**Solution**

1. **TORUS INTERCONNECTION NETWORK**

Torus interconnect is a switch-less topology that can be seen as a [mesh interconnect](https://en.wikipedia.org/wiki/Mesh_networking) with nodes arranged in a [rectilinear](https://en.wikipedia.org/wiki/Rectilinear_grid) array of N = 2, 3, or more dimensions, with processors connected to their [nearest neighbors](https://en.wikipedia.org/wiki/Nearest_neighbor_graph), and corresponding processors on opposite edges of the array connected. [In this lattice](https://en.wikipedia.org/wiki/Torus), each node has 2N connections. This topology got the name from the fact that the lattice formed in this way is topologically homogeneous to an [N-dimensional](https://en.wikipedia.org/wiki/N-dimensional) [torus](https://en.wikipedia.org/wiki/Torus).

The first 3 dimensions of torus topology network are easier to visualize. Below are the descriptions respectively.

* 

Fig1: Illustration of 1D Torus

* 

Fig2: Illustration of 2D Torus

* 

Fig3: Illustration of 3D Torus

* 1D Torus: it is one dimension, *n* nodes are connected in closed loop with each node connected to its 2 nearest neighbors communication can take place in 2 directions, +x and −x. 1D torus is same as [ring interconnection](https://en.wikipedia.org/wiki/Ring_network).
* 2D Torus: it is two-dimension with degree of 4, the nodes are imagined laid out in a two-dimensional rectangular lattice of n rows and n columns, with each node connected to its 4 nearest neighbors, and corresponding nodes on opposite edges connected. The connection of opposite edges can be visualized by rolling the rectangular array into a "tube" to connect two opposite edges and then bending the "tube" into a torus to connect the other two. Communication can take place in 4 directions, +x, −x, +y, and −y. The total nodes of 2D Torus is *n*2
* 3D Torus: it is three dimension, the nodes are imagined in a three-dimensional lattice in the shape of a rectangular prism, with each node connected with its 6 neighbors, with corresponding nodes on opposing faces of the array connected. Each edge consists of *n* nodes. Communication can take place in 6 directions, +x, −x, +y, −y, +z, −z. Each edge of 3D Torus consists of n nodes. The total nodes of 3D Torus is *n*3
* ND Torus: it can have *N* dimension, each node of *N* dimension torus has 2N neighbors and communication can take place in 2N directions. Each edge is consists of n nodes. Total nodes of this torus is *nN*. The main motivation of having higher dimension of torus is t achieve higher bandwidth, lower latency, and higher scalability.

Higher-dimensional arrays are difficult to visualize but we can see from above rule that each higher dimension adds another pair of nearest neighbor connections to each node.

**ADVANTAGES**

* Higher speed, lower latency

Because of the connection of opposite edges, data have more options to travel from one node to another which greatly increased speed.

* Better fairness

In a 4×4 mesh interconnect, the longest distance between nodes is from upper left corner to lower right corner. Each datum takes 6 hops to travel the longest path. But in a 4×4 Torus interconnect, upper left corner can travel to lower right corner with only 2 hops

* Lower energy consumption

Since data tend to travel fewer hops, the energy consumption tends to be lower.

**DISADVANTAGES**

* Complexity of wiring

Extra wires can make the routing process in the physical design phase more difficult. If we want to lay out more wires on chip, it is likely that we need to increase the number of metal layers or decrease density on chip, which is more expensive. Otherwise, the wires that connect opposite edges can be much longer than other wires. This inequality of link lengths can cause problems because of [RC delay](https://en.wikipedia.org/wiki/RC_time_constant).

* Cost

While long wrap-around links may be the easiest way to visualize the connection topology, in practice, restrictions on cable lengths often make long wrap-around links impractical. Instead, directly connected nodes—including nodes that the above visualization places on opposite edges of a grid, connected by a long wrap-around link—are physically placed nearly adjacent to each other in a folded torus network. Every link in the folded torus network is very short—almost as short as the nearest-neighbor links in a simple grid interconnect—and therefore low-latency.

1. **HYPERCUBE INTERCONNECTION NETWORK**

[Hypercube](https://en.wikipedia.org/wiki/Hypercube) networks are a type of [network topology](https://en.wikipedia.org/wiki/Network_topology) used to connect multiple [processors](https://en.wikipedia.org/wiki/Processors) with memory modules and accurately route data. Hypercube networks consist of 2m nodes. These nodes form the vertices of squares to create an internetwork connection. A hypercube is basically a multidimensional [mesh network](https://en.wikipedia.org/wiki/Mesh_networking) with two nodes in each dimension. Due to similarity, such topologies are usually grouped into a k-ary d-dimensional mesh topology family where d represents the number of dimensions and k represents the number of nodes in each dimension.



Fig4: Hypercube Interconnection network

Hypercube interconnection network is formed by connecting N nodes that can be expressed as a power of 2. This means if the network has n nodes it can be expressed as :

**N = 2m**

N = 2 m {\displaystyle N=2^{m}} where m is the number of bits that are required to label the [nodes](https://en.wikipedia.org/wiki/Mesh_node) in the network. So, if there are 4 nodes in the network, 2 bits are needed to represent all the nodes in the [network](https://en.wikipedia.org/wiki/Computer_network). The network is constructed by connecting the nodes that just differ by one bit in their [binary](https://en.wikipedia.org/wiki/Binary_code) representation. This is commonly referred to as Binary labelling. A 3D hypercube internetwork would be a cube with 8 nodes and 12 [edges](https://en.wikipedia.org/wiki/Edge_%28geometry%29). A 4D hypercube network can be created by duplicating two [3D](https://en.wikipedia.org/wiki/Three-dimensional_space) networks, and adding a most significant bit. The new added bit should be ‘0’ for one 3D hypercube and ‘1’ for the other 3D hypercube. The corners of the respective one-bit changed [MSBs](https://en.wikipedia.org/wiki/Most_significant_bit) are connected to create the higher hypercube network. This method can be used to construct any m-bit represented hypercube with (m-1)-bit represented hypercube.