JAIST Repository
https://dspace.jaist.ac.jp/

Title	階層型トーラスネットワーク(HTN)の性能
Author(s)	M.M., Hafizur Rahman
Citation	
Issue Date	2003-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/1688
Rights	
Description	Supervisor:Prof.Susumu Horiguchi, 情報科学研究科, 修士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

Performance of Hierarchical Torus Network: HTN

By M.M. Hafizur Rahman

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Prof. Susumu Horiguchi

March, 2003

Performance of Hierarchical Torus Network: HTN

By M.M. Hafizur Rahman (110129)

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Prof. Susumu Horiguchi

and approved by
Prof. Susumu Horiguchi
Prof. Hong Shen
Prof. Yasushi Hibino

February, 2003 (Submitted)

Copyright © 2003 by M.M. Hafizur Rahman

To Ruba
never ending source
of joy
and happiness.

Abstract:

Hierarchical interconnection networks have raised a great interest in the research community in the last few years and are an emerging standard in the design of interconnection networks for massively parallel computers. In particular, it is suitable for the realization of 3D-wafer stacked implementations.

In this thesis, we propose a new hierarchical interconnection network, called the Hierarchical Torus Network (HTN), as an interconnection network for large scale 3D multicomputers. It consists of Basic Modules (BMs) which are 3D-tori ($m \times m \times m$) and are hierarchically interconnected by 2D-tori ($n \times n$). Both the BMs and the interconnection at higher levels are toroidally connected, hence the name *Hierarchical Torus Network (HTN)*. Architectural details of the HTN and its addressing and routing of messages have also been discussed. We have explored various aspects such as network diameter, cost, average distance, bisection width, peak number of vertical links, and VLSI layout area of the HTN and compared them with those for several commonly used networks for parallel computers. It is shown that the HTN possesses several attractive features including small diameter, small average distance, small number of wires, a particularly small number of vertical links, and an economic layout area.

Wormhole routing is an emerging switching technique for the current generation of multicomputers. We have also used wormhole routing for switching because it requires a small number of buffers and can control data flow in a pipelined fashion, reducing communication overhead. Deadlock-free routing is the most critical issue in wormhole-routed networks and is achieved by using virtual channels. However, the hardware cost increases as the number of virtual channels increases. Deadlock-free routing with a minimal number of virtual channels is needed. In this thesis, we present a deadlock-free routing algorithm for the Hierarchical Torus Network with a minimum number of virtual channels. Dynamic communication performance is simulated for this network. We compare the communication performance of a wormhole-routed HTN with several other networks and show the superiority of the HTN over those networks for parallel computers. Redundancy and yield of HTN are also indicated. Finally, we discuss mapping of advanced application, namely bitonic merge, Fast Fourier Transform (FFT), and finding the maximum value on HTN.

Contents

List of Figures

List of Tables

Chapter 1

"A journey of a thousands miles must begin with a single step."

– Lao-tzu

Introduction

1.1 Introduction

Ever since conventional serial computers were invented, their speed has steadily increased to match the needs of emerging applications. However, the fundamental physical limitation imposed by the speed of light makes it impossible to achieve indefinitely further improvements in the speed of such computers. A natural way to circumvent this situation is to use an ensemble of processors.

Areas requiring great computational speed include numerical modeling and simulation of scientific and engineering problems. These include modeling large DNA structures, global weather forecasting, modeling motion of astronomical bodies, pollution monitoring, fusion energy research, artificial intelligence, and computer vision. In order to solve these grand challenge problems, the goal has been to obtain computer systems capable of computing at the teraflops (10^{12} floating-point operations per second) level. Even the smallest of these problems requires gigaflops (10^9 floating-point operations per second) of performance for hours at a time. The largest problems requires teraflops performance for more than a thousand hours at a time. Computations must be completed within a 'reasonable' time period. Obviously, an execution time of 1 year is always unreasonable. Thus, we need not only teraflops performance but also we need petafloaps (10^{15} floating-point operations per second) or exafloaps (10^{18} floating-point operations per second) of performance. This requires current supercomputer technologies to be upgraded into massively parallel and distributed systems, which satisfy the continuously increasing demand for computing power. The interconnection network is the key element for building massively parallel computers consisting of thousands or millions of processors. A major issue in designing such large-scale multiprocessor systems is the construction of a flexible interconnection network to provide efficient inter-processor communication.

Mature 3-Dimensional (3D) Integrated Circuit (IC) technology has been employed in the development of commercial 3D memory systems. A current challenge is to produce a 3D computer and 3D stacked implementations have been proposed as a new technology for massively parallel computers. Little *et al.* [1] developed a 3D computer consisting of a 32×32 cellular array and organized as a 5-wafer stack. The stack comprised two types

of wafers called accumulator and shifter. The die size of the array was about 1 square inch, and the throughput at 10 MHz, was 600 MOPs (Million Operations per Second). This prototype of a wafer stacked implementation showed that the stacked silicon plane organization provided extremely short paths through the logic sets on various planes of the stack. Furthermore this prototype demonstrated that the technological problems of vertical interconnects could be surmounted. Further development on vertical interconnects was reported by Campbell *et al.* [2] and Carson [3]. Recently, Kurino *et al.* [4] have suggested a highly practical 3D construction technology.

A major obstacle in the design of future 3D computers is the cost in terms of area for vertical interconnects. Each vertical interconnect has an area of $300\mu m \times 300\mu m$. Thus, unconstrained use of interconnects is prohibited. Clearly, an interconnection philosophy which minimizes these vertical links can contribute to the success of a 3D implementation. Jain *et al.* presented a hierarchical interconnection network called TESH (Tori connected mESHes) [5–8], and they concluded that a hierarchical interconnection network minimizes the vertical links for efficient 3D stacked implementations.

An interconnection network [9, 10] is described by its topology, routing algorithm, and flow control mechanism. The topology of the network is the arrangement of its nodes and channels in a graph. Network topology refers to the static arrangement of channels and nodes in an interconnection network, the road-map over which packets travel. Selecting the network topology is the first step in designing a network because the routing strategy and flow control method depend heavily on the topology.

The processing nodes of a massively parallel computer exchange data and synchronize with one another by passing messages over an interconnection network. The interconnection network is often the critical component of a large parallel computer because performance is very sensitive to network latency and throughput. The performance of message passing in multicomputers [15] depends on the routing and switching technique employed in their interconnection network. An efficient and fast switching technique is a basic requirement for getting good performance from an interconnection network. Switching is the mechanism that removes data from an input channel and places data in an output channel. Network latencies are highly dependent on switching technique. Wormhole (WH) [31][32][37] routing has been a popular switching technique in the new generation multicomputers because of its low network latency and lower hardware requirement. In wormhole switching, a message is split up into several flow control digits called flits, which are buffered at each node. When the header flit can not be routed due to the contention in the network, the flow control within the network blocks the message, effectively dilating its length until a link become free. In wormhole switching it is impossible for flits of other packets to cross the path of the current packet.

Routing [9, 10, 34] is the process of determining and prescribing the path or method to use for forwarding message i.e., deciding which output channel an incoming packet is transmitted on. It is the action of moving information across the interconnection network from a source to destination. Flow control [9] deals with the allocation of channels and buffers to a packet as it travels along a path through the interconnection network. A virtual channel [9, 38] is a logical channel with its own flit buffer, control and data

path. Virtual channels can be used to implement a deadlock-free routing algorithm to increase network throughput. Additional virtual channels can be used to avoid congestion in the network, but require physical channel bandwidth. The trade-off between increased network throughput and longer communication latency should be considered when deciding whether to use virtual channels. Additional virtual channels increase the switching complexity, which in turn slightly increases network latency.

Massively parallel computer systems usually have stringent requirements for reliability because of the large investments in such systems as well as the nature of the applications for which they are likely to be used. Fault tolerant [10] networks are essential to the reliability of massively parallel computer systems. A fault-tolerant network has the ability to route information even if certain network components fail. The techniques often used for network fault tolerance are either software-based or hardware-based. In the software-based technique, an adaptive routing algorithm is used, which makes use of multiple paths for a given pair of source and destination to avoid faulty components. In the hardware-based technique, the network is enhanced by additional hardware and provides enough redundancy [42] in the original network design to tolerate a certain number of faults.

Performing a computational problem efficiently on a multicomputer network is a complex task, even when the parallelism in the problem has already been identified. One of the problems that arises in this context is designing parallel algorithms so that their communication requirements can be efficiently supported by the underlying interconnection network. It should be noted that this problem arises because of the limitations of the interconnection network itself, and therefore have a direct influence on the choice of an interconnection network for a given application. Thus, the suitability of a given interconnection network for certain application can often be estimated by studying how efficiently common operations such as sorting, computing the maximum value, bitonic merge, divide-and-conquer, FFT, and broadcasts can be performed on the given network.

1.2 Motivations and Goal

As sequential computers are reaching their limits, a common approach to enhancing their performance is to design parallel computers with off-the-shelf components to exploit the parallelism in problems. Parallel processing with hundreds or thousands of microprocessors has become a viable alternative to conventional supercomputers and mainframes employing a handful of expensive processors. Several commercial machines with hundreds or thousands of processors have reached the market place in the past decade or two. The complexity of an interconnection network often determines the size of a parallel computer that can be constructed. Likewise, the attainable performance of a parallel computer is ultimately limited by the characteristics of the interconnection network. Apparently, one of the critical design issues of parallel computers is the interconnection network which is the backbone for these parallel computers.

Hundreds of different types of networks have been proposed in the past decades. No single network is optimal in all aspects of network performance. Designing new networks remains a topic for intensive investigation, given that there is no clear winner among

existing designs. Careful designers would try to achieve the best out of a good trade-off. However, even such a trade-off can lead to different results in different situations due to emphasis on different parameters. For example, in a non-VLSI environment, the overall performance of the mesh, the tree and the Cube Connected Cycles (CCC) is in ascending order while in VLSI implementation where layout area and wire length are two important parameters, the overall comparison of the above three networks shows that the reverse is true. Thus, the design of interconnection networks is still a very active research area. We believe that this research will continue for decades since parallel and distributed computers are the only solutions for the computational problems challenging human beings in the twenty-first century.

Hierarchical interconnection networks [16] have attracted considerable attention in the research community during the past few years as a means of communication for multiprocessor systems. A hierarchical design approach allows the network to be constructed incrementally starting from one or more basic modules. Hierarchical interconnection networks are intuitively appealing when a large number of nodes are to be connected. For very large scale system, the number of links needed with a conventional network structure such as mesh, torus, and hypercube may become prohibitively large. Hierarchical interconnection networks exploit the locality that exists in communication patterns to allow reduction in the required number of links. Various hierarchical network have been proposed such as TESH [5–8], H3D-Torus [17], H3D-Mesh [18] and so on.

The critical issue in designing interconnection network is to provide efficient interprocessor communication. An interconnection network should transfer a maximum number of messages in the shortest time with minimum cost and maximal reliability. Therefore, the main task of an interconnection network is to transfer information from the source node to the destination node while considering the following points:

- latency as small as possible.
- as many concurrent transfer as possible.
- cost as low as possible.

It has already been shown that a toroidal network has better dynamic communication performance than a mesh network. This is the key motivation for us to consider a hierarchical interconnection network where each level of the network has a toroidal interconnection. Therefore, the *first goal* of this thesis is to propose a new hierarchical interconnection consisting of a toroidal network.

Although the theoretical foundations for constructing large interconnection networks for massively parallel computer system have existed for a long time, the state of the hardware technology has not allowed cost-effective realization until the last decade. However, recent advances in VLSI technology overcome this drawback. Recent progress in VLSI technology achieves VLSI systems on stacked silicon planes. On a stacked silicon plane, part of a massively parallel computer is implemented and these silicon planes are then

interconnected. Jain *et al.* [5–8] have pointed out that hierarchical interconnection networks are suitable for 3D stacked implementations. Hence, the *second goal* of this thesis is to analyze the network performance of 3D-wafer stacked implementations.

Interconnection networks are used by the nodes of parallel computer systems to exchange data and synchronize with each other. Network performance is often critical, as the performance of a massively parallel computer system is sensitive to network latency and throughput. For a network to have good performance, low latency and high throughput must be achieved. Reducing network latency and increasing network throughput is crucial for improving the performance of an interconnection network. Thus, the *third goal* of this thesis is to evaluate the dynamic communication performance of the proposed network along with several other commonly used networks.

As the interconnection network is a critical component of a multicomputer system, methods of achieving fault tolerance in the network has special significance. The failure of a network component can often bring down the entire system, unless adequate measures are provided to tolerate such faults. So, the *fourth goal* of this thesis is to analyze the fault tolerance performance of the proposed network.

In conclusion, the main goal of this thesis is to develop and analyze an efficient hierarchical interconnection network for massively parallel computer systems.

1.3 Contribution of the Thesis

In this thesis, we have proposed a new hierarchical interconnection network called Hierarchical Torus Network (HTN). The basic idea behind this network is that it is a hierarchical interconnection network, where each level is connected as toroidal interconnections.

The contribution of this thesis can be summarized as follows:

- Introduce a new hierarchical interconnection network called HTN for massively parallel computers. Topological properties and the architectural structure of a HTN are also presented.

- Describe various aspects of network features for 3D wafer stacked implementation. Wafer stacked implementation issues for HTN is also described.

- Provide a deadlock-free routing algorithm for HTN. Virtual channels are used to achieve a deadlock-free routing algorithm. Hardware cost increases along with an increase in the number of virtual channels. An investigation into the minimum number of virtual channels for deadlock-free routing algorithm for HTN is also carried out.

- Simulate the dynamic communication performance of the HTN as well as for several commonly used networks for parallel computers. We have conducted several experiments and compared the dynamic communication performance to show the superiority of the HTN over other networks.

- Present the fault tolerance aspects of HTN. Tolerating faults is the key to system survival. Hardware-based fault tolerance techniques provide enough redundancy in the original network design to tolerate a certain number of faulty nodes. Redundancy and yield of HTN are also pointed out.

- Discuss mapping of advanced applications namely bitonic merge algorithm, FFT, and finding the maximum value on HTN.

1.4 Synopsis of the Thesis

After this introductory chapter, the remaining chapters of this thesis are organized as follows:

- Before proposing a new hierarchical interconnection network, it is very important to study existing network topologies, as well as studying how they can be changed into hierarchical interconnection networks. In **chapter 2**, we deal with different network topologies and provides an overview of the properties of many widely-used interconnection network topologies. Different hierarchical interconnection networks are also addressed in **chapter 2**. This chapter is the literature survey.

- In **chapter 3**, we present a new hierarchical interconnection network called Hierarchical Torus Network (HTN). This chapter covers the basic issues in this thesis. It also deals with the topological properties of interconnection networks. This chapter discuses many properties and technology-independent measures commonly used to evaluate and compare different network topologies, and provides a detailed analysis of these properties for HTN. We then discuss various aspects of network features for 3D wafer stacked implementation.

- Because interconnection networks are used for communication between nodes, routing is probably the most important problem in analyzing interconnection networks. In **chapter 4**, we describe the design and implementation issues of deadlock-free routing algorithm in interconnection networks. Some of the basic issues addressed in the chapter include: wormhole routing, virtual channels and channel dependency graphs. Descriptions are mainly provided for developing deadlock-free routing algorithms for HTN. Investigation of the minimum number of virtual channels for deadlock-free routing of HTN is also pointed out.

- **Chapter 5** is dedicated to performance evaluation. Performance evaluation techniques are needed to estimate the performance of a network before it is constructed, compare two or more networks, evaluate design trade-offs, and determine the optimal value of the design parameters. Emphasis is placed in dynamic communication performance and interpretation of simulation results. The interconnection network is a critical component of a multicomputer system and must therefore be designed with some degree of fault tolerance. Fault tolerance in the interconnection network

enables the system to survive the loss of one or more components without a disruption in its operation. Fault tolerance aspects of HTN are also the subject of **chapter 5**. For a network to be useful, it must accommodate a large class of applications. Finally, we discuss some advanced applications such as bitonic merge, FFT, and finding the maximum value on HTN at the end of this chapter.

- In **chapter 6**, we finally conclude this thesis paper with some perspectives. Continuing further research on those directions would be worthwhile.

Chapter 2

"It would appear that we have reached the limits of what it is possible to achieve with such statements, as they tend to sound pretty silly in 5 years."

– John Von Neumann

Interconnection Network for Massively Parallel Computers

2.1 Introduction

In a message-passing multicomputer [15], multiple computers or nodes are connected by an interconnection network and operate under an integrated operating system. Each node is directly connected to a subset of other nodes in the network. Each node is a programmable computer with its own processor, local memory, and other supporting devices. A common component of these nodes is a router, which handles message passing among nodes. Each router has direct links to the router of its neighbors. Usually, two neighboring nodes are connected by a pair of unidirectional channels in opposite directions. A bidirectional channel may also be used to connect two neighboring nodes. In this thesis, henceforth, unless specified, the term computer or processor refers to a node.

The research literature on interconnection networks includes a large number of examples, ranging from a simple bus to hierarchical interconnection networks. Interconnection networks can be broadly divided into two classes: static and dynamic. In the static (or direct) networks, point to point links interconnect the network nodes in some fixed topology. In the dynamic (or indirect) network processing nodes are not directly connected. The communication between any two nodes has to be carried through some switches. In this thesis paper, we have concentrated only on static interconnections. Static networks are also known as direct networks. In this paper, unless and otherwise specified, the interconnection network means direct network. Direct networks have been a popular interconnection architecture for constructing massively parallel computers.

An interconnection network is described by its topology, routing algorithm, and flow control mechanism. The topology of the network is the arrangement of its nodes and channels in a graph, in which vertices represent nodes and edges represent communication links. Network topology refers to the static arrangement of channels and nodes in an interconnection network, the roadmap over which packets travel. Selecting the network

topology is the first step in designing a network because the routing strategy and flow control method depend heavily on the topology.

As the focus of this thesis is on interconnection network, this chapter gives a survey on major interconnection networks proposed through the years and pointing out their potential shortcomings. The rest of this chapter is organized as follows: Section 2.2 outlines the architecture and properties of the most common interconnection network topologies. Hierarchical interconnection networks are the subject of Section 2.3. Finally, conclusions are pointed out in Section 2.4.

2.2 Network Topologies

Several interconnection network topologies have been proposed and studied by researchers as possible candidates for multiprocessor and multicomputer interconnection. These include ring, tree, mesh, torus, hypercube, star graph and many variants of these networks.

With different topologies, researchers are trying to balance performance in different issues and cost parameters. In these topologies, messages may have to traverse some intermediate nodes before reaching the destination node. The routing algorithm determines the paths for message transmissions. Some important interconnection network topologies and their properties have been discussed in this section.

2.2.1 Completely-Connected Network

In a *Completely-connected network* [12], each node has a direct communication link to every other node in the network. Figure 2.1 illustrates a completely-connected network of eight nodes. This network is ideal in the sense that a node can send a message to another node in a single step, since a communication link exists between them. The main disadvantages of a completely-connected network is that the basic cost is high, since there is a large number of communication links.

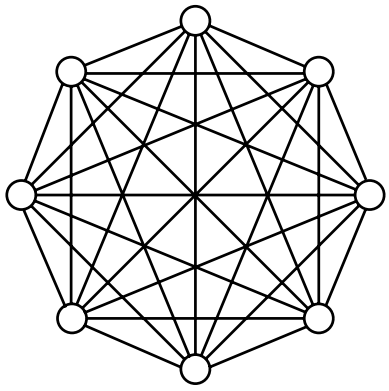


Figure 2.1: A completely-connected network of eight nodes

2.2.2 Star-Connected Network

In a *Star-connected network* [12], one node acts as the central node. Every other node has a communication link connecting it to this node. Figure 2.2 shows a star-connected network of nine nodes. Communication between any pair of nodes is routed through the central node. Hence, the central node is a bottleneck in the star topology.

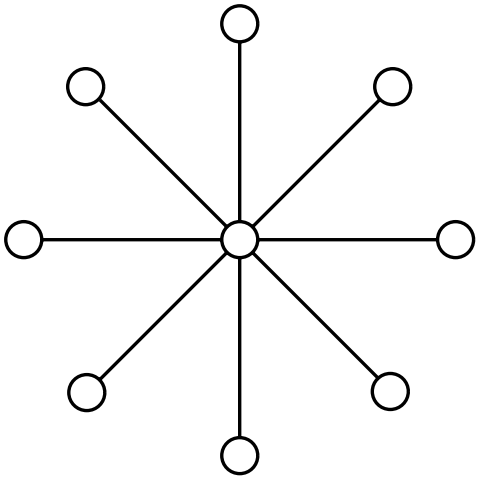


Figure 2.2: A star-connected network of nine nodes

2.2.3 Linear Array and Ring

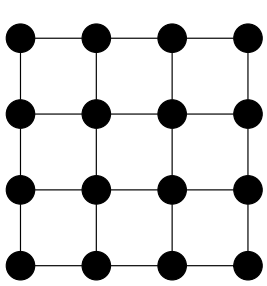
In a *linear array* [12] interconnection network, each node (except the nodes at the ends) has a direct communication link to two other neighboring nodes. It is a simple way to connect nodes in an interconnection network. Figure 2.3.(a) illustrates a four node linear array. A wraparound connection is often provided between the nodes at the ends. A linear array with a wraparound connection is referred to as a *ring* [12] network. Figure 2.3.(b) shows a ring of four nodes. One way of communicating a message between nodes is by repeatedly passing it to the node immediately to the right or left, depending on which direction yields a shorter path, until it reaches its destination. These networks are inherently unreliable as a failure of a single node or link disconnects the network. The usefulness of these topologies is limited to small networks because both the average distance and message densities increase linearly with the number of nodes.

(a) Linear Array **(b) Ring**

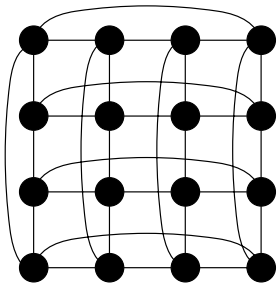
Figure 2.3: A four-node linear array and ring network

2.2.4 Mesh and Torus Network

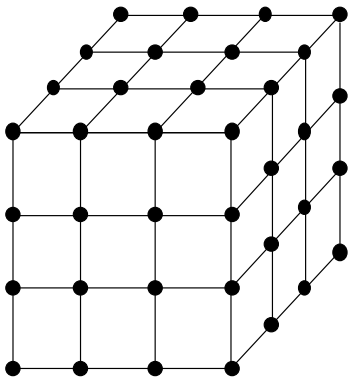
The *2D-mesh* [12] is an extension of the linear array to two dimension. In a 2D-mesh, each node has a direct communication link connecting it to four neighboring nodes except for the nodes on the edges. They only have three neighbors or just two neighbors if they are on the edge or on a corner, respectively. Figure 2.4.(a) illustrates a 2D-mesh of sixteen nodes. If both dimensions of the mesh contain an equal number of nodes, then it is called a square mesh; otherwise it is called a rectangular mesh. Often, the nodes at the periphery are connected by wraparound connections. Such a mesh is called a *torus* [9, 12], as shown in Fig. 2.4.(b). Unlike the mesh, every node in the torus has exactly four neighbors. Common extensions of the 2D-mesh and 2D-torus include 3D-mesh and 3D-torus networks. Figure 2.4.(c) and 2.4.(d) illustrate a 3D-mesh and a 3D-torus network, respectively. A message from one node to another can be routed either in the mesh or in the torus by first sending it along one dimension and then along the other dimension until it reaches its destination.



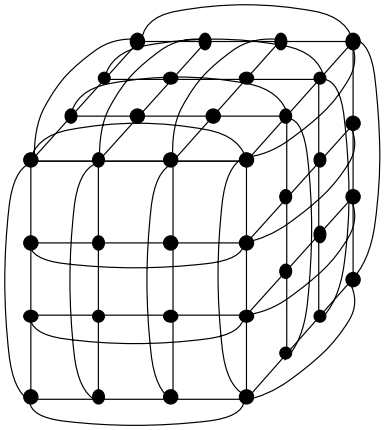
(a) 2D-mesh



(b) 2D-torus



(c) 3D-mesh



(d) 3D-torus

Figure 2.4: Mesh and torus networks.

2.2.5 Hypercube

A *hypercube* [12, 20] is a multidimensional mesh of nodes with exactly two nodes in each dimension. In a hypercube interconnection network, two nodes are connected by a direct link if and only if the binary representation of their labels differ at exactly one bit position. The distance between two nodes along a shortest path is the Hamming distance between their binary position. A message can be routed from a source node to a destination node by passing the message along dimensions if the Hamming distance is one. Hypercube interconnection allows efficient implementation of a large number of parallel algorithms. The main problem of this network is that the number of connections per node increases with the network size.

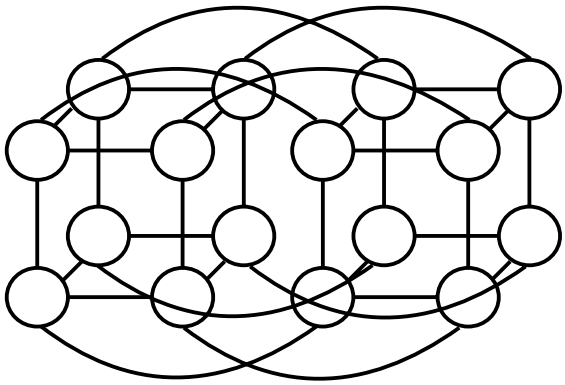


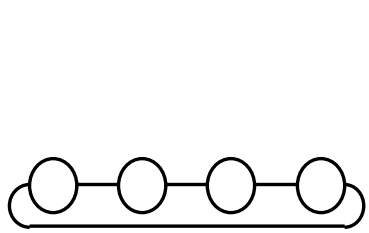
Figure 2.5: A hypercube network of sixteen nodes

2.2.6 k-ary n-cube

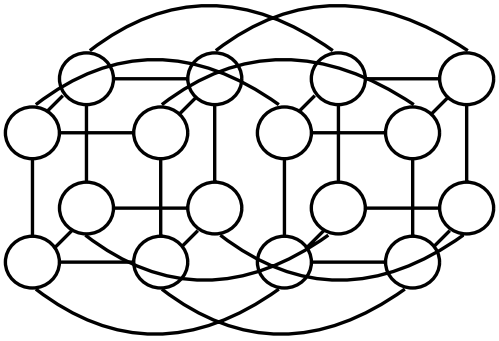
A k-ary n-cubes [26] is defined as a cube with n dimensions and k nodes in each dimension. Here, let n be the dimension of the cubes, k be the radix, and N be the total number of nodes. Dimension, radix, and number of nodes are related by the equation $N = k^n$. A binary n-cube is an example of a k-ary n-cube. Tori are the isomorphic with k-ary n-cubes. 2D-torus and 3D-torus are k-ary 2-cubes and k-ary 3-cubes, respectively. k-ary n-cube network is also known as n-dimensional k-torus [27]. In recent literature, the k-ary n-cube refers to the n-dimensional torus with k nodes in every dimension. Figure 2.6 shows different k-ary n-cube networks. Figure 2.6.(b) is a binary 4-cube networks and 2.6(a), 2.6(c), and 2.6(d) are the isomorphic of k-ary n-cubes.

2.2.7 Tree network

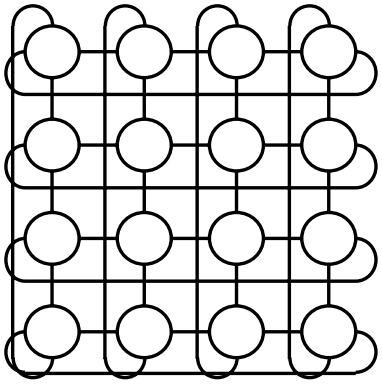
A *tree network* [9, 12] is one in which there is only one path between any pair of nodes. This topology has a root node connected to a certain number of descendant nodes. Each



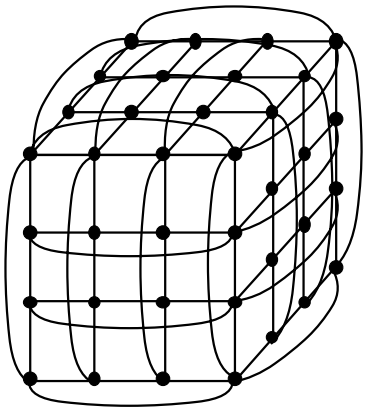
(a) 4-ary 1-cube (Ring)



(b) 2-ary 4-cube (Hypercube)



(c) 4-ary 2-cube (2D-Torus)



(d) 4-ary 3-cube (3D-Torus)

Figure 2.6: Different k-ary n-cube networks

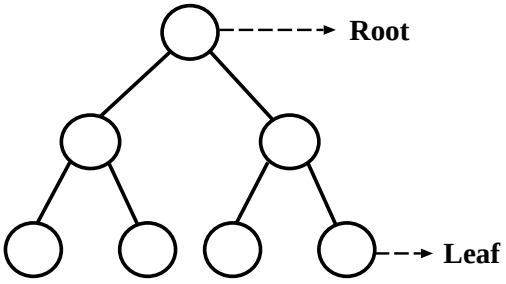


Figure 2.7: Tree topology

of these nodes is in turn connected to a disjoint set of descendants. A node with no descendants is a leaf node. A characteristic property of trees is that every node except the root node has a single parent node. To route a message in a tree, the source node sends the message up the tree until it reaches the node at the root of the smallest subtree containing both the source and destination nodes. Then the message is sent down the tree toward the destination node. In a tree topology, both the diameter and the average distance grows logarithmically with respect to the number of nodes. However, the message density increases more rapidly with uniform traffic. Tree networks suffer from a communication bottleneck at higher levels of the tree. Additionally, there are no alternative paths between any pair of nodes. The most important benefit of tree network is that the cost is $O(N)$. Figure 2.7 illustrates a binary tree network of 7 nodes.

2.3 Hierarchical Interconnection Network

Interconnection networks usually suffer from Little's Law: low cost implies low performance and high performance is obtained at high cost [16]. However, hierarchical interconnection networks [16] provide high performance at low cost by exploring the locality that exists in communication patterns of massively parallel computers. A hierarchical interconnection network provides a plausible alternative way in which several topologies can be integrated together. Hierarchical interconnection networks have attracted considerable attention in the research community during the past few years as a means of communication for multiprocessor systems. They take advantage of the locality of reference in the communication pattern.

For massively parallel computers with millions of nodes, the large diameter of conventional topologies is intolerable. Hierarchical interconnection networks are a cost-effective way to interconnect a large number of nodes. They are also suitable for 3D stacked implementations. A lot of hierarchical interconnection networks have been proposed for this type of massively parallel computer. Most of the networks are viewed as a hierarchy of conventional topologies. A hierarchical interconnection network allows the network to be constructed incrementally starting from one or more basic modules. Some important hierarchical interconnection network and their properties are discussed in this section.

2.3.1 Cube-Connected Cycle

A *cube-connected cycles* [22] can be considered as an n-dimensional hypercube of virtual nodes, where each virtual node is a ring with n nodes, for a total of $n2^n$ nodes. Simply, a node of the hypercube is replaced with an n-node ring such that the n links incident to the original node are distributed among the nodes in the ring. Each node in the ring is connected to a single dimension of the hypercube. Figure 2.8 illustrates a 24-node cube-connected cycles network. In this figure, two links are connected to neighbors in the ring, and one link is connected to a node in another ring through one of the dimensions of the hypercube.

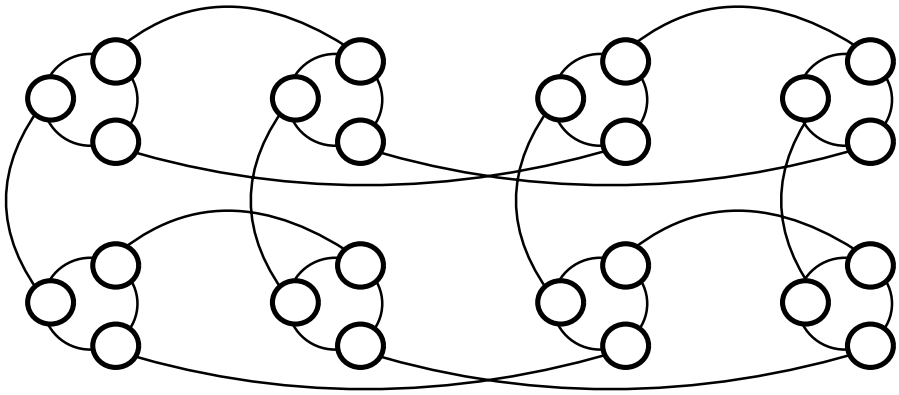


Figure 2.8: Cube-connected cycles of 24 nodes

2.3.2 Star Graph Network

A *star graph* [21] can be informally described as follows. The vertices of the graph are labeled by permutations of n different symbols, usually denoted as 1 to n. A permutation is connected to every other permutation that can be obtained from it by interchanging the first symbol with any of the other symbols. A star graph has $n!$ nodes and node degree is equal to $(n-1)$. Figure 2.9 shows a star graph obtained by permutations of four symbols. Although a star graph has a lower diameter than a hypercube of similar size, routing is more complex.

2.3.3 Recursive Diagonal Torus

The name *Recursive Diagonal Torus (RDT)* [23] itself expresses its characteristics clearly. This novel class of network is composed of a series of recursively structured meshes (tori) with increasing size in the diagonal directions. At first, a 2-dimensional square mesh (torus) will serve as the basis of RDT. Figure 2.10 shows an RDT network structure. It is a class of network which makes use of the advantages of mesh structures and greatly improves the performance of meshes when the number of nodes reaches tens of thousands of nodes.

2.3.4 Shifted Recursive Torus

The *Shifted Recursive Torus (SRT)* [24] consisting of mesh-tori and bypasses the links to shift the tori recursively. The simplest torus interconnection is a ring network. To improve the performance of ring networks, bypass links are added to the ring network under the network constraint that every node has fixed number of links. Figure 2.11 shows a basic interconnection of a standard 1D-SRT consisting of 32 nodes. 1D-SRT can be extended

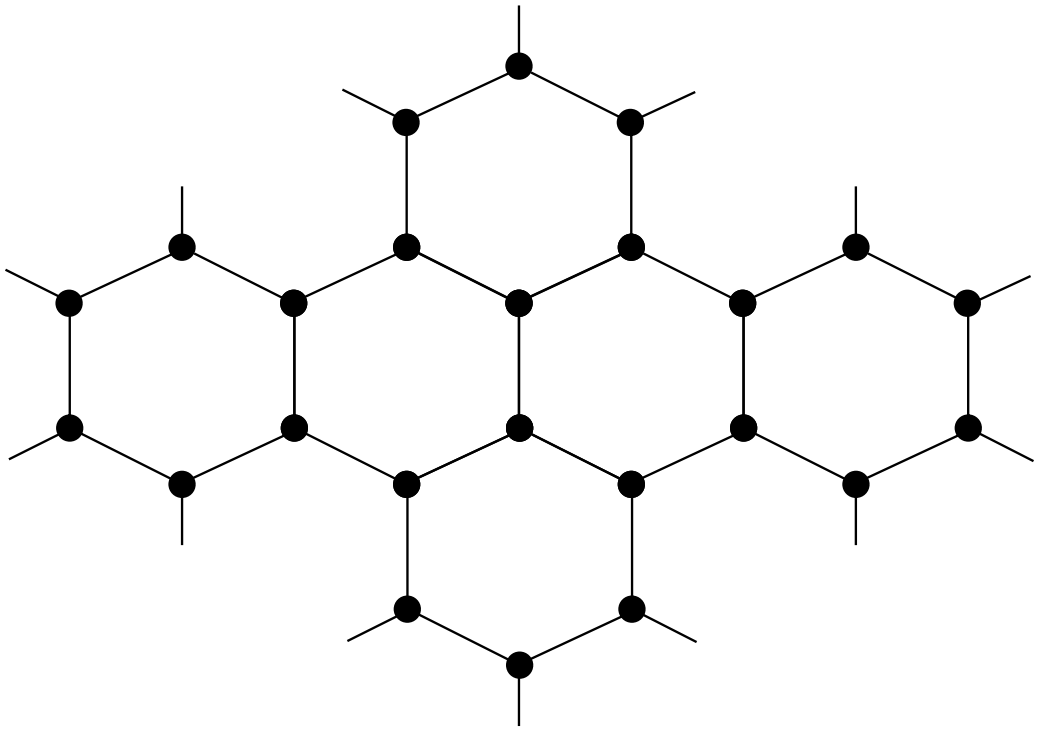


Figure 2.9: Star graph network

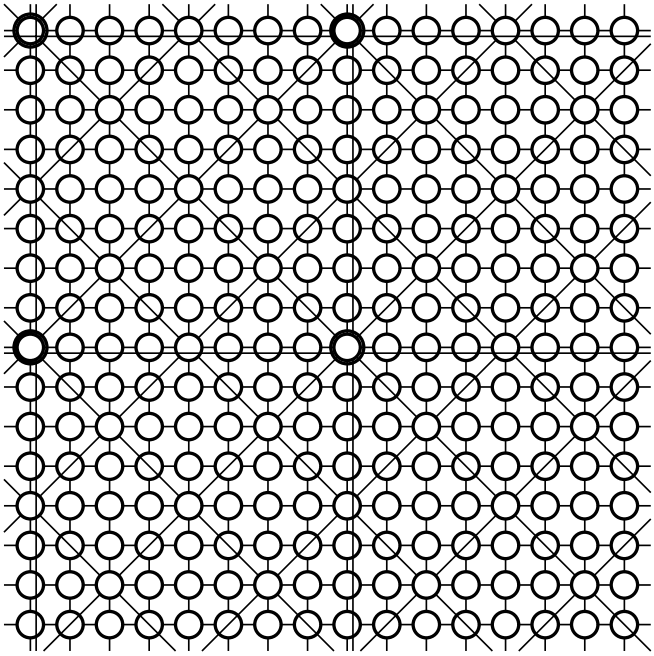


Figure 2.10: Recursive diagonal torus network

to 2D-SRT. The SRT can achieve a smaller diameter with limited number of links per node than the hypercube. It is also an wire-efficient network for VLSI implementation.

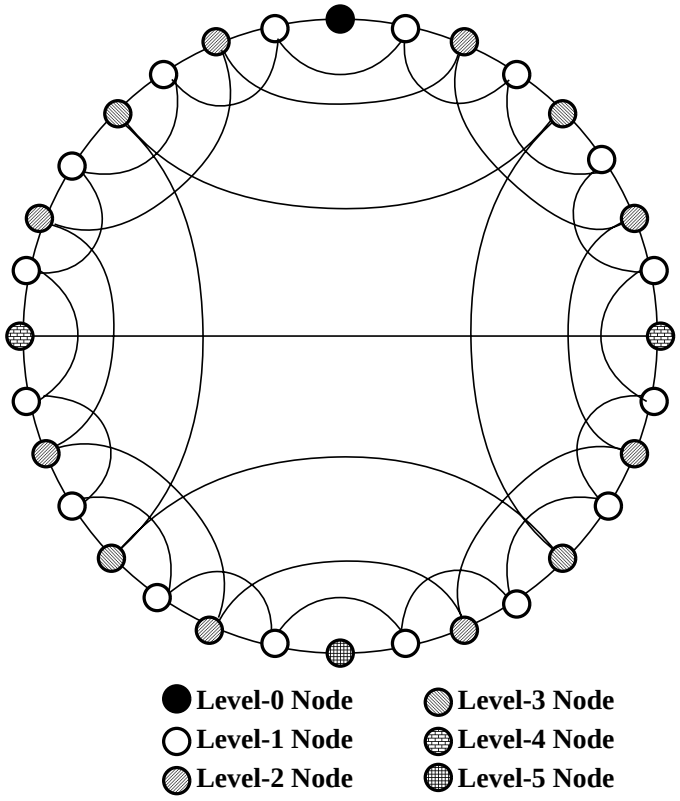


Figure 2.11: Standard 1D-SRT consisting of 32 nodes.

2.3.5 TESH Network

The *TESH (Tori connected mESHes)* [5–8] is a hierarchical interconnection network. It consists of a Basic Module (BM), which is a mesh connection of $(2^m \times 2^m)$ nodes. Successively higher level networks are built by recursively interconnecting lower-level subnetworks in the form of a 2D-torus. Figure 2.12 illustrates the Level-2 interconnection of a TESH consisting of a 2D-torus (4×4) with 16 BMs. It has been shown that TESH possesses several attractive features including small diameter and small numbers of wires, and a particularly small number of vertical links.

2.3.6 H3D-Torus Network

The *Hierarchical 3-Dimensional torus (H3D-torus)* [17] network is defined as a hierarchical interconnection network, which consists of Basic Modules (BM) and the BMs are hierarchically interconnected by a 3D-torus $(n \times n \times n)$ for higher-level interconnection.

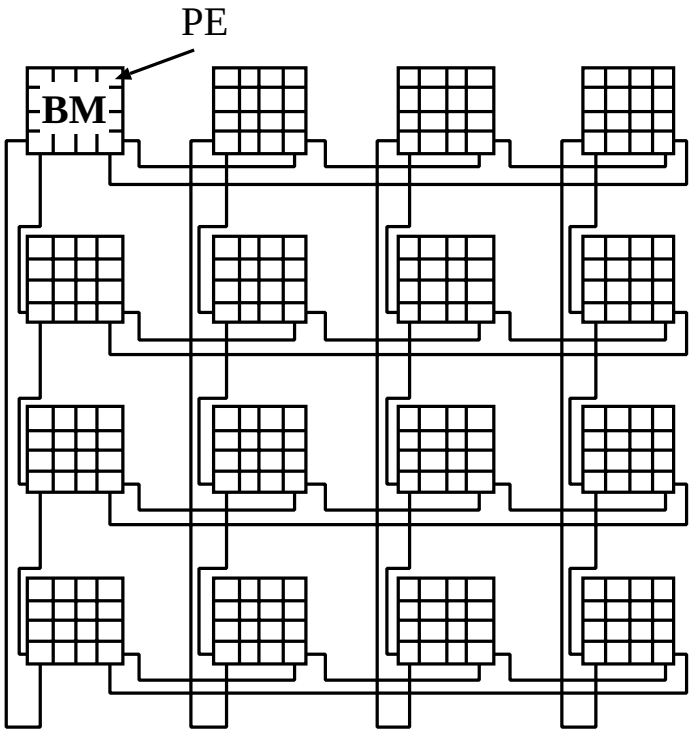


Figure 2.12: Level-2 interconnection of TESH network

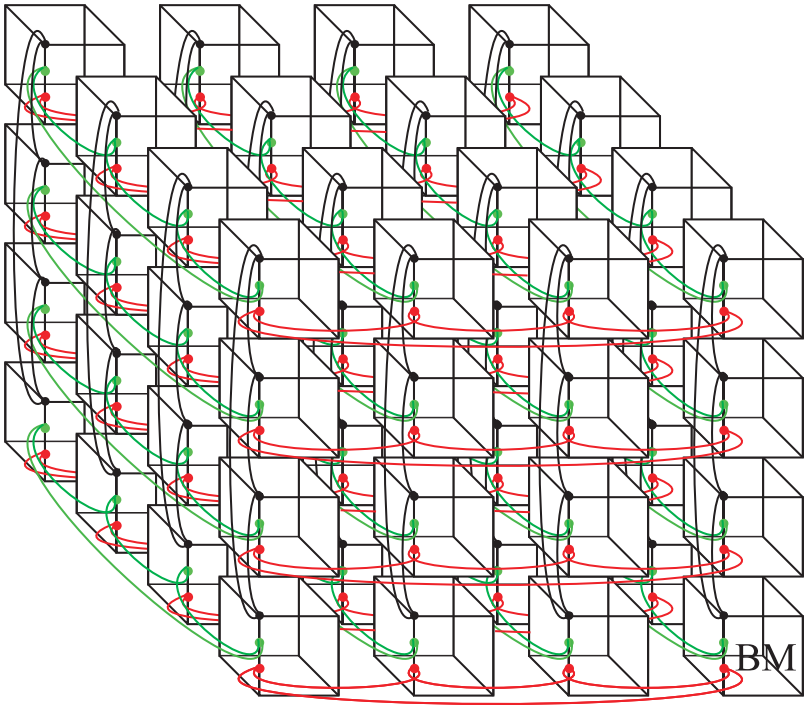


Figure 2.13: Interconnection of a Level-2 H3D-torus network

The BM of the H3D-torus is a 3D-mesh ($m \times m \times m$). Figure 2.13 illustrates the interconnection of a Level-2 H3D-torus network. As shown in Fig. 2.13, Level-2 interconnects the BMs as ($4 \times 4 \times 4$) 3D-torus connection. The Level-3 interconnection is also a 3D-torus connection of 64 Level-2 modules.

2.3.7 H3D-Mesh Network

The *Hierarchical 3-Dimensional mesh (H3D-mesh)* [18] network is defined as a hierarchical interconnection network, which consists of Basic Modules (BM) and the BMs are hierarchically interconnected by a 2D-mesh ($n \times n$) for higher-level interconnection. The BM of the H3D-mesh is a 3D-torus ($m \times m \times m$). Figure 2.14 illustrates the interconnection of a Level-2 H3D-mesh network. As shown in Fig. 2.14, the mesh at Level-2 interconnects the BMs in a (4×4) 2D-mesh. The Level-3 interconnection is also a 2D-mesh connection of 16 Level-2 modules.

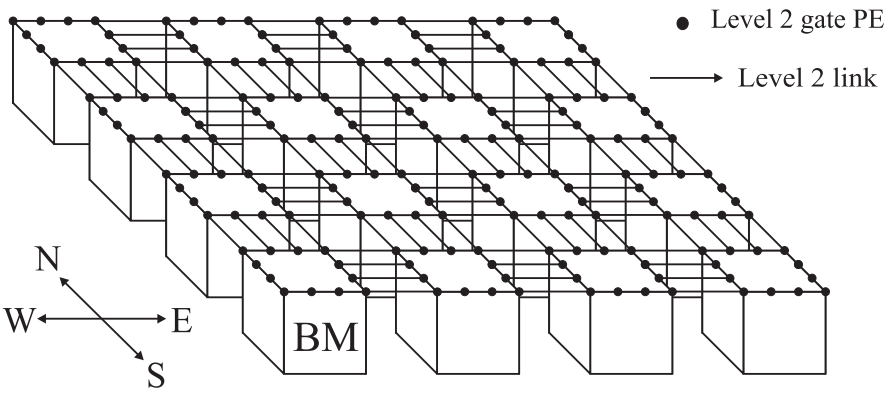


Figure 2.14: Interconnection of a Level-2 H3D-mesh network

Beside these, a lot of hierarchical interconnection networks have been proposed in the literature. For example, Complete Connection of torus hyperCube (CCTCube), Crossed Cube, dBCube, De-Brujin Network, Hyper deBrujin, de-Brujin Connected Torus (BCT), Enhanced Hypercube, Extended Hypercube, Hierarchical Hypercube, Twisted Hypercube, Fibonacci Cube, Polymorphic torus, Ring Tree on Mesh (RTM), and so on.

2.4 Conclusions

In this chapter, we have presented different network topologies including a hierarchical interconnection network. It can be seen that the topological properties of each interconnection network is different from the others, and they are trying to make a trade-off between performance and cost in different aspects. No single network is optimal in all aspects of network performance. Thus, the design of a new interconnection network is an important issue for massively parallel computer system and certainly worthy of research effort.

Chapter 3

"I do not know what I may appear to the world, but to myself I seem to have been only like a boy playing at the seashore, and diverting myself in now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me."

– Sir Isaac Newton

Hierarchical Torus Network

3.1 Introduction

Hundreds of interconnection networks have been proposed in the past decade. Some networks are better than other in some aspects, but worse in others. There is no one ultimate network which is better than all others in all aspects. Designing new networks still remains a topic for intensive investigation, given that there is no clear winner among existing ones. Careful designers would try to achieve the best out of a good trade-off. But even such a trade-off can lead to different results due to emphasis on different parameters in different situations. For example, in a non-VLSI environment, the overall performance of mesh, tree and Cube Connected Cycles (CCC) is in ascending order while in VLSI implementation where layout area and wire length are two important parameters, the overall comparison of the above three networks shows that the reverse is true [13].

Although the theoretical foundations for constructing large scale interconnection networks have existed for a long time, the state of hardware technology did not allow their cost-effective realization. Recent progress in VLSI technology can achieve a VLSI system on stacked silicon planes. A 3D stacked implementation has been proposed as a new technology for the realization of massively parallel computers. A part of a massively parallel computer is implemented on a silicon plane and some of these planes are interconnected by vertical links. Jain *et al.* [5–8] have pointed out that hierarchical interconnection networks are suitable for 3D stacked implementations.

In this chapter, we propose a new hierarchical interconnection network called Hierarchical Torus Network (HTN). Architectural details of the HTN and its addressing and routing are discussed. We also explore various aspects of showing the superiority of the HTN over several commonly used networks for parallel computers.

This chapter is organized as follows: Section 3.2 discusses the architecture of HTN, including addressing and routing. A brief discussion on static network performance of the network is given in Section 3.3. VLSI implementation issues including 3-D construction are addressed next in Section 3.4. Finally, some concluding remarks are given in Section 3.5.

3.2 Architecture of the HTN

Hierarchical Torus Network (HTN) is a hierarchical interconnection network, which consists of Basic Modules (BM) where the BMs are hierarchically interconnected for higher level networks. The BM consists of a 3D-torus network. In this thesis paper, henceforth, unless specified, BM refers to a Level-1 network. Successive higher level networks are built by recursively interconnecting lower level subnetworks in a 2D-torus. Both the BMs and the interconnection of higher levels have a toroidal interconnection. Hence the name Hierarchical Torus Network. Figure 3.1 shows the interconnection philosophy of HTN. This figure illustrates the interconnection of a Level-2 HTN by the basic modules, where the BM is a 3D-torus of $4 \times 4 \times 4$ and Level-2 network is a 2D-torus of 4×4.

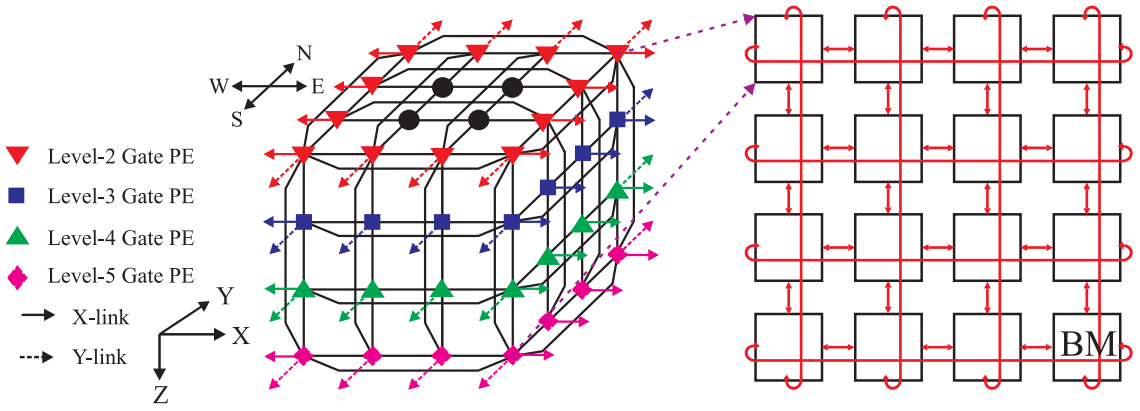


Figure 3.1: Interconnection of HTN

3.2.1 Basic Module

According to the definition, the basic module of the HTN is a 3D-torus network of size $(m \times m \times m)$, where m is a positive integer. m could be any value, however, the preferable one is $m = 2^p$, where p is a positive integer.

The BM has some free ports at the contours of the xy-plane. These free ports are used for higher level interconnection. As seen in Fig.3.2, all ports of the interior PEs are used for intra-BM connections. The exterior PEs, however, have either one or two free ports. These free ports and their associated links are used for inter-BM interconnections to form higher level networks. PEs at the contours of an xy-plane are assigned to higher levels as

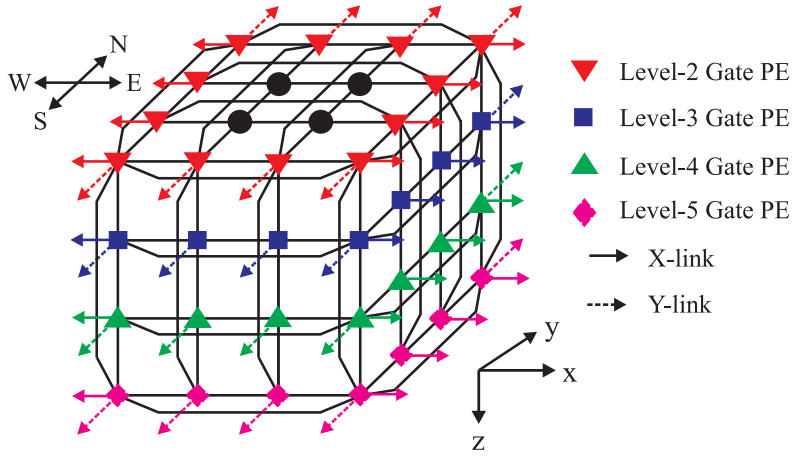


Figure 3.2: Basic Module

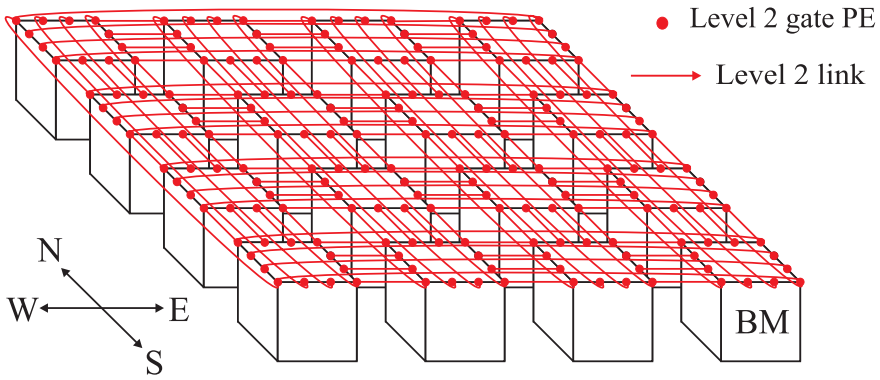


Figure 3.3: Interconnection of a Level-2 HTN

gate nodes. Four links in the North (N), South (S), East (E), and West (W) directions on each contour are interconnected with higher levels. As shown in Fig.3.2, the BM has four links in each direction as defined by:

$$G = \left[\begin{array}{l} 00_S, 01_S, 02_S, 03_S; 00_W, 10_W, 20_W, 30_W; \\ 30_N, 31_N, 32_N, 33_N; 03_E, 13_E, 23_E, 33_E \end{array} \right] \tag{3.1}$$

where, G is the gate node. This equation represents the free links of the BM which is used for higher level connection. Two digits represents PEs at the contours of an xy-plane, where the first digit representing the x-axis and the second digit representing the y-axis. The suffix is used to represents which direction of links is used for higher level connection. N-direction and S-direction links are used in the y-axis interconnection of Level-2. Similarly, the x-axis of Level-2 uses the E-direction and W-direction links.

3.2.2 Higher Level Interconnection

According to the interconnection philosophy of the Hierarchical Torus Network (HTN), successive higher level networks are built by recursively interconnecting lower level subnetworks in a 2D-torus of size $(n \times n)$, where n is also a positive integer. If $n = 4$, a Level-2 subnetwork, for example, can be formed by interconnecting 16 BMs. As shown in Fig. 3.3, each BM is connected to its logically adjacent BMs.

The torus at Level-2 interconnects between the gate nodes in the S-direction and the gate nodes in the N-direction, and between the gate nodes in the E-direction and the gate nodes in the W-direction. The links at Level-2 interconnects gate nodes $[00_S, 01_S, 02_S, 03_S]$ and gate nodes $[30_N, 31_N, 32_N, 33_N]$, and between gate nodes $[00_W, 10_W, 20_W, 30_W]$ and gate nodes $[03_E, 13_E, 23_E, 33_E]$.

Similarly, a Level-3 subnetwork can be formed by interconnecting 16 Level-2 subnetworks, and so on. Thus, Level-L is interconnected as an $(n \times n)$ 2D-torus, where Level-$(L-1)$ is used as subnets of Level-L. Since the Level-3 interconnection includes many BMs as subnets, reasonable numbers of BMs are selected to make the 2D-torus of Level-2 modules. Basic modules with the same co-ordinate position in each Level-2 subnetworks are interconnected by a 2D-torus in a Level-3 interconnection. A similar interconnection rule is applied for higher levels. Figure 3.4 shows the interconnection of Level-3 HTN. As mentioned earlier, a Level-2 network is used as the subnet module of a Level-3 network. In this connection, the first basic module i.e., the BM_{00} from every Level-2 network is selected for the interconnection of a Level-3 HTN.

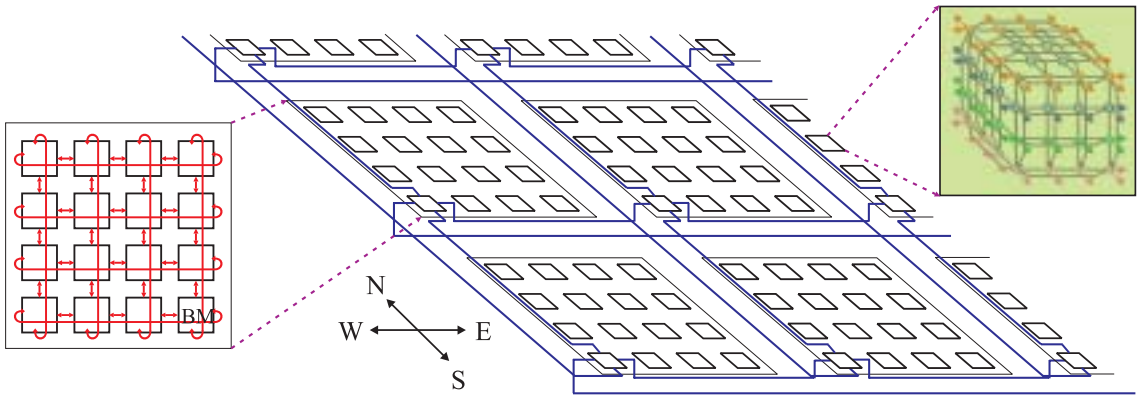


Figure 3.4: Interconnection of a Level-3 HTN

A basic module with $m = 4$ and the higher levels with $n = 4$ are perhaps the most interesting one because it has better granularity than the larger sizes. With $m = 8$, the size of the basic module becomes $(8 \times 8 \times 8)$ with 512 nodes. Correspondingly, with $n = 8$ the second level would have 64 basic modules. In this case, the total number of nodes in Level-2 network is 32768. Clearly, the granularity of the family of networks is rather coarse. In addition to this, the matter of redundancy and reconfiguration becomes more difficult.

Note that the choice of the subnetworks to build a higher level is quite natural. This choice maintains the regularity of the structure of the network and, for reasons which will become apparent in the next section, makes addressing convenient. Several lemmas are stated below, without proof. The proofs are straightforward, and are omitted for brevity.

Lemma 3.1 *A $(m \times m \times m)$ basic module has $4 \times m^2$ free ports for higher level interconnection.*

It is useful to note that for each higher level interconnection, a BM must use $4m(2^q)$ of its free links. $2m(2^q)$ free links for y-direction interconnections and $2m(2^q)$ free links for x-direction interconnections. Here, $q \in \{0, 1, ..., p\}$, is the inter-level connectivity, where $p = \lfloor log_2^m \rfloor$. $q = 0$ leads to minimal inter-level connectivity, while $q = p$ leads to maximum inter-level connectivity. As shown in Fig.3.2, for example, the $(4 \times 4 \times 4)$ BM has 64 free ports. If we chose $q = 0$, then 16 of the free ports and their associated links are used for each higher level interconnection. According to the interconnection philosophy of the higher level network, if $q = 0$, as seen in Fig. 3.2, free ports and their associated links of each contours of the xy-plane are used for each higher level network.

Lemma 3.2 *The highest level network which can be built from $(m \times m \times m)$ basic module is $L_{max} = 2^{p-q} + 1$.*

Setting $q = 0$, for example, leads to the highest possible level that $(m \times m \times m)$ can be interconnected to. The limitation of maximum possible highest level network is not a serious constraint. For the case just considered ($4 \times 4 \times 4$ BM with $q = 0$) a network built with the highest level, Level-5, consists of 4.2 millions PEs.

Lemma 3.3 *The total number of nodes in a network having $(m \times m \times m)$ BMs and $(n \times n)$ higher level is $N = \left[m^3 \times n^{2(L_{max}-1)}\right]$. $L_{max} = (2^{p-q} + 1)$ denotes the highest level of the network.*

Thus, the maximum number of nodes which can be interconnected by the HTN is $N = \left[m^3 \times n^{2(2^{p-q})}\right]$.

3.2.3 Addressing and Routing

PEs in the BM are addressed by three base-m numbers, the first representing the x-axis, the second representing the y-axis, and the last representing the z-axis. PEs at Level-L are addressed by two base-n numbers, the first represents the x-axis and the second represents the y-axis. As shown in Fig. 3.2 PEs in the BM are addressed three base-4 digits since it is a $(4 \times 4 \times 4)$ basic module. The address of a PE at Level-L is represented as shown in Eq. 3.2

$$A^L = \begin{cases} (a_z)(a_y)(a_x) & (a_z, a_y, a_x = 0, 1, ..., m-1) \quad \text{if } L = 1, \text{ i.e., BM} \\ (a_y^L)(a_x^L) & (a_y^L, a_x^L = 0, 1, ..., n-1) \quad \text{if } L \geq 2 \end{cases} \tag{3.2}$$

More generally, in a Level-L HTN, the node address is represented by:

$$\begin{aligned} A &= A^L A^{L-1} A^{L-2} \ ... \ ... \ A^2 A^1 A^0 \\ &= a_n \ a_{n-1} \ a_{n-2} \ a_{n-3} \ ... \ ... \ a_3 \ a_2 \ a_1 \ a_0 \\ &= a_{2L} \ a_{2L-1} \ a_{2L-2} \ a_{2L-3} \ ... \ ... \ a_3 \ a_2 \ a_1 \ a_0 \\ &= (a_{2L} \ a_{2L-1}) \ (a_{2L-2} \ a_{2L-3}) \ ... \\ &\quad ... \ (a_4 \ a_3)(a_2 \ a_1 \ a_0) \end{aligned} \tag{3.3}$$

Here, the total number of digits is $n = 2L$, where, L is the level number and n is the position of the node in the network. The first group contains three digits and the rest of the groups contain two digits. Groups of digits run from group number 1 for Level-1, i.e., the basic module, to group number L for the L-th level. In particular, i-th group $(a_{2i} \ a_{2i-1})$ indicates the location of a Level-$(i-1)$ subnetwork within the i-th group to which the node belongs to; $i = 1, \ 2, \ ... \ ..., \ L$. In a two level network, for example, the address becomes $(a_4 \ a_3 \ a_2 \ a_1 \ a_0)$. The last group of digits $(a_4 \ a_3)$ identifies the BM to which the node belongs to and the first group of digits $(a_2 \ a_1 \ a_0)$ identifies the node within that basic module.

Routing of messages in the HTN is performed from the top level to the bottom level. That is, it is first done at the highest level network, then, after the packet reaches its highest level sub-destination, routing is continued within the subnetwork to the next lower level sub-destination. This process is repeated until the packet arrives at its final destination. When a packet is generated at a source node, the node checks its destination. If the packet is destined to the current BM, the routing is performed within the BM only. If the packet is addressed to another BM, the source node sends the packet to the outlet node which connects the BM to the level at which the routing is performed.

Suppose a packet is to be transported from a source node 0000000 to destination node 1131230. In this case, we see that routing should first be done at Level-3, therefore the source node sends the packet to the Level-3 outlet node 0000130, whereupon the packet is routed at Level-3. After the packet reaches the $(1, 1)$ Level-2 network, then routing within that network is continued until the packet reaches the BM $(3, 1)$. Finally, the packet is routed to its destination node $(2, 3, 0)$ within that BM.

In general, multiple paths exist for routing a packet in the network. Routing a packet at a given level can be performed in different ways. These multiple paths can be useful for an adaptive routing algorithm, where the router may use information about the state of the network and act accordingly. However, we will consider a simple deterministic routing algorithm. Routing at the higher level is performed firstly in the y-direction and then in the x-direction. In BM the routing order is initially in the z-direction, next in the y-direction and finally in the x-direction.

Routing in the HTN is strictly defined by the source node address and the destination node address. Let a source node address be $s_n, s_{n-1}, s_{n-2}, ..., s_1, s_0$, a destination node address be $d_n, d_{n-1}, d_{n-2}, ..., d_1, d_0$, and a routing tag be $t_n, t_{n-1}, t_{n-2}, ..., t_1, t_0$, where, $t_i = d_i - s_i$. The source node address of HTN is expressed as $s = (s_{2L}s_{2L-1}), ..., (s_2, s_1, s_0)$. Similarly, the destination node address is expressed as $d = (d_{2L}d_{2L-1}), ..., (d_2, d_1, d_0)$. Figure 3.5 shows the routing algorithm for the HTN.

Routing Algorithm for HTN

Routing HTN(s,d);
source node address:$s_n, s_{n-1}, s_{n-2}, ..., s_1, s_0$
destination node address: $d_n, d_{n-1}, d_{n-2}, ..., d_1, d_0$
tag: $t_n, t_{n-1}, t_{n-2}, ..., t_1, t_0$
  for $i = n : 3$
    if ($i/2 = 0$ and ($t_i > 0$ or $t_i = -3$)), routedir = North; endif;
    if ($i/2 = 0$ and ($t_i < 0$ or $t_i = 3$)), routedir = South; endif;
    if ($i\%2 = 1$ and ($t_i > 0$ or $t_i = -3$)), routedir = East; endif;
    if ($i\%2 = 1$ and ($t_i < 0$ or $t_i = 3$)), routedir = West; endif;
      while ($t_i \neq 0$) do
        $N_z = outlet_z(s, d, L,$ routedir)
        $N_y = outlet_y(s, d, L,$ routedir)
        $N_x = outlet_x(s, d, L,$ routedir)
        BM_Routing(N_z, N_y, N_x)
        if (routedir = North or East), move packet to next BM; endif;
        if (routedir = South or West), move packet to previous BM; endif;
        $t_i = t_i - 1$;
      endwhile;
  endfor;
    BM_Routing(t_z, t_y, t_x)
end
BM_Routing (t_2, t_1, t_0);
BM_tag t_2, t_1, t_0 = receiving node address (r_2, r_1, r_0) − destination (d_2, d_1, d_0)
  for $i = 2 : 0$
    if ($t_i > 0$ and $t_i \leq 2$) or ($t_i < 0$ and $t_i = -3$), movedir = positive; endif;
    if ($t_i > 0$ and $t_i = 3$) or ($t_i < 0$ and $t_i \geq -2$), movedir = negetive; endif;
    if (movedir = positive and $t_i > 0$), distance $= t_i$; endif;
    if (movedir = positive and $t_i < 0$), distance $= 4 + t_i$; endif;
    if (movedir = negative and $t_i < 0$), distance $= t_i$; endif;
    if (movedir = negative and $t_i > 0$), distance $= -4 + t_i$; endif;
 endfor
  while($t_2 \neq 0$ or distance$_2 \neq 0$) do
    if (movedir = positive), move packet to $+z$ node; distance$_2$ = distance$_2 - 1$; endif;
    if (movedir = negetive), move packet to $-z$ node; distance$_2$ = distance$_2 + 1$; endif;
  endwhile;
  while($t_1 \neq 0$ or distance$_1 \neq 0$) do
    if (movedir = positive), move packet to $+y$ node; distance$_1$ = distance$_1 - 1$; endif;
    if (movedir = negetive), move packet to $-y$ node; distance$_1$ = distance$_1 + 1$; endif;
  endwhile;
  while($t_0 \neq 0$ or distance$_0 \neq 0$) do
    if (movedir = positive), move packet to $+x$ node; distance$_0$ = distance$_0 - 1$; endif;
    if (movedir = negetive), move packet to $-x$ node; distance$_0$ = distance$_0 + 1$; endif;
  endwhile;
end

Figure 3.5: Routing algorithm of HTN

3.3 Static Network Performance

The topology of an interconnection network determines many architectural features of that parallel computer and affects several performance metrics. Although the actual performance of a network depends on many technological and implementation issues. Several topological properties and performance metrics can be used to evaluate and compare different network topologies in a technology-independent manner. Most of these properties are derived from the graph model of the network topology.

In this section, we discuss some of the properties and performance metrics that characterize the cost and performance of an interconnection network. Any design of an interconnection networks is a trade-off of various requirements. Evaluation parameters to be considered include node degree, diameter, cost ($= degree \times diameter$), average distance, bisection width, and connectivity.

3.3.1 Node Degree

The *Node Degree* (ND) is defined as the number of physical channels emanating from a node. This attribute is a measure of the node's I/O complexity. The node degree should kept constant and as small as possible. Small and constant node degree allows simple and low cost routers which amortizes the design cost. Constant degree networks are easy to expand and the cost of the network interface of a node remains unchanged with increasing size of the network. It is also suitable for efficient VLSI implementation. On the other hand, small node degree implies less links and lower connectivity and larger distances. For HTN, the node degree is independent of the network size. Since each node has eight channels, therefor the degree of a node is 8. This is shown in Fig. 3.6.

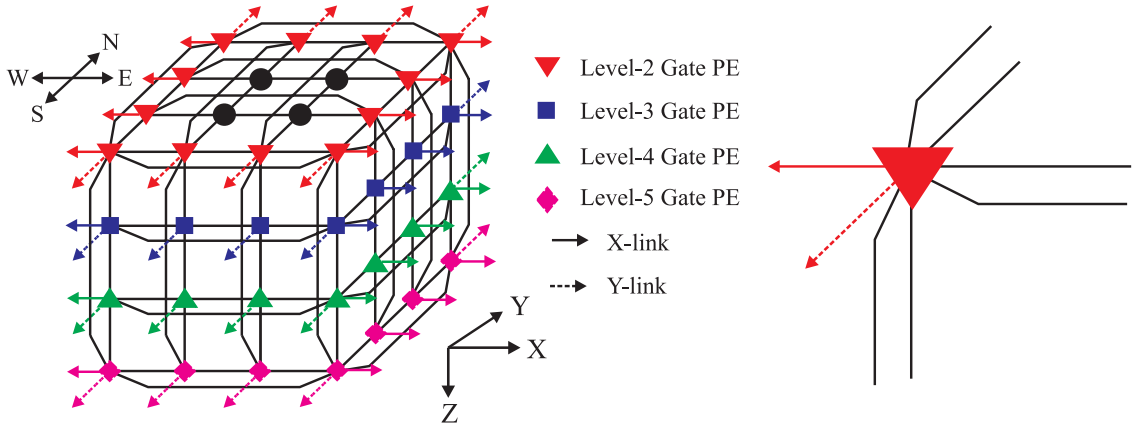


Figure 3.6: Illustration of degree of HTN

3.3.2 Diameter

In an interconnection network, communication between two nodes that are not directly connected must take place through other nodes. The length of a communication path from a given source node to a given destination node is the number of links traversed along the path between the two nodes. Most common network topologies provide multiple paths between pairs of nodes; since the delay is likely to increase with the length of the path, the shortest path is usually preferred for communication. Thus, the length of the shortest path between a given pair of nodes becomes an important metric in the evaluation of the interconnection network. This is captured in the quantity *diameter*, which is defined as the maximum among the lengths of shortest paths between all possible pairs of nodes.

In an interconnection network, the diameter can be computed by finding the maximum among the lengths of the shortest paths between all possible pairs of nodes. Diameter represents the worst-case distance that any message in the network may have to travel if routing is always along the shortest paths. If the message delay is proportional to the number of links traversed, this provides an upper bound on the delay in the absence of any interfering traffic. The diameter of a network directly affects the time for broadcasting a message from one node to all the nodes. The lower the diameter of a network the shorter the time to send a message from one node to the node farthest away from it.

Since reducing the diameter is likely to improve the performance of an interconnection network, the problem of designing interconnection network with low diameter is still a current research topic. In this section, we evaluate the diameter of the HTN and compare it with other network topologies. To evaluate the diameter of the HTN, define the maximum number of steps of routing in each level as follows:

- $D_{BM}^{to\ y-gate-L}$: The maximum number of steps from source node PE in the BM to the gate PE at Level-L in the y-axis.
- $D_{2D-torus}$: The maximum number of steps for an $(n \times n)$ 2D-torus.
- $D_{BM}^{axis\ move}$: The maximum number of steps between the gate PEs during a dimension change from the x-axis to y-axis and vice versa.
- $D_{BM}^{Level\ move}$: The maximum number of steps between the gate PE in the x-axis and the gate PE in the x-axis at Level-L.
- $D_{BM}^{to\ x-gate-2}$: The maximum number of steps between the gate nodes in the x-axis at Level-2 in the BM.

Table 3.1: Diameter of HTN with Level-L

Level (L)	2	3	4	5
No. of PEs	2^{10}	2^{14}	2^{18}	2^{22}
Diameter (D)	19	34	49	64

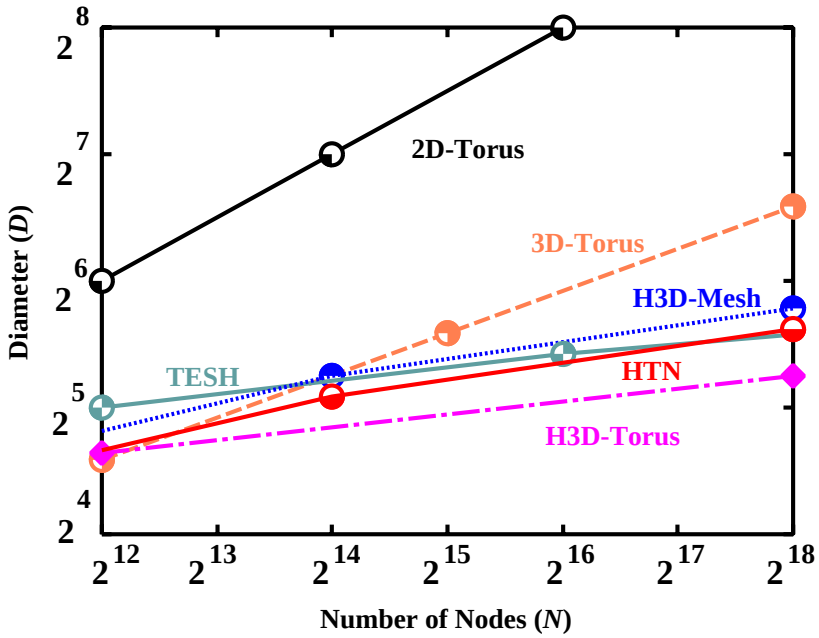


Figure 3.7: Diameter of networks as a function of No. of nodes *(N)*

The routing algorithm gives the diameter of the HTN as follows:

$$\begin{aligned} D &= D_{BM}^{to\ y-gate-L} + (L-1)D_{2D-torus} + (2L-3) \times D_{BM}^{axis\ move} + \\ &\quad (L-2)D_{BM}^{Level\ move} + D_{BM}^{to\ x-gate-2} \end{aligned} \tag{3.4}$$

If we choose, $m = 4$ and $n = 4$, then the values of each distance in the HTN are given by $D_{BM}^{to\ y-gate-L} = 6$, $D_{2D-torus} = 4$, $D_{BM}^{axis\ move} = 3$, $D_{BM}^{Level\ move} = 5$, and $D_{BM}^{to\ x-gate-2} = 6$. The diameter of the HTN with Level-L is summarized in Table 3.1.

After simplification, the diameter D in terms of Level-L is $D = 15L - 11$. The total number of processing elements of the HTN with Level-L is:

$$\begin{aligned} N &= 64\left(16^{L-1}\right) = 64\left(16^{\frac{D+11}{15}-1}\right) = 64\left(16^{\frac{D-4}{15}}\right) \\ D &= 15\left(\log_{16}{}^{\frac{N}{64}}\right) + 4 \\ D &= O\left(\log_{16}{}^{\frac{N}{64}}\right) \approx O\left(\log_{16}{}^{N}\right) \approx \frac{1}{4}O\left(\log_{2}{}^{N}\right) \\ D &\approx O\left(\log N\right) \end{aligned} \tag{3.5}$$

So, the diameter of the HTN is of $O(\log N)$.

Figure 3.7 shows a comparison of network diameter for HTN with several networks. For evaluation of diameter for HTN, we have considered the size of BM is $(4 \times 4 \times 4)$ and the size of higher level network is (4×4). The ordinate indicates the diameter of a network for different sized networks. Each curve stands for a particular family of networks. It is seen that HTN has much smaller diameter than conventional k-ary 2-cube [26] and H3D-mesh network [18] and also smaller than TESH network[5–8] for a medium-sized network.

3.3.3 Cost

Inter-node distance, message traffic density, and fault-tolerance are dependent on the diameter and the degree of a node. The product (*diameter* $\times$ *degree of a node*) is a good criterion to measure the relationship between cost and performance of a multiprocessor system [19, 20]. An interconnection network with a large diameter has a very low message passing bandwidth and a network with a high node degree is very expensive. In addition, a network should be easily expandable; there should be no changes in the basic node configuration as we increase the number of nodes.

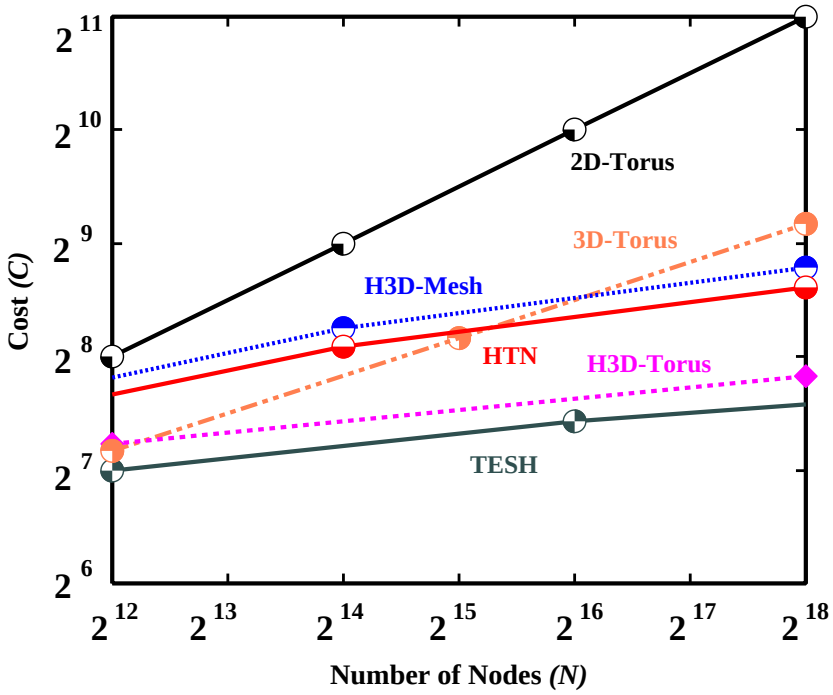


Figure 3.8: Cost of different networks as a function of No. of nodes *(N)*

Figure 3.8 shows a comparison of cost for HTN with several other networks. The ordinate indicates the cost of a network for different sized networks. Each curve stands for a particular family of networks. It is seen that HTN has much smaller cost than conventional k-ary n-cube [26] and H3D-mesh network [18].

3.3.4 Average Distance

Although the diameter is used to compare the network performance of different network topologies, it may not always be indicative of the actual performance of the networks. One reason is that a node in a multicomputer network communicates with many other nodes during the execution of the program; hence, on average, a shorter path than the diameter will be used. Thus, in practice it is more important to measure the distance traveled by an 'average' message. This is captured in the average distance which is defined

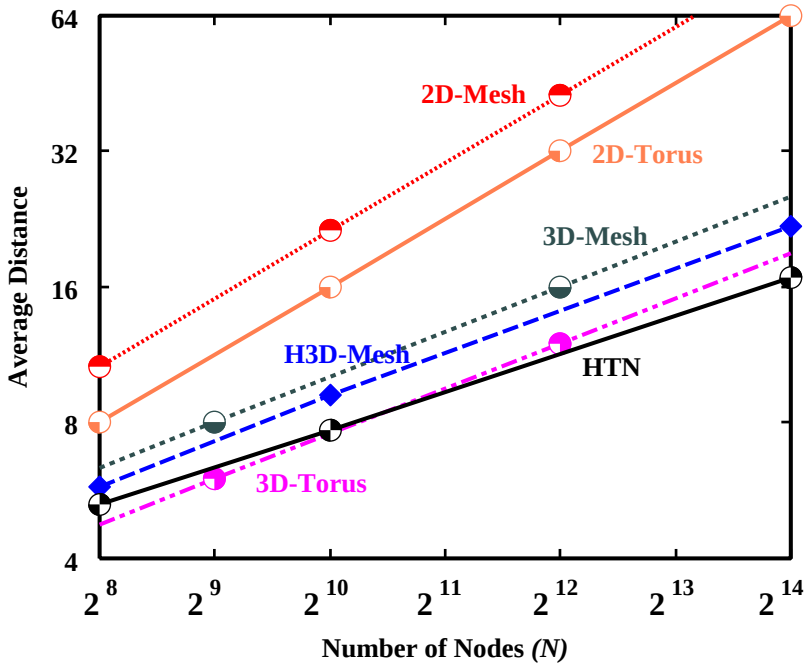


Figure 3.9: Average distance of networks as a function of No. of nodes (N)

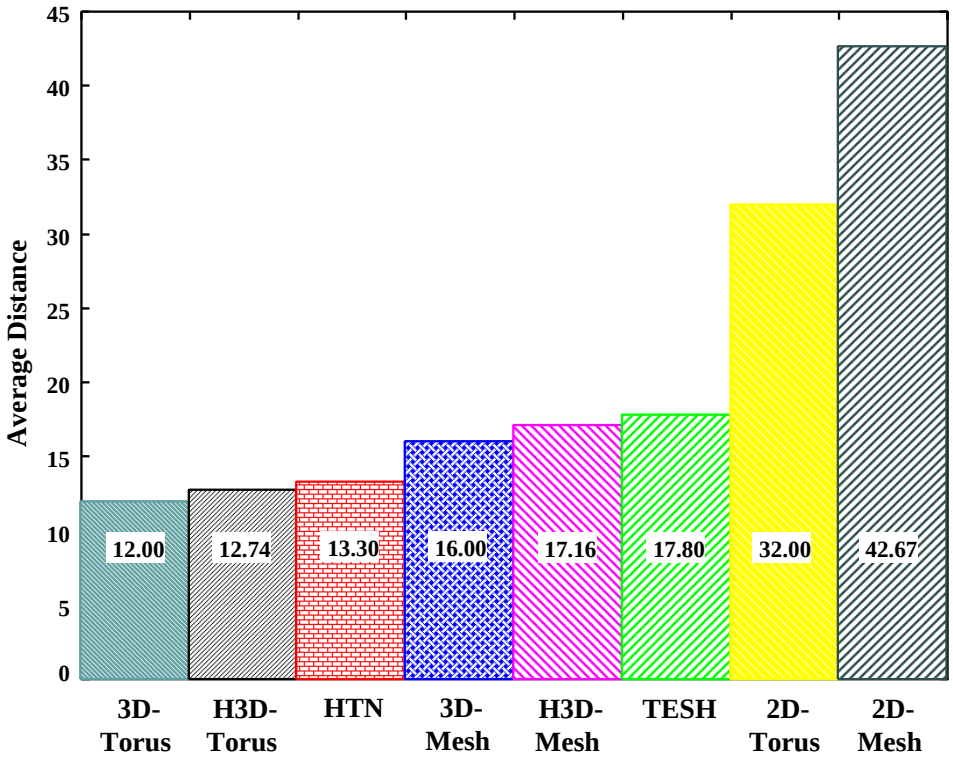


Figure 3.10: Average distance of different networks with 4096 nodes.

as the average of the lengths of all paths followed by messages in the network. While the diameter depends only on the topology of the network, the average distance also depends on the distribution of the messages among the nodes.

The distance between two nodes is defined by the number of hops in the shortest path between those nodes. And the *Average Distance* is the mean distance between all distinct pairs of nodes in a network. For communication-intensive parallel applications, the blocking time, consequently, the communication latency is expected to grow with path length. A small average distance allows small communication latency, especially for distance-sensitive routing, such as store and forward. By waiting for the entire packet to arrive at a router before forwarding to the next hop, store and forward routing results in long delays at each hop. Therefore, a smaller average distance of an interconnection network yields a smaller communication latency of that network. But it is also crucial for distance-insensitive routing, such as wormhole routing, since short distances imply the use of fewer links and buffers, and therefore less communication contention.

We have evaluated the average distance for different conventional topologies by the corresponding formula and of different hierarchical networks by simulation. Figure 3.9 shows a comparison of network average distance between the HTN and several other networks. Figure 3.10 also shows a comparison of average distance of different networks with 4096 nodes. It is seen that HTN has the smaller average distance than TESH [5–8], H3D-mesh [18], and conventional k-ary n-cube [26] networks.

Although the communication performance of a program on a multicomputer depends on the actual times taken for data transfer, diameter and average distance are useful metrics for technology-independent analyses.

3.3.5 Connectivity

The *Connectivity* measures the robustness of a network. It is a measure of the multiplicity of paths between nodes. The connectivity of a network is defined to be the minimum number of nodes or links whose removal causes the network to be disconnected into two or more components. High connectivity improves performance during normal operation by avoiding congested links, and improves fault tolerance. A network is said to be maximally fault-tolerant if its connectivity is equal to the degree of the network. Clearly, connectivity cannot exceed the degree, since the network can be partitioned by removing all neighbors of a specific node. Figure 3.11 shows that removal of 2 links disconnects the 2D-mesh network into two parts. This is the minimum value of links to disjoint the 2D-mesh network. Thus, the connectivity of 2D-mesh network is 2. The connectivity of different networks including the HTN is shown in Table 3.2.

Table 3.2: Comparison of Connectivity for different networks

	2D-mesh	2D-torus	3D-mesh	3D-torus	TESH	H3D-mesh	HTN	H3D-torus
Connectivity	2	4	3	6	2	6	6	3

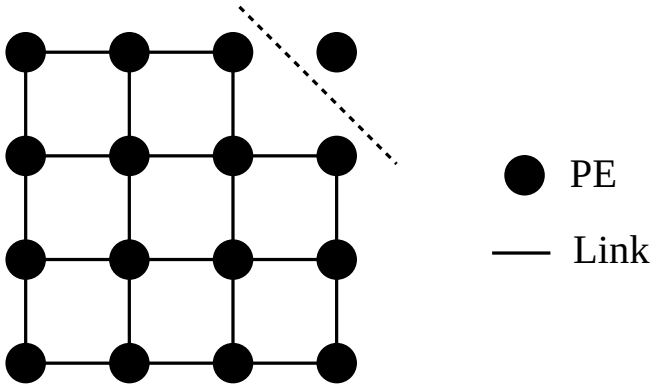


Figure 3.11: Illustration of connectivity for 2D-mesh network.

3.3.6 Bisection Width

The *Bisection width* (BW) is an important topological parameter for interconnection networks and is crucial to their cost of VLSI layout and performance. The bisection width of a network is defined as the minimum number of communication links that have to be removed to partition the network into two equal halves. Bisection width is a measure of the volume of traffic that can be handled by the network. It is also useful in estimating the area required for a VLSI implementation of the network. The bisection width of a ring network is 2, since any partition cuts across only 2 communication links. The bisection width of the HTN is calculated by:

$$BW\ (HTN) \quad = \quad 2^{q+1} \times (m \times n) \tag{3.6}$$

It is calculated by counting the number of links that need to be removed to partition the highest level (Level-L) torus. This equation is valid for higher level networks. We don't consider here the interconnection of basic modules. The basic module is simply a 3D-torus network so its bisection width is $2m^2$.

Many problems can be solved in parallel using *binary divide-and-conquer*: split the input data set into two halves and solve them recursively on both halves of the interconnection network in parallel, and then merge the results in both halves into the final result. Small bisection width implies low bandwidth between both halves and it can slowdown the final merging phase. On the other hand, a large bisection width is undesirable for VLSI design of an interconnection network, since it implies a lot of *extra chip wires*.

Figure 3.12 shows a comparison of bisection width between the HTN and several other networks. The ordinate indicates the bisection width of a network for different sized networks. Each curve stands for a particular family of networks. It is seen that the bisection width of the HTN is larger than TESH [5–8] and smaller than conventional k-ary n-cube [26] networks. Thus, the bisection width of HTN is better than other networks.

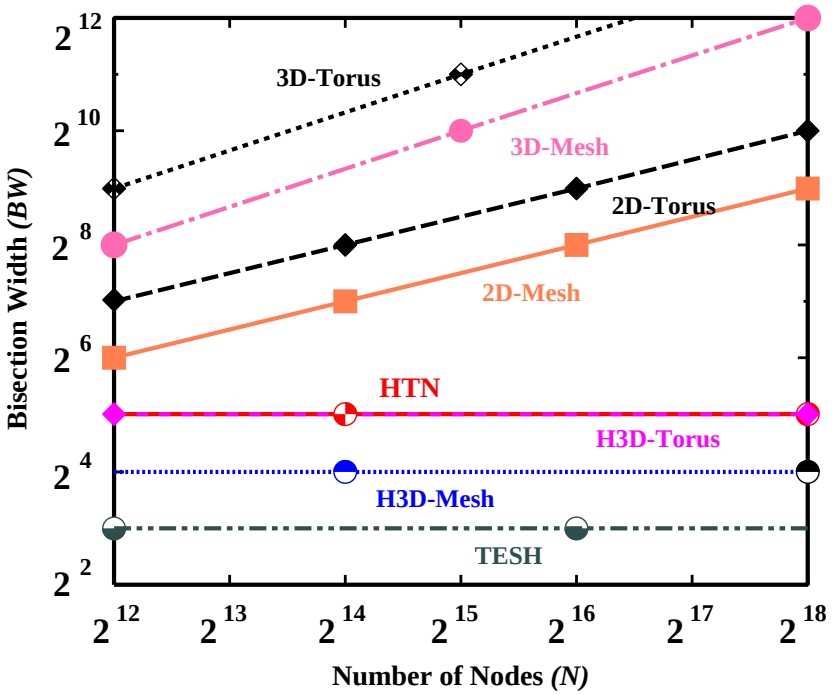


Figure 3.12: Bisection width of networks as a function of No. of nodes *(N)*

3.4 Wafer Stacked Implementation

3.4.1 3D Stacked Implementation

M. Little *et.al.* [1] developed a 3D-computer made of a 32×32 cellular array and organized as a 5 wafer stack containing two types of wafers. A three dimensional stacked implementation is attractive for massively parallel computers [2]. Figure 3.13 illustrates the structure of a 3D stacked implementation. The vertical links between wafers interconnect PEs on adjacent silicon planes. To implement a network in 3D stacked silicon planes, a sub-network is mapped to a silicon plane and sub-networks are interconnected by vertical links between silicon planes. The vertical links between adjacent planes is implemented by wafer feedthroughs and microbridges [1][2] shown in Fig. 3.14.

Vertical links (composed of feedthroughs and microbridges) realize the shortest pass between planes. However, the area of vertical links between silicon planes amount to hundred of μm^2 shown in Fig. 3.14. Thus, unconstrained use is prohibited. An efficient 3D interconnection requires reducing the number of vertical links.

3.4.2 Peak Number of Vertical Links

An important consideration in the design of a new network is the feasibility of 3D implementation. In the implementation of an interconnection network on 3D-stacked planes, one of the most important parameters is the peak number of vertical links C_{peak} between

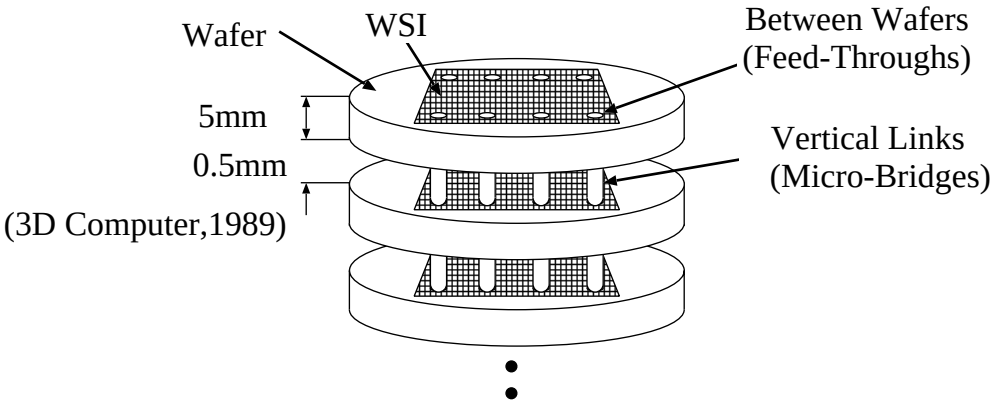


Figure 3.13: Structure of 3D stacked implementation

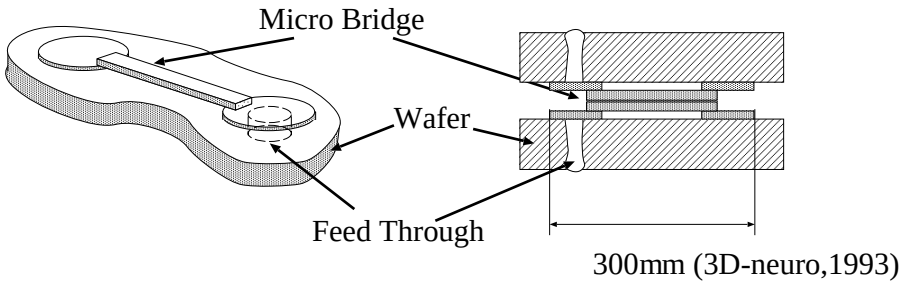


Figure 3.14: Structure of Microbridge and Feedthrough

adjacent silicon planes. The word *peak* here refers to the bulge due to the crowding of wires running between various planes.

As shown in Fig. 3.15, the PEs are placed on a square array on each silicon plane. Let h be the number of silicon planes, M be the number of PEs on each plane, and N be the total number of PEs. The relation between the h, M, N is as follows:

$$N = h \times M \tag{3.7}$$

Figure 3.16 illustrates the 3D stacked implementation of a 2D mesh network with 16 PEs on 4 stacked planes. According to Eq. 3.7, here, $N = 16$, $M = 4$, and $h = 4$. The maximum number of links between wafers is 4. For this case, as shown in Fig. 3.16, the peak number of vertical links is 4. The same scenario is illustrated in Fig. 3.17 for a 2D-torus network with 16 PEs. In this case, the peak number of vertical links is 8. Here, the peak number of vertical links is doubled because of the wraparound connections of the torus network. Thus, the peak number of vertical links of an interconnection network depends upon the architecture of the network. The illustration shown here uses bi-directional links. The peak number of vertical links using unidirectional links becomes twice as much as that using bidirectional links.

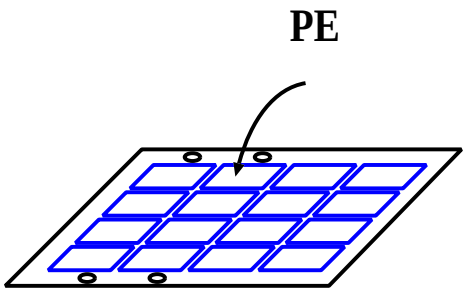


Figure 3.15: PE array in a silicon plane for wafer stacked implementation

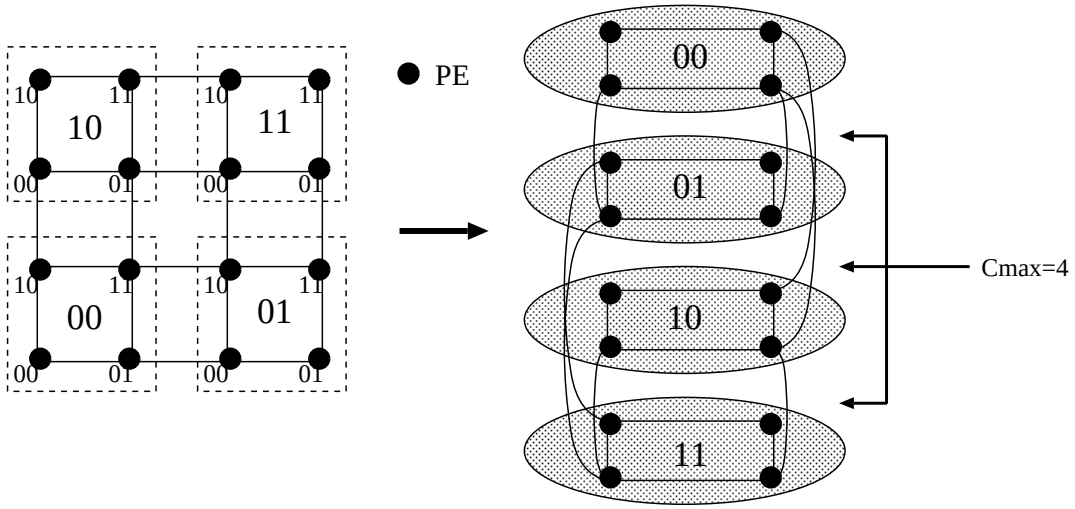


Figure 3.16: Vertical links of 2D-mesh in 3D stacked implementation

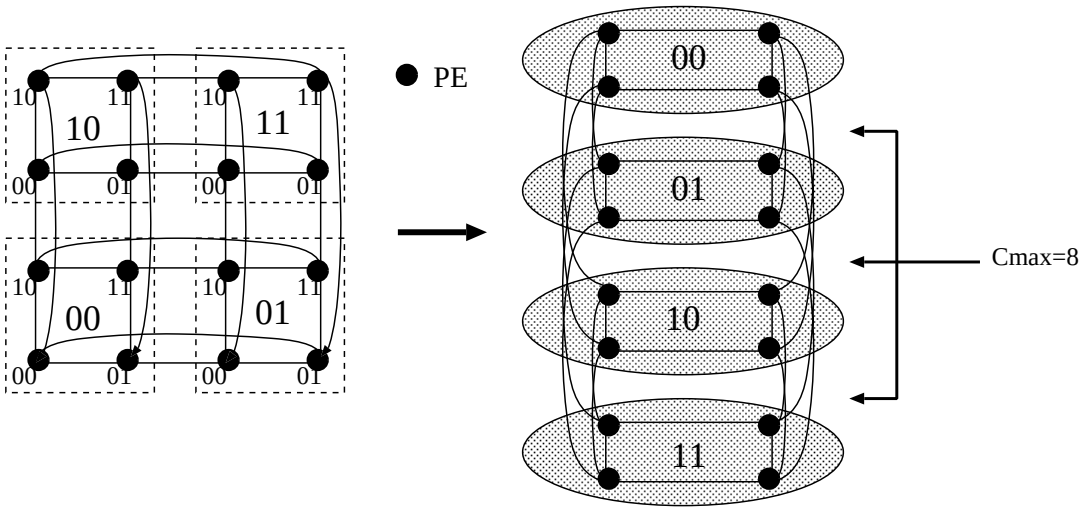


Figure 3.17: Vertical links of 2D-torus in 3D stacked implementation

3.4.3 Peak Number of Vertical Links for HTN

In this section, we evaluate the peak number of vertical links for the Hierarchical Torus Network. The PEs are placed on a square array on each silicon plane. In a hierarchical interconnection network such as HTN, let n be the number of PEs in a BM, g be the number of subnetworks at the Level-$(i-1)$ $(i \leq L \leq L)$. Let $L_d = 1 + \left\lfloor \log_g \frac{M}{n} \right\rfloor$ be the highest level on a plane, and $s = m \bmod n g^{L_d - 1}$ be the number of hierarchies at L_d. The BM of the HTN is a 3D-torus network. It consists of $(m \times m \times m)$ PEs. For instance, if $m = 4$, the BM consist of 64 PEs and is implemented on a silicon plane. Then the mapping of the HTN is given by:

1. HTN (L) is divided into h subnets in the address order.
2. Level-L_d are mapped on s silicon planes. Since interconnection between PEs at higher level is a 2D-torus, the inline scheme is applied to alignment of PEs.
3. The mapping process is repeated until Level-L is reached.

Since HTN interconnects PEs at higher levels through subnets, the peak number of vertical links at Level-L is obtained by summing the peak number of vertical links at lower levels. Thus, the peak number of vertical links C_{peak} is the summation of the peak number when one subnet of Level-i $(1 \leq i \leq L)$ is assigned on a silicon plane. HTN interconnects subnets at different levels using a 2D-torus with the same size as the number of BMs in the subnet. The peak number of vertical links C_{peak} of Level-L_i is given by:

$$\begin{aligned} C_{peak} &= C_{peak}\left(\text{2D} - \text{torus}, 16, S_{Li}\right) \\ &= \begin{cases} 10\left(16^{L_i - 2}\right) & S_{Li} = 1 \\ 12\left(16^{L_i - 2}\right) & S_{Li} = 4 \end{cases} \end{aligned} \tag{3.8}$$

where, S_{Li} is the number of subnets on a silicon plane of Level-L_i. Thus, the peak number of vertical links C_{peak} of HTN is given by:

$$\begin{aligned} C_{peak} &= C_{peak}\left(\text{2D} - \text{torus}, 16, s\right) 16^{L_d - 2} + \sum_{L_i = L_d + 1}^{L} C_{peak}\left(\text{2D} - \text{torus}, 16, 1\right) 16^{L_i - 2} \\ C_{peak} &= C_{peak}\left(\text{2D} - \text{torus}, 16, s\right) 16^{L_d - 2} + \sum_{L_i = L_d + 1}^{L} 5\left(16^{L_i - 2}\right) \end{aligned} \tag{3.9}$$

C_{peak} is calculated for the number of planes $h = 16$. The ordinate indicates the peak number of vertical links C_{peak} needed for different sized networks. Each curve stands for particular set of networks. The 3D-torus network requires the largest C_{peak} among the evaluated 3D networks.

Figure 3.18 shows that the peak number of vertical links C_{peak} for an HTN network is exactly same as the TESH network. Thus, HTN is suitable for medium-sized networks. However, a more quantitative way to compare different network families would be to use a figure of merit such as $f = N/\left(C_{peak} \times D\right)$ [5, 8], where N denotes the total number of nodes, C_{peak} is the peak number of vertical links, and D is the network diameter of the network.

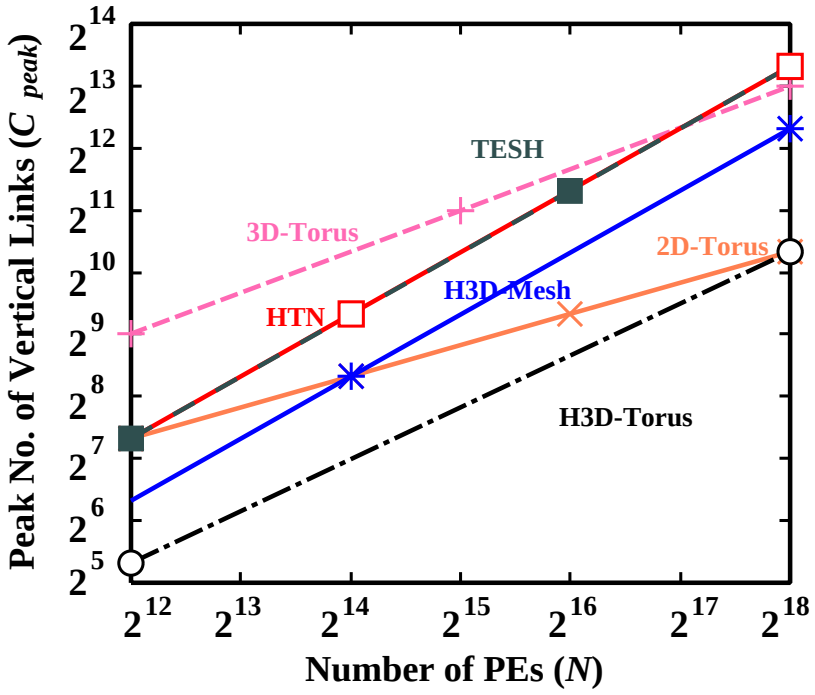


Figure 3.18: A comparison of peak number of vertical links of HTN with other networks

3.4.4 Layout Area

To discuss a suitable network for 3D stacked implementation, layout areas of network have to be evaluated. The following defines formulations of layout area of networks in 3D stacked implementations.

Let $W_{PE} \times W_{PE}$ be the chip size of a PE, $W_{MB} \times W_{MB}$ be the size of a microbridge, W_{line} be the width of a line, and p be the number of lines per link. Figure 3.19 illustrates the layout area of a 2D-torus for the case of $N = 16$, $L = 4$, and $p = 1$.

Fukuda and Horiguchi [30] proposed the layout for CCC (Cube Connected Cycles) using HC links (Hypercube links) in a 3D stacked implementation. The HC links are comprised of the $2^k - 1 = N - 1$ links of a k-hypercube. Since M PEs are allocated into $\sqrt{M} \times \sqrt{M}$ on a silicon plane, $t_x = t_y = \sqrt{M} - 1$. The wiring space is subjected to the maximum number of links in a row and column of the PE array. Let t_x and t_y be the maximum number of links in rows and columns, respectively. The wiring spaces of the links between PE arrays in x-direction and y-direction are as follows:

$$S_{TL,x} = \sqrt{M} t_x p W_{line} \tag{3.10}$$

$$S_{TL,y} = \sqrt{M} t_y p W_{line} \tag{3.11}$$

The same scheme is applied to HTN. The maximum number of torus links t_x, t_y in xy-plane depend on the number of levels on a silicon plane. For one BM on a silicon

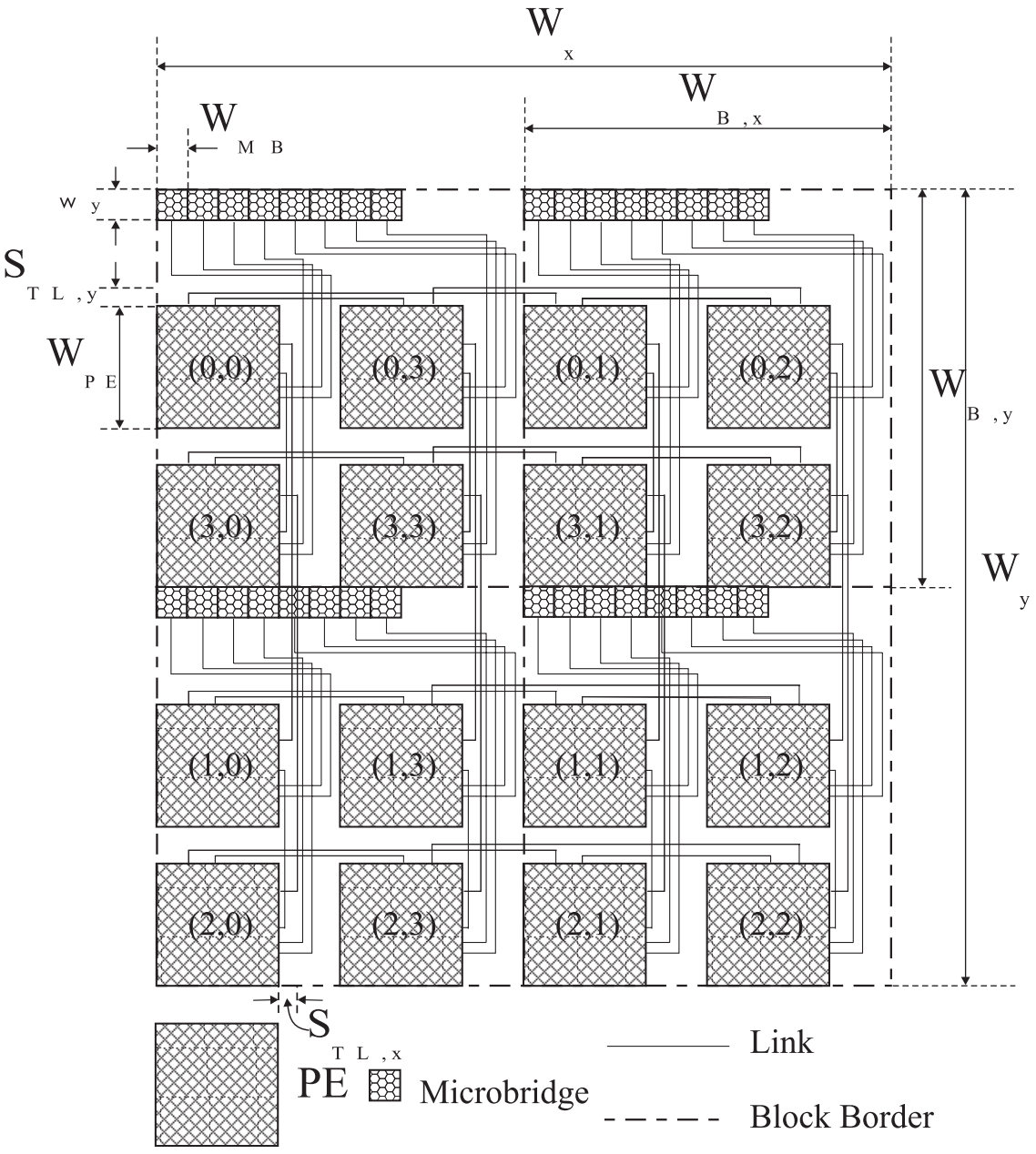


Figure 3.19: Layout Area of 2D-torus for $N = 16$, $L = 4$ and $p = 1$.

plane, we have $t_x = t_y = 8$. Links connected Level-2 are not counted into t_x and t_y. For levels higher than Level-3, the same number of links are required for a 2D-torus consisting of number of BMs in a subnet. The maximum number of torus links for levels higher than 3 is proportional to the number of BMs in a subnet at each level. Therefore, t_x, t_y for the HTN are given by:

$$L_e = \left\lceil \log_{16} \frac{m}{64} \right\rceil \tag{3.12}$$

$$t_x = t_y = \begin{cases} 8 & L_e \leq 2 \\ 8 + 2 \sum_{i=L_e}^{L} \left(4^{i-2}\right) & L_e > 2 \end{cases} \tag{3.13}$$

All the parameters needed to calculate the layout area is in 3D wafer stacked implementation are summarized in Table 3.3. Let us consider an interconnection network with 4096 PEs. For this network, the designer may implement a TESH in 16 silicon planes, and each plane has 16 Level-2 networks. Then, 16 silicon planes are required to build a stack with a total of 4096 nodes. For simplicity of presentation we will ignore the redundancy at various levels in this subsection. These issues are considered in [4]. Suppose the PEs are medium grained processors each with $2.00mm \times 2.00mm$ effective area and $3.00mm \times 3.00mm$ tile area in a $1\mu m$ CMOS technology. The total area required in the plane is $4.8cm \times 4.8cm$. Clearly, a large ULSI chip or a small wafer can easily support the sub-network and its associated control and power wiring.

Two or more high level subnets of the hierarchical network can be placed on larger silicon devices, which would correspondingly reduce the number of silicon planes in the stack. In particular, we assume a part of Level-2 sub-network per plane. Sixteen such planes would require building a stack with 4096 nodes. Given below is an estimation of the vertical wiring needed to interconnect the planes.

- Link width $= \omega$ *bits* bits
- Number of BM in each Level-2 network = 16
- Peak number of Level-3 links per BM = 10
- Number of vertical links $= 16 \times 10 = 160$
- Number of vertical wires$= 160 \times \omega$

For stacked planes the wiring pitch on an area basis is approximately $300\mu m \times 300\mu m$. Since the vertical channel includes multiplexers to connect vertical links between silicon planes, we compute the area needed for the vertical connections of network using $500\mu m \times 500\mu m$.

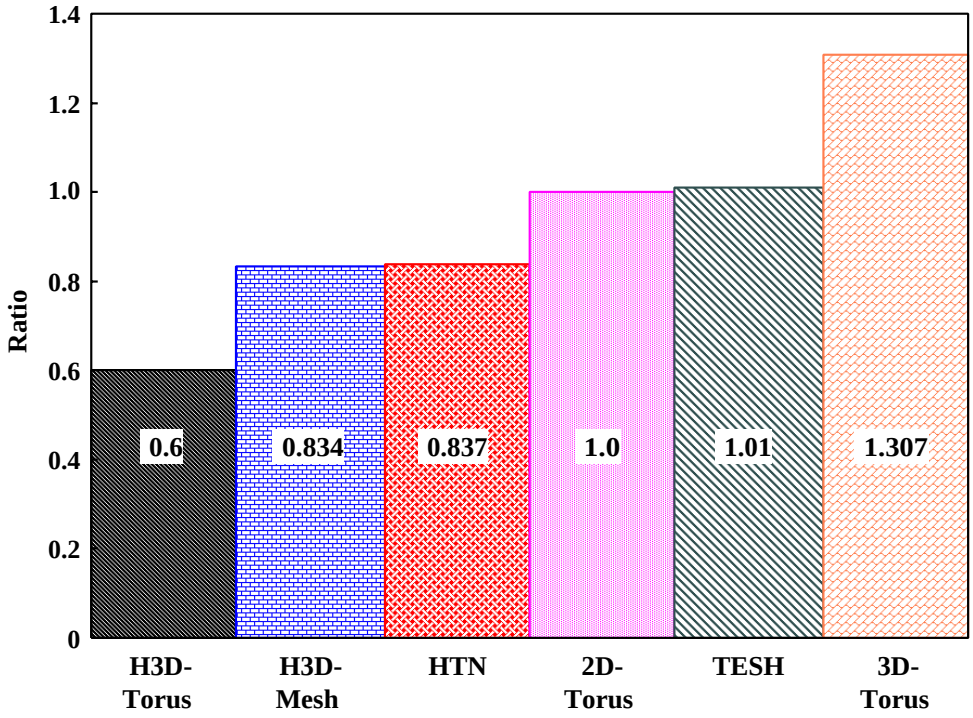


Figure 3.20: Normalized layout area

Table 3.3: Parameters for layout area in 3D stacked implementation

Parameter	Expression
Size of a PE	$W_{PE} \times W_{PE}$
Size of a microbridge	$W_{MB} \times W_{MB}$
No. of lines per links	p
No. of blocks	$\sqrt{L} \times \sqrt{L}$
Peak number of links in R & C	$t_x,\ t_y$
Size of a PE array on a plane	$S_{PE,x} = \sqrt{m}W_{PE},\ S_{PE,y} = \sqrt{m}W_{PE}$
Wiring space of links bet^n PEs	$S_{TL,x} = \sqrt{m}t_x pW_{line},\ S_{TL,x} = \sqrt{m}t_x pW_{line}$
Wiring space of links bet^n micro	$S_{DMB,x} = C_{max}pW_{line},\ S_{DMB,y} = C_{max}pW_{line}$
Size of a block without microbridge	$W'_x = S_{PE,x} + S_{TL,x} + S_{DMB,x},$ $W'_y = S_{PE,y} + S_{TL,y} + S_{DMB,y},$
Block size without microbridge	$W'_{B,x} = \frac{W'_x}{\sqrt{L}},\ W'_{B,x} = \frac{W'_x}{\sqrt{L}}$
Area of microbridge in a block	$w_y = \lceil \frac{cpW_{MB}}{W_{B,y}} \rceil W_{MB}$
Block size with microbridge	$W_{B,x} = W'_{B,x},\ W_{B,x} = W'_{B,x} + w_y$
Total layout size	$W_x = W_{B,x}\sqrt{L},\ W_y = W_{B,y}\sqrt{L}$
Total layout area	$A = W_x \times W_y$

Clearly, the silicon plane described earlier can easily support the wiring needed for the vertical connections. For comparison, the number of vertical wires required for the 2D-torus, 3D-torus, TESH, and HTN with 4096 nodes are considered.

Figure 3.20 shows a normalized layout area by 2D-torus for comparison with several other networks. HTN can be implemented on a smaller silicon area (about 84%) than the silicon area of the 2D-torus. The peak number of vertical links for an HTN is exactly same as the TESH network [5–8], but the layout area is around 17% less.

3.4.5 Maximum Wire Length

The cost of VLSI systems is predominantly that of connecting wires, and the performance is limited by the delay introduced by these interconnections. Thus, to achieve the required performance, the network must make efficient use of the available wires. The length of the longest wire [18] is an important parameter in the design of an interconnection network. The performance of a network is strongly influenced by the longest links.

The operating speed of a network is limited by the physical length of its links. Thus, the maximum length of a wire can be used to describe and compare the maximum physical speeds that the various network topologies can attain. The length of the longest wire may become more important than the diameter of the network. We will assume that all networks have a planar implementation. The formula commonly used to describe the wire length [26] of k-ary n-cubes is:

$$LEN_{MAX}(k,n) \;=\; k^{\frac{n}{2}-1} \tag{3.14}$$

This assumes a square layout of nodes with each side having $\sqrt{N}$ nodes. The above formula underestimates the maximum length because it does not take into account the length of the wrap-around link. For a regular layout, the length of the wrap-around link is given by:

$$LEN_{MAX}(k,n) \;=\; \sqrt{N} - \frac{\sqrt{N}}{k} = k^{(\frac{n}{2}-1)}(k-1) \tag{3.15}$$

A similar formula can be developed for the HTN. The maximum wire length in a regular layout, representing the length of wrap-around links of the highest level torus is

$$LEN_{MAX}(HTN) \;=\; \left(n^{L-1}\right)(m-1) + \left(n^{L-1}-1\right) \tag{3.16}$$

The formula shown in Eq. 3.16 is a conservative estimation. Here we assume that the basic module is realized plane-by-plane as the physical interconnection of a 3D-torus. This is not a real 2D-planner realization. The real 2D-planner realization of a 3D-torus network is shown in Fig. 3.21. Our intuition is that a 3D wafer stacked implementation, can be considered, for brevity, by the equation shown in 3.16.

A comparison of the maximum wire length of different networks with the HTN is shown in Tables 3.4 for 256, 1024 and 4096 nodes. It is shown that the maximum wire length of HTN is smaller than other networks.

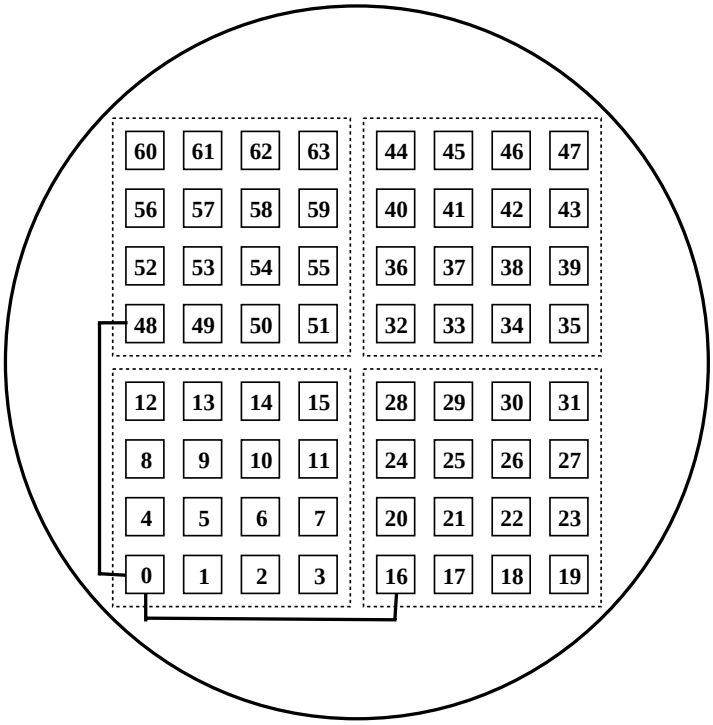


Figure 3.21: 2D-planner realization of 3D-torus network.

Table 3.4: Comparison of maximum wire length of different networks

No. of nodes	Network	Maximum Wire Length
256	2D-Torus	15
	Binary 8-cube	8
	8×2^5 CCC	8
	TESH (2,2,0)	12
	HTN	7
1024	2D-Torus	31
	Binary 10-cube	16
	x	x
	x	x
	HTN	15
4096	2D-Torus	63
	Binary 12-cube	32
	16×2^8 CCC	32
	TESH (2,3,0)	48
	HTN	31

An important advantage of a 3D implementation is the reduction of the length of the interconnects. The longest wires in a planner Level-3 network are the wraparound wires which interconnect the physically farthest Level-2 subnetwork, and there are 63. The length of this longest interconnect is $63 \times width\ of\ a\ processor(in\ tile) = 63 \times 3.0\ mm = 18.90\ cm$. However in the 3D wafer stacked implementation of a Level-3 network described in Section 3.2.2, these long wires run vertically and the longest vertical wire has a length of $16t$ where t is the thickness of a wafer plus the length of the microbridge between wafers is about $0.02\ inch = 0.051\ cm$ and the length of the microbridge is $0.002\ inch = 0.005\ cm$ [2]. Thus, t is about $0.056\ cm$ and the longest vertical wire has a length of $0.90\ cm$. In this case, the longest wires of the Level-3 HTN are the horizontal wires within the Level-2 subnetworks, and there are 15. Now, the longest wire has a length of $15 \times 3.0\ mm = 4.5\ cm$ which gives rise to a factor of 4.20 improvement over the planner implementation.

3.5 Conclusions

In this chapter, we have presented a new hierarchical interconnection network, called Hierarchical Torus Network (HTN) for massively parallel computer systems. The proposed HTN has tremendous potential to be used as an interconnection network for massively parallel computer system since the HTN can connect millions of nodes while keeping good network features. The architecture of the HTN, routing of messages, static network performance, and 3D-integration issues were discussed in detail. From the static network performance, it can be seen that the HTN possesses several attractive features including fixed diameter, small diameter, small average distance, and moderate bisection width. The network is well suited for 3D stacked implementations. It was shown that the peak number of vertical links in 3D-stacked implementations is lower for HTN as compared to other similar networks. Thus, HTN permits efficient VLSI/ULSI/WSI realization. The layout area of HTN in 3D stacked implementations is amenable to 3D implementation. In part, this is due to the fewer numbers of vertical wires needed than almost all other multi-computers networks.

Chapter 4

"There are no traffic jams when you go the extra mile."

– Anonymous

Deadlock-Free Routing for HTN

4.1 Introduction

The basic function of an interconnection network is to transfer information among the communicating nodes of a multiprocessor or multicomputer in an efficient manner. Routing is the act of transfering information across the interconnection network from a source to destination. In a broad sense, *routing* refers to the communication methods and algorithms used to implement this function. The basic issues of routing include: how to set up a communication path in the network, how to choose a path from many alternatives, and how to handle contention for resources in the network.

Routing involves finding a path from a source node to a destination node in a particular topology, and it provides communication performance. By developing a good routing algorithm, both throughput and latency can be improved. Careless design of routing algorithm may cause various problems such as deadlock. Once a deadlock has occurred, the dynamic performance is drastically reduced. Therefore, a deadlock free routing is very essential to achieve good communication performance. The most important issues in the design of a routing algorithm are high throughput, low latency message delivery, avoidance of deadlocks, livelocks, and starvation [34].

Wormhole routing is particularly susceptible to deadlocks. Deadlocks occur as a result of cyclic waits for resources by two or more communication requests. By multiplexing the physical channel into multiple virtual channels and controlling the allocation of virtual channels to communication requests, cyclic wait situation can be prevented. Dally and Seitz [32] showed that deadlocks can occur only if the channel dependency graph of the channel allocation scheme has a directed cycle. The routing algorithm is to be deadlock free if the channel dependency graph has no cycles.

This chapter is organized as follows: Some primitive considerations for deadlock free routing are presented in Section 4.2. The concept of channel dependency graph and its use to detect the deadlock in a network are described in Section 4.3. To make the ring network deadlock-free by using channel dependency graph is also shown in Section 4.3. The deadlock free routing for Hierarchical Torus Network (HTN) and investigation of minimum number of virtual channels to make the HTN deadlock free are described in Section 4.4 and 4.5, respectively. Finally, Section 4.6 concludes this chapter.

4.2 Primitive Considerations

4.2.1 Wormhole Routing

The *wormhole* (WH) [31, 32, 37] routing has been a popular switching technique in new generation of multicomputers because of its low network latency and less hardware requirement. Implementations of WH routing typically divide each message into packets, which are then divided into flits (flow control digits). A flit is the smallest unit of information that a buffer or a channel can accept or refuse. The size of the flits depends on various system parameters. Normally the bits constituting a flit are transmitted in parallel between adjacent nodes.

The header flit of a packet contains the routing information. As the header flit advances along the specified route according to the routing information, the remaining data flits of the packet follow the header flit through the network in a pipelined fashion, as illustrated in Fig. 4.1. Each incoming data flit of a message is simply forwarded along the same output channel as the preceding data flit. The header flit will reserve network resources exclusively for its message and the tail flit will release each resource after it has passed it. Thus, the message will traverse a network like a worm through a hole. By the pipelined nature of WH routing, network latency is insensitive of path length. When the header arrives at an intermediate router, the router immediately forwards the message header to the neighboring router if a usable output channel is available. In wormhole routing, once a packet occupies a channel, the channel is not released until the entire packet passes through the channel. If the header flit is blocked during advancing through the network, the trailing data flits must be blocked also, that is, wait for next channel to be available while holding channels in place.

In wormhole routing, if the required output channel is busy, the message is blocked in place. For example, in Fig. 4.2 illustrates a snapshot of a message being transmitted through routers R_1, R_2, R_3, R_4. At router R_4, message A requires an output channel that is being used by message B. The desired outgoing channel for the message A is not available. Hence, the header flit is buffered in R_4 and the data flits are also buffered in the corresponding router.

The time-space diagram of a wormhole-routed message is shown in Fig. 4.3. The shaded rectangles illustrate the propagation of header flits across the physical channels. The clear rectangles illustrate the propagation of data flits across the physical channel. The unit of flow control in wormhole routing is a single flit and, as a consequence, the use of small buffer. This figure shows the activities of each node over time when a packet is transmitted from a source node S to a destination node D through three intermediate nodes, $I1$, $I2$, and $I3$. The time required to transfer the packet between the source processor and its router, and between the last router and the destination processor, is ignored. The communication latency of the wormhole routing is nearly independent of the distance between the source and destination node. The small buffer requirements and message pipelining enable the construction of routers that are small, compact, and fast.

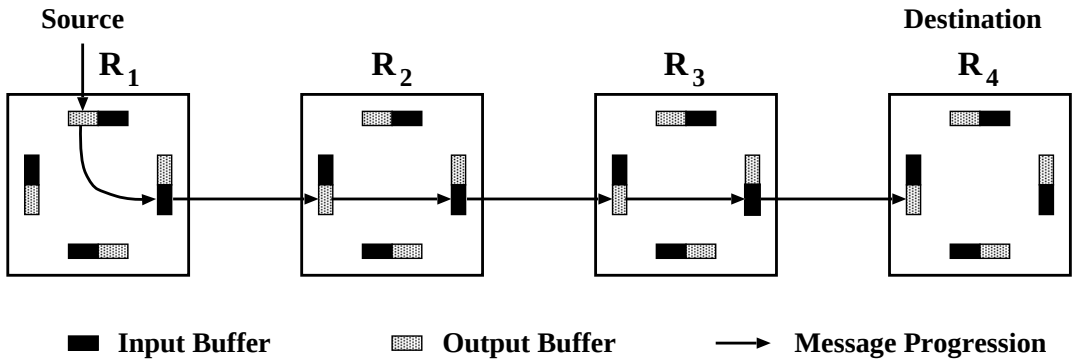


Figure 4.1: Wormhole routing

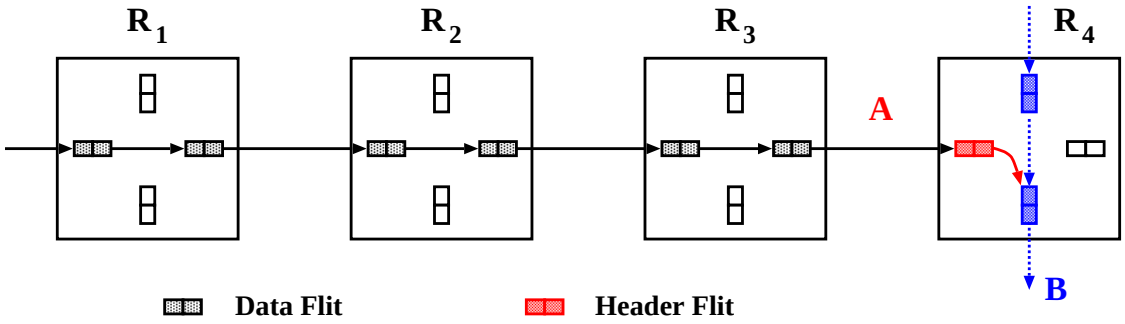


Figure 4.2: An example of the blocked wormhole-routed message

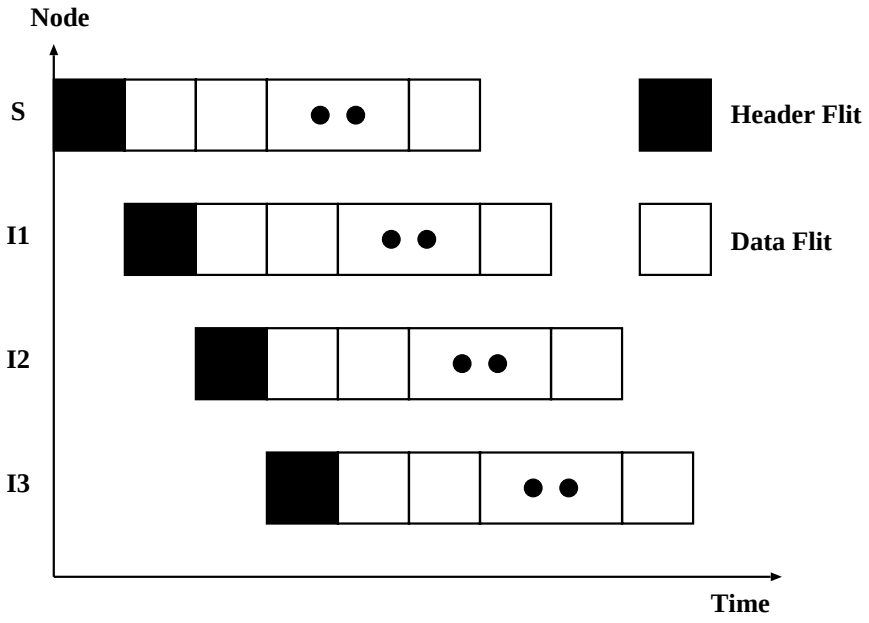


Figure 4.3: Time-space diagram of a wormhole-routed message

4.2.2 Routing Algorithm

An interconnection network topology must allow every node to send packet to every other node. In the absence of complete topology, routing determines the path selected by a message to reach its destination. Messages are divided into packets before transmission. Therefore, message and packet will be used interchangeably. Any routing algorithm for multicomputers should ensure the following:

- Each and every message injected into the network is delivered to its destination.
- Each message delivered to its destination is removed from the network in finite time.

Most of the interconnection networks use distributed routing. In distributed routing, the path is determined in a distributed manner while the packet travels across the network. Each node, upon receiving a packet, decides whether it should be delivered to the local node or forwarded to a neighboring node. In this case, the routing algorithm invoked to determine to which neighbor the packet should be sent. In a practical router design, the routing decision process must be as fast as possible to reduce the network latency. A good routing algorithm should also be easily implemented in hardware. Furthermore, the decision process usually does not require global state information of the network. Providing such information to each router creates additional traffic and requires additional storage space in each router.

Many properties of the interconnection network are a direct consequence of the routing algorithm used. Among these properties the most important ones are as follows:

- *Connectivity:* Ability to route packets from any source node to any destination node.
- *Adaptivity:* Ability to route packets through alternative paths in the presence of contention or faulty components.
- *Deadlock and livelock freedom:* Ability to guarantee that packets will not block or wander across the network forever.
- *Fault tolerance:* Ability to route packets in the presence of faulty components. Although it seems that fault tolerance implies adaptivity, this is not necessarily true. Fault tolerance can be achieved without adaptivity by routing a packet in two or more phases, storing in some intermediate nodes. Fault tolerance also requires some additional hardware mechanisms.

Routing can be classified two classes such as *deterministic* or *adaptive* routing. Under deterministic routing, the path taken by a packet is determined solely by the source and destination pair. Given the same pair of source and destination, the same path is always used regardless of other network condition. This method is also called *oblivious* routing.

On the other hand, adaptive routing takes different path for a given source and destination according to the dynamic network condition such as presence of congested and

faulty channel. Adaptive routing algorithms are classified into *partially adaptive* or *fully adaptive* routing algorithms. Fully adaptive routing algorithms is one in which all possible path between source and destination node are of potential use at the time when the sending of a packet is initiated. Partially adaptive routing algorithms use only a subset of all possible route between source and destination node.

A routing algorithm is said to be *minimal* if the path selected is one of the shortest paths between source and destination pair. Using the minimal routing algorithms, every channel visited will bring the packet closer to the destination. A *nonminimal* routing algorithm allows packets to follow a longer path, usually in response to current network conditions. If nonminimal routing algorithm is used, care must be taken to avoid the situation called livelock, in which a packet continues to be routed through network but never reach the destination.

4.2.3 Deadlock, Livelock, and Starvation

The nodes of an interconnection network send and receive messages or packets through the router. In an interconnection network, packets usually travel across several intermediate nodes before reaching the destination. As each packet whose header has not already arrived at its destination requests some buffers while keeping the buffers currently storing the packet, a deadlock may occurred. A *deadlock* [32, 35] occurs when some packets can not advances toward their destination because the buffers requested by them are full. Every packet is requesting resources held by other packet(s) while holding resources requested by other packet(s). Figure 4.4 shows an example of deadlock in wormhole routing involving four packets. All the packets involved in a deadlocked configuration are blocked forever.

Deadlock is by far the most difficult problem to solve. There are three strategies for deadlock handling: *deadlock prevention*, *deadlock avoidance*, and *deadlock recovery*. In the deadlock prevention, resources are granted to a packet in such a way that a request never leads to a deadlock. It can be achieved by reserving all the required resources before starting packet transmission. In the deadlock avoidance, resources are requested as a packet advances through the network. However, a resource is granted to a packet only if the resulting global state is safe. Finally, the deadlock recovery strategies are optimistic. Deadlock recovery strategies take no action to prohibit deadlock but detect the occurrence of deadlock and resolve the deadlock. This scheme is based on the observation that deadlock is very rare phenomenon in the real world.

A different situation arises when some packets are not able to reach their destination, even if they never block permanently. A packet may be traveling around its destination node, never reaching it because the channels required to do so are occupied by other packets. This situation is known as *livelock* [9]. It can only occur when packets are allowed to follow nonminimal paths.

Livelock is relatively easy to avoid. The simplest way consists of using only minimal path. This restriction usually increases performance in networks using wormhole switching because packets do not occupy more channels than the ones strictly necessary to reach their destination. The main motivation for the use of nonminimal paths is fault tolerance.

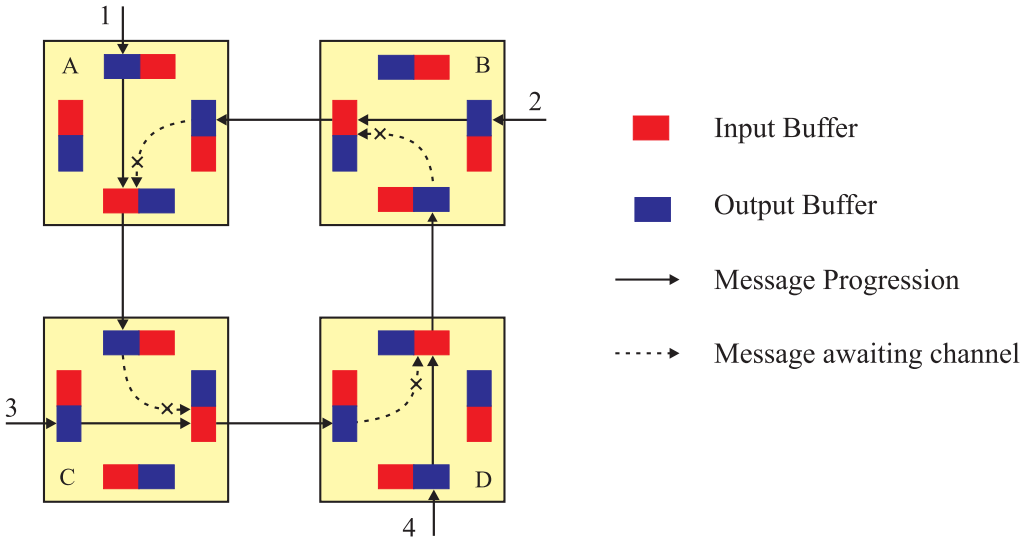


Figure 4.4: An example of deadlock involving four packets

Even when the nonminimal paths are used, livelock can be prevented by limiting the number of misrouting-routing operations.

Another situation may also arise, such that a packet may be permanently stopped if traffic is intense and the resources requested by it are always granted to other packets also requesting them. This situation is known as *starvation* [9] and it usually occurs when an incorrect resource assignment scheme is used to attribute in case of conflict. Starvation can be avoided by allocating resources such as communication channels and buffers in First-In-First-Out (FIFO) order.

Deadlock, livelock, and starvation arise because the number of resources is finite. Additionally, some of these situations may produce the others. For instance, a deadlock permanently blocks some packets. As those packets are occupying some buffers, other packets may require them to reach their destination, being continuously misrouted around their destination node and producing livelock.

4.2.4 Virtual Channel

Deadlock is a network state where no messages can advance because each message requires a channel occupied by another message. Wormhole routing is particularly susceptible to deadlock situation by its nature. Virtual channel was introduced by Dally and Seitz [32, 38] to construct a deadlock-free wormhole routing algorithm. A virtual channel is a logical entity associated with a physical channel used to distinguish multiple data streams traversing the same physical channel. Virtual channels are multiplexed over the physical channel in a demand-driven manner, with bandwidth allocated to each virtual channel as needed. It can be used for solving deadlock problem by imposing restriction on using them to break cyclic dependencies in the network [32].

Figure 4.5 shows the schematic diagram of virtual channels. Each virtual channel is

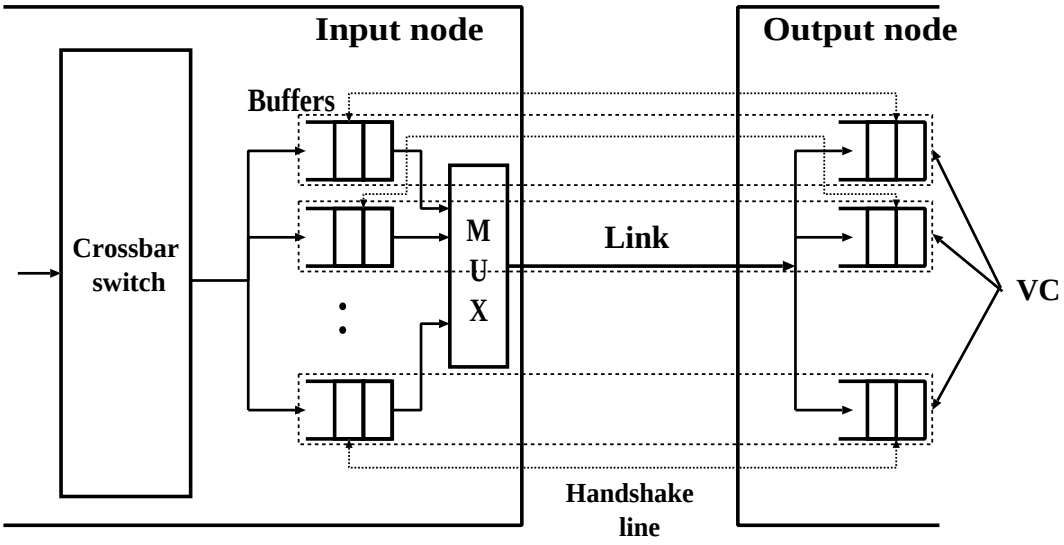


Figure 4.5: Virtual channel

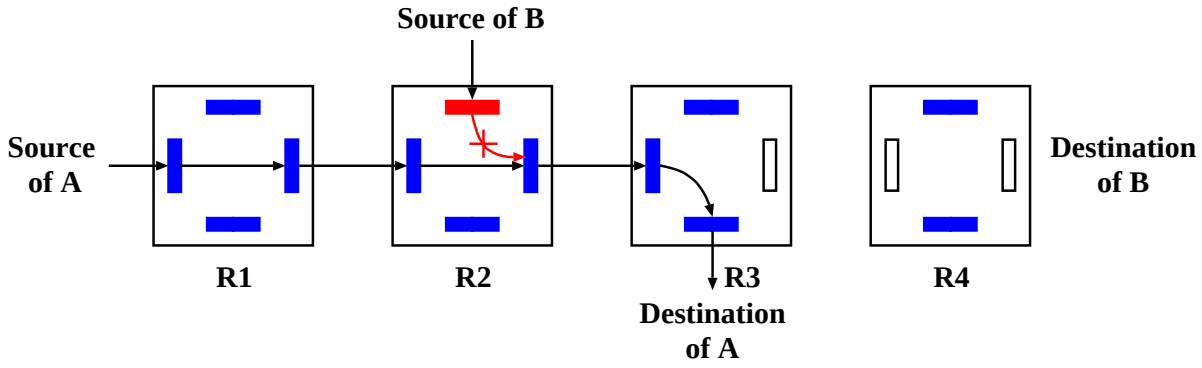


Figure 4.6: Message blocking while physical channels remain idle

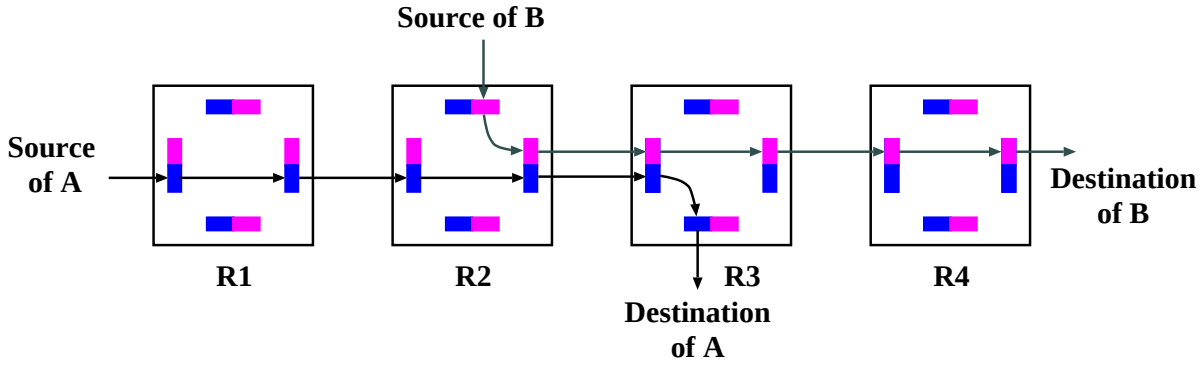


Figure 4.7: Virtual channel allows to pass blocked message

realized by an independently managed pair of message buffers. Handshake signal makes the bridge between input buffer and output buffer. Each message can share the physical channel on a flit-by-flit basis.

Adding virtual channels to an interconnection network is analogous to adding lanes to a street network. A network without virtual channels is composed of one-lane streets. In such a network, a single blocked message blocks all following messages. Adding virtual channels to the network adds lanes to the streets allowing blocked messages to be passed. Adding virtual channels to wormhole-routed networks greatly improves performance because they reduce blocking by acting as bypass lanes for non-blocked messages.

Virtual channel can also be used to improve message latency and network throughput by reducing physical link idle time [38]. By allowing messages to share a physical channel, messages can make progress rather than remain blocked. Splitting each physical channel into several virtual channels increases the number of routing choices, allowing messages to pass blocked messages. For example, Fig. 4.6 shows two messages crossing the physical channel between router $R2$ and $R3$. Only one physical channel is used here. With no virtual channels message A will prevent message B from advancing until the transmission of message A has been completed. In wormhole-routed network, message A may be blocked due to the contention elsewhere in the network while still holding its buffer and preventing message B. In this case, some channels are idle even though there may be other message in the network, e.g., message B can make productive use of these channels.

The problem of idle channels arise because of resource coupling. That is, a channel and a buffer allocated together and released together. Virtual channel decouple allocation of buffers from allocation of channels by providing multiple buffers for each channel in the network. As illustrated in Fig. 4.7, there are two single flit virtual channels multiplexed over each physical channel. By multiplexing the two messages on a flit-by-flit basis, both messages continue to make progress. The overall time a message spends blocked at a router waiting for a free channel is reduced leading to an overall reduction in individual message latency. The network throughput will also increased due to increased physical channel utilization.

4.3 Channel Dependency Graph

The theoretical model of deadlock avoidance presented in this chapter relies on the concept of channel dependency graph [32][33]. Deterministic, Dimension Order Routing is used for evaluation the dynamic communication performance of the HTN. Thus, we keep the study of channel dependency graph on dimension order routing only.

When a packet is holding a channel, and then it requests the use of another channel, there is a dependency between those channels. Both channels are in one of the paths that may be followed by the packet. If wormhole switching is used, those channels are not necessarily adjacent because a packet may hold several channels simultaneously. Channel dependency graph is used for detecting deadlocks in a wormhole routing algorithm.

The behavior of packets regarding is different depending on routing choice of each node. With dimension order routing, packets have a single routing option at each node.

Consider a set of packets such that every packet in the set has reserved a channel and it requests a channel held by another packet in the set. Obviously, that channel can not be granted, and that situation will last forever. Thus, it is necessary to remove all the cyclic dependencies between channels to prevent deadlock.

Definition 4.1 *An interconnection network I is a strongly connected directed graph, $I = G(N, C)$. The vertices (N) of the graph represent the set of processing nodes. The edge (C) of the graph represent the set of communication channels.*

Definition 4.2 *A routing function $R : C \times N \rightarrow C$ maps the current channel c_c and destination node n_d to the next channel c_n on the route from c_c to n_d, $R(c_c, n_d) = c_n$. A channel is not allowed to route itself, $c_c \neq c_n$.*

Definition 4.3 *A channel dependency graph D for a given interconnection network I and routing function R, is a directed graph, $D = G(C, E)$. The vertices of D are the channels of the interconnection network I. The edges of D are the pairs of channels (c_i, c_j) such that there is a channel dependency from c_i to c_j.*

The edges are determined by the following equation

$$E = \{(c_i, c_j) | R(c_i, n) = c_j \text{ for some } n \in N\}. \tag{4.1}$$

Definition 4.4 *A configuration is an assignment of a list of nodes to each queue. The number of flits in the queue for channel c_i will be denoted as $size(c_i)$. If the first flit in the queue for channel c_i is destined for node n_d, then $head(c_i) = n_d$. A configuration is legal if*

$$\forall c_i \in C, size(c_i) \leq cap(c_i). \tag{4.2}$$

Here, $cap(c_i)$ be the capacity of the queue of channel c_i, $size(c_i)$ be the number of flits enqueued for channel c_i, and $head(c_i)$ be the destination of the header flit enqueued for channel c_i.

Definition 4.5 *A deadlock configuration for a routing function R is a nonempty legal configuration of channel queues $\ni$*

$$\forall c_i \in C, (head(c_i) \neq d_i \text{ and } c_j = R(c_i, n) = size(c_j) = cap(c_j)). \tag{4.3}$$

In this configuration no flit is one step from its destination and no flit can advance because the queue for the next channel is full. A routing function R is *deadlock free* on an interconnection network I if no deadlock configuration exists for that function on that network.

Theorem 4.1 *A routing function **R** (deterministic) for an interconnection network **I** is deadlock free if and only if there are no cycles in the channel dependency graph **D**.*

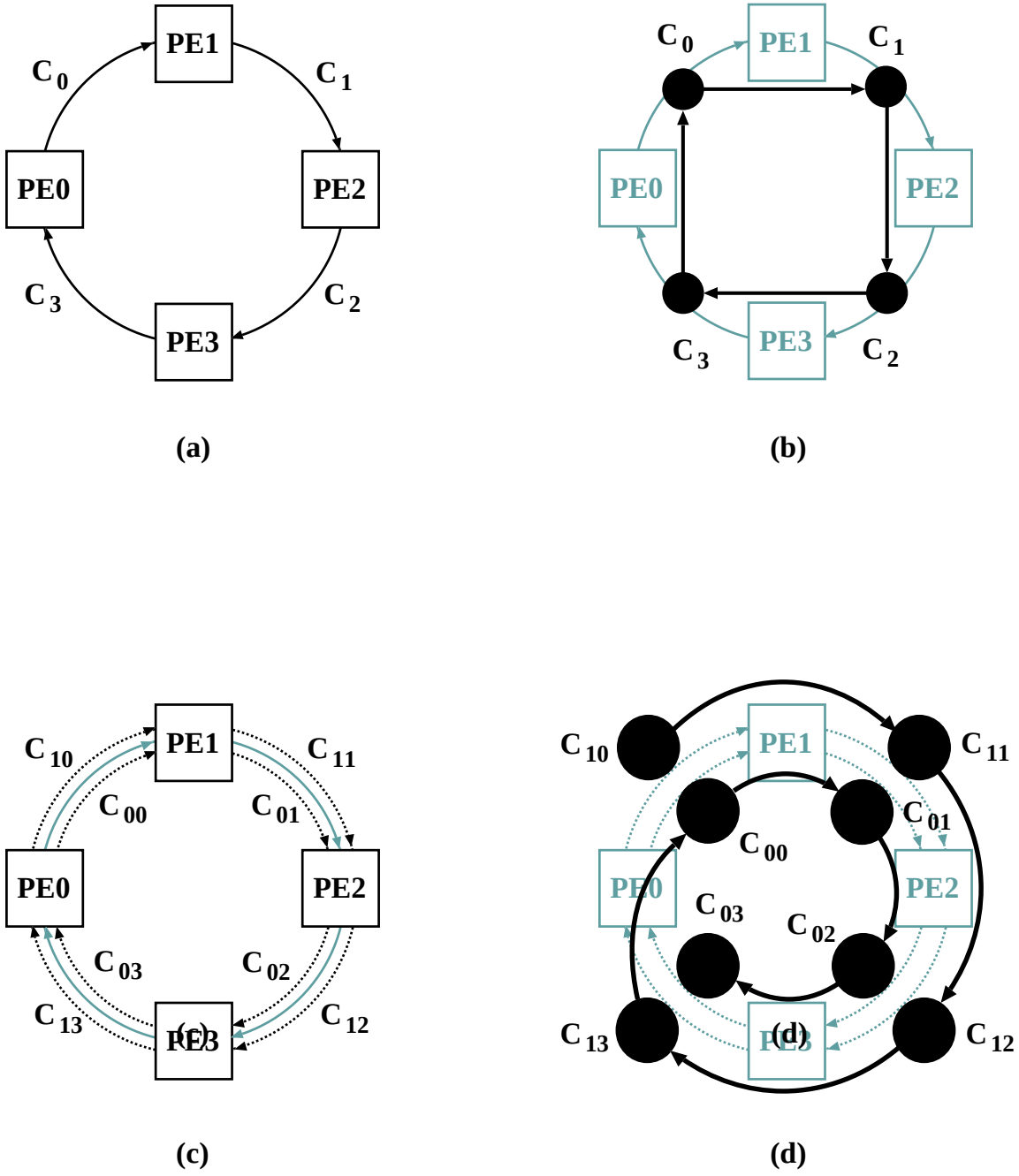


Figure 4.8: (a) A ring network with unidirectional channels. (b) The associated channel dependency graph contains a cycle. (c) Each physical channel is logically split into two virtual channels. (d) A modified channel dependency graph without cycles.

In Theorem 4.1 *if-and-only-if* relationship is used between the deadlock and the cycles in the channel dependency graph. The network becomes deadlock free by breaking the cycles. Splitting each physical channel along a cycle into multiple virtual channels and then restricting the routing so that the dependence between the virtual channels is acyclic.

Figure 4.8.(a) illustrates the phenomena of channel dependency graph and breaking the cycle by using virtual channels in a four-noded uni-directional ring network. The nodes are denoted by n_i, $i = \{0, 1, 2, 3\}$. A unidirectional channel connecting each pair of adjacent nodes. Let, c_i, $i = \{0, 1, 2, 3\}$ be the outgoing channel from node n_i. In this case, it is easy to define a routing function. It can be stated as follows: If a current node n_i is equal to the destination node n_j, store the packet. Otherwise, use c_i, $\forall j \neq i$. The channel dependency graph of Fig. 4.8.(a) is shown in Fig. 4.8.(b). There is a cyclic dependency between c_i channel. Effectively, a packet at node n_0 destined for n_2 can reserve c_0 and then request c_1. A packet at node n_1 destined for n_3 can reserve c_1 and then request c_2. A packet at node n_2 destined for n_0 can reserve c_2 and then request c_3. Finally, one packet at node n_3 destined for n_1 can reserve c_3 and then request c_0. A configuration containing the above-mentioned packets is deadlocked because every packet has reserved one channel and is waiting for a channel occupied by another packet. This deadlock configuration is illustrated in Fig. 4.8.(b) by channel dependency graph.

Now consider that every physical channel c_i is split into two virtual channels, c_{0i} and c_{1i}, as shown in Fig. 4.8.(c). c_{0i} is the first virtual channel of physical channel i. Similarly, c_{1i} is the second virtual channel of physical channel i. Now, the new routing function can be stated as follows: If the current node n_i is equal to the destination node n_j, store the packet. Otherwise, use c_{0i}, if $j < i$ or c_{1i}, if $j > i$. As can be seen, the cyclic dependency has been removed because after using channel c_{03}, node n_0 is reached. This phenomenon of breaking the cyclic dependency is illustrated in 4.8.(d).

There is no deadlock configuration after using 2 channels and restricted routing function. If there were a packet stored in the queue of channel c_{12}, it would be destined for n_3 and flits could advance. So c_{12} must be empty. Also, if there were a packet stored in the queue of channel c_{11}, it would be destined for n_2 or n_3. As c_{12} is empty, flits could also advance and c_{11} must be empty. If there were a packet stored in the queue of c_{10}, it would be destined for n_1, n_2, or n_3. As c_{11} and c_{12} are empty, flits could also advance and c_{10} must be empty. Similarly it can be shown that the remaining channels can be emptied.

4.3.1 Deadlock Configuration of Mesh and Torus Networks

In mesh interconnection networks, cyclic dependency can occur due to the inter-dimensional turns made by the messages [35]. All the possible turns a message can make are shown in Fig. 4.9. The deadlock situation in mesh network are prevented by proper routing algorithms.

In torus interconnection network, end to end nodes are connected by wrap-around connections. Due to this wrap around connection, besides inter-dimensional turns, cyclic dependency can also occur in the same dimension [35]. The possible deadlock configurations for a torus network are shown in Fig. 4.10. Additional virtual channel is required to break the wrap-around dependencies of the torus network.

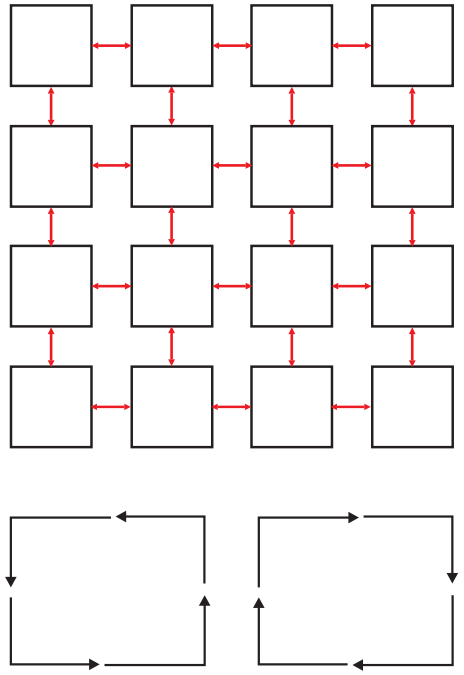


Figure 4.9: Deadlock configuration in mesh network

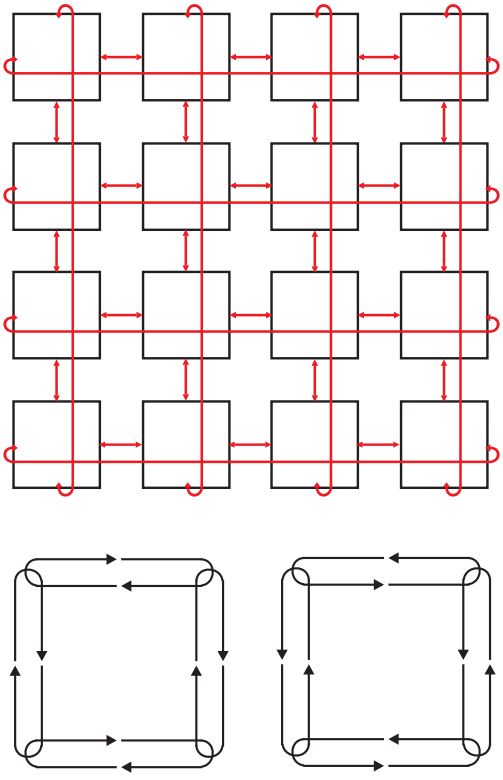


Figure 4.10: Deadlock configuration in torus network

4.4 Deadlock-Free Routing for HTN

4.4.1 Necessity of Deadlock-Free Routing

Interconnection network routing algorithms aim to minimize message blocking by efficiently utilizing network virtual channel and physical channel resources while ensuring deadlock freedom. Deadlock in an interconnection network is the situation in which some messages can not advance forever because of blocking by other messages. If a deadlock occurs, packet delivery is delayed indefinitely. In addition to this message delivery is also reduced. In short, once a deadlock has occurred, the dynamic performance is drastically reduced. A good wormhole routing algorithm must reduce network latency as much as possible without deadlock.

4.4.2 Dimension Order Routing

Dimension order routing algorithm is very popular and receives several names, like XY routing [37] (for 2D-mesh network) or *e-cube* [32] (for Hypercube network). In the dimension order routing, the header of the message contains the address of the destination relative to the current location. It is updated after each transmission. The routing mechanism is described as: first, determine whether the packet has reached its destination or not and second, if this is not the case, determine the lowest dimension in which the message still has to move and route the message in that dimension. The dimension order routing algorithm determines the only route for the source and destination pair. Figure 4.11 illustrates the mechanism of dimension order routing in a 2D-mesh network.

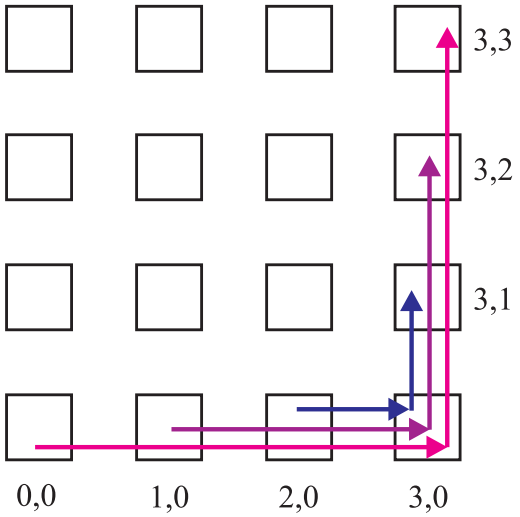


Figure 4.11: A set of routing paths created by the dimension order routing in a 2D-mesh network

Deadlocks are mainly avoided by using a proper routing algorithm within the network. One approach to designing a deadlock-free routing algorithm for a WH switched network

is to ensure that cycles are avoided in the channel dependency graph [32]. This can be achieved by assigning each channel a unique number and allocating channels to packet in strictly ascending or descending order. Routing is restricted to visiting the channel in order (ascending or descending) to eliminate cycles in the graph. If the routing restriction disconnects the network, physical channels are split into virtual channels to connect the network again.

In the dimension order routing, each packet is routed in one dimension at a time, arriving at the proper coordinate in each dimension before proceeding to the next dimension. By enforcing a strictly monotonic order on the dimension traversed, deadlock-free routing is guaranteed.

The network load and the availability of network resources do not influence the routing of a message. Thus, the dimension order routing algorithm is exceedingly simple to implement in hardware and provides low latency and high bandwidth. Additionally, switches can be decomposed into smaller and faster switches, thus increasing speed. It is well suited for uniform traffic distribution. The main disadvantages of deterministic routing is that it cannot respond to dynamic network condition.

4.4.3 Routing Algorithm for HTN

As mentioned earlier, in this thesis paper, we use wormhole routing for switching and popular deterministic, dimension order routing algorithm for routing messages.

We recall the routing algorithm stated in Section 3.2.3. The HTN performs routing of messages from the top level to the bottom level as in TESH [5, 36]. We divide the routing path of Level-L HTN into three phases, such as *phase-1*, *phase-2*, and *phase-3*. In phase-1 and phase-3, intra-BM communication is performed. Phase-1 is for source-BM and phase-3 is for destination-BM. In phase-2, inter-BM communication is performed. In phase-1, messages are transferred from source node to the outlet node of source-BM for higher level transfer. In phase-2, messages are transferred from the outlet node of source-BM to the inlet node of destination-BM, i.e., higher level communication is performed. Phase-2 is again divided into sub-phases, which is described in below. In phase-3, messages are transferred from inlet node of destination-BM to the destination of the message.

The above taxonomy of routing algorithm for the HTN is summarized in the following way:

- *Phase 1:* Intra-BM transfer path from source PE to the face of the BM.
- *Phase 2:* Higher level transfer path

  sub-phase $2.i.1$: Intra-BM transfer to the outlet PE of Level $(L-i)$ through the y-link.

  sub-phase $2.i.2$: Inter-BM transfer of Level $(L-i)$ through the y-link.

  sub-phase $2.i.3$: Intra-BM transfer to the outlet PE of Level $(L-i)$ through the x-link.

sub-phase $2.i.4$: Inter-BM transfer of Level $(L-i)$ through the x-link.
Here, $0 \leq i \leq (L-2)$.

- *Phase 3:* Intra-BM transfer path from the outlet of the inter-BM transfer path to the destination PE.

Routing of the HTN is strictly defined by the source node address and the destination node address. Let a source node address be $s_n, s_{n-1}, s_{n-2}, ..., s_1, s_0$, a destination node address be $d_n, d_{n-1}, d_{n-2}, ..., d_1, d_0$, and a routing tag be $t_n, t_{n-1}, t_{n-2}, ..., t_1, t_0$, where, $t_i = d_i - s_i$. The source node address of HTN is expressed as:

$$\begin{aligned} s &= s_{2L}, s_{2L-1}, s_{2L-2}, ..., s_2, s_1, s_0 \\ &= (s_{2L}s_{2L-1}), ..., (s_2, s_1, s_0) \end{aligned} \tag{4.4}$$

Similarly, the destination node address is expressed as:

$$\begin{aligned} d &= d_{2L}, d_{2L-1}, d_{2L-2}, ..., d_2, d_1, d_0 \\ &= (d_{2L}d_{2L-1}), ..., (d_2, d_1, d_0) \end{aligned} \tag{4.5}$$

Hence, $n = 2L$, where, L is the level number and n is the position of the node. Figure 4.12 shows the routing algorithm for the HTN.

As an example, consider the routing between $PE_{(0,0)(3,0,0)}$ and $PE_{(3,2)(2,3,0)}$. At first, in phase-1 routing, the packets move to the outlet node $PE_{(0,0)(0,0,0)}$ of the source-BM. Next, in phase-2 routing, the packets move to the node whose address in the y-axis is the same. The packets are transferred to node $PE_{(3,0)(0,3,0)}$. Then, in x-axis routing of phase-2, the packets are transferred from $PE_{(3,0)(0,3,0)}$ to $PE_{(3,2)(0,3,0)}$. Finally, in phase-3 routing, the routing is performed in the destination-BM and the packets are moved to the destination $PE_{(3,2)(2,3,0)}$. The complete route is $PE_{(0,0)(3,0,0)} \rightarrow PE_{(0,0)(0,0,0)} \rightarrow PE_{(3,0)(0,3,0)} \rightarrow PE_{(3,0)(0,3,3)} \rightarrow PE_{(3,1)(0,3,0)} \rightarrow PE_{(3,1)(0,3,3)} \rightarrow PE_{(3,2)(0,3,0)} \rightarrow PE_{(3,2)(1,3,0)} \rightarrow PE_{(3,2)(2,3,0)}$. The above mentioned scenario is illustrated in Fig. 4.13.

4.4.4 Deadlock-free Routing

A deadlock-free routing algorithm can be constructed for an arbitrary interconnection networks by introducing virtual channels. In this section, we investigate the number of virtual channels required to make the routing algorithm deadlock-free for the HTN. We also present a proof that the HTN is deadlock-free by these virtual channels. The proposed routing algorithm enforce some routing restrictions to avoid deadlocks [32][33]

As dimension order routing is used in HTN, routing at the higher level is performed firstly in the y-direction and then in the x-direction. In a basic module (BM), the routing order is initially in the z-direction, then in the y-direction, and finally in the x-direction.

The interconnection of the BM and the higher level network of HTN is toroidal connection. By using the following lemma and corollary, the number of virtual channels required to make deadlock free routing of HTN is evaluated.

Routing Algorithm for HTN

Routing HTN(s,d);
source node address:$s_n, s_{n-1}, s_{n-2}, ..., s_1, s_0$
destination node address: $d_n, d_{n-1}, d_{n-2}, ..., d_1, d_0$
tag: $t_n, t_{n-1}, t_{n-2}, ..., t_1, t_0$
  for $i = n : 3$
    if ($i/2 = 0$ and ($t_i > 0$ or $t_i = -3$)), routedir = North; endif;
    if ($i/2 = 0$ and ($t_i < 0$ or $t_i = 3$)), routedir = South; endif;
    if ($i\%2 = 1$ and ($t_i > 0$ or $t_i = -3$)), routedir = East; endif;
    if ($i\%2 = 1$ and ($t_i < 0$ or $t_i = 3$)), routedir = West; endif;
      while ($t_i \neq 0$) do
        $N_z = outlet_z(s, d, L,$ routedir)
        $N_y = outlet_y(s, d, L,$ routedir)
        $N_x = outlet_x(s, d, L,$ routedir)
        BM_Routing(N_z, N_y, N_x)
        if (routedir = North or East), move packet to next BM; endif;
        if (routedir = South or West), move packet to previous BM; endif;
        $t_i = t_i - 1$;
      endwhile;
  endfor;
    BM_Routing(t_z, t_y, t_x)
end
BM_Routing (t_2, t_1, t_0);
BM_tag t_2, t_1, t_0 = receiving node address (r_2, r_1, r_0) − destination (d_2, d_1, d_0)
  for $i = 2 : 0$
    if ($t_i > 0$ and $t_i \leq 2$) or ($t_i < 0$ and $t_i = -3$), movedir = positive; endif;
    if ($t_i > 0$ and $t_i = 3$) or ($t_i < 0$ and $t_i \geq -2$), movedir = negetive; endif;
    if (movedir = positive and $t_i > 0$), distance $= t_i$; endif;
    if (movedir = positive and $t_i < 0$), distance $= 4 + t_i$; endif;
    if (movedir = negative and $t_i < 0$), distance $= t_i$; endif;
    if (movedir = negative and $t_i > 0$), distance $= -4 + t_i$; endif;
 endfor
  while($t_2 \neq 0$ or distance$_2 \neq 0$) do
    if (movedir = positive), move packet to $+z$ node; distance$_2$ = distance$_2 - 1$; endif;
    if (movedir = negetive), move packet to $-z$ node; distance$_2$ = distance$_2 + 1$; endif;
  endwhile;
  while($t_1 \neq 0$ or distance$_1 \neq 0$) do
    if (movedir = positive), move packet to $+y$ node; distance$_1$ = distance$_1 - 1$; endif;
    if (movedir = negetive), move packet to $-y$ node; distance$_1$ = distance$_1 + 1$; endif;
  endwhile;
  while($t_0 \neq 0$ or distance$_0 \neq 0$) do
    if (movedir = positive), move packet to $+x$ node; distance$_0$ = distance$_0 - 1$; endif;
    if (movedir = negetive), move packet to $-x$ node; distance$_0$ = distance$_0 + 1$; endif;
  endwhile;
end

Figure 4.12: Routing algorithm of HTN

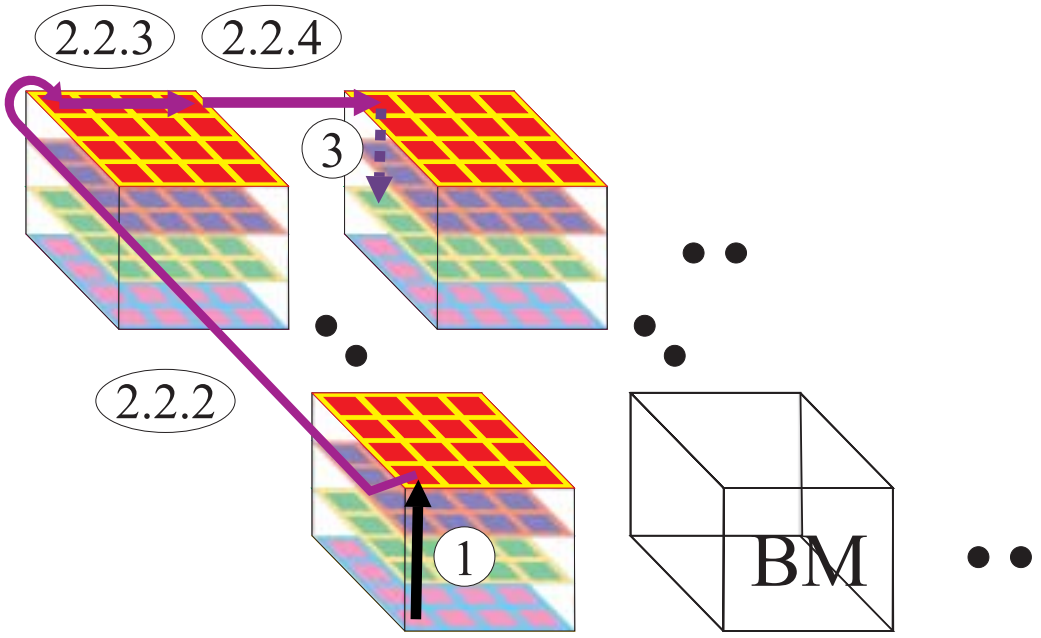


Figure 4.13: An example of message routing in HTN

Lemma 4.1 *If a message is routed in the order $z \rightarrow y \rightarrow x$ in a 3D-torus network, then the network is deadlock-free with 2 virtual channels.*

Proof: In torus interconnection networks, cyclic dependencies can occur in two ways. Firstly, due to the inter-dimensional turns made by the messages. Secondly, due to wrap-around connection in the same direction. In order to avoid these cyclic dependencies, we need two virtual channels, one for inter-dimensional turns and another for wrap-around connections. Initially, messages are routed over virtual channel 0. Then, messages are routed over virtual channel 1 if the packet is going to use a wrap-around channel. The channels are allocated as shown in Eq. 4.6 for a 3D-torus network. Enforcing this routing restriction and use of virtual channels avoid cyclic dependencies. Thus, deadlock freeness is proved.

$$C = \begin{cases} (l, vc, n_2), & z+ \text{ channel}, \\ (l, vc, 4 - n_2), & z- \text{ channel}, \\ (l, vc, n_1), & y+ \text{ channel}, \\ (l, vc, 4 - n_1), & y- \text{ channel}, \\ (l, vc, n_0), & x+ \text{ channel}, \\ (l, vc, 4 - n_0), & x- \text{ channel} \end{cases} \tag{4.6}$$

Here, $l = \{0, 1, 2, 3, 4, 5\}$ are the links used in the BM, $l = \{0, 1\}$,$l = \{2, 3\}$, and $l = \{4, 5\}$ are the used in the z–direction, y–direction, and x–direction, respectively. $vc = \{0, 1\}$ are virtual channels, and n_0, n_1, and n_2 are the PE addresses in the BM.

Corollary 4.1 *If the message is routed in the $y \rightarrow x$ direction in a 2D-torus network, then the network is deadlock-free with 2 virtual channels.*

Proof: If the channels are allocated as shown in Eq. 4.7 for a 2D-torus network then the deadlock freeness is proved.

$$C = \begin{cases} (l, vc, n_{2L}), & y+ \text{ channel}, \\ (l, vc, 4 - n_{2L}), & y- \text{ channel}, \\ (l, vc, n_{2L-1}), & x+ \text{ channel}, \\ (l, vc, 4 - n_{2L-1}), & x- \text{ channel} \end{cases} \tag{4.7}$$

Here, $l = \{6, 7\}$ are the links used for higher-level interconnection, $vc = \{0, 1\}$ are virtual channels, and n_{2L} and n_{2L-1} are the PE addresses in the higher level where L is the level number. Links-6 is used in the interconnection of East and West direction, and Links-7 is used in the interconnection of North and South direction.

Theorem 4.2 *A Hierarchical Torus Network (HTN) with 6 virtual channels is deadlock-free.*

Proof: Both the BM and the higher levels of the HTN have a toroidal interconnection. In phase-1 and phase-3 routing, packets are routed in the source-BM and destination-BM, respectively. The BM of the HTN is a 3D-torus network. According to Lemma 4.1, the number of necessary virtual channels for each of phase-1 and phase-3 is 2. Intra-BM links between inter-BM links on the xy-plane of the BM are used in sub-phase 2.i.1 and sub-phase 2.i.3. Sub-phase 2.i.1 and sub-phase 2.i.3 utilizes channels over intra-BM links. They share either the channels of phase-1 or phase-3. Nodes at the contours of xy-plane are assigned to each high level as gate nodes. Links at the edge of the BM are used in sub-phase 2.i.2 and sub-phase 2.i.4, and these links form a 2D-torus network. This 2D-torus network is the higher-level interconnection of the HTN. According to Corollary 4.1, the number of necessary virtual channels for this 2D-torus network is 2.

Therefore, the total number of necessary virtual channels for the whole network is 6.

4.5 Minimum Number of Virtual Channels

Virtual channel is used to solve the problems of deadlock in wormhole-routed interconnection networks. The use of virtual channels requires a careful analyses in order to maximize the benefits and minimize the drawbacks. Dally [38] has been pointed out that the larger the number of virtual channels, the higher the performance achieved. Aoyama and Chien [41] compare the cost of adding virtual channels for wormhole switched networks. They have been shown that a 30% penalty in router speed per extra virtual channel.

Multiplexing large number of virtual channels on a physical channel reduces the bandwidth of individual virtual channel. As the consequence, it will reduce the data rate of individual messages and increase the message latency. Crossbar controller speed which is determined by the routinhg algorithm is also reduced for additional virtual channels.

Increasing the number of virtual channel makes the link controllers more complex since they must support the arbitration between multiple virtual channel for physical channel. Again, the hardware cost is increasing along with the increase of number of virtual channels. A good wormhole routing algorithm must reduce network latency as much as possible without excessive hardware requirement. Thus, the deadlock free routing algorithm with minimum number of virtual channels is needed. In this section, we have investigated the minimum number of virtual channels for deadlock-free routing of the HTN.

Theorem 4.3 *A Hierarchical Torus Network (HTN) with* 2 *virtual channels is deadlock-free.*

Proof: Both the BM and the higher levels of the HTN have a toroidal interconnection. In phase-1 and phase-3 routing, packets are routed in the source-BM and destination-BM, respectively. The BM of the HTN is a 3D-torus network. According to Lemma 4.1, the number of necessary virtual channels for phase-1 and phase-3 is 2. Intra-BM links between inter-BM links on the xy-plane of the BM are used in sub-phase $2.i.1$ and sub-phase $2.i.3$. Sub-phase $2.i.1$ and sub-phase $2.i.3$ utilizes channels over intra-BM links. They share either the channels of phase-1 or phase-3. PEs at the contours of xy-plane are assigned to each high level as gate nodes. Links at the edge of the BM are used in sub-phase $2.i.2$ and sub-phase $2.i.4$, and these links form a 2D-torus network. This 2D-torus network is the higher-level interconnection of the HTN. According to Corollary 4.1, the number of necessary virtual channels for this 2D-torus network is also 2. The main idea is that messages are routed over one virtual channel. Then, messages are switched over the other virtual channel if the packet is going to use a wrap-around connection.

Therefore, the total number of necessary virtual channels for the whole network is 2.

4.6 Conclusions

This chapter prsented the concept of deadlock in interconnection network and the requirements for deadlock free routing algorithms. In addition, we examined the deadlock-free routing algorithm for HTN by using virtual channels. Wormhole routing is used for switching, because it requires the small number of buffers and can control data flow as pipelined fashion reducing communication overhead. Dimension order routing for message passing because of its simplicity. Investigation of number of virtual channels required for deadlock free routing of HTN is also studied in this Chapter. At first, it is shown that 6 virtual channels per physical channel guarantees the deadlock-free routing algorithm for HTN. However, the hardware cost is high for this large number of virtual channels. In order to reduce the hardware cost, we have investigated the minimum number of virtual channel for deadlock free routing of HTN. It has been proved that 2 virtual channels per physical channel are sufficient for the deadlock-free routing algorithm of the HTN and this is the minimum value.

Chapter 5

"There are three principal means of acquiring knowledge... observation of nature, reflection, and experimentation. Observation collects facts; Reflection combines them; Experimentation verifies the result of that combination."
– Denis Diderot

Performance of the HTN

5.1 Introduction

The design of an interconnection network for a multicomputer inevitably involves tradeoffs between performance and cost. The goal in the design of an interconnection network is to achieve the highest performance at a given cost, or to minimize the cost subject to given performance constraints. Thus, there is a requirement to estimate the performance of the network before it is actually constructed or integrated into the system. Performance evaluation of an interconnection networks includes: the hardware cost, the message delay, the network throughput, the potential for fault tolerance, the embedding capability, and the partitioning capability.

In an interconnection network, the hardware cost is expressed in terms of number of links and node degree. The message delay is expressed theoretically by the network diameter and the average distance. Diameter is the upper bound of message delay. In practice, the message delay is expressed by the average latency of messages between the time of their departure from their source and the time of their arrival at their destination. The network throughput is measured by the average number of delivered messages per unit of time. The fault tolerance of a network is the number of elements that can fail without the network becoming disconnected. In other words, a fault tolerant network has the ability to route information even if certain network components (processors, switches, links) fail. The embedding capability is the ability of a network to efficiently emulate other topological structures. The partitioning capability is the ability of a network to subdivide into subgraphs of identical topological structures.

The details of hardware cost is addressed in Chapter 3. This Chapter deals with the performance issue of Hierarchical Torus Network (HTN). Dynamic communication performance (message delay and network throughput), system yield by providing redundancy, and suitability of some advanced applications are studied in this Chapter. The embedding

issue and partitioning issue are beyond the scope of this paper. These two issues are kept in mind for future research.

The outline of this chapter is as follows: Section 5.2 discusses the dynamic communication performance of HTN as well as its applicability to several commonly used networks for parallel computers. The critical issues of fault tolerance and yield are examined in Section 5.3. Next, mapping of of some advanced applications such as bitonic merge algorithm, Fast Fourier Transform (FFT) and finding the maximum to the HTN is considered briefly in Section 5.4. Finally, some concluding remarks of this chapter are given in Section 5.5.

5.2 Dynamic Communication Performance

The overall performance of a multicomputer system is affected by the performance of the interconnection network as well as the performance of the node. Continuing advances in VLSI/WSI technology promise to deliver more power to the individual node processor. On the other hand, the low performance of the communication network severely limits the speed of entire multicomputer system [39]. The principal performance measure of multicomputer communication network is message latency and network throughput. For the network to have good performance, low latency and high throughput must be achieved.

5.2.1 Performance Metrics

The performance of an interconnection network is characterized by two parameters – message latency and network throughput. Message latency refers to the time elapsed by messages as they traverse from source to destination, from the instant when the first flit is injected to the network at the source till when the last flit of the message is received at the destination. Network throughput, on the other hand, refers to the number of messages delivered per unit of time through the network.

The general definition of latency is vague, and can be interpreted in different ways. If the study only considers the network hardware, latency is usually defined as the time elapsed since the message header is injected into the network at the source node until the last unit of information is received at the destination node. If the study also considers the injection queues, the queuing at the source node is added to the latency. This queuing time is usually negligible unless the network is close to its saturation point. When the messaging layer is also being considered, latency is defined as the time elapsed since the system call to send a message is initiated at the source node until the system call to receive that message returns control to the user program at the destination node. Unless otherwise stated, the message latency presented in this chapter, is the average value of the time elapsed since the message header is injected into the network at the source node until the last unit of the data flit is received at the destination node. Latency is measured in time units. However, when comparing several design choices, the absolute value is not important. As the comparison is performed by computer simulation, latency is measured in simulator clock cycles.

Throughput is the maximum amount of information delivered per unit of time. It can also be defined as the maximum traffic accepted by the network, where traffic is the amount of information delivered per unit of time. Throughput could be measured in messages per second or messages per clock cycle, depending on whether absolute or relative timing is used. However, throughput would depend on message and network size. So, throughput is usually normalized, dividing it by message size and network size. When the throughput is compared by computer simulation, and wormhole routing is used for switching, throughput can be measured in flits per node and clock cycle.

In wormhole routing, packets are buffered in a distributed manner. It use small buffer. In this system with finite buffers, packets may be discarded because of lack of buffer-space. Thus, the network becomes unstable beyond certain sustainable throughput, resulting in a steep rise in the message latency, and the throughput might even decrease in this region with a further increase of messages in the network.

5.2.2 Simulation Parameter

The simulation parameters affecting simulation results are as follows:

1. **Network Topology**
   Including our proposed HTN, we have been considered TESH, H3D-mesh, H3D-torus, 2D-mesh, and 2D-torus networks for the purpose of comparison.

2. **Network Size**
   We have used 256, 512, and 1024 noded networks for simulation.

3. **Switching**
   We use wormhole routing algorithm for switching technique. This is an emerging switching technique for current generation multicomputers.

4. **Routing Strategy**
   We have chosen dimension order routing algorithm because it is representative of routing strategy for existing multicomputers. The dimension order routing algorithm provides the only route for the source and destination pair.

5. **Number of Virtual Channels**
   2 virtual channels per physical channel is simulated, which is the minimum number of virtual channels required to implement the deadlock-free routing algorithm for HTN. We have also used 3, 4, and 6 virtual channels.

6. **Arbitration of Virtual Channels**
   The virtual channels are arbitrated by round-robin algorithm.

7. **Message Length**
   Three messages are used for simulation, such as short message (16 Flits), medium message (64 Flits), and long message (256 Flits). 2 flits are used as a header flit.

8. **Traffic Pattern**
   Network performance is largely affected by the message traffic pattern. The message traffic pattern is used in our simulation is uniform traffic pattern. In this pattern, every nodes send messages every other nodes with equal probability. That is, source and destination are randomly selected.

9. **Simulation Time**
   Simulation time is 20000 cycles. In each clock cycle, one flit is transferred from input buffer to output buffer or from output buffer to input buffer. Thus, for transferring data between two nodes it takes two clock cycles.

5.2.3 Experimental Result

In this section, we have shown some simulation results of algorithm proposed in Chapter 4. Figure 5.1 shows a comparison of dynamic communication performance of HTN with H3D-mesh network and mesh network under uniform-random traffic pattern. The inter-level connectivity is 0. We have used as much virtual channels as possible to make the corresponding network deadlock free. We have emphasized on deadlock free routing only. The use of virtual channel is not uniform for different networks. We have used 1, 5, and 6 virtual channels for mesh network, H3D-mesh network, and HTN, respectively. Thus, a fair comparison of the communication performance of these families of interconnection networks is not shown in Fig. 5.1.

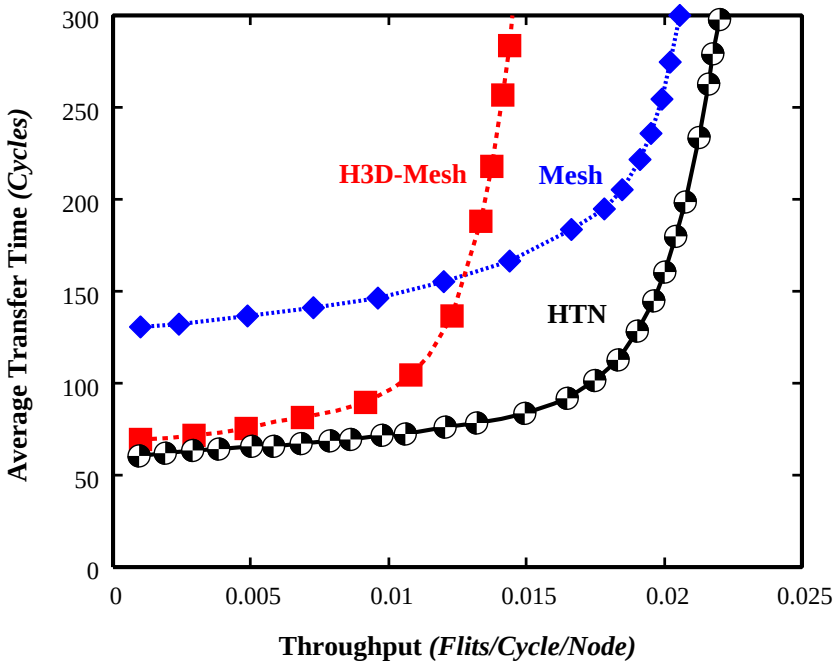


Figure 5.1: Dynamic communication performance of different networks with dimension order routing algorithm for short message: 1024 nodes, different virtual channels.

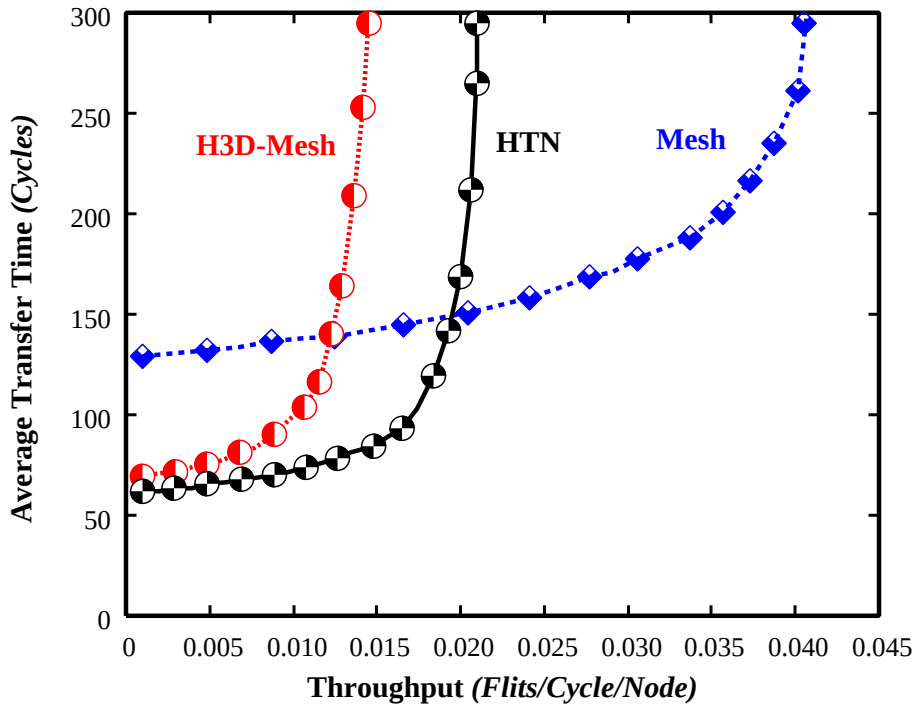


Figure 5.2: Dynamic communication performance of different networks with dimension order routing algorithm for short message: 1024 nodes, 3 virtual channels.

We have tried to reduce the number of virtual channels for deadlock free routing of HTN and H3D-mesh network. In fact, we have made a fair comparison of dynamic communication performance between different networks. Figure 5.2, shows a comparison of dynamic communication performance between HTN, H3D-mesh network, and mesh network with 3 virtual channels.

Figure 5.2 shows that the communication performance of HTN is better than H3D-mesh network but the maximum throughput of HTN is less than mesh network. To improve the throughput of HTN, inter-level connectivity (q) of HTN is increased from 0 to 1. With this condition, we have evaluated the dynamic communication performance of HTN, H3D-mesh and mesh networks and presented in Fig. 5.3 and 5.4. It is shown that the dynamic communication performance of HTN with 3 virtual channels is better than H3D-mesh and mesh networks. Under uniform traffic pattern, the message latency of HTN is lower and the network throughput is higher than H3D-mesh and mesh networks.

The hardware cost of an interconnection network is increased along with the increase of number of virtual channels. That's why, we have investigated the minimum number of virtual channels for deadlock free routing of HTN. 2 virtual channel per physical channel is the optimal number of virtual channels required to implement the deadlock free routing of HTN. We have simulated the dynamic communication performance of different networks with 2 virtual channels.

In Fig. 5.5, we can see that the communication performance of HTN with 2 virtual

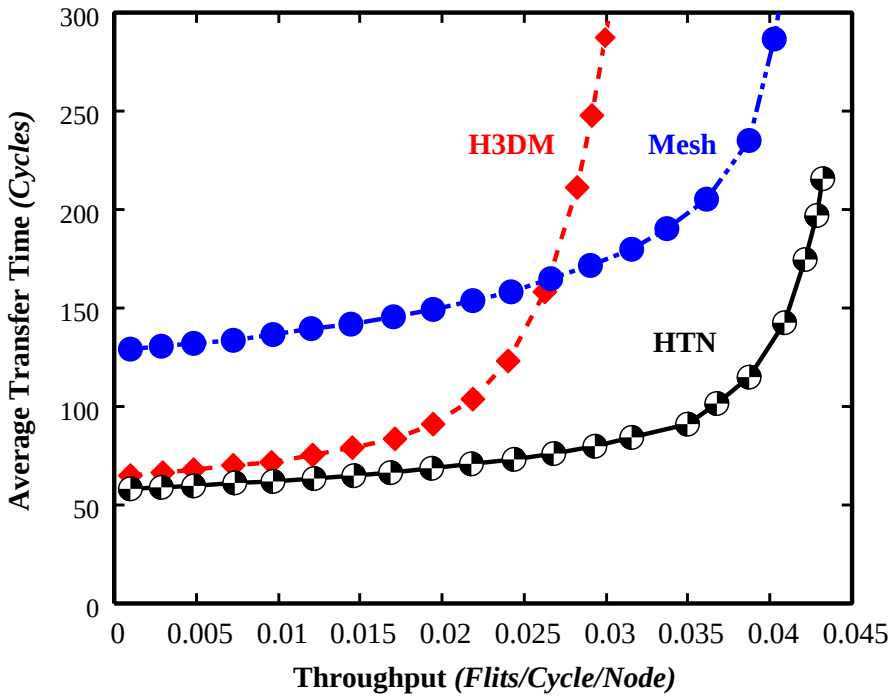


Figure 5.3: Dynamic communication performance of different networks with dimension order routing algorithm for short message: 1024 nodes, 3 virtual channels.

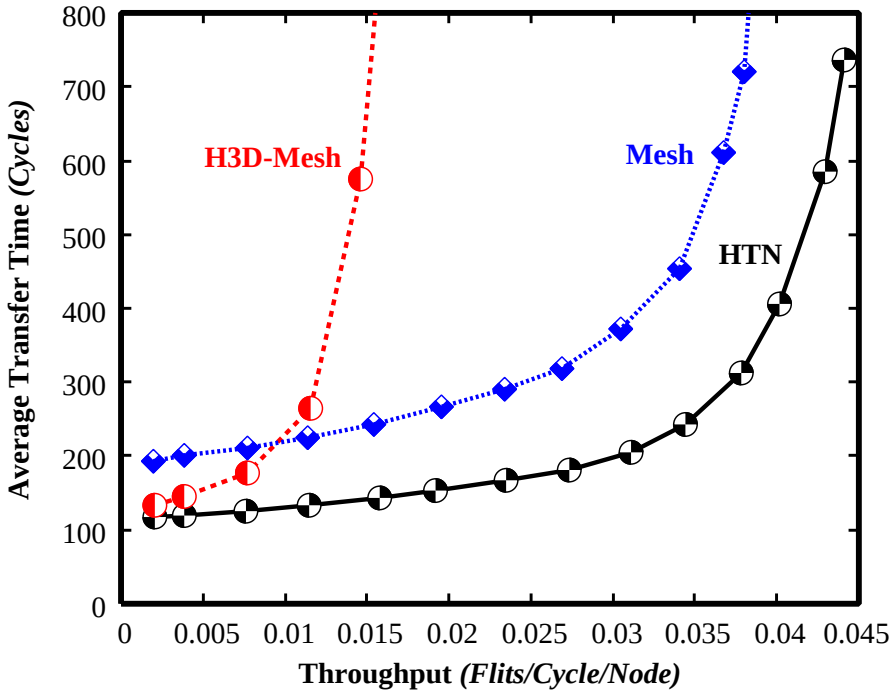


Figure 5.4: Dynamic communication performance of different networks with dimension order routing algorithm for medium message: 1024 nodes, 3 virtual channels.

channels is better than H3D-mesh, mesh and torus networks. Under uniform traffic pattern, the message latency of HTN is slightly lower than H3D-mesh network but it is far lower than mesh and torus networks. The maximum throughput of HTN is higher than H3D-mesh, mesh, and torus networks. Figure 5.6 and 5.7 show the communication performance for medium and long messages. From Fig. 5.5, 5.6, and 5.7, it has been seen that the dynamic communication performance of HTN is significantly better than H3D-mesh, mesh and torus networks.

A comparison of the communication performance between different hierarchical interconnection networks such as HTN, TESH, H3D-mesh, and H3D-torus is not an easy task because each networks have different architecture for interconnection. That's why, it is hardly match the total number of nodes of different networks. According to Lemma 3.3, the total number of nodes in the HTN is $N = \left[m^3 \times n^{2(L-1)}\right]$. If $m = 4$, $n = 2$, and $L = 2$, then the total number of nodes is 256. Level-2 TESH, Level-2 H3DM, 16×16 mesh, and 16×16 torus networks have 256 nodes.

It has already been shown that the communication performance of HTN is better than mesh and torus networks and it is also better than hierarchical interconnection network H3D-mesh network. In Fig. 5.8 and 5.9, we can see that the message latency of HTN is remarkably lower than TESH network. The maximum throughput of HTN is far higher than TESH network. Figure 5.9 is a little modification of Fig. 5.8, which contains the graph of different hierarchical networks. We have also evaluated the dynamic communication performance of these hierarchical interconnection networks for short message and 3 virtual channels. The result is plotted in Fig. 5.10. Simulation is also carried out for medium and long messages. These results are also plotted in Fig. 5.11 and 5.12.

Figure 5.9, 5.10, 5.11, and 5.12 compare the communication performance of different hierarchical interconnection networks for short, medium, and long messages, respectively. It is shown in these figures that the communication performance of HTN is significantly better than H3D-mesh and TESH networks. Here it is also mentioned that the inter-level connectivity of different hierarchical interconnection networks is 1 ($q = 1$).

As mentioned earlier that it is hardly match the number of nodes of different networks. We have considered a rectangular HTN as shown in Fig. 5.13.(a) to match the total number of nodes with H3D-torus network. The structure of the H3D-torus is also shown in Fig. 5.13.(b). We have evaluated the dynamic communication performance of HTN and H3D-torus network of 512 nodes. The comparison of communication performance between HTN and H3D-torus network is shown in Fig. 5.14. Here the inter-level connectivity is 0 and 3 virtual channels are used for simulation. From the graph, it is seen that the average transfer time of HTN is lower than H3D-torus network and the maximum throughput of HTN is far higher than H3D-torus network. Thus, the dynamic communication performance of HTN is far better than H3D-torus network.

Figure 5.15 shows the throughput as a function of probability of packet generation rate. As shown in figure 5.15, throughput of the proposed routing algorithm on HTN declines later than H3D-mesh, mesh, and torus networks, because it saturates later than those networks. Therefore, the throughput of HTN is better than H3D-mesh, mesh, and torus networks.

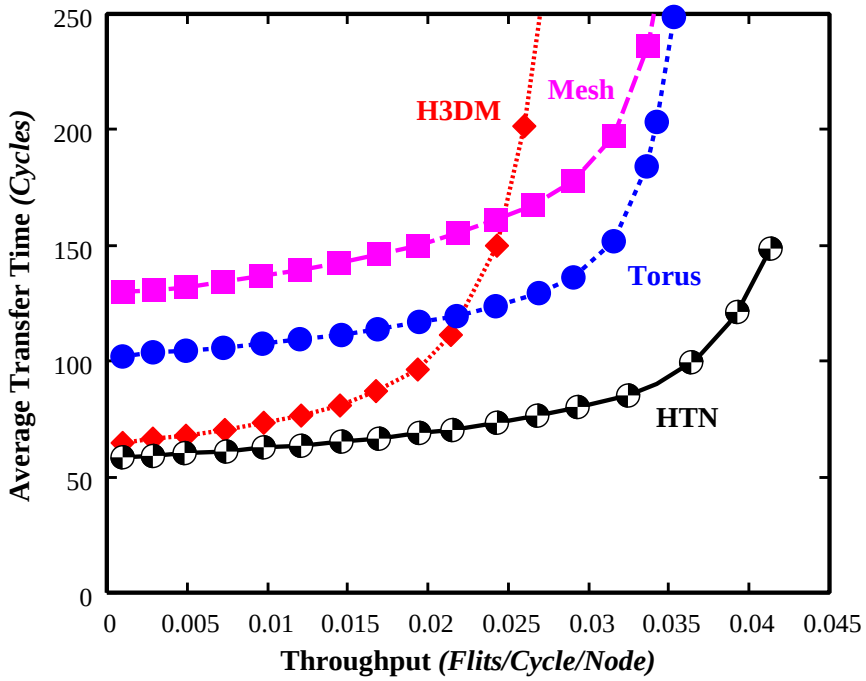


Figure 5.5: Dynamic communication performance of different networks with dimension order routing algorithm for short message: 1024 nodes, 2 virtual channels.

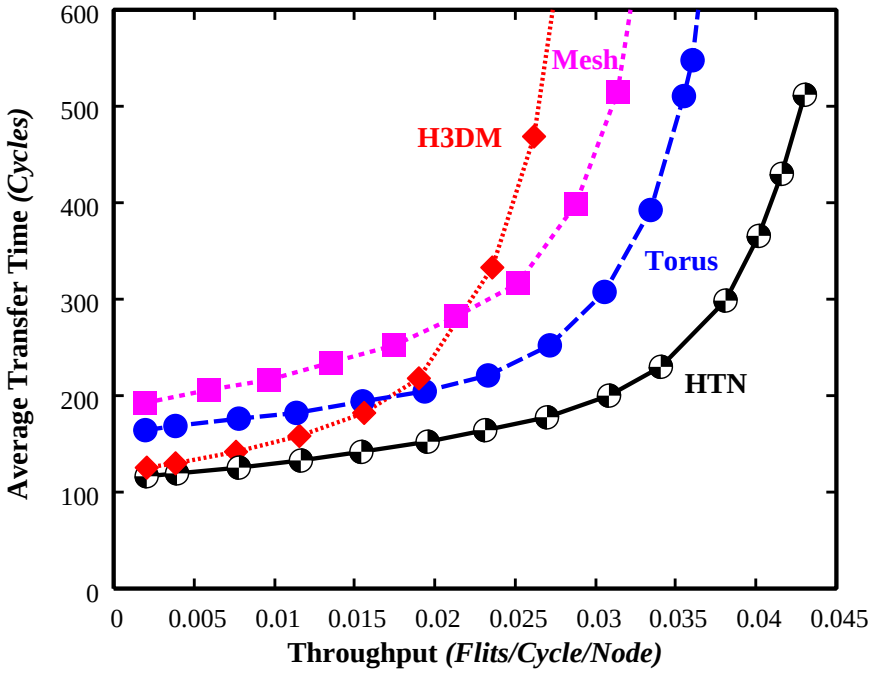


Figure 5.6: Dynamic communication performance of different networks with dimension order routing algorithm for medium message: 1024 nodes, 2 virtual channels.

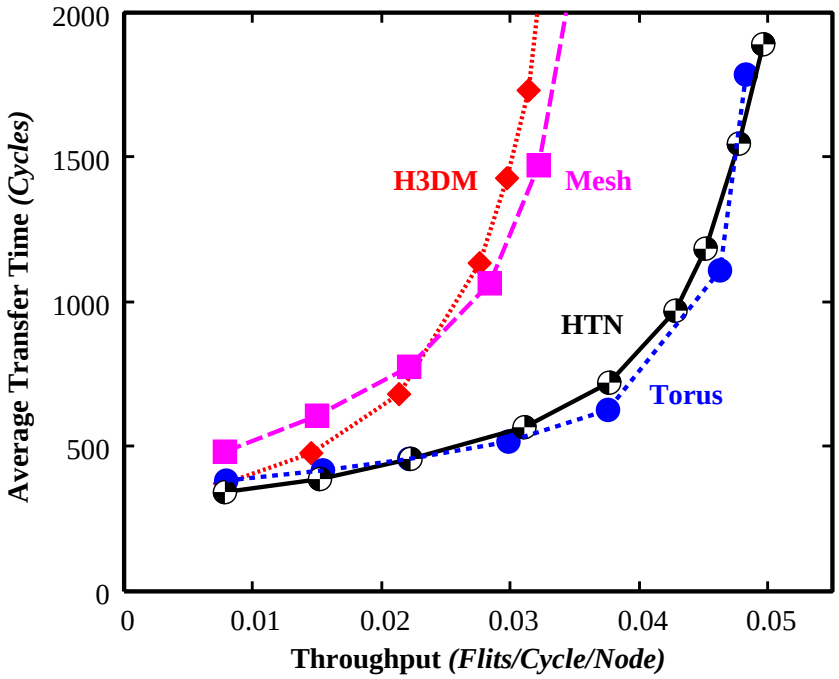


Figure 5.7: Dynamic communication performance of different networks with dimension order routing algorithm for long message: 1024 nodes, 2 virtual channels.

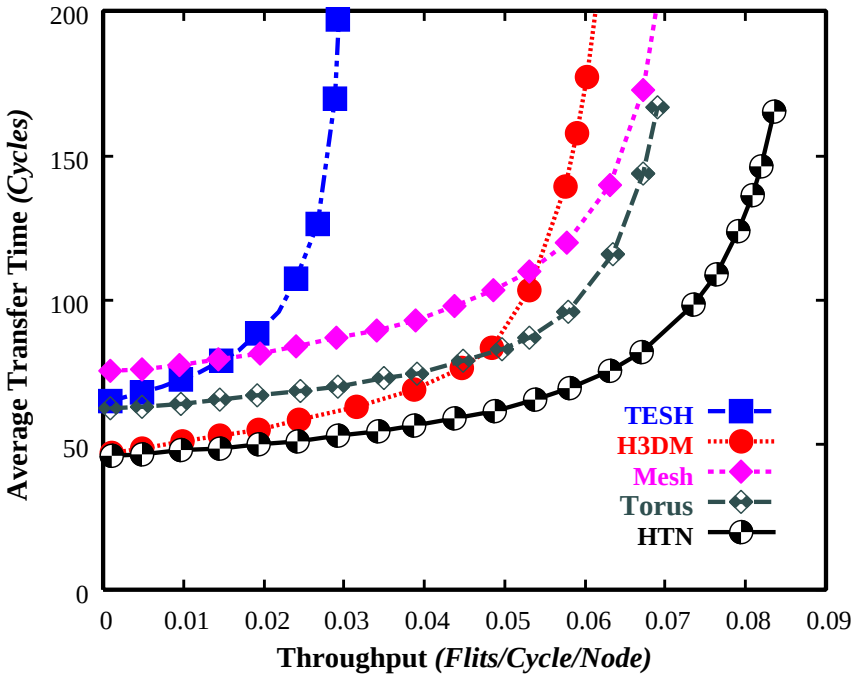


Figure 5.8: Dynamic communication performance of different networks with dimension order routing algorithm for short message: 256 nodes, 2 virtual channels.

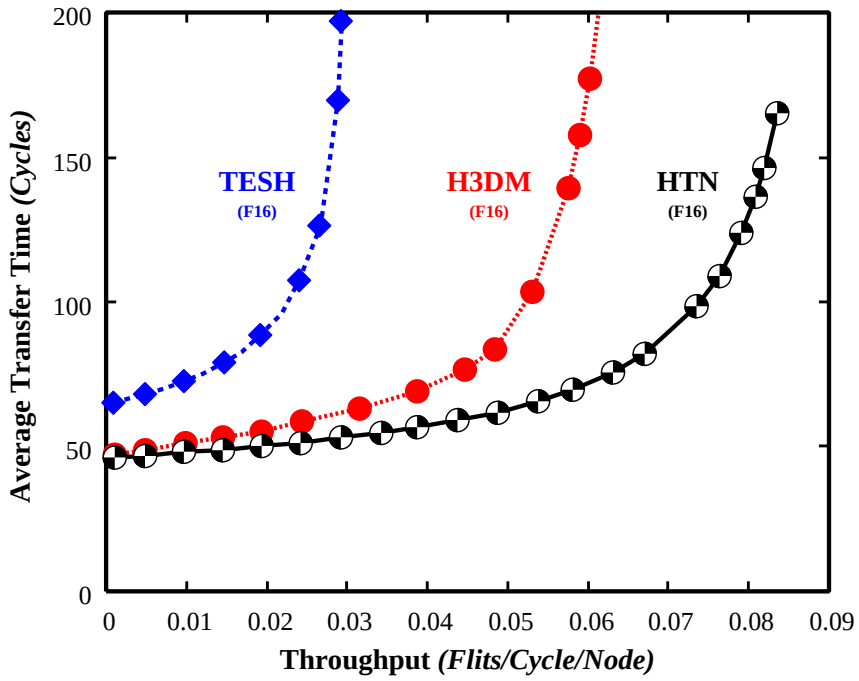


Figure 5.9: Dynamic communication performance of different hierarchical network with dimension order routing algorithm for short message: 256 nodes, 2 virtual channels.

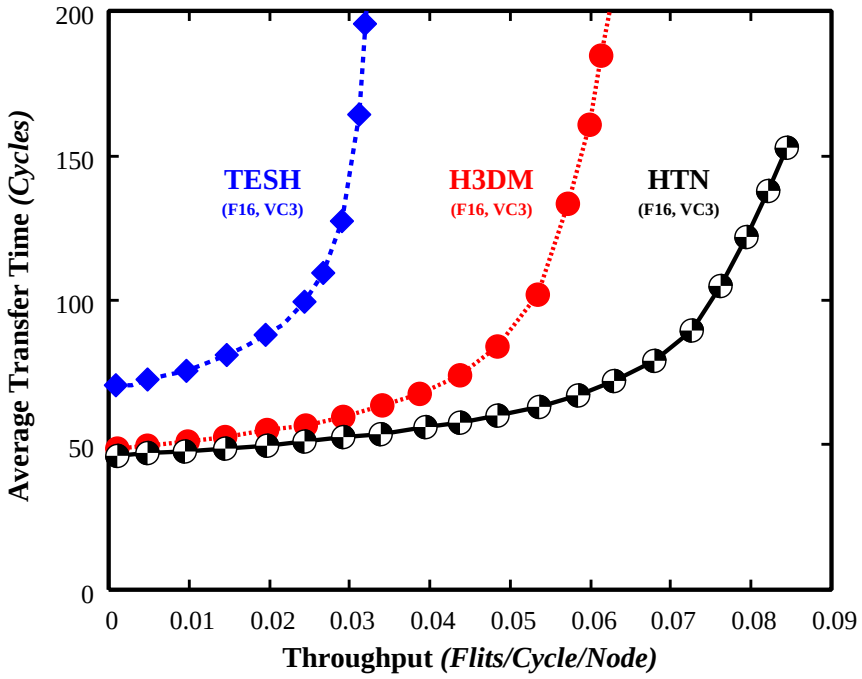


Figure 5.10: Dynamic communication performance of different hierarchical network with dimension order routing algorithm for short message: 256 nodes, 3 virtual channels.

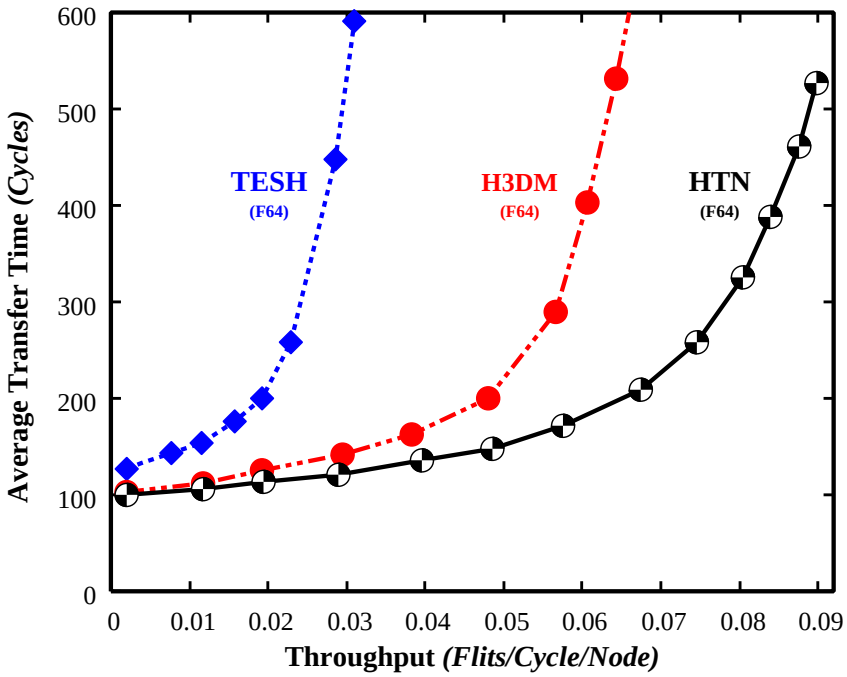


Figure 5.11: Dynamic communication performance of different hierarchical network with dimension order routing algorithm for medium message: 256 nodes, 2 virtual channels.

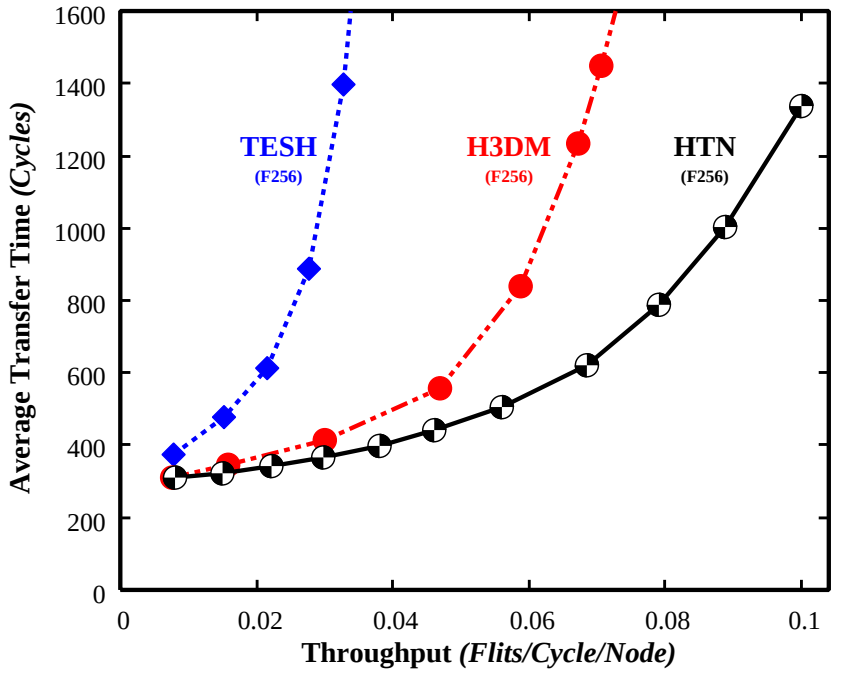


Figure 5.12: Dynamic communication performance of different hierarchical network with dimension order routing algorithm for long message: 256 nodes, 2 virtual channels.

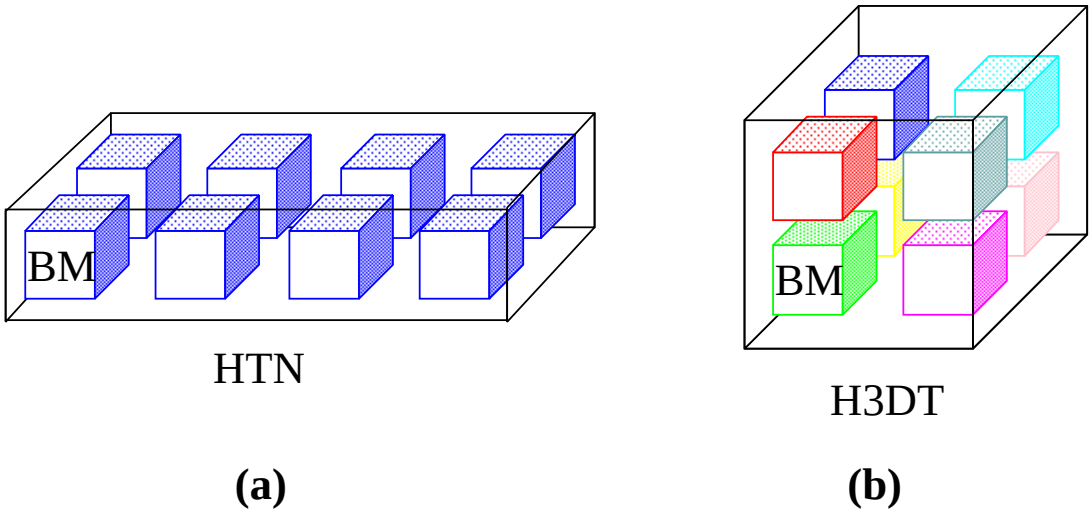


Figure 5.13: (a) An HTN with 512 nodes. (b) An H3D-torus network with 512 nodes.

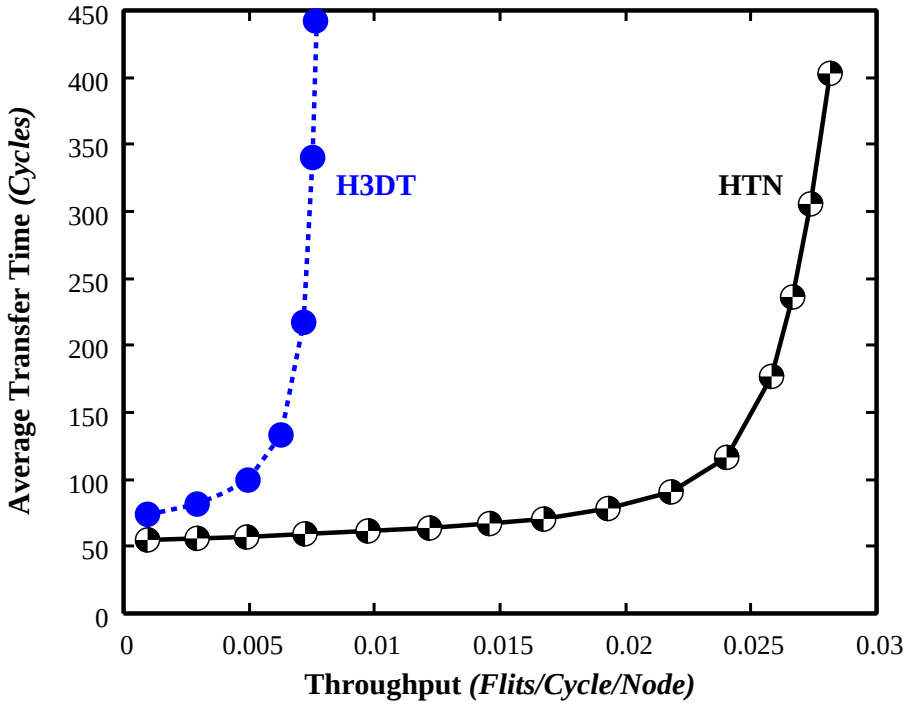


Figure 5.14: Dynamic communication performance between HTN and H3D-torus network with dimension order routing algorithm for short message: 512 nodes, 2 virtual channels.

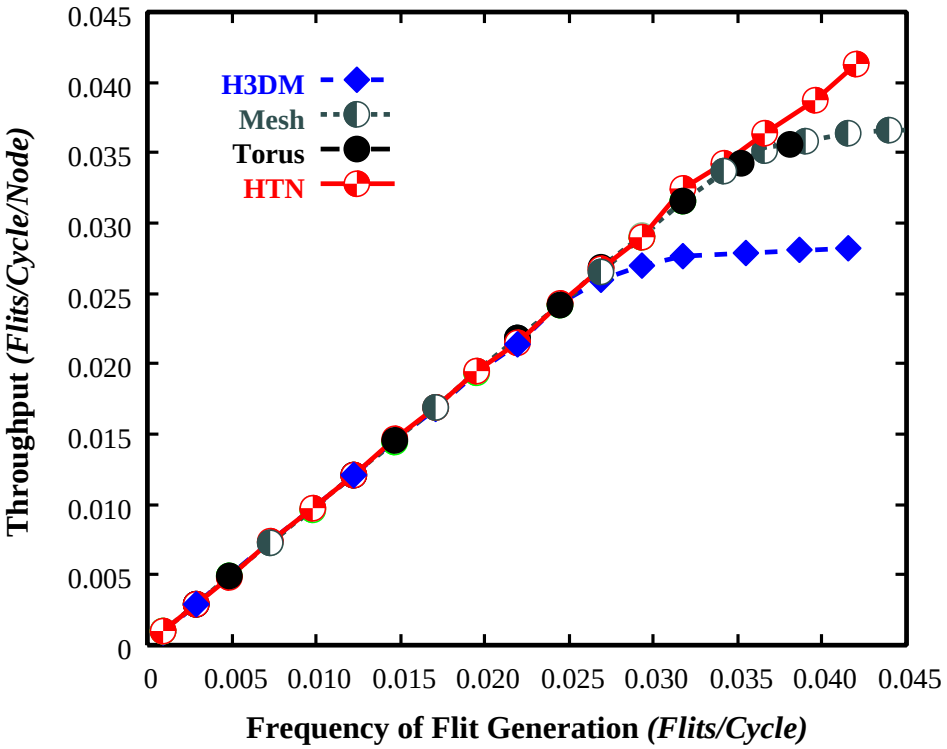


Figure 5.15: Effect of Flits generation rate for short message: 1024 nodes, 2 virtual channels, and

5.2.4 Effect of Message Length

In this section, we have analyzed the effect of message length on performance. Figure 5.16 and 5.17 show the average message latency divided by message length for uniform traffic pattern. In Fig. 5.16, we have considered 3 virtual channels. Here, the dynamic communication performance is simulated for short and medium messages. It is shown that the dynamic communication performance of medium message is better than short message for all networks. For the sake of clarity, Fig. 5.17 shows only the curve of HTN with short, medium, and long messages. In this case, 2 virtual channels per physical channel is used for evaluating dynamic communication performance.

The average message latency is smaller for long messages. The reason is that wormhole switching is used. Thus, they are pipelined in nature. Path setup time is amortized among more flits when messages are long. Moreover, data flit can advance faster than message headers because headers have to be take the routing decision. Hence, headers have to wait for the routing control unit to compute the output channel, and possibly waiting for the output channel to become free. Therefore, when the header reaches the destination node, data flits advance faster, thus favoring long message.

Figure 5.17 shows that average transfer time is decreasing and the maximum throughput is increasing in Hierarchical Torus Network (HTN) along with the increase of message length.

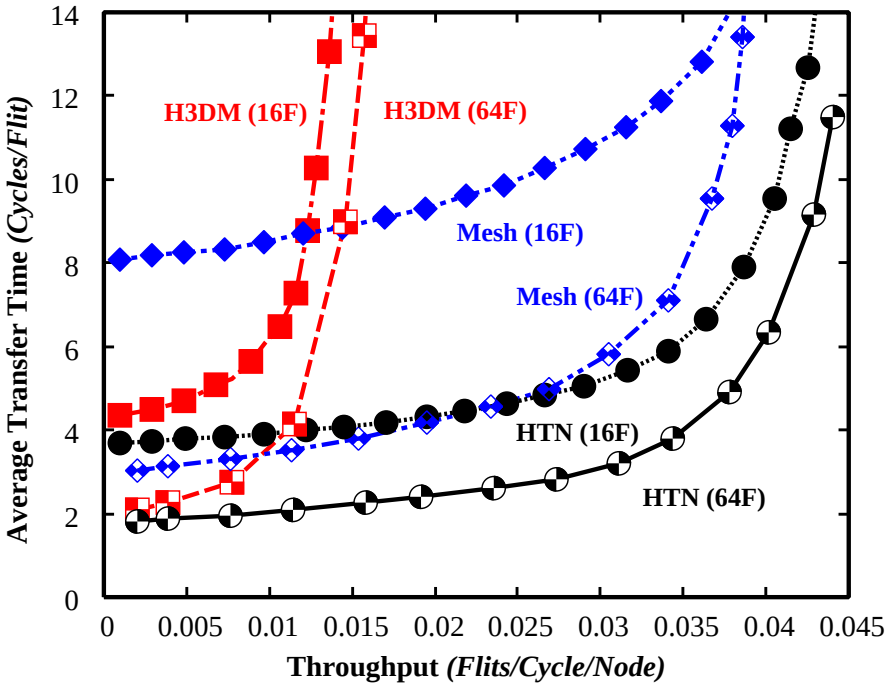


Figure 5.16: Average message latency divided by message length vs. network throughput of different networks: 1024 nodes, 3 virtual channels.

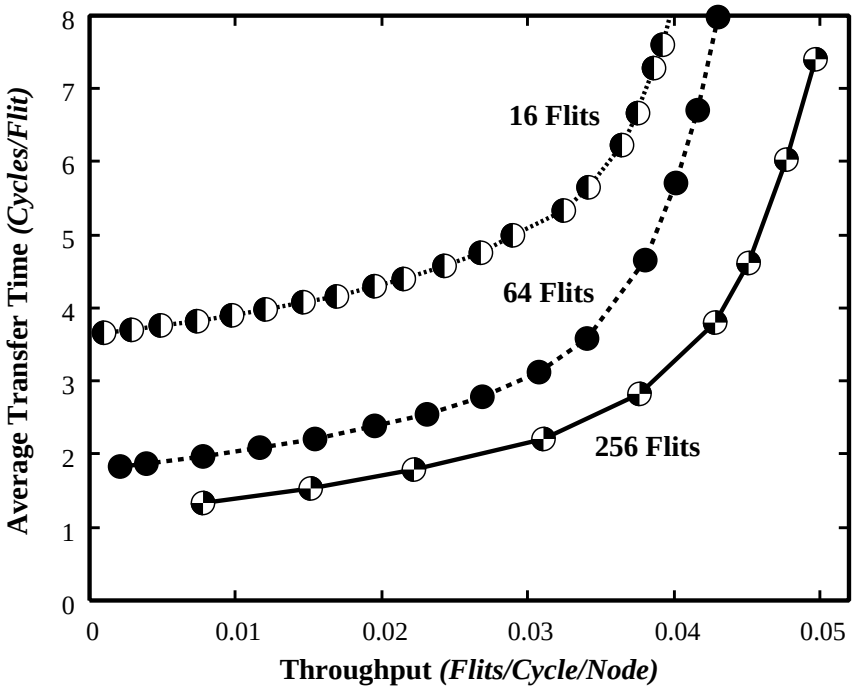


Figure 5.17: Average message latency divided by message length vs. network throughput of HTN: 1024 nodes, 2 virtual channels.

5.2.5 Effect of the Number of Virtual Channel

Splitting each physical channel into several virtual channels increases the number of routing choices, allowing messages to pass blocked messages. On the other hand, flits from several messages are multiplexed onto the same physical channel, slowing down both messages. The effect of increasing the number of virtual channels has been analyzed in [38] for deterministic routing algorithm on a 2D-mesh network.

In [38], it has been pointed out that the larger the number of virtual channels, the higher the performance achieved. Unfortunately, this is not true for all the topologies and network traffic conditions [40]. The first approach may be choosing the optimal number of virtual channels for a given network. However, that optimal number depends on several parameters, including network traffic. In this section, we have examined the effect of the number of virtual channels on the performance of the HTN.

Figure 5.18 shows the behavior of the dimension order routing algorithm with 2, 3, 4, and 6 virtual channels per physical channel on the HTN. The dynamic communication performance of HTN with different virtual channels is almost same. Adding more virtual channels does not increase communication performance considerably. Because the buffer size is smaller and blocked message occupy more channels. As a result, throughput increases by a very small amount. Also, message latency slightly increases when adding virtual channels.

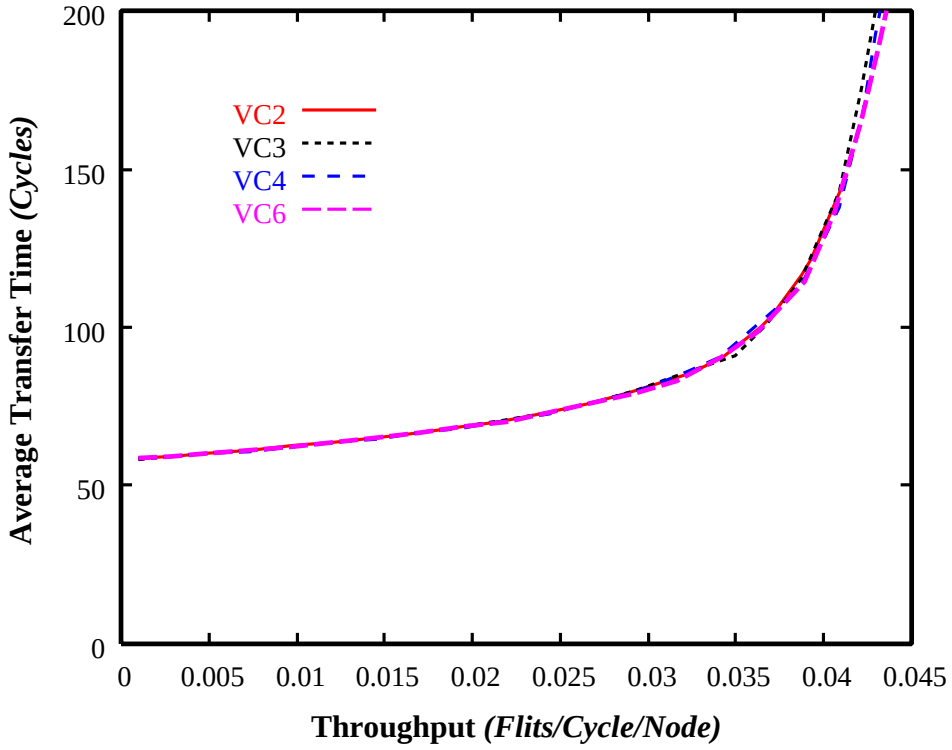


Figure 5.18: Effect of the number of virtual channels on dimension order routing algorithm for HTN: 1024 nodes, 16 flits

5.3 Fault Tolerance Performance

Performance and fault tolerance are two dominant issues facing the design of interconnection networks for massively parallel computers. Fault tolerance is the ability of the network to function in the presence of component failures. However, techniques used to realize fault tolerance are often at the expense of considerable performance degradation. Conversely, making high-performance communication techniques resilient to network faults posses challenging problems. In this section, we have pointed out the redundancy and yield of the Hierarchical Torus Network (HTN).

5.3.1 Path Substitution

Faulty PEs are replaced by means of substitution paths. A substitution path is defined as a sequence of adjacent PEs beginning with a faulty PE and ending in a spare PE. It must contain exactly one faulty PE, one spare PE, and zero or more healthy PEs. Two substitution paths are disjoint if they don't share a PE. Basic types of substitution paths, depicted in Fig. 5.19, are Shift, L-shaped, and Diagonal [42]. A *Shift* substitution path replaces a faulty PE by shifting the PEs along a single row or column. An *L-Shaped* substitution path borrows spare nodes from other rows or columns to replace the faulty PE by the spare PE of that borrowed rows or columns. A *Diagonal* substitution path replaces a faulty node by a spare node that are surrounded by faulty nodes.

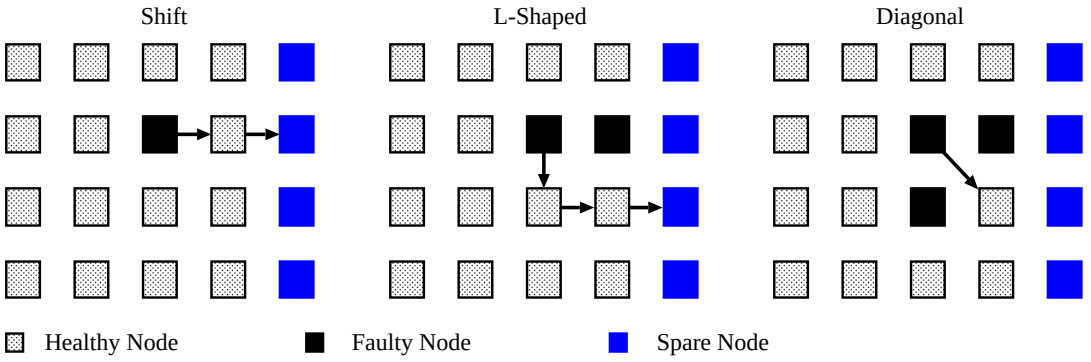


Figure 5.19: Substitution paths

5.3.2 Redundancy and Yield

The Hierarchical Torus Network (HTN) is implemented with redundancy at each level of hierarchy, i.e., basic modules have spare PEs, the Level-2 network has spare BMs, the Level-3 network has spare Level-2 subnetworks and so on. This is devised for fault tolerance. This hierarchical redundancy scheme is portrayed in Fig. 5.20.

Let us focus on the basic module. The BM of HTN is an $(m \times m \times m)$ 3D-torus network. A redundant BM includes m columns of spare nodes as well as the necessary

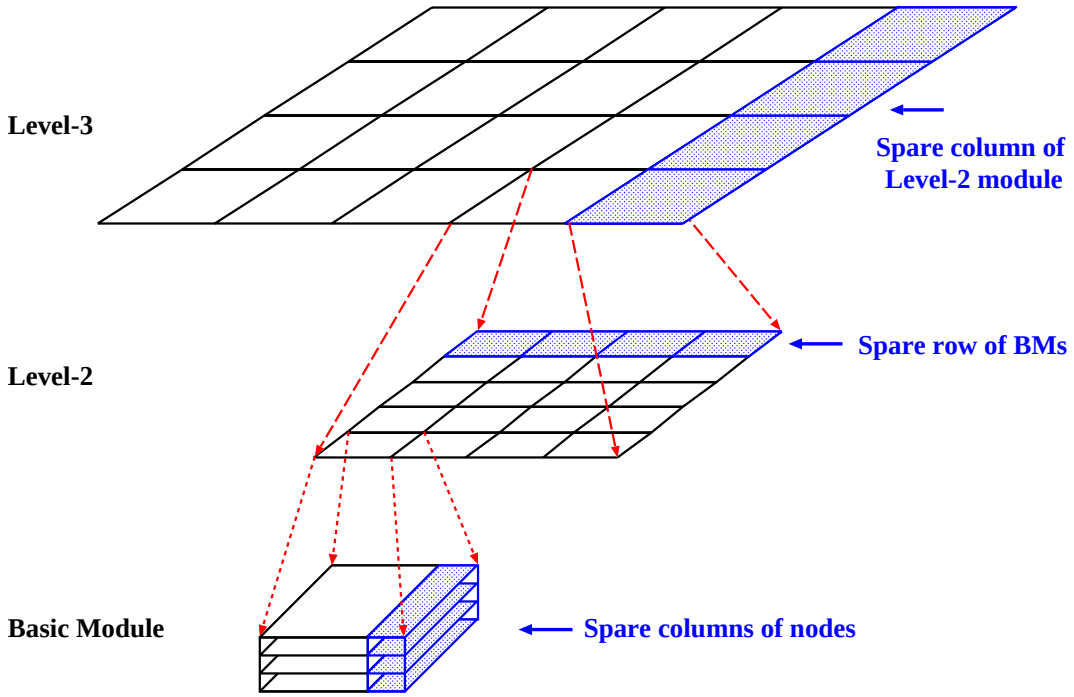


Figure 5.20: Hierarchical redundancy

switches to reconfigure the module using the healthy nodes. We need more switches to reconfigure the BM since it is a 3D-torus network. To reduce the number of switches and reconfiguration complexity we have restricted the reconfiguration strategy. Each plane in the xy-plane of the BM includes a column of spare nodes to replace the faulty nodes of that plane using the healthy nodes of the spare column. Each plane in the xy-plane of the BM is a 2D-torus network. For example, in a $(4 \times 4 \times 4)$ redundant basic module with the 4 columns of spare nodes have $5 \times 4 \times 4 = 80$ PEs, while only 64 are needed. Thus, each plane has $5 \times 4 = 20$ PEs, while only 16 are needed. With this arrangement, a maximum of 4 faulty PEs per plane can be tolerated through replacement by the spare PEs, as illustrated in Fig. 5.21, 5.22, and 5.23. Needless to say that more than 4 PE per plane turn out to be defective or faulty in which the BM is declared as non-reconfigurable. In fact, switches and links must also be taken into account. The switch states useful for reconfiguration, are (a) no connect, (b) north-to-south and east-to-west, (c) north-to-west and south-to-east, and (d) north-to-east and south-to-west connects.

At the second level, a row of spare BMs is provided so that a faulty BM may be replaced by one of the spare BMs. We should remark that the scheme can be improved upon by mutual sharing of the spare column between adjacent planes of the basic module and adjacent BMs. This can enhance the harvesting even further.

To assess the effectiveness of the fault tolerance strategy, let us compute the yield of the system. Assuming that the defect distribution in the PEs, switches and links is Poisson. Yield is defined as the probability of obtaining a fault free network. The yield of a basic

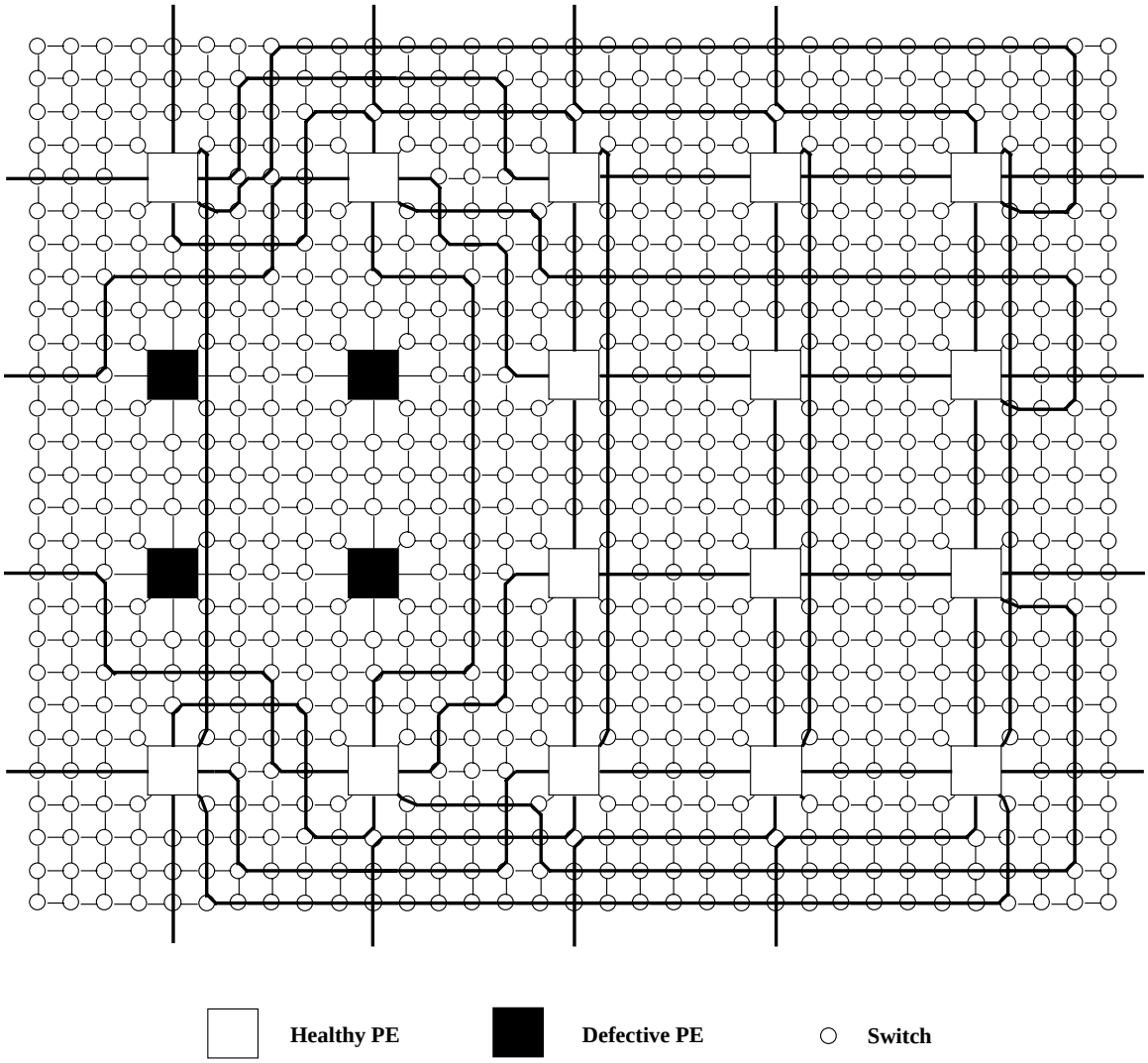


Figure 5.21: Reconfiguration of a BM in the presence of faulty PEs.

module is given by:

$$Y_{BM} \quad = \quad Y_{PE's} \times Y_{switches,\ links} \tag{5.1}$$

That is the probability of a healthy BM is estimated as the product of (a) the probability of having a minimum of 64 healthy PEs and (b) the probability of all links and switches being healthy. This is a conservative estimation. In a $(4 \times 4 \times 4)$ BM, a maximum of 4 faulty PEs can be tolerated in each plane, the yield of the PEs per plane of the BM can be estimated by Eq. 5.2. 4 planes are available in that BM. Thus, the yield of the PEs per BM can be estimated by Eq. 5.3. Here, Y_{node} is the probability that a PE is fault free.

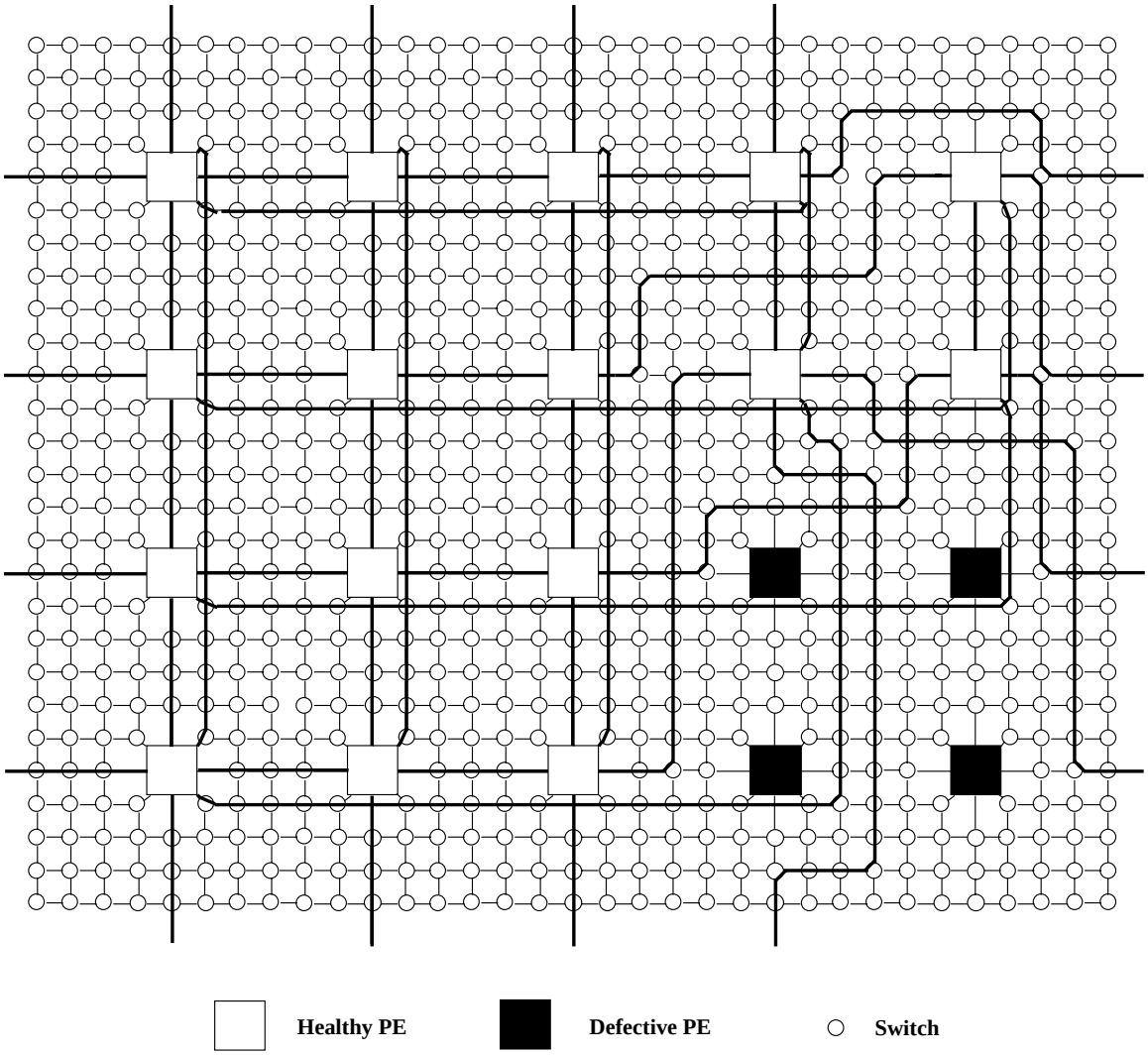


Figure 5.22: Reconfiguration of a BM in the presence of faulty PEs.

$$Y_{PE's} = \sum_{k=16}^{20} \Pr\{k \text{ PE's are good}\} \tag{5.2}$$

$$Y_{PE's} = \left[\sum_{k=16}^{20} \binom{20}{k} Y_{node}^{k} (1 - Y_{node})^{20-k}\right]^{4} \tag{5.3}$$

Consider that the size of the PE is 2×2 *mm square* in 0.25 micron CMOS technology, then $Y_{node} = e^{-0.04D}$, where D denotes the fault density per *cm square*. Assuming a channel width of 1.0 *mm* between PEs, the tile area is 9.00 mm^2. Since the PE area is 4 mm^2, the total channel area per tile becomes 5.00 mm^2. It is reasonable to assume

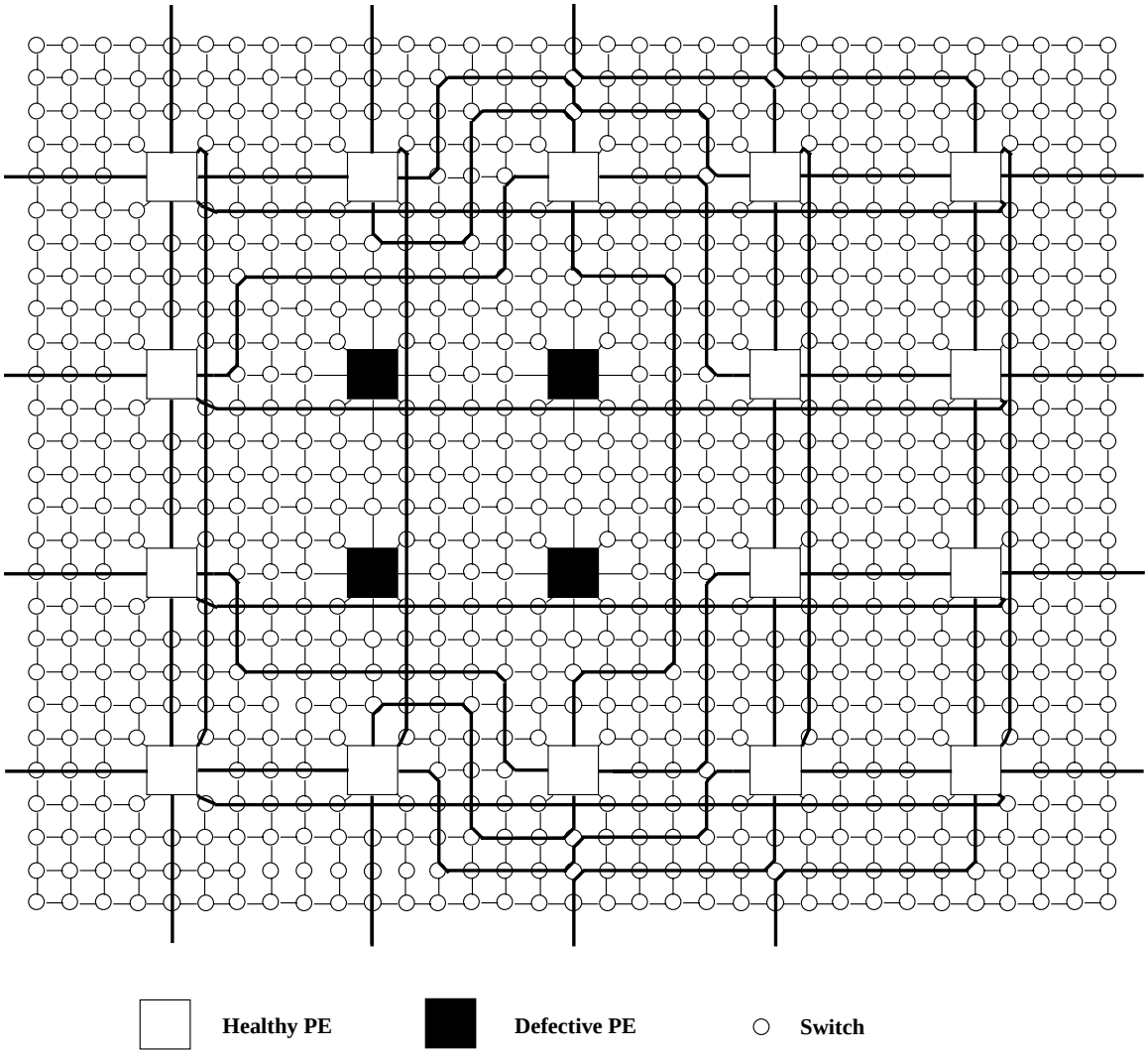


Figure 5.23: Reconfiguration of a BM in the presence of faulty PEs.

that the area occupied by the switches and links is about 30% of the channel area [6]; further, we assume that 50% of this is critical area. Therefore, the critical area used by the switches and links per tile is 0.75 mm^2. As stated above, each redundant planes of the BM consists of 20 PEs or titles. Thus, the total critical area of switches and links per plane is $20 \times 0.75 = 15.00\ mm^2 = 0.15\ cm^2$. Therefore the yield of the switches and links is computed as:

$$\begin{aligned} Y_{switches,\ links} &= \left[e^{-Critical\ Area\ of\ Switches\ \&\ Links \times D}\right]^4 \\ &= \left[e^{-0.15D}\right]^4 \end{aligned} \tag{5.4}$$

For the second level, as mentioned above, a row of spare BMs is provided. Therefore,

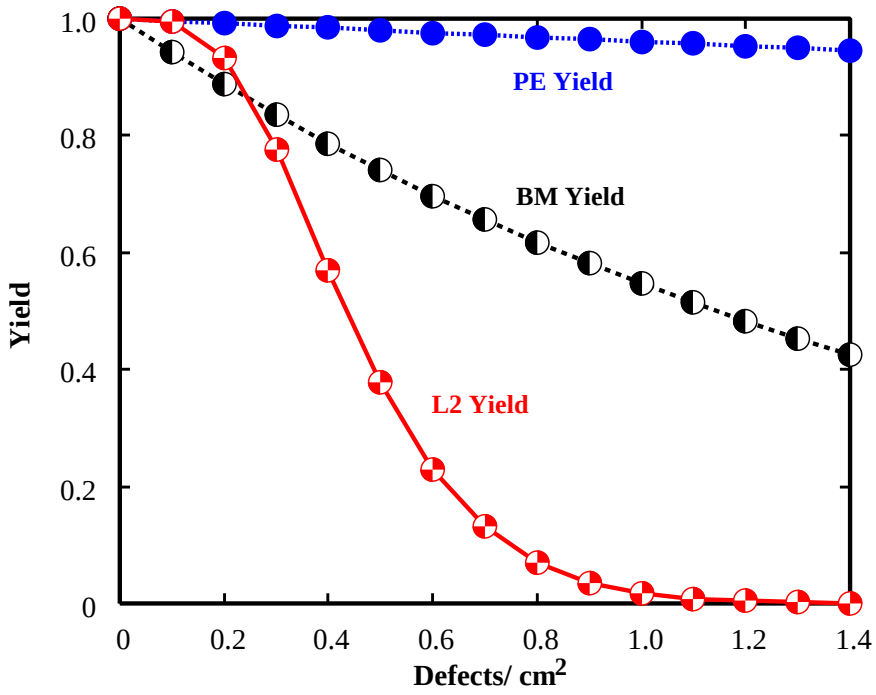


Figure 5.24: Yield for BM and Level-2 network vs. fault density with spare node

the second level network yield becomes as follows:

$$Y_{second\_level} = \sum_{k=16}^{20} \binom{20}{k} Y_{BM}^{k} \left(1 - Y_{BM}\right)^{20-k} \tag{5.5}$$

For different fault densities with and without spare elements the yield is shown in Fig. 5.24 and Fig. 5.25, respectively. It is shown that the use of an additional column of nodes in each plane of the BM, a spare row of BMs for each second level network, a spare column of second level subnetworks at the third level, and so on, results in remarkable enhancement of network yield. Thus, with a 25% redundancy at each level, the yield at the second level is estimated to range from 0.995 to 0.378 corresponding to the fault density ranging from 0.10 $defects/cm^2$ to 0.50 $defects/cm^2$. Thus, the yielding of HTN is satisfactory.

5.4 Application Mapping

One of the desirable attributes of designing interconnection network is convenient mapping of applications, specially those with regularity of computations in the designed network [30]. Mapping an algorithm to an interconnection network involve mapping computations to processors so that the algorithm runs efficiently. There are two cases of mapping algorithms to interconnection networks. In the first case, a known parallel algorithm

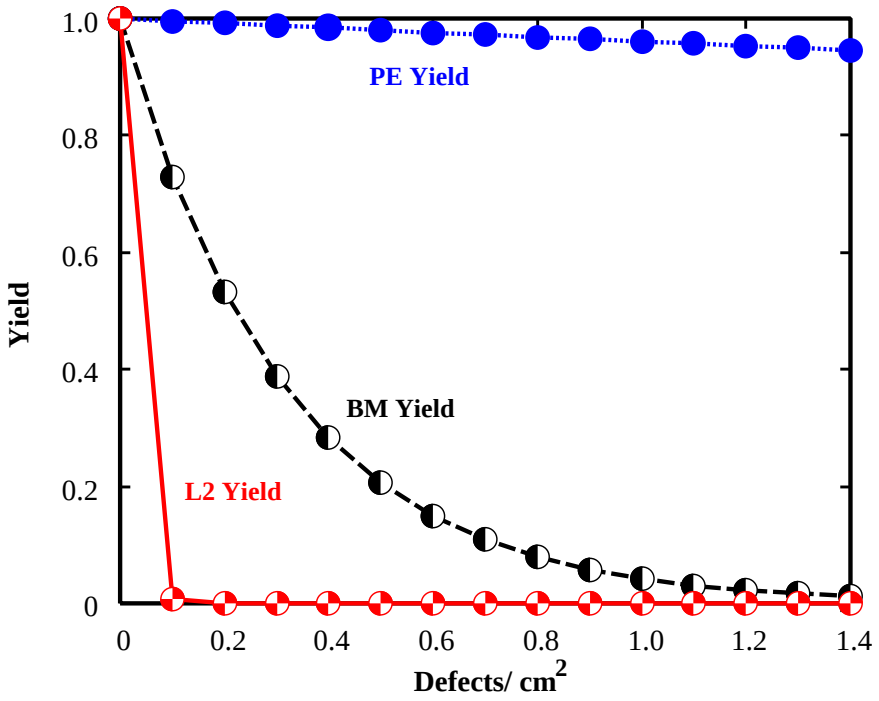


Figure 5.25: Yield for BM and Level-2 network vs. fault density without spare node

for solving the problem is mapped to the interconnection network to achieve maximum parallelism. This does not modifying the computation behavior of the algorithm. In many cases, however, the communication behavior of the algorithm can not be supported efficiently by the given network, and a new parallel algorithm may need to be designed to take the advantages of the underlying network. Thus, mapping also includes the design of parallel algorithms for a specific interconnection network.

In this section, we will discuss the mapping of advanced applications, namely bitonic merge, Fast Fourier Transform (FFT) and finding the maximum.

5.4.1 Converge and Diverge

Several interesting applications involve an input vector consisting of N pieces of data and utilize a divide-and-conquer scheme. Therefore, it is useful to first map the CONVERGE and DIVERGE function to the hierarchical networks.

Let N be an integer number, where $N = 2^k$, the value of data be $d[m]$, where $(m = 0, 1, 2, \ldots \ldots \ldots N-1)$.

DIVERGE function executes an operation between $2^0, 2^1, 2^2, \ldots \ldots \ldots, 2^{k-2}, 2^{k-1}$. On the other hand, CONVERGE function executes an operation between $2^{k-1}, 2^{k-2}, \ldots \ldots \ldots, 2^1, 2^0$. CONVERGE and DIVERGE functions are defined by:

CONVERGE();

for $j = k-1 : -1 : 0$

```
      for 0 ≤ m ≤ N − 1 do in parallel
        if a_j = 0,OPERATION(m, m + 2^j); endif;
      endfor;
    endfor;
  end;

  DIVERGE();
    for j = 0 : k − 1
      for 0 ≤ m ≤ N − 1 do in parallel
        if a_j = 0,OPERATION(m, m + 2^j); endif;
      endfor;
    endfor;
  end;
```

Where a_j be the j-th bit of the binary representation of m, OPERATION$(m, m+2^j)$ is an operation between $d[m]$ and $d[m+2^j]$. Figure 5.26 illustrates the execution of converge on a 4×4 2D-mesh.

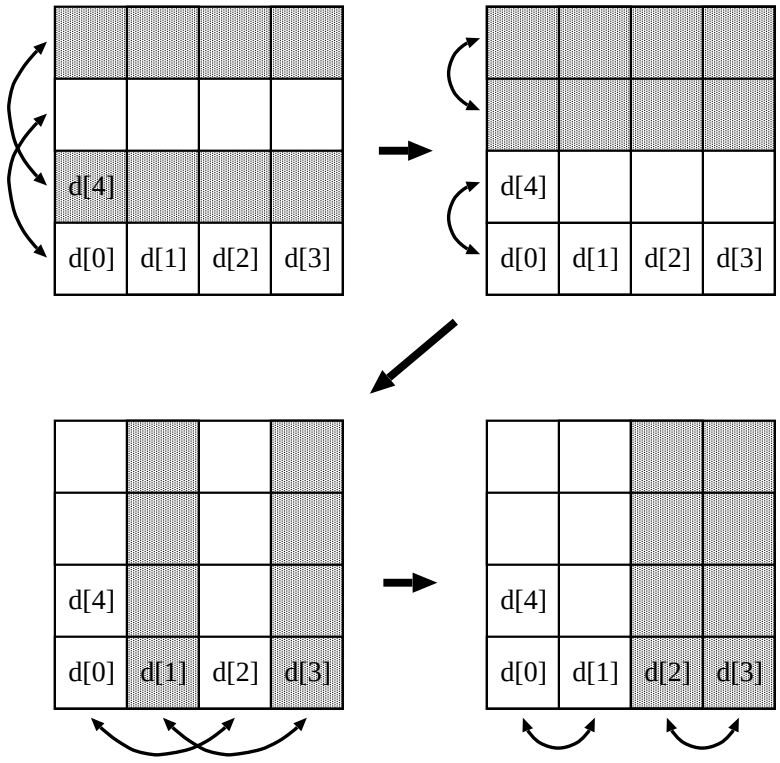


Figure 5.26: CONVERGE on a 4×4 2D-mesh

5.4.2 Converge and Diverge on HTN

First we begin by mapping CONVERGE on a basic module of HTN. CONVERGE on a BM of HTN is given as follows:

CONVERGE_BM();
for each element m in z-axis(${a_z}^1 = 0, 1$) do in parallel
  OPERATION($z[m, i], z[m, i + 2]$);
endfor;
for each element m in z-axis(${a_z}^1 = 0, 2$) do in parallel
  OPERATION($z[m, i], z[m, i + 1]$);
endfor; for each element m in y-axis(${a_y}^1 = 0, 1$) do in parallel
  OPERATION($y[m, i], y[m, i + 2]$);
endfor;
for each element m in y-axis(${a_y}^1 = 0, 2$) do in parallel
  OPERATION($y[m, i], y[m, i + 1]$);
endfor; for each element m in x-axis(${a_x}^1 = 0, 1$) do in parallel
  OPERATION($x[m, i], x[m, i + 2]$);
endfor;
for each element m in x-axis(${a_x}^1 = 0, 2$) do in parallel
  OPERATION($x[m, i], x[m, i + 1]$);
endfor;

Similarly the DIVERGE function can be obtained if the CONVERGE function is executed in reverse order. Both the CONVERGE and DIVERGE function can also be executed by higher level networks. Now we discuss the CONVERGE of higher level network of HTN.

CONVERGE_HTN-L();
for each element m in y-axis(${a_y}^L = 0, 1$) do in parallel
  OPERATION($y[m, i], y[m, i + 2]$);
endfor;
for each element m in y-axis(${a_y}^L = 0, 2$) do in parallel
  OPERATION($y[m, i], y[m, i + 1]$);
endfor; for each element m in x-axis(${a_x}^L = 0, 1$) do in parallel
  OPERATION($x[m, i], x[m, i + 2]$);
endfor;
for each element m in x-axis(${a_x}^L = 0, 2$) do in parallel
  OPERATION($x[m, i], x[m, i + 1]$);
endfor;

Here, the x-axis and y-axis at higher levels correspond a sequence of subnets. CONVERGE operation at Level-2 executes between BMs. CONVERGE operation at Level-3 executes between Level-2 networks.

5.4.3 Bitonic Merge

Sorting is an important operation for arranging data in many applications. Many efficient sorting algorithms have been developed over the years, using a variety of techniques. Sorting involves comparing different pairs of data items and rearranging them.

In this section, we discuss the bitonic merge [44, 45] and estimate the processing time of bitonic merging algorithm on HTN. The following definition and theorem provide the background of bitonic merge.

Definition 5.1 *A sequence* $a_1, a_2, a_3, ..., a_{2n}$ *is said to be bitonic if either*

1. *there is an integer* $1 \leq j \leq 2n$ *such that* $a_1 \leq a_2 \leq a_3 \leq ..... \leq a_j \geq a_{j+1} \geq a_{j+2} \geq ..... \geq a_{2n}$
2. *the sequence does not initially satisfy condition (1) but can be shifted cyclically until condition (1) is satisfied.*

For example, $\{1, 3, 5, 6, 7, 9, 4, 2\}$ is a bitonic sequence as it satisfies condition (1). Similarly, the sequence $\{7, 8, 6, 4, 3, 1, 2, 5\}$, which does not satisfy condition (1), is also bitonic as it can be shifted cyclically to obtain $\{2, 5, 7, 8, 6, 4, 3, 1\}$.

Theorem 5.1 *Let* $\{a_1, a_2, a_3, ..., a_{2n}\}$ *be a bitonic sequence. If* $d_i = \min(a_i, a_{n+i})$ *and* $e_i = \max(a_i, a_{n+i})$ *for* $1 \leq i \leq n$, *then*

1. $\{d_1, d_2, d_3, ..., d_n\}$ *and* $\{e_1, e_2, e_3, ..., e_n\}$ *are each bitonic, and*
2. $\max(d_1, d_2, d_3, ..., d_n) \leq \min(e_1, e_2, e_3, ..., e_n)$.

The bitonic merging algorithm sorts a bitonic sequence in either ascending or descending order. Its routine falls in the class of either CONVERGE or DIVERGE functions. The operation to make a bitonic list is given by:

```
OPERATION(m,m+2^j)
{
  move R1(m + 2^j) to R2(m);
  if a_{j+1}=0,
    [R1(m), R2(m)]=[min{R1(m), R2(m)},max{R1(m), R2(m)}] ;
  else
    [R1(m), R2(m)]=[max{R1(m), R2(m)},min{R1(m), R2(m)}];
  endif;
  move R2(m) to R1(m + 2^j);
}
```

According to theorem 5.1.(1), the bitonic merge operation becomes as follows:

```
OPERATION(m,m+2^j)
{
```

move $R_1(m+2^j)$ to $R_2(m)$;
$[R_1(m), R_2(m)]$=[min$\{R_1(m), R_2(m)\}$, max$\{R_1(m), R_2(m)\}$];
move $R_2(m)$ to $R_1(m+2^j)$;
}

This algorithm has two important steps. The 'move step' is used to transfer data from one node to another, and the comparison (min, max) step.

5.4.4 Fast Fourier Transform (FFT)

Fast Fourier Transform (FFT) has wide and variety applications and it requires very high speed computation. FFT is an efficient algorithm for the computation of the discrete Fourier transform:

$$X[k] = \sum_{k=0}^{N-1} x[n]\ W_N^{nk} \qquad k = 0, 1, 2, \ldots\ \ldots,\ N-1 \tag{5.6}$$

where, $W_N = exp(\frac{-j2\pi}{N})$. The basic operation of the decimation in frequency FFT algorithm is the following butterfly operation:

$$\begin{aligned} x_m[p] &= x_{m-1}[p] + x_{m-1}[q] \\ x_m[q] &= (x_{m-1}[p] - x_{m-1}[q])W_n^r \end{aligned} \tag{5.7}$$

here, $k = \log N$, and m is the number of stages for butterfly.

Since the two input data of a butterfly in the m-th stage are 2^{k-m} locations apart, the FFT algorithm falls in the CONVERGE function class. Thus, the CONVERGE function is modified to perform an FFT algorithm as follows:

OPERATION(m,m+2^j)
{
move $R_1(m+2^j)$ to $R_2(m)$;
$Temp(m) = R_1(m) + R_2(m)$;
$R_2(m) = (R_1(m) - R_2(m)) * {W_N}^r$;
$R_1(m) = Temp(m)$;
move $R_2(m)$ to $R_1(m+2^j)$;
}

The total time required to perform the FFT on N pieces of data is the same as the time required to sort N pieces of data by the bitonic merge algorithm except that T_{OPER} is different. It is the time required to perform the operations in the middle 3 lines of the above pseudo-code.

5.4.5 Finding the Maximum

To find the maximum or minimum value among a large number of data, CONVERGE is iteratively used. However, the recursive execution is not necessary for finding the maximum. Therefore, the CONVERGE operation becomes simpler as follows:

CONVERGE();
for $j = k-1 : -1 : 0$
  for $0 \leq m \leq \frac{N}{2}^{k-j-1} - 1$ do in parallel
    if $a_j = 0$, **OPERATION**$(m, m+2^j)$; endif;
  endfor;
endfor;
end;

Thus, the operation to find the maximum is given by:
OPERATION$(m, m+2^j)$
{
  move $R_1(m+2^j)$ to $R_2(m)$;
  $R_1(m) = \max[R_1(m), R_2(m)]$;
}
The total operation to find the maximum is given by:
MAX
{
  CONVERGE_BM();
  for $j = 2 : L$
    if $a_x^{[j-1:0]} = 0$ *and* $a_y^{[j-1:0]} = 0$ *and* $a_z^{[j-1:0]} = 0$,**OPERATION**$(m, m+2^j)$;endif;
      CONVERGE_network-j();
    endif;
  endfor;
}

5.4.6 Processing Time

To estimate the processing time of an application on interconnection network of massively parallel computers, we define communication time and execution time. Communication time is defined as the time required for unit distance routing-step, i.e., moving one data of a sequence from a node to one of its neighboring nodes. Execution time is defined as the time required for execution of one OPERATION. CONVERGE and DIVERGE functions require $logN$ steps. Thus, the total time required for an application is as follows:

$$T = \alpha \times T_{move} + logN \times T_{OPER} \tag{5.8}$$

Where,

$$\begin{aligned} \alpha &= \text{Total number of communication steps.} \\ T_{move} &= \text{Transfer time between adjacent nodes.} \\ T_{OPER} &= \text{Execution time for OPERATION.} \end{aligned}$$

The performance of an application mapping is discussed by using the total number of communication steps in an interconnection network. The total number of communication steps of different networks are obtained as follows:

2D-Torus

$$\begin{aligned} S_{bitonic}^{2D-Torus} &= 4 \sum_{j=1}^{log\sqrt{N}} \left(\frac{\sqrt{N}}{2^j} \right) \\ &= 4\left(\sqrt{N} - 1\right) \end{aligned} \tag{5.9}$$

$$S_{max}^{2D-Torus} = 2\left(\sqrt{N} - 1\right) \tag{5.10}$$

3D-Torus

$$\begin{aligned} S_{bitonic}^{3D-Torus} &= 6 \sum_{j=1}^{logN^{\frac{1}{3}}} \left(\frac{N^{\frac{1}{3}}}{2^j} \right) \\ &= 6\left(N^{\frac{1}{3}} - 1\right) \end{aligned} \tag{5.11}$$

$$S_{max}^{3D-Torus} = 3\left(N^{\frac{1}{3}} - 1\right) \tag{5.12}$$

The total communication steps for finding the maximum is half of the bitonic merge, because the finding of maximum of maximum does not necessary for return data in 2D-torus and 3D-torus networks.

H3D-Torus

$$S_{bitonic}^{H3D-Torus} = 6(L-1) \times \left(\sum_{j=1}^{logN_{HL}^{\frac{1}{3}}} \frac{N_{sn}^{\frac{1}{3}}}{2^j} \right) \times N_{BM} + S_{bitonic}^{BM} \tag{5.13}$$

$$S_{max}^{H3D-Torus} = 6(L-1) \times \left(\sum_{j=1}^{logN_{HL}^{\frac{1}{3}}} \frac{N_{sn}^{\frac{1}{3}}}{2^j} \right) + S_{bitonic}^{BM} \tag{5.14}$$

L = Level number.
N_{HL} = Size of the higher level network.
N_{BM} = Size of the basic module.
$S^{BM}_{bitonic}$ = Number of communication steps in basic module.

If the size of the basic module is $(4 \times 4 \times 4)$ and the size of the higher level network is $(4 \times 4 \times 4)$, then

$$S^{H3D-Torus}_{bitonic} = 1152(L-1) + 18 \tag{5.15}$$

$$S^{H3D-Torus}_{max} = 18L \tag{5.16}$$

H3D-Mesh

$$S^{H3D-Mesh}_{bitonic} = 4(L-1) \times \sum_{j=1}^{log\sqrt{N_{HL}}} \left(\frac{\sqrt{N_{HL}}}{2^j} \right) \times \frac{N_{BM}}{4} + S^{BM}_{bitonic} \tag{5.17}$$

$$S^{H3D-Mesh}_{max} = 4(L-1) \times \sum_{j=1}^{log\sqrt{N_{HL}}} \left(\frac{\sqrt{N_{HL}}}{2^j} \right) + S^{BM}_{bitonic} \tag{5.18}$$

If the size of the basic module is $(4 \times 4 \times 4)$ and the size of the higher level network is (4×4), then

$$S^{H3D-Mesh}_{bitonic} = 192(L-1) + 18 \tag{5.19}$$

$$S^{H3D-Mesh}_{max} = 12(L-1) + 18 \tag{5.20}$$

HTN

$$S^{HTN}_{bitonic} = 4(L-1) \times \left(\sum_{j=1}^{logn} \frac{n}{2^j} \right) \times \frac{m^3}{m \times q} + S^{BM}_{bitonic} \tag{5.21}$$

$$S^{HTN}_{max} = 4(L-1) \times \left(\sum_{j=1}^{logn} \frac{n}{2^j} \right) + S^{BM}_{bitonic} \tag{5.22}$$

If $m = 4$ and $n = 4$, i.e., the size of the basic module is $(4 \times 4 \times 4)$ and the size of the higher level network is (4×4), then

$$S^{HTN}_{bitonic} = 192(L-1) + 18 \tag{5.23}$$

$$S^{HTN}_{max} = 12(L-1) + 18 \tag{5.24}$$

5.4.7 Mapping Performance

Table 5.1 summarizes the total number of communication steps on a network for bitonic merge, FFT, and finding the maximum.

Table 5.1: The total number of communication steps on a network for bitonic merge, FFT, and finding the maximum.

	Bitonic Merge	FFT	Finding the maximum
2D-torus	$4(\sqrt{N}-1)$	$4(\sqrt{N}-1)$	$2(\sqrt{N}-1)$
3D-torus	$6(\sqrt{N}-1)$	$6(\sqrt{N}-1)$	$3(\sqrt{N}-1)$
H3D-torus	$1152(L-1)+18$	$1152(L-1)+18$	$18L$
H3D-mesh	$192(L-1)+18$	$192(L-1)+18$	$12(L-1)+18$
HTN	$192(L-1)+18$	$192(L-1)+18$	$12(L-1)+18$

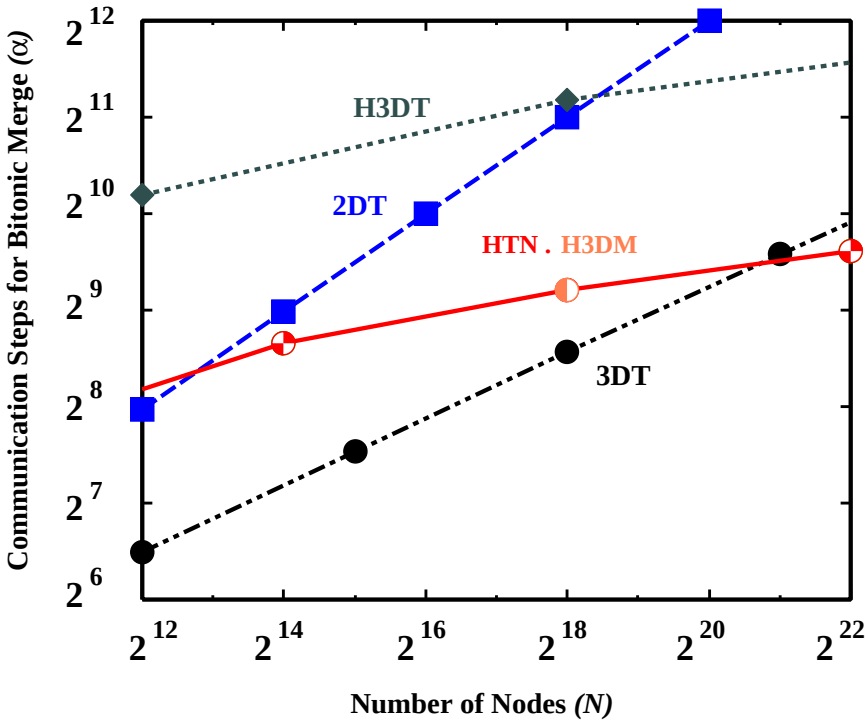


Figure 5.27: The total number of communication steps of the bitonic merge in different networks

Figure 5.27 shows the total number of communication steps of bitonic merge on different networks. Hierarchical networks execute this operation using $O(logN)$ communication steps. Since bitonic merge requires a large number of communication between all PEs

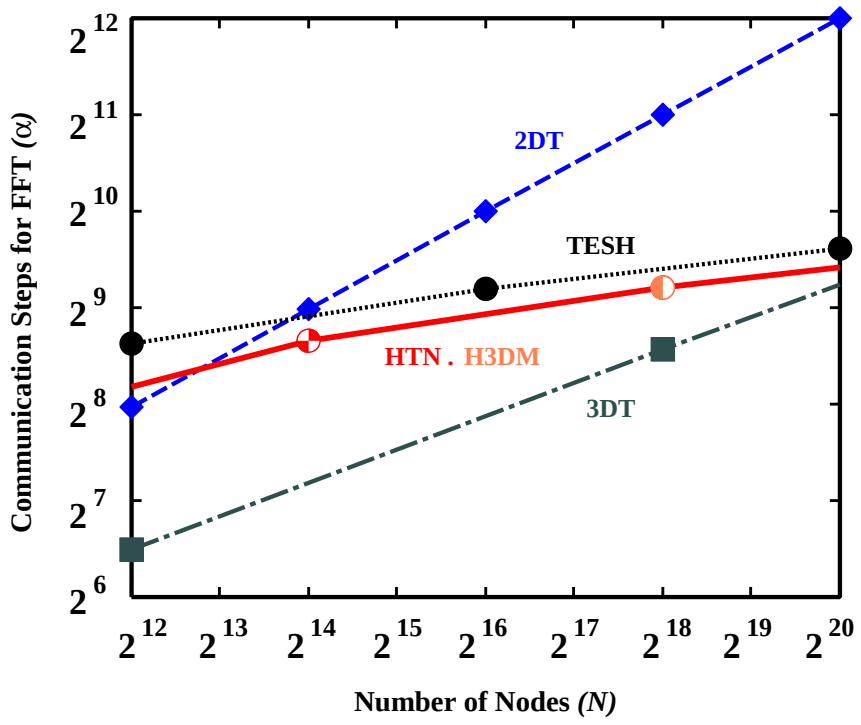


Figure 5.28: The total number of communication steps of the FFT in different networks

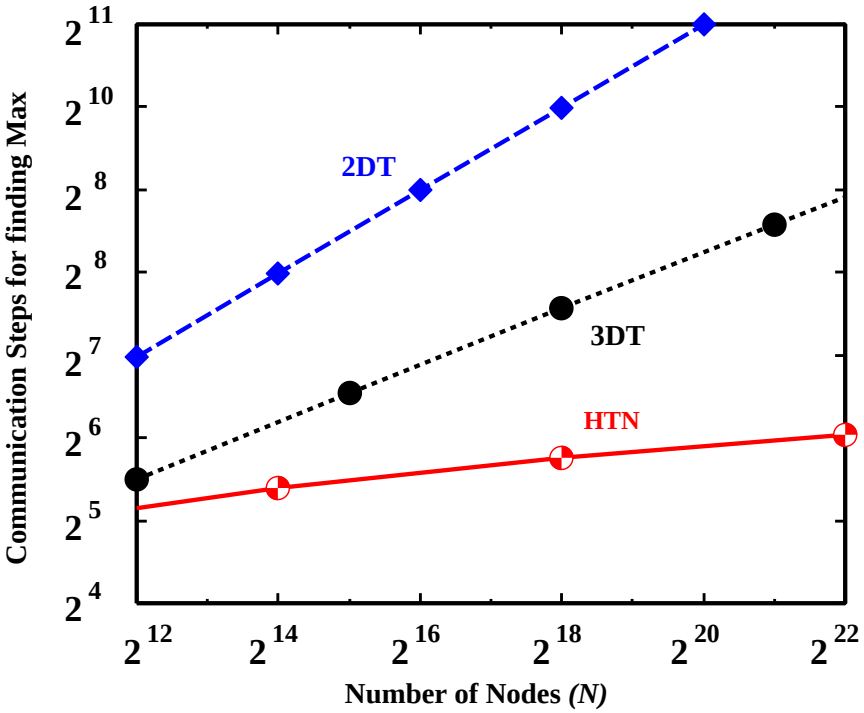


Figure 5.29: The total number of communication steps for finding the maximum in different networks

at each step, the total number of communication steps in HTN is larger than 2D-torus and 3D-torus for thousands of PEs. For millions of PEs, however, HTN shows better performance than 2D-torus and 3D-torus networks. Also, HTN shows better performance than H3D-torus network. This Fig. 5.27 also represents the communication steps of FFT on different networks. Figure 5.28 also shows the communication steps of FFT on different networks. The total communication steps of FFT on HTN is smaller than TESH network and k-ary n-cube networks. Thus, the total communication steps of bitonic merge on HTN is smaller than TESH network.

Figure 5.29 shows the total number of communication steps for finding the maximum. Here it is shown that HTN shows better performance than 2D-torus and 3D-torus networks.

5.5 Conclusions

In this chapter, we have studied the performance of Hierarchical Torus Network (HTN), analyzed the effect of network traffic and dependencies on some evaluation standards. Dynamic communication performance is simulated for HTN as well as several other networks for parallel computers. It is seen that the average transfer time of HTN is smaller than mesh, torus, TESH, H3D-mesh, and H3D-torus networks. Maximum throughput of HTN is also higher than those networks. The comparison is made under uniform traffic pattern and it reveals that HTN outperforms mesh, torus, TESH, H3D-mesh, and H3D-torus networks. Redundancy and yield of HTN are also indicated. Finally, we have discussed mapping of advanced application namely bitonic merge algorithm, FFT, and finding the maximum value on HTN. It is shown that the number of communication steps of these advanced applications on HTN is smaller than other networks.

Chapter 6

"Still round the corner there may wait, A new road or a secret gate."

– J.R.R. Tolkein

Conclusions

6.1 Conclusions

Parallel computers are generally built from processing elements which are interconnected using a network. A new hierarchical interconnection network, Hierarchical Torus Network (HTN) for massively parallel computer systems have been proposed in this thesis paper. Using the proposed HTN, millions of nodes can be connected together with retaining good network properties. Any design of an interconnection network is a trade-off of various evaluation standards. We studied various aspects of HTN in order to obtain a better interconnection network.

The interconnection philosophy of the HTN, routing of messages in the network, static network performance, and 3D-integration issues were discussed in detail. From the static network performance, it is seen that the HTN possesses several attractive features including fixed degree, small diameter, and small average distance. High bisection width matches the communication requirements for a given computational rate. On the other hand, a large bisection width is undesirable for a VLSI design of the interconnection network, since it implies a lot of extra chip wires. The bisection width of the HTN is higher than TESH and H3D-mesh network, equal to H3D-torus network and smaller than conventional topologies, respectively.

The network is well suited to 3D wafer stacked implementations. It is shown that the peak number of vertical links in a 3D wafer stacked implementation is quite small for HTN as compared to other similar networks. Thus, HTN permits efficient VLSI/ULSI/WSI realization. The layout area of HTN in a 3D wafer stacked implementation is amenable to 3D implementation, due to the smaller number of vertical wires needed than almost all other multi-computer networks.

We have used wormhole routing for switching, because it requires a small number of buffers and can control data flow in a pipelined fashion, reducing communication overhead. The characteristics of wormhole routing makes it particularly susceptible to deadlock. We have presented a deadlock-free routing algorithm for HTN using virtual channels. We used a deterministic dimension order routing algorithm for message passing. A dimension order routing algorithm is exceedingly simple and provides low latency and high bandwidth. It

is well-suited for uniform traffic patterns. Firstly, we have shown that six virtual channels per physical channel are sufficient for deadlock-free routing algorithm of an HTN. However, the hardware cost is high for this large number of virtual channels. In order to reduce the hardware cost, we have investigated the minimum number of virtual channels for deadlock-free routing of HTN. It has been proven that two virtual channels per physical channel are sufficient for deadlock-free routing algorithm of the HTN and this is the optimum value.

An interconnection network should transfer a maximum number of messages in the shortest time with minimum cost and maximal reliability. Interprocessor communication is a critical issue for interconnection networks. For good performance of a network, low latency and high throughput are indispensable. We have evaluated the dynamic communication performance of HTN as well as several other commonly used networks and hierarchical interconnection networks for parallel computers by dimension order routing under uniform traffic pattern. Evaluation of dynamic communication performance was carried out by computer simulation. The simulation studies were performed for short (16 Flits), medium (64 Flits), and long (256 flits) messages. It has been shown that the average transfer time of an HTN is smaller than mesh, torus, TESH, H3D-mesh, and H3D-torus networks. Maximum throughput of the HTN is also higher than those networks. The comparison of dynamic communication performance reveals that HTN outperforms mesh, torus, TESH, H3D-mesh, and H3D-torus networks.

Fault tolerant networks are essential for the reliability of massively parallel computer systems. Introducing redundancy in a network is a well-known technique for fault tolerance in interconnection networks. We have evaluated the redundancy and yield of HTN. It is shown that the yield of HTN is satisfactory with 25% redundancy.

The interconnection network used in a multicomputer in a parallel computer system plays a key role in determining how fast applications can be executed on the system. To show the suitability of the HTN, we have discussed mapping of advanced applications such as bitonic merge, Fast Fourier Transform (FFT), and finding the maximum value on HTN. It is shown that the number of communication steps of different advanced applications on HTN is better than other networks.

6.2 Future Works

This thesis focuses on the performance of interconnection network, the importance of other issues should be kept in mind. The issues of further exploration are as follows:

- We have evaluated the dynamic communication performance for 256-node, 512-node, and 1024-node networks. According to the interconnection philosophy of HTN, millions of nodes can be connected. We are planning to evaluate the dynamic communication performance of the HTN for Level-3 networks or more using this proposed routing algorithm.

- We limited our attention to deterministic routing algorithms and uniform traffic patterns for evaluating dynamic communication performance. However, the adaptive

routing algorithm gives several routing options and selects one of them according to the state of the network. It is automatically adjusted to compensate for network changes such as traffic patterns, channel availability, or equipment failures. Eventually, it will increase the dynamic communication performance. We are planning to use an adaptive routing algorithm to assess the performance improvement of HTN. We plan to extend our analyses to other traffic patterns.

- It has been observed that large machines tend to be less reliable than smaller ones and that parallel computers tended to be quite large. As such they could stand to benefit from fault-tolerant capabilities. Redundancy and yield of HTN is presented in this thesis but is not sufficient for fault tolerance. Therefore, we would like to develop fault-tolerant routing algorithms for an HTN with faulty nodes.

- The suitability of an interconnection network depends on how many algorithms have been optimally mapped onto the network. For instance, two dimensional interconnection networks (Mesh, Torus) are suited to image processing, computer vision, matrix computations, and computational geometry applications. We have only discussed a FFT on HTN. We would like to study some other advanced applications on HTN.

- The ability of a network topology to simulate another topology efficiently is often of considerable interest while designing parallel algorithms for a multicomputer network. The communication patterns in some parallel algorithms inherently favor certain topologies – for example, matrix operations often map naturally to a mesh network and divide-and-conquer algorithms are easily implemented on a hypercube. Thus, we plan to investigate the embedding of other frequently used topologies into the HTN.

Of course, we would like to see the Hierarchical Torus Network made fully operational and in use by the research community.

Acknowledgments

First and foremost, I wish to express my sincere and profound gratitude and profuse thanks to my supervisor Prof. Susumu Horiguchi of School of Information Science at Japan Advanced Institute Institute of Science and Technology (JAIST) for his constant encouragement and kind guidance during this research work. I am deeply grateful to Prof. Susumu Horiguchi for his invaluable discussions, fruitful comments,valuable suggestion, patient supervision, and his constructive criticism during my studies. His warm support helped me a lot.

I also owe my gratitude to Prof. Hong Shen and Prof. Yasushi Hibino for gladly agreeing to serve as members of my thesis examination committee and for providing helpful advice.

I would also like to thank Associate Prof. Toru Abe, Research Associate Ryoko Hayashi and Masaru Fukushi of JAIST for their helpful discussion, comments, and suggestions during this research work. In particular, Mr. Masaru Fukushi has given me a lot of helpful advice to reach the valuable insights in Wafer Stacked Implementation of Massively Parallel Computers. I would like to continue my sincere and heartiest thanks to Mr. Yasuyuki Miura of Communication Research Laboratory, Tokyo, Japan for his kind help during his stay in the Multimedia laboratory.

The author is also highly obliged to Prof. Hiroakira Ono of School of Information Science at JAIST for his kindness, and valuable and discussion during the research of sub-theme. I would like to take this opportunity to thank all the teaching staff at the school of Information Science at JAIST, who has benefited me with a worth of knowledge.

I would like to express my gratefulness to the Ministry of Education, Science, Sports, and Culture, Japan for providing me with the opportunity to study in Japan and conduct my research. I owe a great deal to Y. Hayakawa, H. Shima & P. Ruchira sensei of Kanazawa university and H. Sasaki & E. Horiguchi sensei of JAIST for teaching me Japanese language. Without them, I would certainly have run into much trouble.

It is my great pleasure to do research work with the members of Multimedia System laboratory. I devote my sincere thanks and appreciation to my lab-mates. I can not begin to express how much I have appreciated and their support and friendship through the years. I will keep them in my mind for their help and assistance.

Although far away in Bangladesh, I would also like to say thank to my parents, who encouraged me to succeed by teaching me not to fear failure. Last, but not least, I can never hope to thank my dear Ruba enough for her love, care, and unending support. Without the blessing of Almighty Allah and prayers of my family, I certainly would have given up long ago.

Bibliography

[1] M.J. Little, J. Grinberg, S.P. Laub, J.G. Nash, and M.W. Yung, "3-D Computer", *IEEE Int'l. Conf. on Wafer Scale Integration,* pp. 55-64, 1989.

[2] M.L. Campbell, S.T. Toborg, and S.L. Taylor, "3-D Wafer Stack Neuro-computing", *IEEE Int'l. Conf. on Wafer Scale Integration,* pp. 67-74, 1993

[3] J. Carson,"The Emergence of Stacked 3D Silicon and Impacts on Microelectronics System Integration," *IEEE Int'l Conf. on Innovative Systems in Silicon*, pp.1-8, Aug. 1996.

[4] H. Kurino, T. Matsumoto, K.H. Yu, N. Miyakawa, H. Itani, H. Tsukamoto, and M. Koyanagi, "Three-dimensional Integration Technology for Real Time Micro Vision Systems", *IEEE Int'l. Conf. on Innovative system in Silicon* pp.203-212, 1997.

[5] V.K. Jain, T. Ghirmai, and S. Horiguchi, "TESH: A New Hierarchical Interconnection Network for Massively Parallel Computing", *IEICE Transactions*, vol.E80-D, No.9, pp.837-846, 1997.

[6] V.K. Jain, T. Ghirmai, and S. Horiguchi, "Reconfiguration and Yield for TESH: A New Hierarchical Interconnection Network for 3-D Integration", *Proc. IEEE Int'l. Conf. SISI*, pp.288-297, 1996.

[7] V.K. Jain and S. Horiguchi, "VLSI Considerations for TESH: A New Hierarchical Interconnection Network for 3-D Integration", *IEEE Trans on VLSI Systems*, vol.6, No.3, pp. 346-353, 1998.

[8] S. Horiguchi, T. Ooki, and V.K. Jain, "Network Performance of TESH: A New Hierarchical Interconnection Network for 3-D Integration", *Proc. of IASTED. International Conf on Parallel and Distributed Computing and Networks*, pp. 170-175, Brisbane, Australis, 1998.

[9] Jose Duato, Sudhakar Yalamanchili, and Lionel Ni, "Interconnection Network: an Engineering Approach", *IEEE Computer Society Press*, 1997.

[10] A. Verma and C.S. Raghavendra, "Interconnection Networks for Multiprocessors and Multicomputers: Theory and Practice", *IEEE Computer Society Press*, 1994.

[11] Isaac D. Scherson and Abdou S. Youssef, "Interconnection Networks for High-Performance Parallel Computers", *IEEE Computer Society Press*, 1994.

[12] Vipin Kumar, Ananth Grama, Anshul Gupta, and George Karypis, "Introduction to Parallel Computing Design and Analysis of Algorithms", *The Benjamin/Cummings Publishing Company, Inc.*, pp.30-38, 1994.

[13] Guihai Chen, "A tutorial on Interconnection Networks," *http://cs.nju.edu.cn/ gchen/teaching/fpc/fpc99.html*

[14] L.M. Ni and D.K. Panda, "Sea of Interconnection Networks: What's Your Choice?," In *A Report of the ICPP'94 Panel*, 1994.

[15] W.C. Athas and C.L. Seitz, "Multicomputers: Message-Passing Computers," *IEEE Computer*, vo.21, No.8, pp.9-24, Aug. 1988.

[16] Y.R. Potlapalli, "Trends in Interconnection Network Topologies: Hierarchical Networks", *Int'l. Conf. on Parallel Processing Workshop*, pp. 24-29, 1995.

[17] S. Horiguchi and T. Ooki, "Hierarchical 3D-Torus Interconnection network for Massively Parallel Computers", *JAIST Research Report, IS-RR-2000-022*, pp. 1-15, ISSN 0918-7553, 2000.

[18] S. Horiguchi, "New Interconnection for massively Parallel and Distributed System," *Research Report, Grant-in-Aid Scientific Research, Project Number: 09044150*, JAIST, pp. 1-72,1999.

[19] J. Mohan Kumar and L. M. Patnaik, "Extended Hypercube: A hierarchical Interconnection Networks of Hypercubes," *IEEE Trans on Parallel and Destributed Systems*, vol.3, No.1, 1992

[20] L.N. Bhuyan and D.P. Aggarwal, "Generalized hypercube and hyperbus structurtes for a computer network," *IEEE Trans on Computers*, vol.C-33, pp. 323-333, Apr. 1984.

[21] S.B. Akers and B. Krishnamurthy, "A group-Theoretic Model for Symmetric Interconnection Network", *IEEE Trans. on Computers*, vol.38, No.4, pp.555-566, April 1989.

[22] Franco P. Preparata and Jean Vuillemin, "The Cube-Connected Cycles: A Versatile Network for Parallel Computation", *Communications of the ACM*, vol.24, No.5, pp.300-309, May 1981.

[23] Y. Yang, A. Funahashi, A. Jouraku, H. Nisji, H. Amano, and T. Sueyoshi, "Recursive Diagonal Torus: An Interconnection Network for Massively Parallel Computers", *IEEE Trans. on Parallel and Distributed Systems*, vol.12, No.7, pp. 701-715, Jul. 2001.

[24] Y. Inoguchi and S. Horiguchi, "Shifted Recursive Torus Interconnection for High Performance Computing", *Proc. HPC Asia '97*, pp. 61-66, Seoul, Korea, April,28 - May,02, 1997.

[25] A. Agarwal, "Limits on Interconnection Network Performance," *IEEE Trans. on Parallel and Distributed System*, vol.2, No.4, pp.398-411, Oct. 1991.

[26] W.J. Dally , "Performance Analysis of k-ary n-cube Interconnection Networks", *IEEE Trans. on Computers*, vol. 39, No.6, pp.775-785, June 1990.

[27] L. Gravano, G. Pifarre, P. Berman, and J. Sanj, "Adaptive Deadlock- and Livelock-Free Routing with All Minimal Paths in Torus Networks", *IEEE Trans. on Parallel and Distributed Systems*, vol.5, No.12, pp.1233-1251, June 1994.

[28] Daniel A. Reed and Drik C. Grunwald, "The Performance of Multicomputer Interconnection Networks," *IEEE Computer*, pp.63-73, June 1987.

[29] S. Horiguchi, "Wafer Scale Integration", *Proc.* 6^{th} *Int'l Microelectronics Conference*, pp. 51-58, 1990.

[30] S. Horiguchi and S. Fukuda, "A Hierarchical Redundant Cube-Connected Cycles for WSI yield enhancement", *Proc. IEEE Int'l. Conf. Wafer Scale Integration*, pp. 163-171, 1995.

[31] W.J. Dally and C.L. Seitz, "The Torus Routing Chip," *Journal of Distributed Computing,* vol.1, No.3, pp. 187-196, 1986.

[32] W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. on Computers* vol.C-36, No.5, pp. 547-553, May 1987.

[33] Jose Duato, "A new theory of Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Trans. on Parallel and Distributed Systems*, vol.4., No.12, pp.1320-1331, 1993.

[34] S.A. Felperin, L. Gravano, G. Pifarre, and J.L. Sanz, "Routing techniques for massively parallel communication." *Proc. IEEE,* vol.79, No.4, pp. 488-503, 1991.

[35] J. Upadhyay, V. Varavithya, and P. Mohapatra, "Routing Algorithm for Torus Network." *Int'l Conf. on High Performance Computing,* pp.773-778, 1995.

[36] Y.Miura and S.Horiguchi,"A Deadlock-Free Routing for Hierarchical Interconnection Network: TESH," *Proc. of the 4th Int'l. Conf. on High Performance Computing in Asia-Pacific Region*, pp. 128–133, 2000.

[37] L.M.Ni and P.K.McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *Proc of the IEEE*, Vol.81, No.2, pp. 62-76, 1993.

[38] W.J.Dally, "Virtual-Channel Flow Control," *IEEE Trans on Parallel and Destributed Systems*, vol.3, No.2, 1992

[39] Xiaoding Zhang, "System Effets of Interprocessor Communication Latency in Multicomputers," *IEEE Micro*, vol.11, No.2, pp.12-55, April 1991.

[40] H.S. azad, L.M. Mackenzie, and M.O. Khaoua, "The Effect of the Number of Virtual Channels on the Performance of Wormhole-Routed Mesh Interconnection Networks," *Proc. of UKPEW2000*, July 24-25, 2000.

[41] Kazuhiro Aoyama and Andrew A. Chien, "The cost of Adaptivity and Virtual Lanes in a Wormhole Router," *Journal of VLSI Design*, vol.2, No.4, 1995.

[42] B.M. Maziarz and V.K. Jain, "Autometic Reconfiguration and Yield of the TESH Multicomputer Network," *IEEE Trans. on Computers*, vol.51, No.8, pp.963-972, Aug. 2002.

[43] R. Muller and Q.F. Stout, "Parallel Algorithms for Regular Architectures: Meshes and Pyramids", *The MIT Press*, Chap. 5, 1996.

[44] D. Nassimi and S. Sahni, "Bitonic Sort on a Mesh-connected Parallel Computer," *IEEE Trans. on Computers*, vol.c-27, No.1, pp.2-5, Jan. 1979.

[45] Selim G. Akl, "Parallel Sorting Algorithms," *Academic Press*, vol.11, No.2, pp.17-37 & 81-108, 1985.

Publications

- **Journal:**

1. M.M. Hafizur Rahman and S. Horiguchi, "HTN: A New Hierarchical Interconnection Network for Massively Parallel Computers," *Accepted for publication in the IEICE Transactions, special issue on PDCAT2002*, Japan, Sept., 2003.

- **International Conferences:**

1. M.M. Hafizur Rahman and S. Horiguchi, "Network Performance of Hierarchical Torus Network (HTN)," *Proc. of the 3rd Int'l. Conf. on Parallel and Distributed Computing, Applications and Technologies (PDCAT02)*, pp.122-129, Kanazawa, Japan, Sept. 03-06, 2002.
2. M.M. Hafizur Rahman and S. Horiguchi, "A Deadlock-Free Routing Algorithm of Hierarchical Torus Network: HTN," *Proc. of the 6th High Performance Computing in Asia Pacific Region (HPC Asia)*, pp.114-119, Bangalore, India, Dec. 16-19, 2002.
3. M.M. Hafizur Rahman, Yasuyuki Miura and S. Horiguchi, "Dynamic Communication Performance of Hierarchical Interconnection Network: H3D-Mesh," *Proc. of the 2nd Int'l. Conf. on Electrical & Computer Engineering (ICECE)*, pp.352-355, Dhaka, Bangladesh, Dec. 26-28, 2002.

- **National Conference:**

1. M.M. Hafizur Rahman and S. Horiguchi, "Dimension Order Routing on Hierarchical Torus Networks" *Proc. of the JCHC of IEE, Japan*, pp.241, Fukui, Japan, Sept. 18-19, 2002.