. Addition of a new processor requires only one additional link.
3. All links (hops) in linear array can be of same length.

   Therefore, estimation of communication delay is straight forward.

On the other hand, major drawbacks of a linear array are -

1. It is of poor fault tolerance. A failure causes disruption in communication.
2. Bandwidth in linear array can be a bottleneck.

0.3.3.2 Two-dimensional array (mesh)

Common two-dimensional interconnection network is obtained

by arranging N processors into an $m \times m$ array, where $m = \sqrt{N}$ (Figure 20).

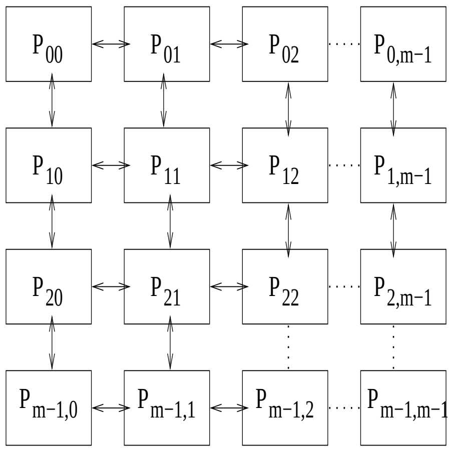


Figure 20: Two-dimensional mesh

Here, each intermediate processor node is connected to four neighbors.

Following are key features of two-dimentional mesh architecture-

1. All links (hops) in such network have same length.
2. Cost: is proportional to number of processor nodes (N).
3. Scalability is poorer than linear array structure.
4. Maximum network delay is O($\sqrt{N}$).
5. Better reliability than linear array. More than one path from source to destination enable better tolerance against failure in link.

Similar to two-dimensional array, a K-dimensional mesh can also be constructed.
Here, each intermediate processor node has 2^K neighbors.
Maximum delay, in such a network, is O($N^{\frac{1}{K}}$).

0.3.3.3 Binary tree connection

Here, processors form a complete binary tree.

Each intermediate processor node is having three neighbors (Figure 21).

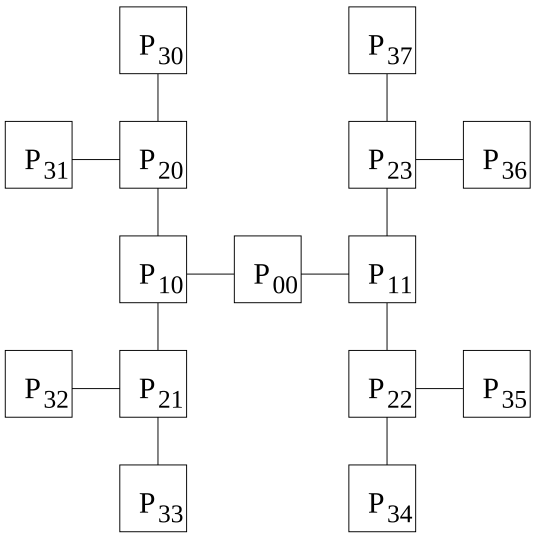


Figure 21: Tree network

Cost of a tree network is O(N), where N is number of processor nodes.

Maximum delay in communication between two processors is $O(\log_2 N)$.

0.3.3.4 Cube connection

Figure 22(a) shows a 3-cube network, conventionally referred to as cube network.

The 3-cube network has 8 nodes.

A 4-cube (hypercube) network is shown in Figure 22(b). It has 16 nodes.

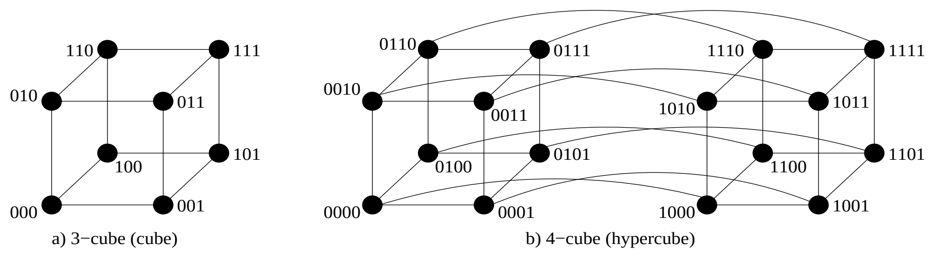


Figure 22: Hypercube network

If number of processors in an SIMD machine is $N = 2^q$, where $q \geq 1$,

then a q-dimensional cube or hypercube can be obtained.

Each node of q-dimensional hypercube is having q neighbors.

Neighbor of P$_k$ is P$_l$, where l is obtained by complementing a single bit of k.

If source P$_k$ = P$_{100}$, in 3-cube, then its three neighbors are

Cube$_2$(100) = P$_{000}$, Cube$_1$(100) = P$_{110}$, and Cube$_0$(100) = P$_{101}$.

That is,

If source is P$_k$ $(p_{q-1}p_{q-2}\cdots p_i \cdots p_1p_0)$; k = 0, 1, $\cdots$, 2^q-1, then its i^{th} neighbor is

Cube$_i$(P$_k$ $(p_{q-1}p_{q-2}\cdots p_i \cdots p_1p_0)$) = P$_l$$(p_{q-1}p_{q-2}\cdots p_i' \cdots p_1p_0)$; $\forall_{i=0\ to\ (q-1)}$

0.3.3.5 Shuffle network

Let consider N processors P_0, P_1, P_2, $\cdots$, P_{N-1}; N is a power of 2.

In perfect shuffle interconnection, there is an one-way link P_i to P_j, where

$$\begin{aligned} j &= 2i, \text{ for } 0 \le i \le \tfrac{N}{2} - 1 \\ &= 2i+1-N, \text{ for } \tfrac{N}{2} \le i \le N - 1 \end{aligned}$$

It defines that binary representation of j is obtained by

Cyclically 1-bit left shifting the binary representation of i.

That is, $j_{m-1}j_{m-2}\cdots j_1j_0$ = Shuffle($i_{m-1}i_{m-2}\cdots i_1i_0$) = $i_{m-2}i_{m-3}\cdots i_1i_0i_{m-1}$.

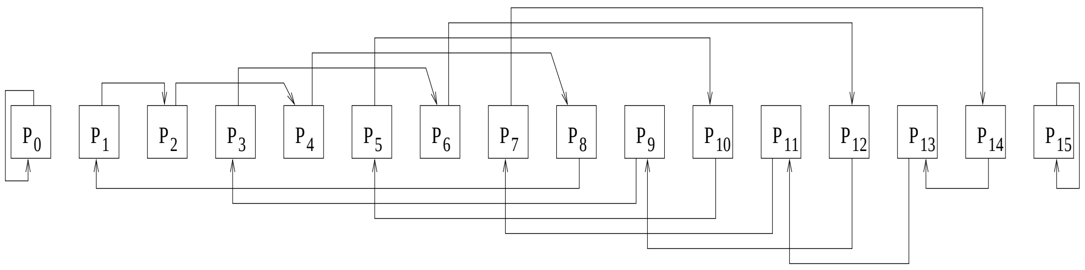


Figure 23: Perfect shuffle network

Figure 23 is an example of perfect shuffle network.

Example 0.1 *Let consider a system with 16 nodes of Figure 23.*

The shuffle operation S(0100) $\rightarrow$ 1000 at node 4 enables transfer of data to node 8.

Shuffle-exchange network

In addition to shuffle link, there is a 2-way link between every even numbered PE and its successor (—— dotted line).

That is, there is an additional exchange link from P_i to P_j, where

$$j_{m-1}j_{m-2}\cdots j_1j_0 = \text{Exchange}(i_{m-1}i_{m-2}\cdots i_1i_0) = i_{m-1}i_{m-2}\cdots i_1i'_0$$

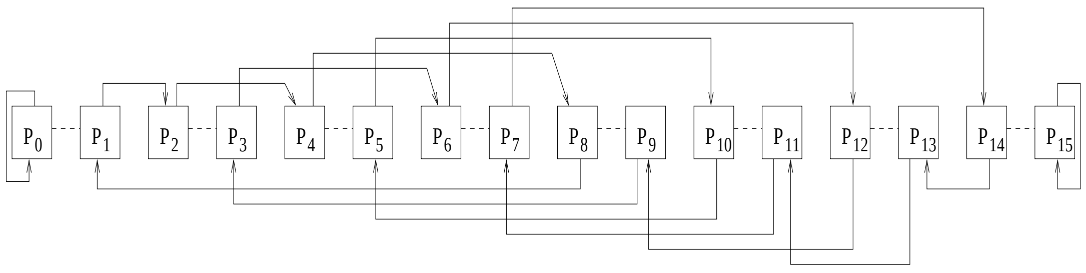


Figure 24: Shuffle-Exchange network

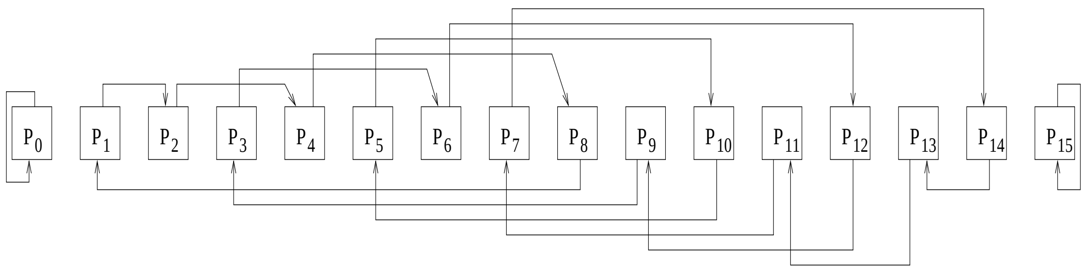


Figure 25: Perfect shuffle network

Figure 24 is an example of shuffle-exchange network.

Example 0.2 *Consider Figure 24.*

Shuffle operation S(0010) $\rightarrow$ 0100 at node 2 enables transfer of data to node 4.

Similarly, E(0100) $\rightarrow$ 0101 enables node 4 to node 5 data transfer.