Hierarchical Interconnection Networks for Massively Parallel Computers

by

M.M. Hafizur Rahman

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Professor Susumu Horiguchi

School of Information Science
Japan Advanced Institute of Science and Technology

March, 2006

Abstract

The most critical component in determining the ultimate performance potential of a multicomputer is its interconnection network. In this dissertation, we propose a new hierarchical interconnection network, called the Hierarchical Torus Network (HTN) for large scale 3D multicomputers. It consists of multiple 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 **H**ierarchical **T**orus **N**etwork **(HTN)**. To reduce the vertical links between silicon planes, we consider higher-level networks as 2D-torus instead of 3D-torus network.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 other networks. It is shown that the HTN possesses several attractive features including constant node degree, small diameter, small average distance, high arc-connectivity, better bisection width, small number of wires, a particularly small number of vertical links, and an economic layout area.

We have used wormhole routing for switching because it has low buffering requirements, and more importantly, it makes latency independent of the message distance. Deadlock-free routing is the most critical issue in wormhole-routed networks and is achieved by using virtual channels (VCs). Since the hardware cost increases as the number of VCs increases, the unconstrained use of VCs is cost-prohibitive. In this dissertation, we present a deadlock-free routing algorithm for the HTN with a minimum number of VCs. By using the dimension-order routing and various traffic patterns, we have evaluated the dynamic communication performance of the HTN as well as other networks. HTN yields low latency and high throughput, which are indispensable for high performance massively parallel computers. It is also shown that the impact of non-uniform traffic patterns on the HTN is less than on the other networks. We have also described a suite of low-cost adaptive routers, LS, CS, and LS+CS with dimension order routing, analyzed their cost, and evaluated the dynamic communication performance for the HTN. The hardware cost for the LS, CS, and LS+CS algorithms is exactly equal to dimension order routing. The only overhead imposed is router delay for header selection. The dynamic communication performance using LS+CS algorithm is better than when the other algorithms are used. Therefore, an HTN with the LS+CS algorithm is a good choice for future massively parallel computers.

A fault tolerant network is very essential for the reliability of massively parallel computer systems. We have presented a hierarchical redundancy approach to reconfigure a faulty node by redundant node for the HTN. With a 25% redundancy, the system yield at the BM and second level are satisfactory. To show the suitability of the HTN, we have discussed mapping of some commonly used advanced applications. It is shown that the number of communication steps for applications mapping on the HTN is lower than for conventional and other hierarchical interconnection networks.

Pruning technique reduces the wiring complexity. We have explored the 3D-WSI implementation aspect of pruned-HTN. It is shown that the peak number of vertical links and layout area of pruned HTN in 3D-WSI is less than that of non-pruned HTN. To show the versatility of torus-torus combination for hierarchical networks, we have modified two other hierarchical networks (H3D-torus and TESH) using torus-torus networks.

To Taha
never dull
never boring
always beloved

Acknowledgments

First of all, I would like to thank the almighty Allah (SWT) who has again blessed me to pass another important passage in my life.

I wish to express my sincere and profound gratitude and profuse thanks to my supervisor Prof. **Susumu Horiguchi** of GSIS at Tohoku university for his constant encouragement and kind guidance during this research work. I am deeply grateful to him for his enthusiasm and insight, which have made research a most enjoyable and fulfilling experience. His phenomenal depth of knowledge and ability to discern the key points of the research problem inspire me a lot. I am deeply indebted to him for his patient supervision and warm support. I am both incredibly proud and immensely lucky to become one of his students.

I wish to continue my gratitude to Prof. Y. Hibino, Prof. M. Kaneko, Prof. T. Matsuzawa, and Prof. Y. Inoguchi for gladly agreeing to serve as members of my dissertation committee and for providing helpful advice.

I am highly obliged to Prof. Yasushi Inoguchi of School of Information Science at JAIST for his kindness, and valuable suggestion 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 also like to thank Asso. Prof. X. Jiang and Research Associate Masaru Fukushi of Tohoku university and Ryoko Hayashi of Kanazawa Institute of Technology for their helpful discussion, comments, and suggestions during this research work. In particular, Mr. 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 thanks to Dr. Yasuyuki Miura of Communication Research Laboratory, Tokyo for his kind help.

I am indebted to the to the Ministry of Education, Science, Sports, and Culture, Japan for the financial support I have been provided with for staying and studying here at JAIST. I would like to continue thanks the japanese language teachers, especially Etsuko Horiguchi sensei who teach 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 firmware laboratory. I devote my sincere thanks and appreciation to my lab-mates. I can not 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 from Bangladesh, I would also like to say thanks 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 and our sweet son Taha, enough for their continuous support and encouragement. They were putting up with many dreary evenings and weekends when I was doing this research work. Their tolerance an understanding made it possible for me to complete this endeavor.

Contents

List of Figures

List of Tables

Chapter 1

"A journey of a thousands miles must begin with a single step."
– Lao-tzu (604 BC–531 BC)

Introduction

1.1 Introduction

Parallelism exists everywhere, in our society and in nature. For example, many workers in an auto factory must work together (in parallel) to produce a car; a university might have hundreds of professors and researchers simultaneously carrying out teaching and research activities; and a student, while listening to a lecture, might at the same time be thinking about how to have a fun party on the weekend. All these are instances in which more than one activity, either working or thinking, is going on in parallel.

In the case of production lines in a factory, the purpose of exploiting the advantages of parallelism by using many workers is to increase productivity or output. Two important factors will affect the output of these parallel processes. First, everyone involved must be doing some useful work simultaneously, so that the whole task can be completed in a shorter time. Second, there must be proper and efficient coordination among all workers. Without any coordination, the whole factory will in chaos, while too much coordination or inefficient coordination, will result in lower efficiency on the production lines.

Parallel computing for solving large-scale scientific and engineering problems is also characterized by these factors. The rationale for using parallelism in scientific computing is to increase the throughput of a computer by making it perform more than one operation at the same time.

A traditional sequential computer has one central processing unit (CPU) that handles basic numerical operations sequentially, i.e., one operation at a time. But the number of computations involved in large scale scientific problems is so huge that even the fastest sequential computers cannot produce solutions in a reasonable amount of time. There are basically two ways to make a computer work faster: either make a faster CPU, so that an operation will take less time to finish, or put more than one CPU in the computer so that multiple operations can be performed simultaneously. The latter is called parallel processing because there are multiple processing units working in parallel in the computer. Due to the difficulties in wire fabrication and the limit of the speed of light, it is becoming more and more difficult to build faster single-processor computers. On the other hand it is relatively easy to put a number of processors (not necessary the fastest) together such that the aggregate performance is significantly increased.

Research in parallel computing, both in hardware and software, has made dramatic progress in the last decade. However, in science and engineering, there is an apparently insatiable demand for ever-greater processing power to solve the “grand challenges”. These grand challenge problems include modeling global climate change over long periods, global ocean circulation, the evolution of galaxies, the atomic structures of materials, the efficiency of combustion in an engine, air flow over the surfaces of vehicles, damage due to impacts, pollution monitoring, the behavior of microscopic electronic devices, analysis of the human genome, and computer vision. In years to come, new grand challenges will come into view. Even the smallest of these problems requires Gigaflops (10^9 floating-point operations per second) of performance for hours at a time, while the largest requires Teraflops (10^{12} floating-point operations per second) of performance for more than a thousand hours at a time. Computations must be completed within a ‘reasonable’ time period; the definition of ‘reasonable’ can vary, but obviously, an execution time of 1 year is always unreasonable. Thus, we need not only Teraflops performance but also Petafloaps (10^{15} floating-point operations per second) or even Exafloaps (10^{18} floating-point operations per second) of performance. Achieving this level of performance requires current parallel computer technologies to be upgraded into massively parallel and distributed systems. Indeed, such computers are arguably the only feasible way of achieving the enormous computational power [1] required to solve the grand challenges of the future.

Parallel systems can be based on either a shared-memory or a distributed-memory model. In shared-memory architectures, known as multiprocessors, all processors may access a shared memory; in those based on the distributed-memory model, known as multicomputers; processors communicate by interchanging messages. The latter, in particular, have experienced rapid development during the last decade [2] because of their superior scalability. Such systems are organized as an ensemble of nodes, each with its own processor, local memory and other supporting devices, comprising a processing element (PE), along with a router or switching element (SE) which communicates with other nodes via an interconnection network. 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.

1.2 Interconnection Networks

An interconnection network is a crucial component of a multicomputer because the overall system performance is very sensitive to network latency and throughput [3, 4]. Networks employ a variety of topologies that can be classified into two broad categories: indirect and direct [4, 5]. In indirect networks, the nodes are connected to other nodes (or memory banks in a shared-memory architecture) through multiple intermediate stages of switching elements (SE). Many experimental and commercial parallel machines have employed indirect interconnection networks [5, 6], such as Hitachi SR2201 [7, 8], Cedar [9], Cray X/Y-MP, DEC’s GIGA switch and Cenju-3, IBM’s RP3 [10] and SP2 [11], Thinking Machine CM-5 [12] and Meiko CS-2. Examples of indirect networks include crossbar [7], bus [13] and multistage interconnection networks (MINs) [9].

In direct networks (also called point-to-point networks) each node has a point-to-point or direct connection to some of the other nodes (known as its neighbors) for direct communication between processors. Direct interconnection networks have been widely employed by recent machines [5]. From the scalability point of view, direct networks are preferred. Moreover, direct networks can exploit locality in traffic more effectively. Consequently, most recent multicomputers employ these networks, including the Intel iPSC [14–16], Intel Delta [17], Intel Paragon [18], Cosmic Cube [19], nCUBE [20–22], MIT Alewife [23], J-machine [24, 25] and M-Machine [26], iWARP [27], Stanford DASH [28], Stanford FLASH [29], Cray T3D [30], Cray T3E [31], and SGI Origin [32]. In this study we focus on direct interconnection networks. In this dissertation, henceforth, "interconnection network" will refer to a "direct interconnection network" unless otherwise specified.

It is important to design parallel computers using an interconnection network topology that can scale up to a large number of processors and that is capable of supporting fast communication and data sharing among processors. Several aspects of interconnection networks have been investigated. Much of the early literature deals with their topological and functional properties. As systems with thousands of processing elements began to emerge, many other aspects such as implementation and wiring complexity, flow control, routing algorithms, performance evaluation, and fault tolerance issues became important along with the topological properties. An interconnection network [5, 33, 34] is described by its topology, routing algorithm, and flow control mechanism. To design a parallel computer efficiently using an interconnection network, we need to study the following issues.

Network Topology: Interconnection networks are composed of a set of shared nodes and channels, and the topology of the network refers to the arrangement of these nodes and channels. The topology of an interconnection network is the arrangement of its nodes and channels in a graph; analogous to a roadmap . The channels (like roads) carry packets (like cars) from one router node (intersection) to another. 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 [35] depends on the routing and switching technique employed in their interconnection network.

Routing Algorithm: Once a topology has been chosen, there can be many possible paths (sequences of nodes and channels) that a message could take through the network to reach its destination. Routing determines which of these possible paths a message actually takes. The routing algorithm [5, 33, 36] employed by a network determines the path taken by a packet from a source node to its destination node. In some networks there is only a single route from each source to each destination, whereas in others, there are many possible paths. When there are many paths, a good routing algorithm balances the load uniformly across the channel regardless of the offered traffic pattern [6]. Continuing our roadmap analogy, while the topology provides the roadmap, the roads and intersections, the routing method steers the car, making the decision on which way to turn at each intersection. Just as in routing cars on a road, it is important to distribute the traffic – to balance the load across different roads rather than having one road become congested while parallel roads are empty.

Switching Method: An efficient and fast switching technique is a basic requirement for getting good performance from an interconnection network. Switching is the mechanism that moves data from an input channel and to an output channel. Network latencies are highly dependent on switching technique. Wormhole (WH) switching (also known as wormhole routing) [4, 37, 38] has become the dominant switching technique used in contemporary multicomputers. This is because it has low buffering requirements, and more importantly, it makes latency independent of the message distance [5, 6]. In WH routing, a message is divided into flits (flow control units) for transmission and flow control, and each channel buffer need only be one flit in length. The first flit of a message, the header flit, includes the routing information; it is followed by the data flits in pipelined fashion. If the header cannot be routed in the network due to contention for resources (buffers and channels), the data flits are also blocked in situ, keeping all the allocated buffers occupied. In WH routing it is impossible for flits of other packets to cross the path of the current packet.

Flow Control: Flow control manages the allocation of resources to packets as they progress along their route. The key resources in most interconnection networks are the channels and buffers. Channels are used to transport packets between nodes. Buffers are storage devices implemented within the nodes, such as registers or memories, which allow the packet to be held temporarily at the nodes. Continuing our analogy: the topology determines the roadmap, the routing method steers the car, and the flow control controls the traffic light, determining when a car can advance over the next stretch of road (channels) or when it must pull off into a parking lot (buffer) to allow other cars to pass. To realize the performance potential of the topology and routing method, the flow control strategy must avoid resource conflicts that can hold a channel idle [6].

Since wormhole routing relies on a blocking mechanism for flow control, deadlock can occur because of cyclic dependencies over network resources during message routing. A good flow control strategy should reduce congestion, be fair, and avoid deadlock [6]. Virtual channels [38, 39] were originally introduced to solve the problem of deadlock in wormhole-routed networks. Flow control concerns techniques for dealing with contention among multiple messages for the same channels and buffers [40]. Flow control techniques are very dependent on the switching scheme employed [41]; in wormhole routing, the most common flow control strategy is the use of virtual channels.

Fault-Tolerance: Massively parallel computer systems usually have stringent reliability requirements 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 [33] networks are essential to the reliability of massively parallel computer systems. Interconnection networks are often composed of thousands of components – nodes, channels, routers, and connectors – that collectively have failure rates higher than is acceptable for the application. High availability or reliability is usually achieved by some form of fault-tolerance, which enables the system to survive the loss of one or more components without disruption in its operation. The specific fault-tolerance methods used in the system play an important role in the availability, reliability, and the performance of the overall system. A fault-tolerant network has the ability to route information even if certain network components fail.

The failure of a network component can often bring down the entire system, unless adequate measures are provided to tolerate such failures. Faults can occur in a massively parallel computer system in the PE, memory, I/O system, or the router and channels of the interconnection network, and to be effective a fault tolerant scheme must address each of these subsystems. The techniques 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 to tolerate a certain number of faults.

Algorithms and Applications: The interconnection network used in a massively parallel computer system plays a key role in determining how fast applications can be executed on the system. Any non-trivial parallel program running on a multiprocessor requires some amount of information exchange among the processors; in many cases, the time for such communication is a significant part of the overall running time of the program. Therefore, it is important to design algorithms and applications to use the interconnection network efficiently.

Performing a computational problem efficiently on a multicomputer network is a complex task, even when the parallelism in the problem has already been identified [33]. Some of the problems that arise in this context are: distributing shared data among the nodes in a shared-memory machine so that they can be accessed in parallel without conflicts; allocating processes (or computations) to nodes to match their communication needs to the interconnection network; designing parallel algorithms so that their communication requirements can be efficiently supported by the underlying network; balancing the load among the processors in the system; and so on. It should be noted that most of these problems arise 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.

Most parallel algorithms are designed so that they have some regular types of data movement among nodes. These data movement may involve frequently-used permutations such as shuffle or matrix transpose, global communication operations such as broadcasting, or it may be expressed in terms of more elementary operations such as finding the maximum value, sorting, etc. How fast such a communication operation can be executed on an interconnection networks determines the performance of the parallel algorithm on that multiprocessor. Thus, the suitability of a given interconnection network for certain applications can often be estimated by studying how efficiently these common operations, such as sorting, computing the maximum value, bitonic merge, divide-and-conquer, Fast Fourier Transform (FFT), and broadcasts, can be performed on the given network.

Implementation: 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 wafer-stacked integration has been proposed as a new technology for massively parallel computers. Little *et al.* [43] developed a 3D computer consisting of a 32×32 cellular array 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). Implementation of a wafer-stacked prototype 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.* [44] and Carson [45]. Recently, Kurino *et al.* [46] have suggested a highly practical 3D construction technology.

A major obstacle in the design of future 3D computers is the cost in terms of the area required for vertical interconnects. Each vertical interconnect has an area of $300\mu m \times 300\mu m$. Thus, the unconstrained use of interconnects is not cost-effective in parallel computer implementation. 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) [47–50], and they concluded that a hierarchical interconnection network minimizes the vertical links for efficient 3D wafer-stacked integration.

1.3 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 advantages of parallelism for solving 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 the parallel computer that can be constructed. Likewise, the attainable performance of a parallel computer is ultimately limited by the characteristics of the interconnection network. Clearly, one of the critical design issues for parallel computers is the interconnection network, which is the backbone for these computers.

Hundreds of various types of interconnection networks have been proposed in past decades. No single network is optimal in all aspects of network performance. Designing new interconnection networks remains a topic for intensive investigation, given that there is no clear winner among those that currently exist. Careful designers 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; 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 that will challenge human beings in the twenty-first century.

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

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

Hierarchical interconnection networks [51, 52] have attracted considerable attention in the research community 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. The large diameter of conventional topologies is completely infeasible for very large scale systems, while hierarchical interconnection networks [51] are a cost-effective way to interconnect a large number of nodes. A variety of hyper-cube based hierarchical interconnection networks such as Folded Cube [53], Twisted Cube [54], Extended Cube [55], Enhanced Cube [56], Reduced Cube [57], Generalized Hypercube [58] , and so on have been proposed, but for large scale multicomputer systems, the number of physical links in these systems becomes prohibitively large. To alleviate this problem, several k-ary n-cube-based hierarchical interconnection networks such as TESH [47–50], H3D-Mesh [59, 60], H3D-torus [61, 62], and Cube Connected Cycles (CCC) [63] have been proposed. However, the dynamic communication performance of these networks is still very low, especially in terms of network throughput.

It has already been shown that a torus network has better dynamic communication performance than a mesh network [64]. This is the key motivation that led us to consider a hierarchical interconnection network, in which both the basic module and the interconnection of higher levels have toroidal interconnections. Therefore, the *first goal* of this dissertation is to propose a new hierarchical interconnection network consisting of torus networks.

Although the theoretical foundations for constructing large interconnection networks for massively parallel computer system have existed for a long time, the state of hardware technology has not allowed their cost-effective realization until the last decade. However, recent advances in VLSI technology have overcome this drawback, achieving VLSI systems on stacked silicon planes [65, 66]. On a stacked silicon plane, part of a massively parallel computer is implemented and these silicon planes are then interconnected. Jain *et al.* [47–50] have pointed out that hierarchical interconnection networks are suitable for 3D wafer-stacked integration. Hence, the *second goal* of this dissertation is to analyze the proposed network's performance on 3D wafer-stacked integration.

In massively parallel computers, an ensemble of nodes works in concert to solve large application problems. The nodes communicate data and coordinate their efforts by sending and receiving messages in the multicomputer through a router, using a routing algorithm. Efficient routing, featuring low latency and high throughput, is critical to the performance of interconnection networks, and achieving the low latency and high throughput are indispensable for high-performance massively parallel computers. Therefore, the *third goal* of this dissertation is to evaluate the dynamic communication performance of the proposed network using both deterministic and adaptive routing algorithms. Adaptive routing allows paths to be chosen dynamically based on router status [67–71]. We will assess the improvement of the dynamic communication performance of our proposed network, using a suite of low cost adaptive routing algorithm, over the performance of a dimension order routing algorithm.

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

One of the desirable attributes of designing interconnection network is the convenient mapping of applications, especially those with regularity of computations, in the designed network [72]. Therefore the *fifth goal* of this dissertation is to investigate how efficiently some common and advanced applications such as bitonic merge, finding the maximum value, and FFT can be mapped on our proposed interconnection network.

The wiring complexity of an interconnection network is the total number of links in the overall network. The wiring complexity of the system is an important issue, since the silicon area is limited and networks are generally wiring intensive [73, 74]. The links of a richly connected network can be removed in a periodic fashion by pruning, for reduced complexity and, hence, increased performance. Therefore, the *sixth goal* of this dissertation is to prune our proposed network and analyze its 3D-WSI realization.

Finally, we will modify some other hierarchical interconnection networks using our main idea (torus-torus hierarchical interconnection network) and evaluate both their static network performance and dynamic communication performance. In conclusion, the main goal of this dissertation is to develop and analyze an efficient hierarchical interconnection network for massively parallel computer systems.

1.4 Contribution of the Dissertation

In this dissertation, we have proposed a new hierarchical interconnection network called the Hierarchical Torus Network (HTN). The basic idea behind this new network is that it is a hierarchical interconnection network, in which each level is connected by toroidal interconnections.

The contributions of this dissertation 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 issue for HTN is also described.

- Provide a deadlock-free dimension order routing algorithm for HTN. Virtual channels are used to achieve a deadlock-free routing algorithm. Since the hardware cost increases as the number of virtual channels increases, the unconstrained use of virtual channels is not cost-effective in parallel computers. A deadlock-free routing algorithm for our proposed HTN with a minimum number of virtual channels is investigated.

- Simulate the dynamic communication performance of the HTN as well as for several commonly used networks for parallel computers using dimension order routing. We have conducted several experiments and compared its dynamic communication performance with that of several other networks, to demonstrate the superiority of the HTN.

- Propose a suite of low-cost adaptive routing algorithms for the efficient use of physical links and virtual channels, to improve the dynamic communication performance of the HTN. We present a conservative estimate of hardware cost and delay for the proposed routing algorithms and compare it with that of a dimension order routing algorithm.

- Prove the freedom from deadlock of the proposed adaptive routing algorithms, using 3 virtual channels. To evaluate the dynamic communication performance of an HTN we used the proposed adaptive routing algorithms under various traffic patterns and compared the resulting performance with that of the dimension order routing algorithm.

- Present the fault tolerance aspects of HTN. Tolerating faults is the key to system survival. Derive a theoretical estimate of system yield for the HTN as a function of defect density with a reconfiguration approach by hardware redundancy.

- Investigate the versatility of the HTN by mapping some primitive applications such as bitonic merge, finding the maximum value, and FFT on it and comparing its mapping performance to that of other conventional and hierarchical interconnection networks.

- Prune the proposed HTN to reduce the wiring complexity, and analyze its 3D-WSI implementation.

- Modify some other hierarchical interconnection networks using our main idea (torus-torus hierarchical interconnection network) and analyze the cost-performance trade-offs.

1.5 Synopsis of the Dissertation

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

- Before proposing a new hierarchical interconnection network, it is very important to study existing network topologies, as along with studying how they can be changed into hierarchical interconnection networks. In **chapter 2**, we deal with various network topologies and provide an overview of the properties of many widely-used interconnection networks. Different hierarchical interconnection networks are also addressed in **Chapter ??**. This chapter is the literature survey and presents the current state-of-the-art of k-ary n-cube based hierarchical interconnection networks.

- In **Chapter 3**, we present a new hierarchical interconnection network called Hierarchical Torus Network (HTN). In this chapter, we describe the architectural details of the HTN, including addressing of nodes. This chapter discusses 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 the 3D wafer-stacked implementation for our proposed HTN.

- Because interconnection networks are used for communication and coordination between nodes, routing is probably the most important problem in analyzing interconnection networks. **Chapter 4** presents an in-depth analysis of the routing algorithm of our proposed HTN. We describe deadlock free dimension order routing and a suite of low-cost adaptive routing algorithms for the HTN. An investigation of the minimum number of virtual channels required for deadlock-free routing in an HTN is also described. The dynamic communication performance of an HTN is evaluated using dimension-order routing and our proposed adaptive routing algorithms under various traffic patterns. We present a conservative estimate of hardware costs and router delay for the proposed adaptive routing algorithms and compare it with a dimension order routing algorithm.

- The interconnection network is a critical component of a multicomputer system and must therefore be designed with some degree of fault-tolerance. In **Chapter 5**, the reconfiguration of faulty nodes by redundant nodes is presented for the HTN. A hierarchical redundancy approach is explored, in which redundancy is provided at each level of the network. An expression for yield is presented considering the redundant circuit and estimate the yield of the HTN. For a network to be useful, it must accommodate a large class of applications. At the end of **Chapter 5**, we discuss some advanced applications such as bitonic merge, FFT, and finding the maximum value on HTN.

- Next, in **Chapter 6**, we prune our proposed HTN to reduce the total number of physical links. The architecture of the pruned HTN and its 3D wafer-stacked implementation issue are discussed in detail.

- **Chapter 7** contains some modified hierarchical interconnection networks based on our main idea (torus-torus hierarchical interconnection network). The basic structure, addressing of nodes, routing of messages, static network performance, and dynamic communication performance of those modified networks are presented in this chapter.

- In **Chapter 8**, we conclude this dissertation with some perspectives and outline some directions for future work, considering areas that might be worthy of further research.

Chapter 2

"Science is a wonderful thing if one does not have to earn ones' living at it"
– Albert Einstein (1879–1955)

Interconnection Networks for Massively Parallel Computers

2.1 Introduction

Network topology refers to the static arrangement of channels and nodes in an interconnection network – the road over which the packet travel. Selecting the interconnection network topology is the first step in designing a multicomputer network because routing strategy and flow-control method depend heavily on the network topology. A roadmap is needed before a route can be selected and the traversal of that route scheduled.

Interconnection networks are used in massively parallel computers to allow cooperation between several nodes (processing elements). The research literature on interconnection networks embodies studies of a large number of interconnection networks, ranging from a simple bus to hierarchical interconnection networks. They have been proposed in terms of their graph theoric properties. Interconnection networks can be broadly divided into two classes: direct and indirect. In the direct (or static) networks, point to point links interconnect the nodes according to network topology. In the indirect (or dynamic) network nodes are not directly connected; the communication between any two nodes has to be carried through some switches. In this dissertation, we have concentrated only on direct networks.

In a message-passing multicomputer [35], 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. These nodes may have different functional capabilities. For example, the set of nodes may contain vector processors, graphics processors, and I/O processors. 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. Although the function of a router can be performed by the local processor, dedicated routers have been used in high-performance multicomputers, allowing overlapped computation and communication within each node. As the number of nodes in the system increases, the total communication bandwidth, memory bandwidth, and processing capability of the system also increase, Thus, direct networks have been a popular interconnection network architecture for massively parallel computers. In this dissertation, henceforth, unless specified, the term computer or processor refers to a node.

Direct network topologies are derived from graphs in which nodes represent processors and edges represent the dedicated links between processors. A network is symmetric if it is isomorphic to itself with any node labeled as origin. Almost all direct networks studied in the literature have some degree of symmetry. Such a symmetric network has many advantages. First, it allows the network to be constructed from simple building blocks and expanded in a modular fashion. Second, a symmetric network facilitates the use of simple routing algorithms. Third, it is easier to develop efficient computational algorithms for multiprocessor interconnected by a symmetric network. Finally, it makes the network easier to model and analyze.

An important advantage of many regular static interconnection networks is their modularity. The degree of a node in these networks either remains fixed regardless of the size of the network, or grows very slowly with network size (e.g. as a logarithmic function). This allows very large networks to be constructed from simple building blocks. Due to their regularity and modularity, direct interconnection networks are often used in massively parallel computers. Some of these networks have been used in commercial multicomputers or research prototype. Table 2.1 lists the types of interconnection network employed in several representative commercial and experimental machines.

Machine	Network Topology
Connection Machine CM-5 [12]	Fat-Tree
Connection Machine CM-2	Hypercube
Intel iPSC-2 [15, 16]	Hypercube
nCUBE [20–22]	Hypercube
SGI Origin 2000 [32]	Hypercube
Intel Paragon [18]	2D mesh
Intel Touchstone Delta [17]	2D mesh
MIT Reliable Router [75]	2D mesh
Stanford DASH [28]	2D mesh
MIT J-Machine [24, 25]	3D mesh
MIT M-Machine [26]	3D mesh
Michigan HARTS	Hexagonal mesh
KSR first level ring [5]	Ring
Illinois Illiac IV	2D torus
Chaos Router [76]	2D torus
iWARP [27]	2D torus
Cray T3D [30]	3D torus
Cray T3E [31]	3D torus
Tera Computer	Incomplete 3D torus

Table 2.1: A collection of types of interconnection networks used in commercial and experimental parallel computers

The main focus of this dissertation is on hierarchical interconnection networks for massively parallel computers. This is why this chapter provides a survey of various types of interconnection networks which have been proposed through the years and pointing out their potential shortcomings. The rest of this chapter is organized as follows: Section 2.3 outlines the architecture and properties of the most common interconnection network topologies. Hierarchical interconnection networks are the subject of Section 2.4. Finally, conclusions are pointed out in Section 2.5.

2.2 Definitions

In distributed-memory architectures, processing elements are linked together by an interconnection network. In this section, we define the terminology which will be used throughout the discussion of interconnection networks.

2.2.1 Fundamental Definitions

Definition 2.1 ***(Node)*** *A node is that basic functional unit from which a massively parallel computer system is constructed. Nodes consist of its processing element, local memory, other supporting devices, and a router.*

Definition 2.2 ***(Communication Link)*** *A communication link is a direct connection from one node to another node. Communication links may unidirectional or bidirectional.*

Definition 2.3 ***(Buffers)*** *Buffers are the memory reserved at each node for data elements which require transmission over the network. Injection buffers hold data waiting to enter the network (from local memory). Delivery buffers hold data waiting to leave the network (to local memory).*

Definition 2.4 ***(Interconnection Network)*** *An interconnection network is a graph in which the vertices are nodes and the edges are communication links. The edges are directed if the communication links are undirectional; they are undirected if the links are bidirectional.*

These definitions form the basis for the discussion of interconnection networks. In this dissertation, we will often use the term *node*, *link*, and *network* as shorthand for processing element, communication link, and interconnection network. We now define some topological characteristics of interconnection networks.

2.2.2 Topological Characteristics of Interconnection Networks

The definitions in this section are well known characteristics of interconnection networks. More details about the topological characteristics will be provided in Chapter 3.

Definition 2.5 ***(Degree)*** *The degree of a network is the maximum number of links originating from any node.*

Definition 2.6 ***(Distance)*** *The distance between a pair of nodes is the smallest number of links that must be traversed to get from one node to the other.*

Definition 2.7 ***(Diameter)*** *The network diameter is the maximum distance between any pair of nodes.*

Definition 2.8 ***(Bisection Width)*** *The bisection width of the network is the minimum number of links that must be removed to split the network into two equal halves.*

A low degree for each node (i.e., a small number of connections) is a requirement for scalability. A small diameter is desirable because it is the lower bound for worst-case node-to-node communication time. A large bisection width is desirable because it defines the maximum bandwidth available between two halves of the network. These features will be used later to assess the benefits of different interconnection networks.

2.2.3 Layout Characteristics of Interconnection Networks

While the topological characteristics provide useful information with respect to routing algorithm about the underlying graph for a network, layout characteristics provide insight into fabrication cost and scalability.

Definition 2.9 ***(Maximum Edge Length)*** *The maximum edge length (wire length) is the longest link between a pair of nodes when the network is laid out in a plane (or in a 3D volume)*

Definition 2.10 ***(Network Area (or Volume))*** *This is the minimum area (or volume) consumed by the network when laid out in a plane (or in a 3D volume)*

Definition 2.11 ***(Link Width)*** *The link width is the width in bits of each link connecting a pair of nodes.*

In VLSI and other technologies, a small maximum edge length is desirable in order to avoid long signal propagation delays. Similarly, a small area is desirable to limit fabrication costs. For a fixed fabrication cost, the number of nodes (and the achievable parallelism) is higher for networks with a smaller network area, assuming fabrication cost is proportional to area.

2.2.4 Dynamic Communication Performance Metrices

The dynamic communication performance of an interconnection network depends on the routing algorithm. We now define several metrices for mrasuring the performance of a routing algorithm.

Definition 2.12 ***(Latency)*** *Latency is the time a packet spends in the network, excluding the time it waits in the injection and delivery buffers. Latency is measured in clock cycles.*

Definition 2.13 ***(Throughput)*** *Throughput describes the rate of data transfer. It is the number of flits accepted by delivery buffers per cycle per node, at steady state.*

In general, latency and throughput are used to describe the performance of routing algorithms in an interconnection network.

2.3 Interconnection Network Topologies

The choice of a particular interconnection network for a massively parallel computer system is based on various factors. One must begin by considering the types of problems for which it will be used. If a specific problem is targeted, an ideal network may exist. However, if the goal is to solve any problem, a more general network may be appropriate.

Depending on the scalability requirements, the topological and layout characteristics of the network can create performance and fabrication cost constraints. Obviously, the degree of scalability required also depends on the computational capacity required for the system and the power of the individual processing elements.

Of course, the routing algorithm for an interconnection network will also affect the decision. We will discuss the routing algorithm details in Chapter 4.

In this section, we present some important interconnection networks used in massively parallel computers and discuss their properties. There are many more networks than we present here, so we will limit our descriptions to some of the more common ones.

2.3.1 Completely-Connected Networks

In a *Completely-connected network* [77], each node is directly connected to every other node. For an N node network, each node has degree $N-1$. The diameter is 1 and the bisection width is $\left\lceil \frac{N}{2} \right\rceil \times \left\lfloor \frac{N}{2} \right\rfloor$. The small diameter and the large bisection width are very desirable. But the high degree of each node makes these networks impractical for more than a few nodes (an N node network has $N \times (N-1)$ unidirectional links. As a result, sparse networks are used for all large scale parallel computers. Figure 2.1 shows some examples of completely connected networks.

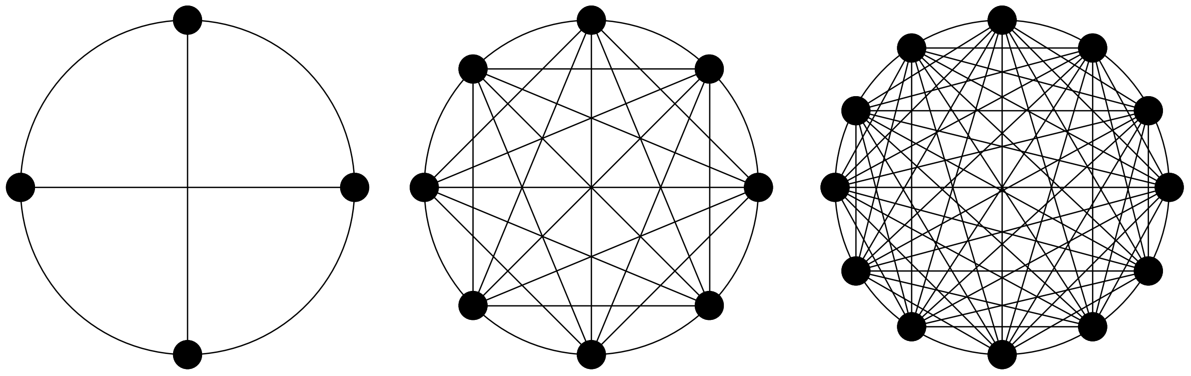


Figure 2.1: Completely-connected networks for $N=4$, $N=8$, and $N=12$.

2.3.2 Star Networks

Star-Connected Network

In a *Star-connected network* [77], 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

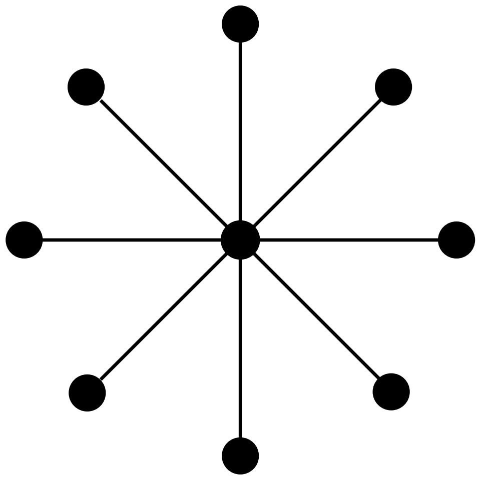


Figure 2.2: A star-connected network of nine nodes

network of nine nodes. For an N node network, the terminal nodes are only linked to the central node so their degree is 1. The central node is linked to all the other processors, so its degree is $N-1$. The diameter and bisection width are 2 and 1, respectively. Communication between any pair of nodes is routed through the central node. Hence, the central node is a bottleneck in the star-connected network. The bottleneck at the central node makes it impractical to have many nodes. Another, more serious, problem is that if the central node fail, the whole network fails, along with all access to peripherals.

Star Graph Network

A *star graph* [78, 79] 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 $N = n!$ nodes and node degree is equal to $(n-1)$. The diameter is $\left\lfloor \frac{(3n-1)}{2} \right\rfloor$. Figure 2.3 shows a star graph obtained by permutations of four symbols. The degree and the diameter of the star graph grow at a much slower rate with the number of nodes. It is vertex and edge symmetric, maximally fault tolerant, and strongly resilent. The main disadvantage of the star graph is that the routing of message in the star graph is more complex.

2.3.3 Tree Networks

A *tree network* [5, 77] 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 of these nodes is in turn connected to a disjoint set of descendants. A node with no descendants is called leaf node. A characteristic property of trees is that every node but the root has a single parent node. Therefore, tree network contains no cycles. The connectivity for a complete binary tree with $N = 2^d - 1$ nodes is:

$$i \rightarrow \left\{ \ (2i, 2i+1) \qquad \text{for} \ \ 0 \leq i < (2^d - 1) \right. \tag{2.1}$$

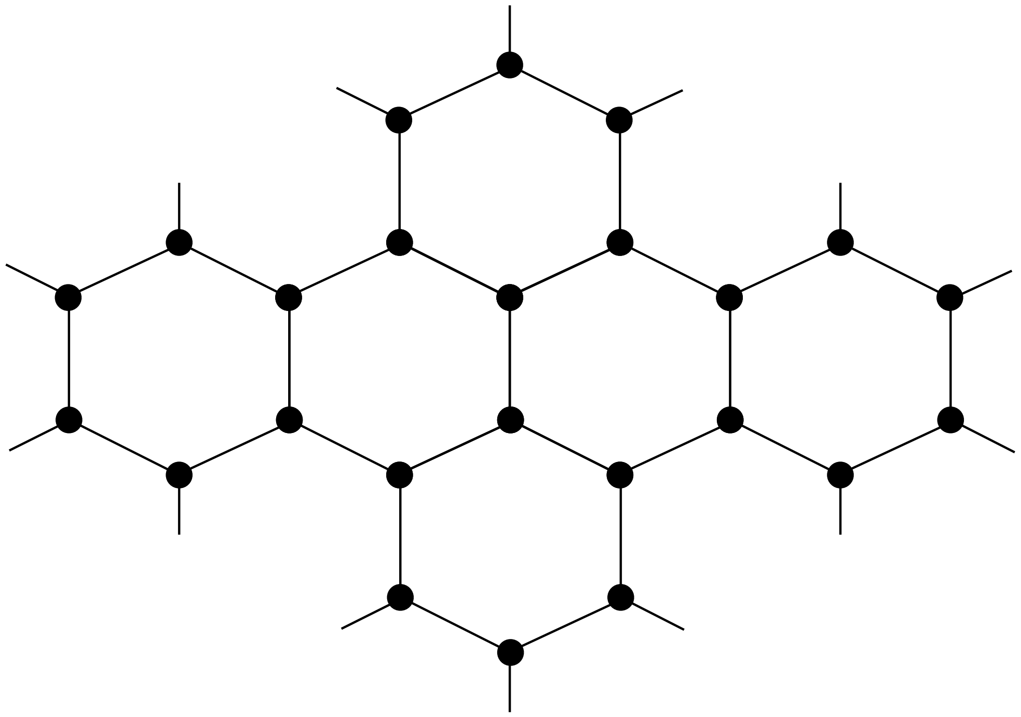


Figure 2.3: Star graph network

Figure 2.4 shows a binary tree network with 15 nodes. Networks based on trees can be used to achieve diameters which are logarithmic in N. The diameter of a binary tree network is $\left(2 \times \left\lfloor \log_2^N \right\rfloor\right)$. Unfortunately the bisection width is 1, making it difficult to move data across the network. However, the network used in an early parallel computer DADO is based on the binary-tree network [77].

A related network, the *X-tree* network, improves the bisection width to $\left\lceil \log_2^N \right\rceil$ by connecting the nodes at each level, as shown in Figure 2.5. It also improves nearest-neighbor communication, for some cases. The connectivity is:

$$i \rightarrow \begin{cases} (2i, 2i+1) & \text{for } \; 0 \leq i < (2^d - 1) \\ (i+1) & \text{for } \; 2^j \leq i < (2^{j+1} - 1), 1 < j < d) \end{cases} \tag{2.2}$$

Several other networks are also based on binary tree. The *mesh-of-trees* network [80, 81] is formed by placing binary trees on the top of each row and column of a mesh and then deleting the original edges of the mesh. Meshes of trees have both small diameter and large bisection width. Figure 2.6 provides an example of mesh-of-tree. The *tree-of-mesh* [80] is a binary tree with the nodes replaced by meshes and the edges replaced by links between the nodes of the meshes.

The most important drawback of tree networks is that the root node and the nodes close to it suffers from a communication bottleneck. Additionally, there are no alternative paths between any pairs of nodes. The bottleneck can be removed by allocating a higher channel bandwidth to channels located close to the root node. The shorter the distance to the root node, the higher the channel bandwidth. However, using channels with different

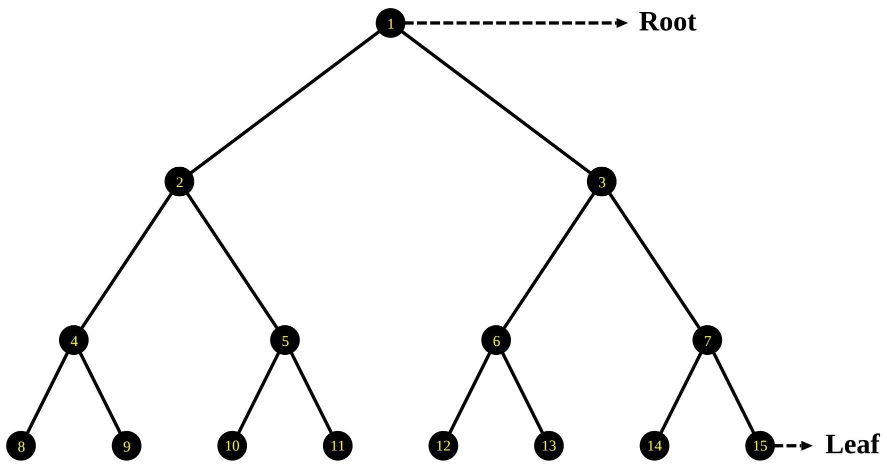


Figure 2.4: A 15 node binary tree network

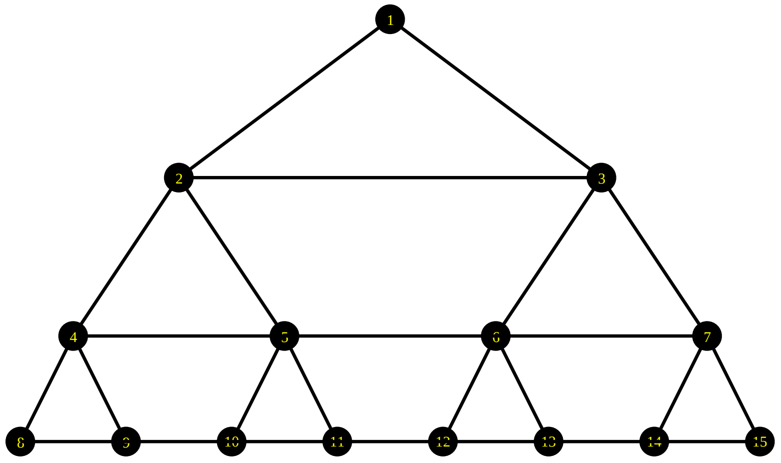


Figure 2.5: A 15 node X-tree network

bandwidth is not practical, specially when message transmission is pipelined. *Fat-tree* [82] , based on the tree-of-meshes network, offer a good solution to the communication bottleneck by providing more link bandwidth closer to the root of the tree. Also, fat-trees are area universal, meaning that a fat-tree which uses VLSI layout area A can emulate any other network using the same area with at most polymorphic slowdown. Figure 2.7 provides an example of a binary fat tree. The network used in the Connection Machine CM-5 is based on the fat-tree [12].

2.3.4 Hypercubic Networks

The hypercubic family of networks provides both a small diameter and a large bisection width. All of the hypercubic networks are derived from the binary cube[1]. The hypercubic networks have been studied for several decades. In [83], Schwartz reviews the early literature on these networks. In [81], Leighton provides an excellent descriptions of the

[1]Binary cubes are also known as *boolean cubes* or *hypercubes*

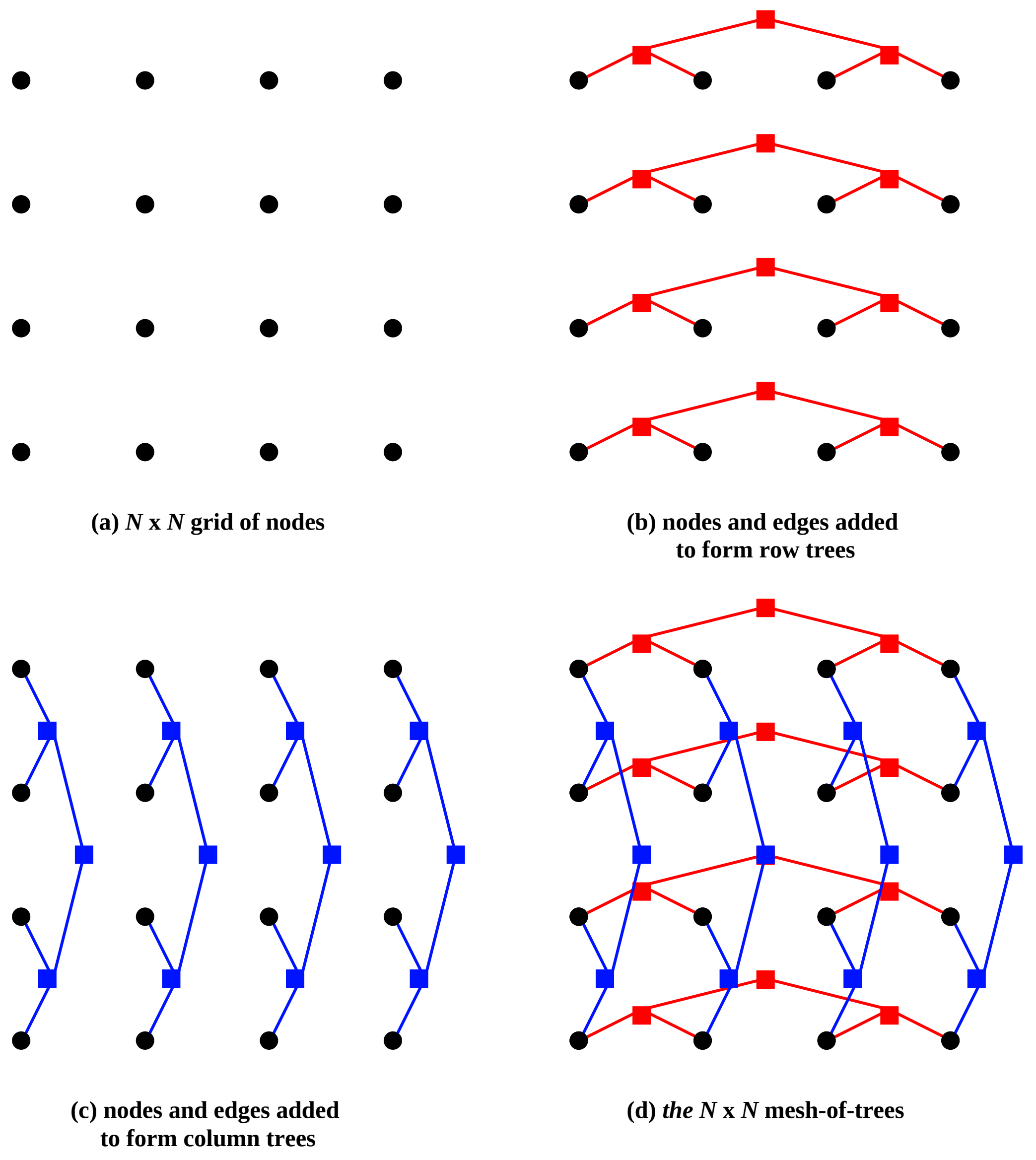

(a) *N* x *N* grid of nodes
(b) nodes and edges added to form row trees
(c) nodes and edges added to form column trees
(d) *the N* x *N* mesh-of-trees

Figure 2.6: The 2D mesh-of-trees. Leaf nodes from the original grid are denoted with black circles. Nodes added to form row trees are denoted with red squares, and nodes added to form column trees are denoted with blue squares

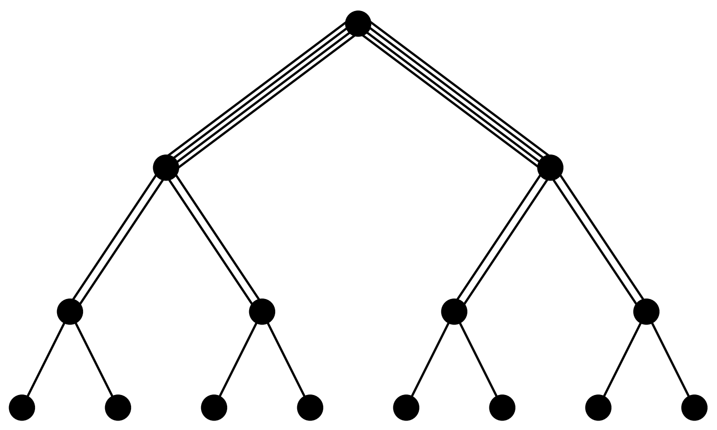


Figure 2.7: A fat-tree network

major types of hypercubic networks and their properties.

The Binary Cube Network

Generally, the *binary cube* network is well known as *hypercube* network. It is one of the most versatile and efficient networks yet discovered for parallel computation. It is well suited for both special-purpose and general purpose tasks, and it can efficiently simulate any other network of the same size. The n-dimensional binary cube (hypercube) [58, 77, 83] has $N = 2^n$ nodes with the following connectivity:

$$i \rightarrow \left\{ \; (i \oplus 2^j) \qquad \text{for } \; 0 \leq i < 2^n, \, 0 \leq j < n \right. \tag{2.3}$$

The symbol $\oplus$ denotes the bitwise exclusive-OR operation. Nodes are labeled using n-bit binary numbers 0 to $2^n - 1$. Two nodes are connected by a direct link if and only if the binary representation of their labels differ at exactly one bit position. A binary cube can be constructed recursively. A zero-dimensional binary cube is a single node; a one-dimensional binary cube is constructed by connecting two zero-dimensional binary cube; a two-dimensional binary cube is constructed by connecting two one-dimensional binary cube, and so on. In general, a $(n + 1)$-dimensional binary cube is constructed by connecting two d-dimensional binary cube. Figure 2.8 illustrates the binary cubes of dimension zero to four.

The key advantages of the binary cube are its small diameter $(\log_2^N)$ and its large bisection width $\left(\frac{N}{2}\right)$. Again binary cube allows efficient implementation of a large number of parallel algorithms. However, it has several drawbacks. The degree of the nodes is $(\log_2^N)$, and although it grows fairly slowly with N, the degree is not bounded. Also, the large number of links per node tend to limit the link width, as compared to mesh and torus networks. Lastly, it requires large fabrication area, so scaling the network to large sizes is very costly.

Despite difficulties involved in the fabrication of binary cubes, several systems have been based on this architecture. These include systems from Intel IPSC [14, 15], nCUBE [20–22] and Thinking Machines (CM-1, CM-2)

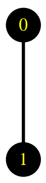
Remark: Note that the n-dimensional binary cube is actually an k-ary n-cube with the extent of each dimension equal to two, i.e., it is 2-ary n-cube network.

In order to circumvent the difficulties associated with the node degree in binary cubes, several variants of hypercube have been devised that have similar computational properties but bounded degree. These are the butterfly, the cube-connected cycles, and the Benes network.

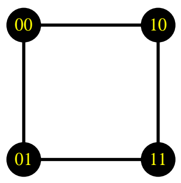
The Cube-Connected Cycles Network

The *cube-connected cycles (CCC)* [63] has fixed degree and preserves many of the properties of the binary cube. It can be constructed from the binary cube by replacing each node with an n node ring, resulting in an $n2^n$ node network. Each node in in the ring is linked to a different dimension. The nodes are labeled by a pair consisting of the original binary cube node label and the position of the ring. The connectivity is:

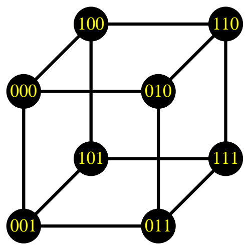
(a) 0-D Hypercube



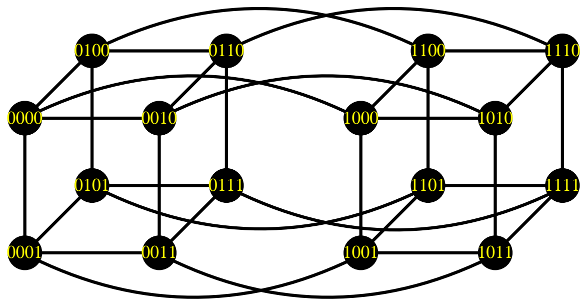
(b) 1-D Hypercube



(c) 2-D Hypercube



(d) 3-D Hypercube



(e) 4-D Hypercube

Figure 2.8: The Binary cube networks of zero, one, two, three, and four dimensions, the nodes are labeled using n-bit binary numbers.

$$(i, j) \rightarrow \begin{cases} (i \oplus 2^j, j) & \text{for } \; 0 \leq i < 2^n, \, 0 \leq j < n \\ (i, (j \pm 1) \bmod n) & \text{for } \; 0 \leq i < 2^n, \, 0 \leq j < n \end{cases} \tag{2.4}$$

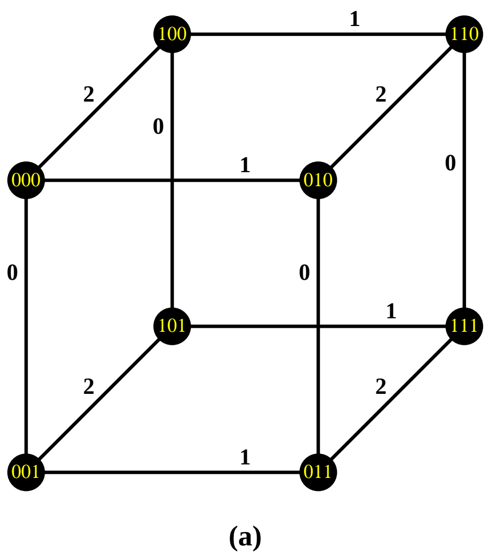
Figure 2.9 illustrates an example of the CCC network and the corresponding binary cube network. In this figure, two links are connected to neighbors in the ring, and one link is connected to a node in another ring through one of the dimensions of the binary cube (hypercube). The diameter of the n-dimensional CCC network is $2n - 1 + \left\lfloor \frac{n}{2} \right\rfloor$, and the bisection width is $\frac{N}{2}$.

Other Hypercubic Network

There are many other hypercubic networks. One class, shuffle-type networks, includes *shuffle-exchange* and *de-Bruijn* networks. A larger class, butterfly-type networks, includes the *omega*, *flip*, *baseline*, *banyan*, and *delta* networks, as well as several butterfly variants. These are indirect networks, which is out of scope of this dissertation.

2.3.5 Array Networks

Array networks are the simplest networks for parallel computation. They support a rich class of interesting and important parallel algorithms. Low dimensional arrays are a



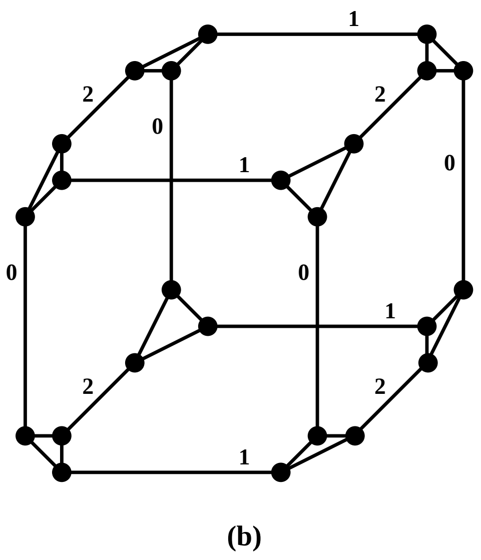


Figure 2.9: (a) The 3-dimensional binary cube network (b) The 3-dimensional CCC. Labels for individual nodes in the CCC are binary cube node label and the adjacent link label.

common network because they scale well and provide a natural mapping for many data structures.

Linear Array and Ring Networks

A *linear array* [77] is the simplest sparse network. Each node (except the nodes at the ends) has a direct communication link to two other neighboring nodes. Nodes are linked together in a straight line. The interconnection rules for an N node linear array is:

$$i \rightarrow \begin{cases} i+1 & \text{for } \ i=0 \\ i\pm 1 & \text{for } \ 0<i<N-1 \\ i-1 & \text{for } \ i=N-1 \end{cases} \tag{2.5}$$

The degree, diameter, and bisection width for an N node linear array are 2, $(N-1)$, and 1, respectively. The network scale well, but the diameter is large and the bisection width is small. The linear array is inherently unreliable as a failure of a single node or link disconnects the network. By linking the first and last nodes by an wraparound link, we can form a *ring* [77] network . The interconnection rules for an N node ring becomes:

$$i \rightarrow \left\{ \ (i\pm 1) \bmod N \qquad \text{for } \ 0 \le i < N-1 \right. \tag{2.6}$$

Here, the degree, diameter, and bisection width for an N node ring network are 2, $\left(\left\lfloor \frac{N}{2} \right\rfloor\right)$, and 2, respectively. The diameter of a ring network is smaller and the bisection width is larger than those of linear array network; however, the diameter is still large and the bisection width is small. Unfortunately, the large diameter and small bisection width

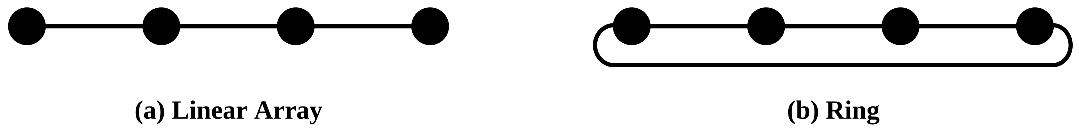
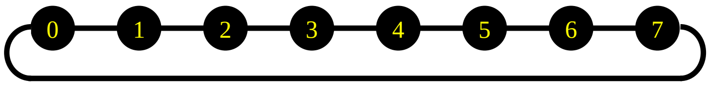


Figure 2.10: A four-node linear array and ring network



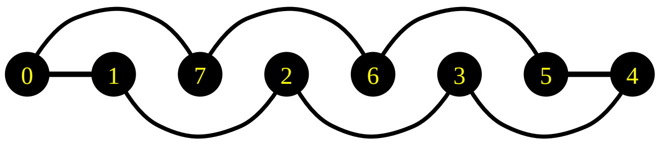


Figure 2.11: A layout for a ring network which minimizes link lengths ($N = 8$).

are still significant drawbacks for this type of network. Figure 2.10 shows an example of linear arrays and rings. The ring is employed in KSR 1st-level ring [5].

A potential problem with ring networks is signal propagation time. The obvious layout shown in Figure 2.10 will have one link with length proportional to the number of nodes. One could set up the nodes in a circle, but this too may become impractical as N increases. A better solution is shown in Figure 2.11, which limits the length of links without affecting scalability.

n–Dimensional Mesh and Torus Networks

Networks based on linear arrays and rings suffer from a large diameter and a small bisection width. These problems can be reduced by extending the network to more than one dimension In fact, linear arrays and rings are actually special cases of the more general n–dimensional mesh and torus networks.

The n*–dimensional mesh* [84] is defined by a set of extents, $k_0, k_1, k_2,$ ... ..., k_{n-1} and has $N = k_0 \times k_1 \times k_2 \times$ ... ..., $\times k_{n-1}$ nodes. Nodes are labeled by n-tuples, with values corresponding to node's offset in each dimension with respect to node $(0, 0, ..., 0)$. Node $X = (x_0, x_1, ..., x_{n-1})$ is valid node if $0 \leq i < k_i$, for $0 \leq i < n$. In dimension i, $0 \leq i < n$, the interconnection for node X is:

$$(x_0, x_1, ..., x_i, ..., x_{n-2}, x_{n-1}) \rightarrow \begin{cases} (x_0, x_1, ..., x_{i+1}, ..., x_{n-2}, x_{n-1}) & \text{if } \; x_i < k_i - 1 \\ (x_0, x_1, ..., x_{i-1}, ..., x_{n-2}, x_{n-1}) & \text{if } \; x_i > 0 \end{cases} \tag{2.7}$$

If we assume the extents of all of the mesh's dimensions have the same even value (i.e., $k_i = k$, for $0 \leq i < n$), then the k^n node mesh network will have a diameter of $n(k-1)$ and a bisection width of k^{n-1}.

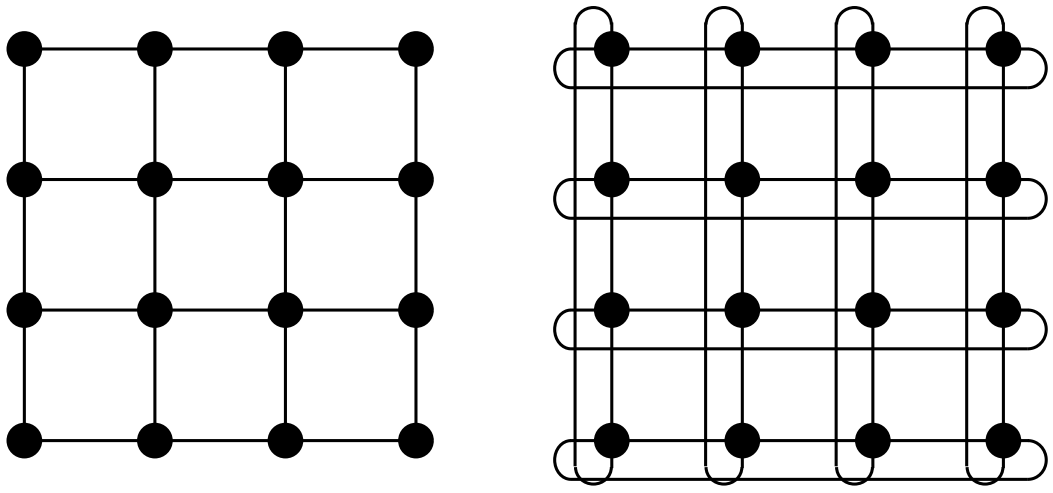


Figure 2.12: 2D mesh and torus networks with 4 nodes in each dimension (a) 2D-mesh (b) 2D-torus networks

The nodes at the periphery in the mesh network are connected by wraparound links. Such a mesh network is called a *wraparound mesh* or a *torus* network [6, 77]. Thus, the n–dimensional torus is identical to the mesh, except for some additional connectivity. Unlike the mesh, with these wraparound links, every node is connected to its $2n$ neighbors. In dimension i, $0 \leq i < n$, the interconnection for node X is:

$$(x_0, x_1, ..., x_i, ..., x_{n-2}, x_{n-1}) \rightarrow \begin{cases} (x_0, x_1, ..., (x_{i+1}) \bmod k_i, ..., x_{n-2}, x_{n-1}) \\ (x_0, x_1, ..., (x_{i-1}) \bmod k_i, ..., x_{n-2}, x_{n-1}) \end{cases} \tag{2.8}$$

If we assume the extents of a torus all have the same, even value (i.e., $k_i = k$, for $0 \leq i < n$), then the k^n node torus network will have a diameter of $\left(n \times \frac{k}{2}\right)$ and a bisection width of $(2 \times k^{n-1})$.

Figure 2.12 and 2.13 provide some examples of mesh and torus networks. In theory, higher dimensional mesh and torus networks should be very desirable, at least from the perspective of diameter and bisection width. However in practice, engineering constraints such as the network area (or volume) and the link width become a problem at higher dimensions. As a result, mesh and torus networks with $n > 3$ are not common in parallel system, with the exception of network architectures based on the binary cube. Many commercially available parallel computers are based on low dimensional mesh and torus networks. These include 2D-mesh such as Intel Paragon [18] and Touchstone Delta [17], MIT Reliable Router [75], and Stanford DASH [28]; 3D-mesh such as MIT J-Machine [24, 25] and M-Machine [26]; 2D-torus such as Chaos Router [76] and CMU iWARP [27]; and 3D-torus such as Cray T3D [30] and Cray T3E [31].

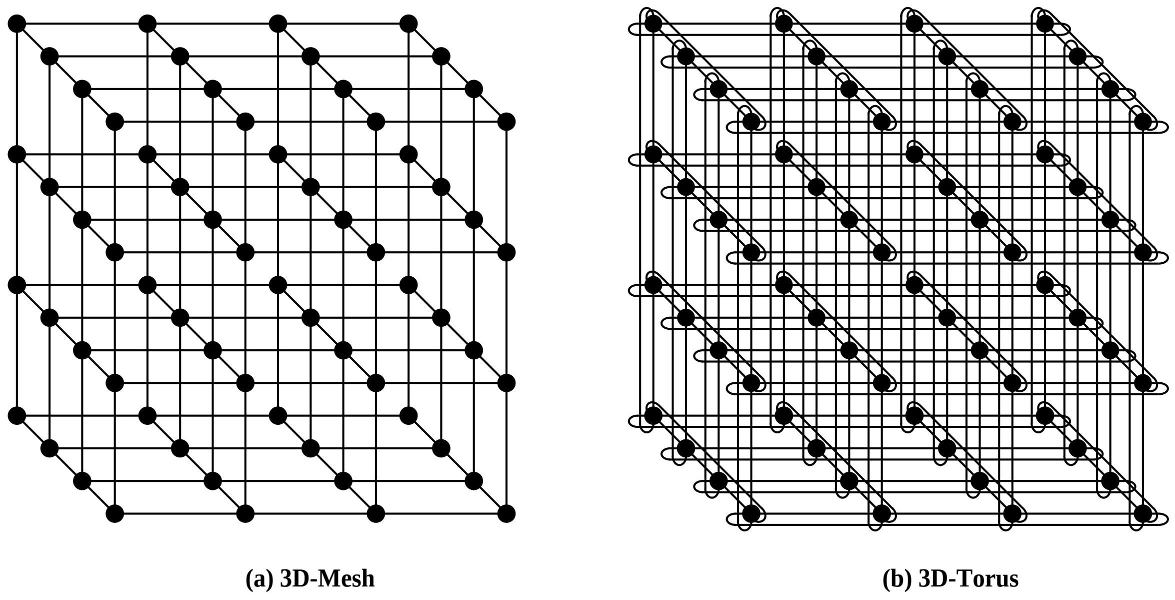


Figure 2.13: 3D mesh and torus networks with 4 nodes in each dimension (a) 3D-mesh and (b) 3D-torus networks

k-ary n-cube Networks

A k-ary n-cubes [64] 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 $N = k^n$. Torus networks 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 [85]. In recent literature, the k-ary n-cube refers to the n-dimensional torus with k nodes in every dimension. k-ary n-cubes have many desirable topological properties including ease of implementation, modularity, symmetry, low diameter and node degree, plus an ability to exploit locality exhibited by many parallel applications [86]. k-ary n-cubes are suited to a variety of applications including matrix computation, image processing and problems whose task graphs can be embedded naturally into the topology [87]. Figure 2.12(b) and 2.13(b) represent 4-ary 2-cube and 4-ary 3-cube networks, respectively. A binary n-cube[2] is also an example of a k-ary n-cube with the extent of each dimension equal to 2.

2.4 Hierarchical Interconnection Network (HIN)

Interconnection networks usually suffer from Little's Law: low cost implies low performance and high performance is obtained at high cost [51]. However, hierarchical interconnection networks [51] provide high performance at low cost by exploring the locality that exists in communication patterns of massively parallel computers. A hierarchical interconnection network (HIN) provides a plausible alternative way in which several topologies

[2]2-ary n-cube is well known as hypercube.

can be integrated together. HINs have attracted considerable attention in the research community during the past few years as a means of communication for multicomputer systems. They take advantage of the locality of reference in the communication pattern. For massively parallel computers with millions of nodes, the total number of physical links and large diameter of conventional topologies are completely infeasible. Hierarchical interconnection networks [51, 52] are a cost-effective way to interconnect a large number of nodes. They are also suitable for 3D wafer-stacked implementations.

The HIN can be seen as an attempt to combine the advantages of various conventional interconnection network topologies together. We have classified the conventional networks as completely-connected, array, tree, and hypercubic networks. It is very hard to categorize the HINs like the conventional ones because it combines a variety of networks together. However, we have classified the HINs as completely-connected network based HIN, array based HIN, tree based HIN, and hypercube network based HIN.

In this section, we present some important HINs for massively parallel computers and discuss their properties. There are many more HINs than we present here, so we will limit our descriptions to some of the more common ones.

2.4.1 Completely-Connected Network based HIN

Multi-level Fully-Connected Networks

A *Multi-level Fully-Connected (MFC)* networks [88] begin with N_2 identical copies of a nucleus with N_1 nodes, where $N_2 \leq N_1 + 1$ and the nucleus may be any connected-network or hyper-network. Each nucleus copy is viewed as a level-2 cluster and connected to each other level-2 cluster via at least one inter-cluster link. The resultant network is called level-2 MFC network. To construct a level-L MFC network, we use N_L identical copies of a level-$(L-1)$ MFC network with $\prod_{i=1}^{L-1}(N_i)$ nodes as level-L clusters, and connect each level-L cluster to each of the other level-L clusters via at least one level-L inter-cluster link, where $N_L \leq \prod_{i=1}^{L-1}(N_i) + 1$. According to this recursive construction, an arbitrarily large MFC networks using any type of nucleus can be constructed. A level-L MFC network based on nucleus G is denoted by MFC(L, G), in particular, MFC$(1, G)$ is the nucleus G. The MFC is superior over fully-connected network, however, the wiring complexity is still very high. Figure 2.14 illustrates a level-2 MFC network with $N_1 = N_2 = 8$.

Swapped Network

A major characteristics of *swapped network* [89] is that the address of each neighbor of a node is obtained by swapping two equal-length bit-strings in the node address. The use of bit-string swapping as the rule for connectivity establishes swapped networks as a sub-class of multi-level fully-connected (MFC) networks.

The swapped network $Sw(G)$, derived from the n-node nucleus or basis graph G, is an n^2-node graph with n copies of G (clusters) numbered 0 to $n-1$, so that node i in cluster j $(N_{i,j})$ is connected to node j of cluster i $(N_{j,i})$ for all $i \neq j$ and $0 \leq i, j \leq n-1$.

A swapped network is characterized by its nucleus graph, number of hierarchical levels, number of clusters in each level, and the number of links connecting each pair of clusters. A swapped network that has L levels and uses the graph G as its nucleus is called a G-based level-L swapped network, and is denoted by $SN(L, G)$. The nucleus or basis graph

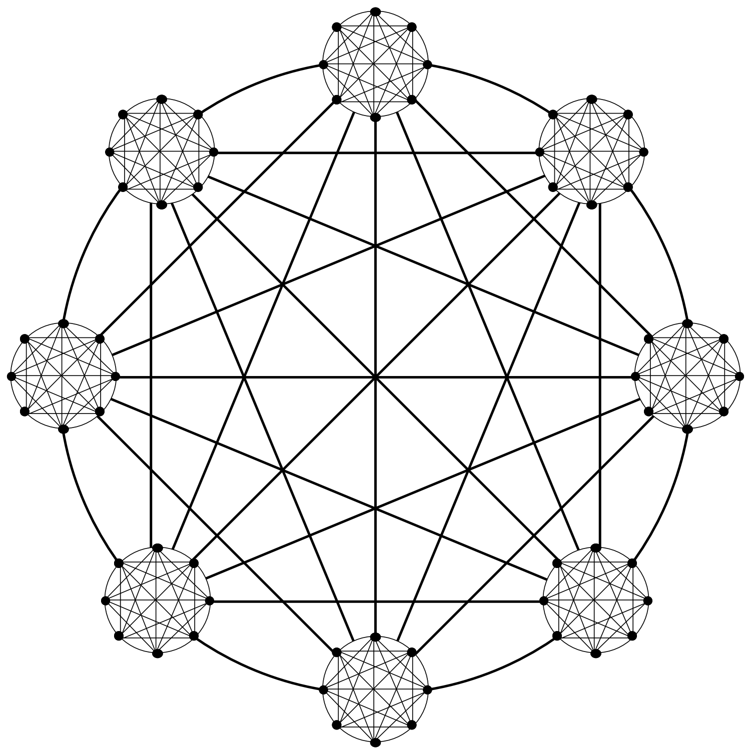


Figure 2.14: A level-2 MFC network with 8 clusters and the cluster size is 8.

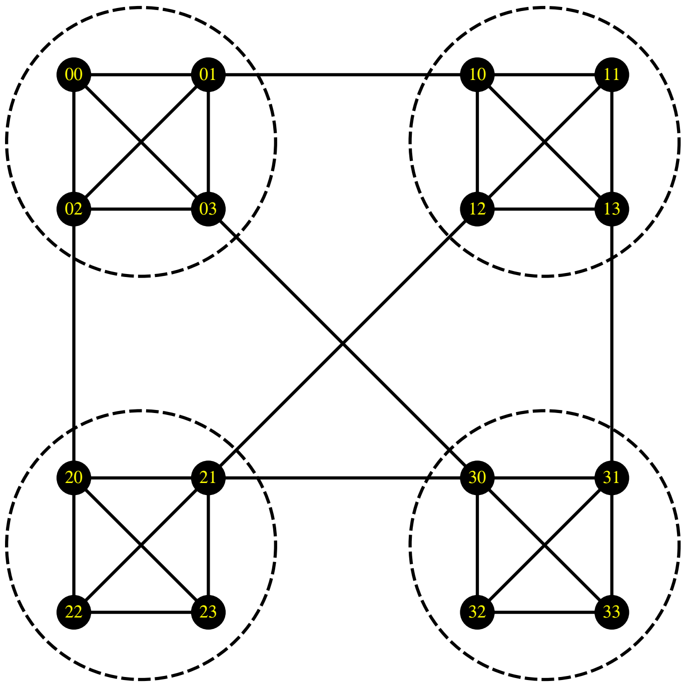


Figure 2.15: An example of 16-node swapped network with the 4-node complete graph as its basis.

may be any kind of network such as hypercube, completely-connected network, mesh, and etc. Figure 2.15 shows a 16-node swapped network with the 4-node complete graph as its basis.

2.4.2 Tree Network based HIN

Pyramid Network

A *pyramid network* [90] is a 4-ary tree where each level is connected as a mesh network. It is an attempt to combine the advantages of mesh networks with those of tree networks. A pyramid network of size k^2 is a complete 4 ary rooted tree of height $d(=\log_2^k)$ augmented with additional interprocessor links so that the nodes in every tree levels form a 2D-mesh network [91]. A pyramid of size k^2 has at its base a 2D-Mesh network containing k^2 nodes. The total number of nodes in a pyramid of size k^2 is $N = \frac{4k^2-1}{3}$ organized in $d+1$ levels. The levels of the pyramid are numbered in ascending order such that the base has level number 0, and the single processor at the apex of the pyramid has level number d. Every interior node is connected to nine other nodes: one parent at level-$(i+1)$ (provided that $i \leq d-1$), four mesh neighbors at the same level, and and four children at level-$(i-1)$ (provided that $i \geq 1$). Figure 2.16 illustrates a pyramid network of size 16.

The node degree is constant and it is 9. The advantage of the pyramid over the 2D-mesh is that the pyramid reduces the diameter of the network. When a message must travel from one side of the mesh to the other, fewer link traversal are required if the message travels up and down the tree rather than across the mesh. The diameter of a pyramid of size k^2 is $2\log_2^k$. The addition of tree links does not give the pyramid a significantly higher bisection width than a 2D-mesh network. The bisection width of a pyramid of size k^2 is $2k$. In addition, the length of the longest link in the pyramid network is an increasing function of the network size.

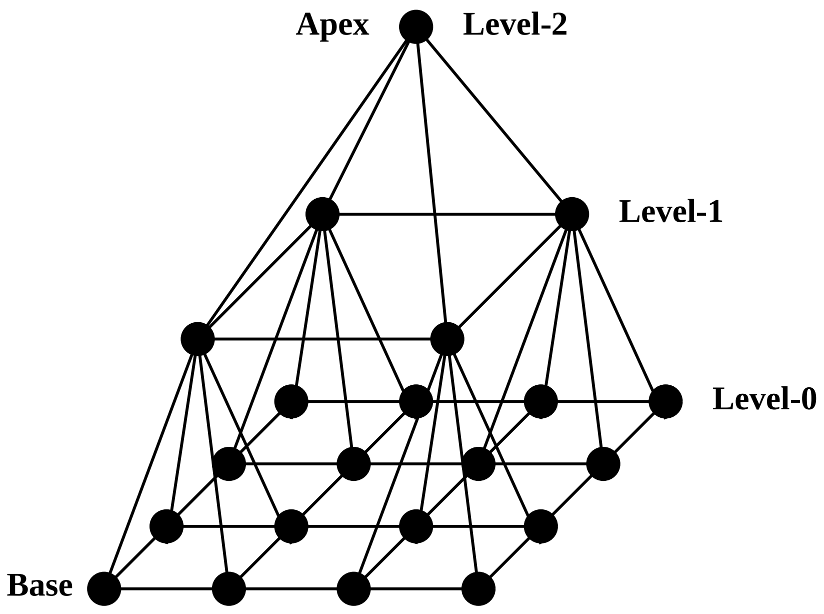


Figure 2.16: A pyramid network of 16 node

Hierarchical Cliques Network

Hierarchical Cliques (HiC) network [92] incorporates positive features of the completely-connected network and tree network. It is a k-ary tree, modified to enhance local connectivity in a hierarchical, modular fashion.

$HiC_{(k,h)}$ is a k-ary tree of height h modified so that groups of nodes on the same level form cliques. Figure 2.17 illustrates the structure and addressing scheme of an HiC with $k = 4$ and $h = 3$. The root node has 4 children which form a clique; in Figure 2.17 nodes of a clique are shown enclosed in dotted oval. The key features of the hierarchical cliques are low degree, low diameter (logarithmic), self routing, versatile embedding, good fault tolerance, and strong resilience. However, the upper level networks are still congested because of tree network.

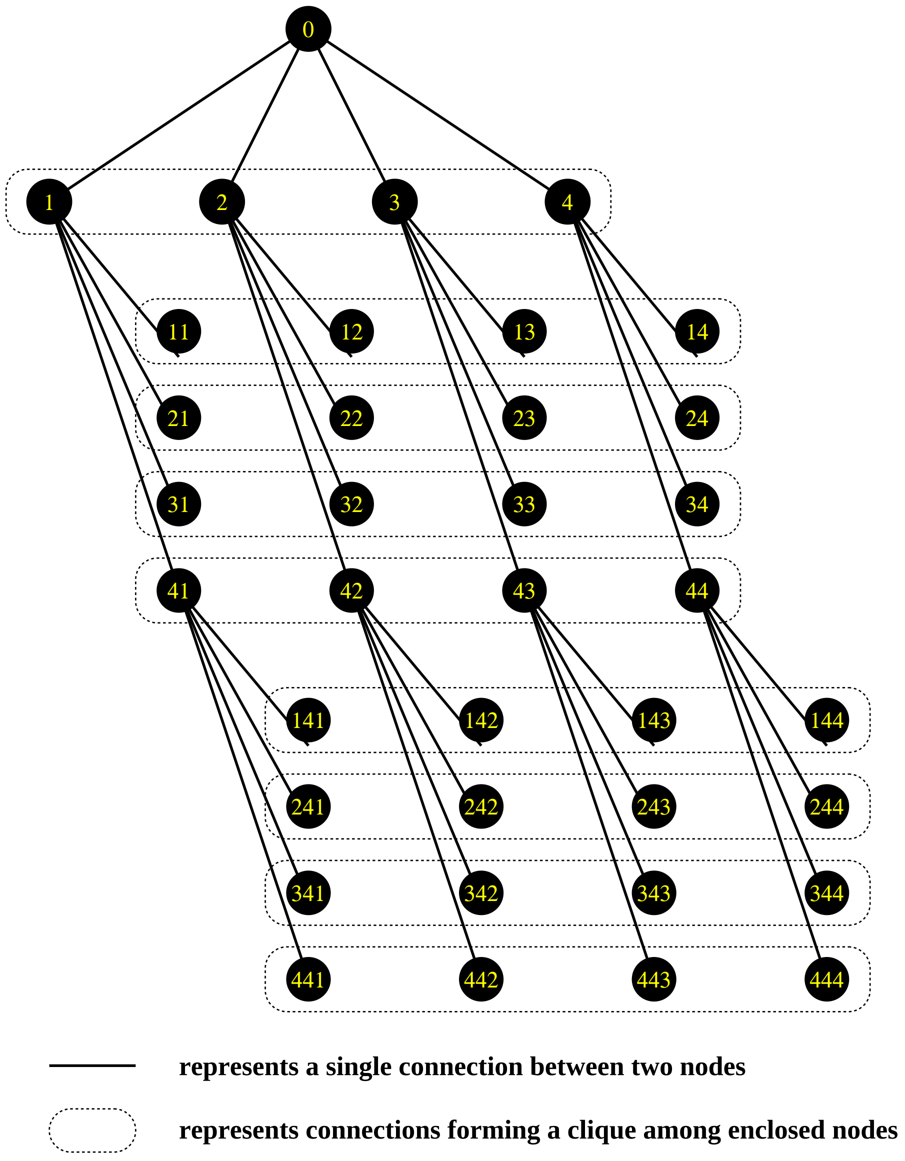

0
1
2
3
4
11
12
13
14
21
22
23
24
31
32
33
34
41
42
43
44
141
142
143
144
241
242
243
244
341
342
343
344
441
442
443
444
represents a single connection between two nodes
represents connections forming a clique among enclosed nodes

Figure 2.17: A hierarchical clique network

2.4.3 Hypercube Network based HIN

Fibonacci Cube

Fibonacci cube [93] is a type of incomplete hypercube networks. They are subgraphs of hypercube networks induced by nodes that have no two consecutive 1's in their binary representation. More precisely, a Fibonacci cubes Γ_n of dimension n is an undirected network of f_n nodes, each labeled by $n-2$ binary numbers such that no two 1's occur consecutively (where f_n is the n-th Fibonacci number defined as follows: $f_0 = 0$, $f_1 = 1$, and $f_n = f_{n-1} + f_{n-2}$ for $n \geq 2$). Two nodes are connected if and only if their labels differ in exactly one bit position. Fibonacci cubes Γ_n is a network recursively connecting two disjoint subgraphs Γ_{n-1} and Γ_{n-2}. Each node in Γ_{n-2} is connected to a counterpart node in Γ_{n-1}. As the basis, Γ_0 is an empty graph and Γ_1 is a graph with a single node. Figure 2.18 illustrates the interconnection of Fibonacci cubes.

Fibonacci cube has very attractive recurrent structure which are essential in developing fault-tolerant schemes. It contains about $\frac{1}{5}$ fewer links than the hypercube for the same number of nodes. Thus it reduces the wiring complexity of hypercube. Therefore, it requires less layout area for VLSI/WSI implementation. One potential shortcoming of the Fibonacci cube is that (unlike the hypercube) its node degrees are not homogeneous, which results implementation difficulties. The asymmetry in structure and relative sparsity in connections, the communication delay is higher than that of the hypercube network.

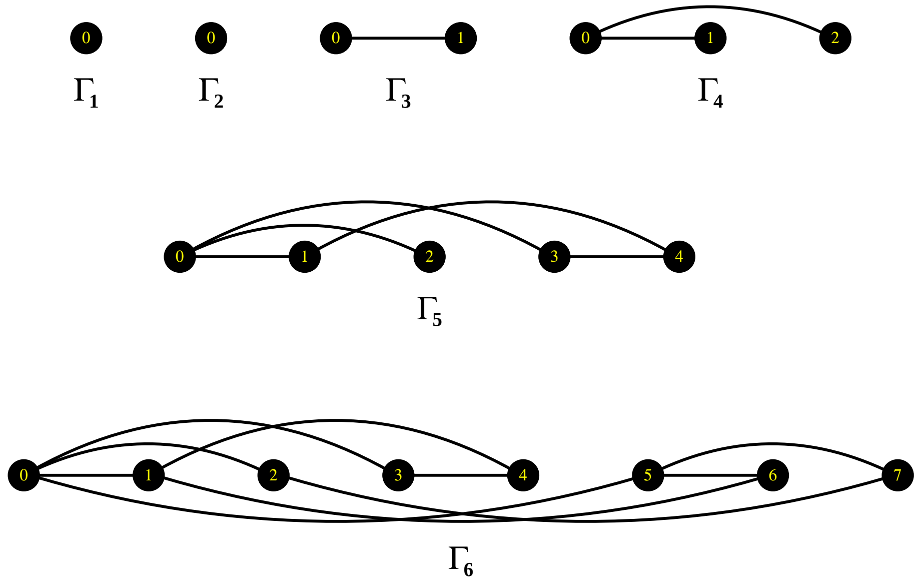


Figure 2.18: Fibonacci cubes

Hierarchical Cubic Networks

To resolve the non-planarity and inability to grow incrementally, a new class of interconnection network, called *Hierarchical Cubic Networks (HCN)* [94] has been proposed.

An HCN uses hypercubes as basic modules (BM). BM also referred to as a cluster. The HCN(n, n) has 2^n clusters, where each cluster is a n-dimensional hypercube. Each node in the HCN(n, n) has $(n + 1)$ links connected to it. Among these, n links (local links) are used within the cluster to form the hypercube. The additional link (external link) is used to connect the clusters. Each node is represented using a pair of numbers (I, J), where I is a n-bit cluster number, and J is a n-bit address of the node within a cluster. Clusters are interconnected using the external links to form the HCN(n, n) using the following rules:

```
for $I = 0$ to $(2^n - 1)$ do
  for $J = 0$ to $(2^n - 1)$ do
      if $(I \neq J)$ then
          connect $((I, J)(J, I))$;
          /* non-diameter link */
      else
          connect $((I, I)(\bar{I}, \bar{I}))$;
          /* diameter link */
```

$\bar{I}$ is the bitwise complement of the n-bit value of I. The external link between nodes (I, I) and $(\bar{I}, \bar{I})$, where $0 \leq I \leq (2^n - 1)$, is called a diameter link. An external link that is not a diameter link is called non-diameter link. Figures 2.19 depicts a HCN$(2, 2)$ network.

The HCN has about three-fourths the diameter of a comparable hypercube, although the number of links per node is about half of the hypercube. However, the total number of links $(2^{2n-1} \times (n + 1))$ of the HCN is still very high for thousands of nodes.

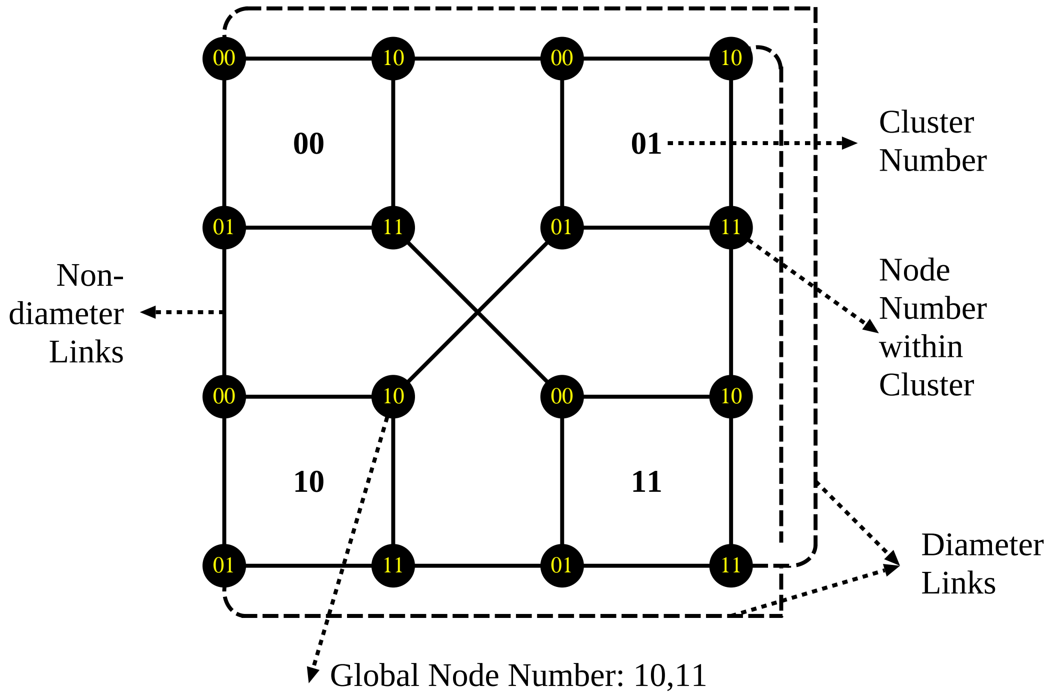


Figure 2.19: A HCN(2,2) network

2.4.4 Array Network based HIN

Recursive Diagonal Torus Network

The name *Recursive Diagonal Torus (RDT)* [95] 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 2D square mesh (torus) will serve as the basis of RDT. Figure 2.20 shows an RDT network structure. It is a class of network which makes use of the advantages of mesh (tori) structures and greatly improves the performance of meshes (toruses) when the number of nodes reaches tens of thousands of nodes.

The RDT possesses several attractive features including a small diameter that is almost half that of the hypercube. Although the congestion on upper rank torus sometimes degrades the performance under random traffic, the RDT provides much better dynamic communication performance than that of the 2D/3D torus in most cases even under hot-spot traffic. The main drawback of the RDT is that due to presence of diagonal links, it is difficult to implement a fault-tolerant network.

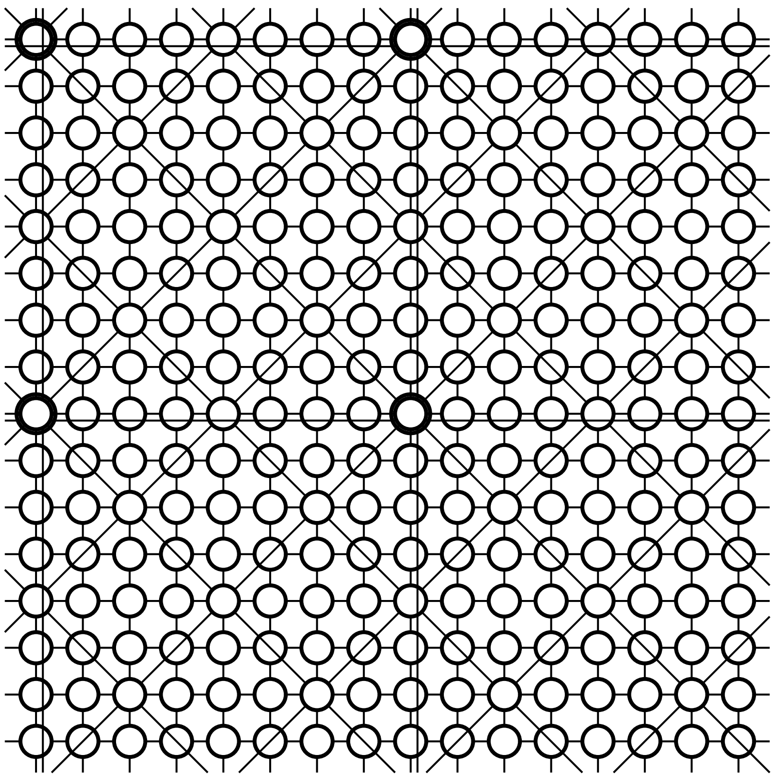


Figure 2.20: Recursive diagonal torus network

Shifted Recursive Torus Network

The RDT has a simple architecture and its dynamic communication performance is high. However, the wiring of the RDT is complex and it is difficult to implement a fault tolerant RDT due to diagonal links. To reduce the wiring complexity and to achieve a simple fault

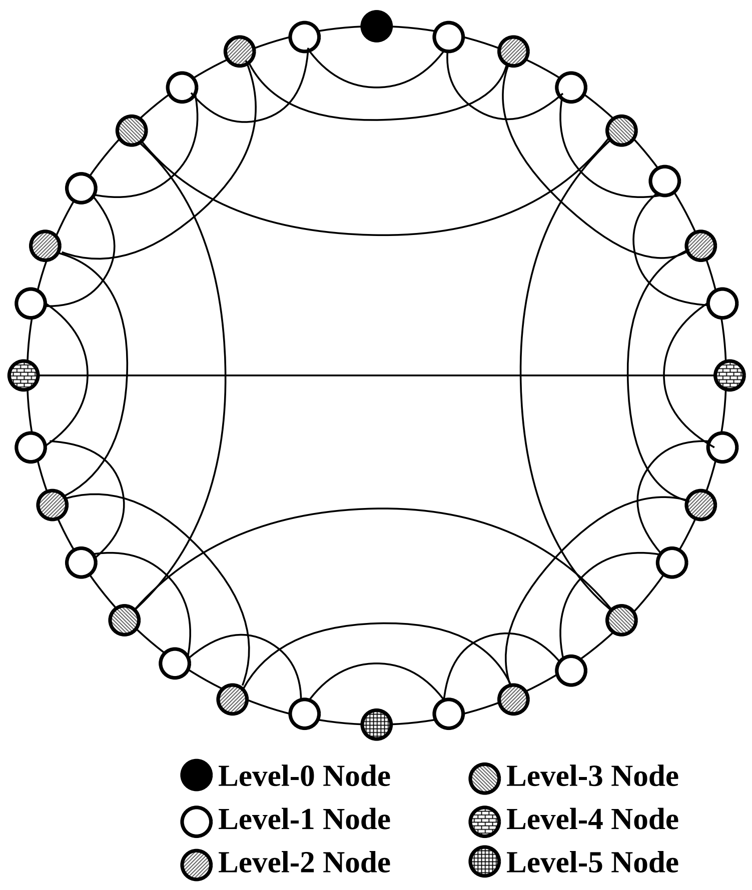


Figure 2.21: Standard 1D-SRT consisting of 32 nodes.

tolerant network, a new hierarchical network called *Shifted Recursive Torus (SRT)* [96] has been proposed.

The *Shifted Recursive Torus (SRT)* [96] 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 constraint that every node has fixed number of links. Figure 2.21 shows a basic interconnection of a standard 1D-SRT consisting of 32 nodes. 1D-SRT can be extended to 2D-SRT. The SRT has a smaller diameter with limited number of links per node than the hypercube. It is also an wire-efficient network for VLSI implementation.

TESH Network

The *TESH (Tori connected mESHes)* [47–50] is a hierarchical interconnection network which consists of basic module (BM) and the BMs are hierarchically interconnected for higher level networks. At the lowest tier, the Level-1 network – also called BM, consists of a mesh network of size $(2^m \times 2^m)$, where m is a positive integer. Successively higher level networks are built by recursively interconnecting the immediate lower-level $(2^m \times 2^m)$ subnetworks in the form of a 2D-torus. Figure 2.22 illustrates a Level-2 TESH network considering $m = 2$. It is a 2D-torus (4×4) network of 16 BMs. The Level-3 network is also a 2D-torus, built with $(2^m \times 2^m)$ Level-2 subnetworks. A similar interconnection rule is applied for higher level network.

TESH possesses several attractive features including low degree, small diameter, and low average distance. Particularly, it has smaller number of vertical links between silicon wafers. TESH is a suitable network for 3D wafer-stacked integration. Also in 3D-WSI, it shortens the longest links of the higher level networks. However, the main demerit of the TESH network is that the dynamic communication performance of the TESH network is lower than conventional torus network, especially in terms of network throughput.

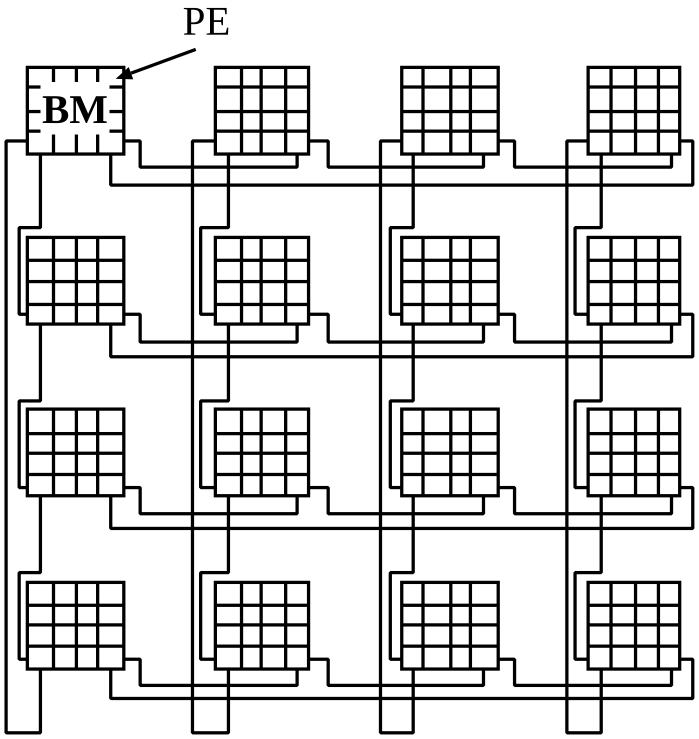


Figure 2.22: Level-2 interconnection of TESH network

H3D-Torus Network

TESH network is a suitable network for medium size network ($N \leq 4096$) and requires the smallest number of vertical links in 3D wafer-stacked implementation [47, 50]. It has been shown that TESH possesses several attractive features including small diameter, small number of wires (particularly, small number of vertical links), and low layout area for a few thousands of nodes [47]. However, the restriction in vertical links increases the diameter of the TESH network for tens of thousands of nodes. H3D-torus network overcomes this problem.

An *Hierarchical 3-Dimensional torus (H3D-torus)* [61, 62] has been put forward as a new interconnection network for large-scale 3D multicomputers. The H3DT network consists of multiple basic modules (BM) which are 3D-mesh of size $(m \times m \times m)$. The BMs are hierarchically interconnected by a 3D-torus of size $(n \times n \times n)$ to build higher level networks. Nodes at the four vertical edges of the BM are used for higher level interconnection. As illustrated in Figure 2.23, a Level-2 H3D-torus network, for example, can be formed by interconnecting 64 BMs as a $(4 \times 4 \times 4)$ 3D-torus network. The Level-3 interconnection is also a 3D-torus connection of 64 Level-2 modules. A similar interconnection rule is applied for higher level networks.

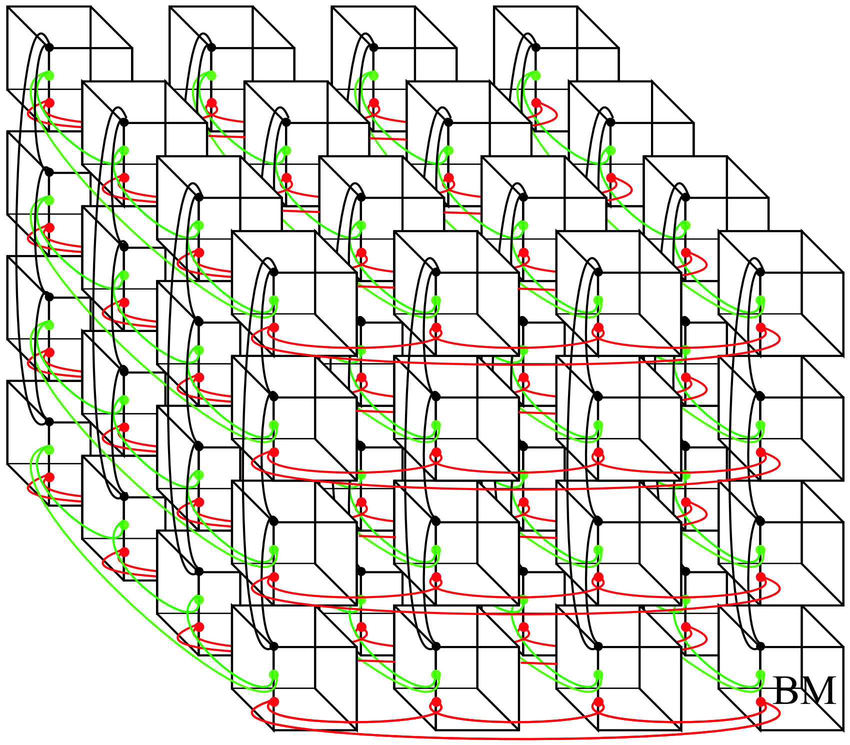


Figure 2.23: Interconnection of a Level-2 H3D-torus network

In [62], it is shown that H3D-torus network possesses several attractive features including small diameter, small number of wires – in particular small number of vertical links, and economic layout area for tens of thousands of nodes. The diameter is far better than any other conventional and hierarchical networks. The layout area of the H3D-torus in 3D-WSI is amenable to 3D implementation. However, the restricted use of physical links between basic modules in the higher level networks reduces the dynamic communication performance of this network. Its dynamic communication performance, especially in terms of network throughput, is far lower than that of conventional and other hierarchical networks [97].

H3D-Mesh Network

Due to restricted use of physical links between basic modules in the higher level networks, the inter-BM links are concentrated, which in turn reduces the network throughput. If we increase the inter-BM links, then the benefit (smaller number of vertical links and economic layout area for 3D-WSI) of H3D-torus network reduces and even vanishes. H3D-mesh network overcomes this problem. The inter-BM links of the H3D-mesh network is higher than that of the H3D-torus network. To reduce the peak number of vertical links between silicon planes, 2D-mesh have been considered as higher level network instead of 3D-torus network.

The *Hierarchical 3-Dimensional mesh (H3D-mesh)* network [59, 60] 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 network of size ($m \times m \times m$). Nodes at the contours of the xy-plane of the BM are used for higher-level interconnection. Figure 2.24 illustrates the interconnection of a Level-2 H3D-mesh network. As shown in Figure 2.24, the mesh at Level-2 interconnects the BMs in a (4×4) 2D-mesh.

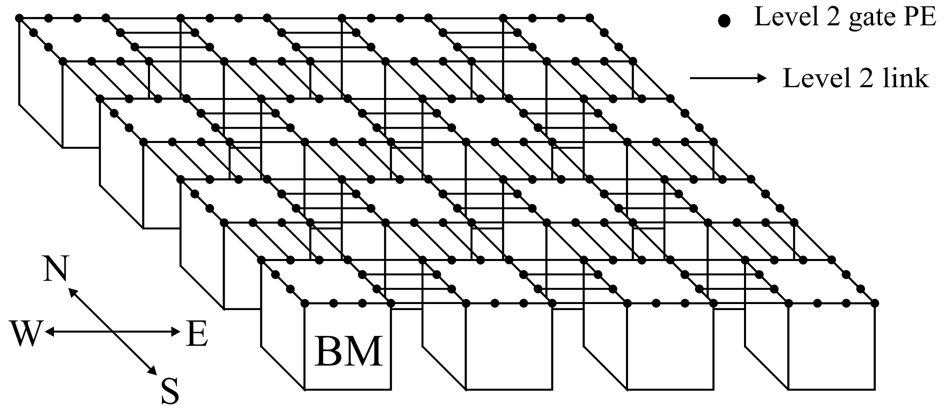


Figure 2.24: Interconnection of a Level-2 H3D-mesh network

Four links in the north *(N)*, south *(S)*, east *(E)*, and west *(W)* directions on each contour are interconnected for higher-level networks. Using the links in the N-direction and S-direction, the y-direction of the mesh at Level-2 is interconnected. Similarly, using the links in the E-direction and W-direction, the x-direction of the mesh at Level-2 is interconnected. Then the mesh interconnection between BMs at Level-2 is complete. The Level-3 interconnection is also a 2D-mesh connection of 16 Level-2 modules. A similar interconnection rule is applied for higher-level networks.

In [59], it is shown that H3D-mesh network possesses several attractive features including small diameter, small number of wires – in particular small number of vertical links, and economic layout area. The layout area of the H3D-torus in 3D-WSI is amenable to 3D implementation. However, they are worse than those of H3D-torus network. In [60], we have evaluated the dynamic communication performance of the H3D-mesh network by computer simulation. It is seen that the throughput of H3D-mesh is better than that of H3D-torus network. However, it is still lower than that of conventional and other hierarchical networks.

Beside these, a lot of hierarchical interconnection networks have been proposed in the literature. For example, based on completely connected network Complete Connection of Torus hyperCube (CCTCube), based on tree network Folded Fat H-Tree [98], based on star network Star-Connected Cycle [99], based on hypercube network Crossed Cube, Folded cube [53], Twisted hypercube [54], Extended hypercube [55], Enhanced hypercube [56], Reduced cube [57], Hierarchical hypercube [100], Metacube [101], and Extended Fibonacci cube [102], based on cube-connected cycles Extended Cube Connected Cycles (ECCC)

[103], based on de-Bruijn network de-Bruijn Cube (dBCube) [104], Hyper deBruijn [105], and de-Bruijn Connected Torus (BCT) [106], based on ring network chordal ring [107], barrel shifter, and Recursive circulant [108], based on k-ary n-cube network Express Cube [109], Polymorphic torus [110], Recursive Torus Network (n-RTN), and Ring Tree on Mesh (RTM) have been proposed in the literature.

2.5 Conclusions

In this chapter, we have described a variety of interconnection networks which have been proposed, and in some cases, implemented for distributed-memory parallel computers. In particular, a number of recently developed systems have been based on mesh and torus networks. The binary cube, though no longer as popular as a few years ago, remains well-suited for a myriad of efficient algorithms, and thus remains a topology of interest. Using mesh, torus, and hypercube a large number of hierarchical interconnection networks have been proposed to minimize the cost and maximize the performance. Over the past few years, hierarchical interconnection networks have begun to receive a great deal of interest.

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 various 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

"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 (1713–1784)

Hierarchical Torus Network (HTN)

3.1 Introduction

Hundreds of interconnection networks have been proposed in the last two decades. 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 [112].

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.* [47–50] have pointed out that hierarchical interconnection networks are suitable for 3D stacked implementations.

It has already been shown that a torus network has better dynamic communication performance than that of a mesh network [64]. This is the key motivation for us to consider a hierarchical interconnection network, in which both the basic module and the interconnection of higher levels have toroidal interconnections. 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 describes the basic structure of the 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

The HTN is a hierarchical interconnection network consisting of BM that are hierarchically interconnected for higher level networks. The BM consists of a 3D-torus network. In this dissertation, unless specified otherwise, 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 basic modules and the higher level networks have a toroidal interconnection. Hence, we use the name *"Hierarchical Torus Network (HTN)"*. To reduce the peak number of vertical links between silicon planes, we consider higher-level networks as 2D-toroidal connections instead of 3D-toroidal connections, despite the fact that a 3D-torus has better performance than a 2D-torus network. Moreover, in [111], it is shown that lower dimensional k-ary n-cubes outperform their higher dimensional counterparts under a bisection bandwidth constraint for deterministic routing and uniform traffic. The HTN is attractive since its hierarchical architecture permits systematic expansion of millions of nodes. Figure 3.1 shows the interconnection philosophy of HTN. This figure illustrates the interconnection of a Level-2 HTN using basic modules, where the BM is a 3D-torus of size $(4 \times 4 \times 4)$ and Level-2 network is a 2D-torus of size (4×4).

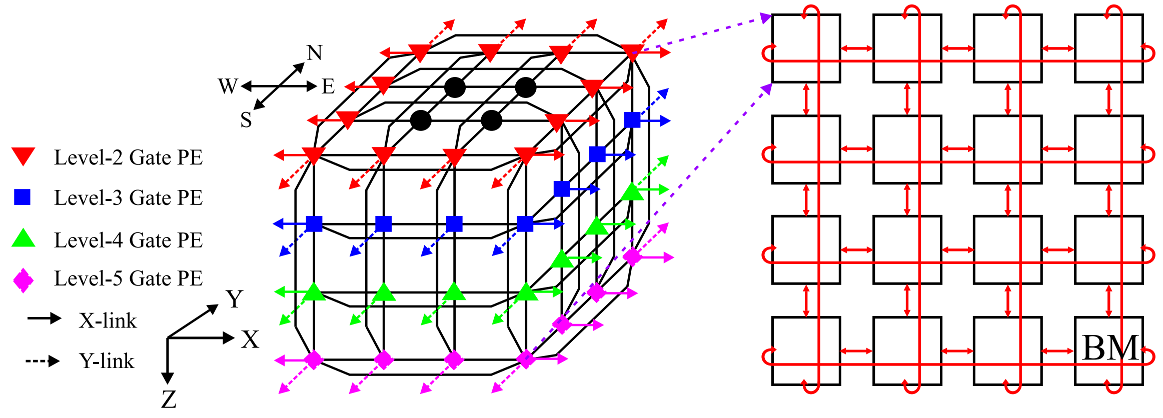


Figure 3.1: Interconnection of HTN

3.2.1 Basic Module

According to the definition, the BM 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 also a positive integer.

The BM, a $(4 \times 4 \times 4)$ torus, is shown in Figure 3.2, has some additional free ports at the contours of the xy-plane. These free ports are used for higher level interconnection.

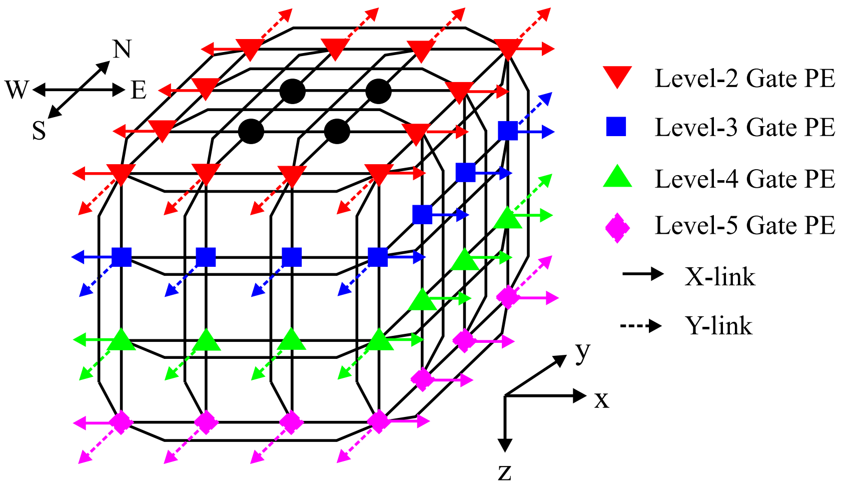


Figure 3.2: Basic module of the HTN

All free ports, typically one or two, of the exterior Processing Elements (PEs) are used for inter-BM connections to form higher level networks. All ports of the interior PEs are used for intra-BM connections. PEs at the contours of an xy-plane are assigned to higher levels as 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 Figure 3.2, the BM has four links in each direction as defined by:

$$G = \begin{bmatrix} 00_S, 01_S, 02_S, 03_S; 00_W, 10_W, 20_W, 30_W; \\ 30_N, 31_N, 32_N, 33_N; 03_E, 13_E, 23_E, 33_E \end{bmatrix}$$

where, G is the gate node. This equation represents the free links of the BM, which are used for higher level interconnection. $(0, 1, 2, 3 \quad \text{for } m = 4)$ represents PEs at the contours of an xy-plane, where the first digit represents the x-axis and the second digit represents the y-axis. The suffix is used to represent which direction of links is used for higher level connection. N-direction and S-direction links are used in the y-axis interconnection of higher level network. E-direction and W-direction links are used in the x axis interconnection of Level-2.

3.2.2 Higher Level Interconnection

According to the interconnection rule of the 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. Figure 3.3 shows the Level-2 subnetwork, which consists of a (4×4) torus and can be formed by interconnecting 16 BMs. As shown in Figure 3.3, each BM is connected by using G nodes to its logically adjacent BMs.

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

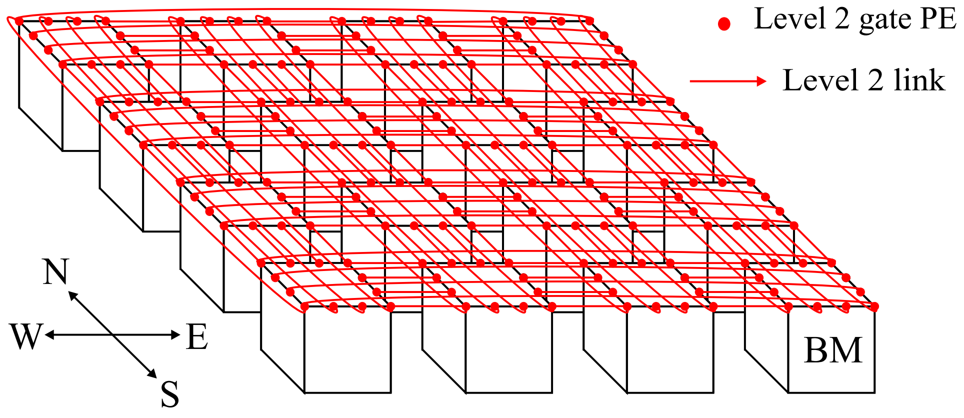


Figure 3.3: Interconnection of a Level-2 HTN

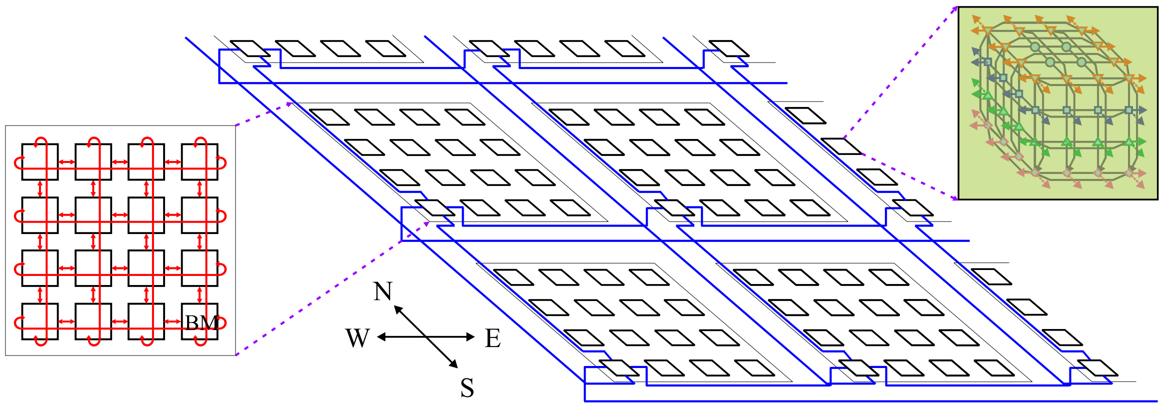


Figure 3.4: Interconnection of a Level-3 HTN

Similarly, a Level-3 network can be formed by interconnecting 16 Level-2 subnetworks, and so on. Thus, Level-L is interconnected as a 2D-torus, in which Level-$(L-1)$ is used as subnets. Level-3 or higher level network interconnects many BMs. BMs with the same co-ordinate position in each Level-2 subnetwork 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 BM i.e., the BM$(0,0)$ from every Level-2 network is selected for the interconnection of a Level-3 HTN.

A BM with $m = 4$ and the higher levels with $n = 4$ is perhaps the most interesting network size because it has better granularity than the larger sizes. With $m = 8$, the size of the BM becomes $(8 \times 8 \times 8)$ with 512 nodes. Correspondingly, with $n = 8$, the second level would have 64 BMs. In this case, the total number of nodes in a Level-2 network is $32,768$. Clearly, the granularity of the family of networks is rather coarse. In addition, 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 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 the sake of 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 Figure 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 Figure 3.2, free ports and their associated links at each contour 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 $L_{max} = 2^{p-q} + 1$.*

If the size of the BM is $(4 \times 4 \times 4)$ then $p = \lfloor log_2^4 \rfloor = 2$. With $q = 0$, $L_{max} = 2^{2-0} + 1 = 5$. Level-5 is the highest possible level that $(4 \times 4 \times 4)$ BM can be interconnected.

Lemma 3.3 *If the size of the BM is $(m \times m \times m)$ and the size of the higher level network is $(n \times n)$, then the total number of nodes in a Level-L network is $N = \left[m^3 \times n^{2(L-1)}\right]$.*

If $m = 4$, and $n = 4$, Level-2 and Level-3 networks have 1024 and 16384 nodes, respectively. The maximum number of nodes in a network having $(m \times m \times m)$ BMs and $(n \times n)$ higher level networks is $N_{max} = \left[m^3 \times n^{2(L_{max}-1)}\right]$. $L_{max} = (2^{p-q} + 1)$ denotes the highest level of the network. Putting the value of L_{max}, $N_{max} = \left[m^3 \times n^{2(2^{p-q})}\right]$.

The limitation of having maximum possible highest level network is not a serious constraint. For the case just considered $(4 \times 4 \times 4)$ BM with $q = 0$, the highest level of the HTN is Level-5. However, this level-5 network consists of 4.2 millions nodes.

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 representing the x-axis and the second representing the y-axis. The address of a PE at Level-L HTN is represented as shown in Equation 3.1.

$$A^L = \begin{cases} (a_z)(a_y)(a_x) & \text{if } L = 1, \text{ i.e., BM} \\ (a_y^L)(a_x^L) & \text{if } L \geq 2 \end{cases} \tag{3.1}$$

Here, $(a_z, a_y, a_x = 0, 1, ..., m-1)$ and $(a_y^L, a_x^L = 0, 1, ..., n-1)$. More generally, in a Level-L HTN, the node address is represented by:

$$\begin{aligned} A &= A^L A^{L-1} A^{L-2} \dots\dots\dots A^2 A^1 \\ &= a_\alpha\ a_{\alpha-1}\ a_{\alpha-2}\ a_{\alpha-3} \dots\dots\dots a_3\ a_2\ a_1\ a_0 \\ &= a_{2L}\ a_{2L-1}\ a_{2L-2}\ a_{2L-3} \dots\dots\dots a_3\ a_2\ a_1\ a_0 \\ &= (a_{2L}\ a_{2L-1})\ (a_{2L-2}\ a_{2L-3}) \dots\dots\dots\dots\dots (a_4\ a_3)\,(a_2\ a_1\ a_0) \end{aligned} \tag{3.2}$$

Here, the total number of digits is $\alpha = 2L + 1$, where L is the level number. 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 BM, 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; $2 \leq i \leq L$. In a two-level network, for example, the address becomes $A = (a_4\ a_3)\ (a_2\ a_1\ a_0)$. The last group of digits $(a_4\ a_3)$ identifies the BM to which the node belongs, and the first group of digits $(a_2\ a_1\ a_0)$ identifies the node within that BM.

Routing of messages in the HTN is performed from top to bottom. That is, it is first done at the highest level network; then, after the packet reaches its highest level sub-destination, routing continues 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's destination is 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.

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, a good routing algorithm should be easy to implement in hardware. Deterministic, dimension order routing algorithm is used by most existing multicomputers due to its simplicity. We have also considered the dimension order routing algorithm for the HTN. Routing at the higher level is performed first in the y-direction and then in the x-direction. In a BM, the routing order is initially in the z-direction, next in the y-direction, and finally in the x-direction.

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.

Routing in the HTN is strictly defined by the source node address and the destination node address. Let a source node address be $s_\alpha, s_{\alpha-1}, s_{\alpha-2}, ..., s_1, s_0$, a destination node address be $d_\alpha, d_{\alpha-1}, d_{\alpha-2}, ..., d_1, d_0$, and a routing tag be $t_\alpha, t_{\alpha-1}, t_{\alpha-2}, ..., t_1, t_0$, where $t_i = d_i - s_i$. The source node address of HTN is expressed as $s = (s_{2L}, s_{2L-1}), (s_{2L-2}, s_{2L-3}), ...,$ (s_2, s_1, s_0). Similarly, the destination node address is expressed as $d = (d_{2L}, d_{2L-1})$, $(d_{2L-2}, d_{2L-3}), ..., (d_2, d_1, d_0)$. Figure 3.5 shows the routing algorithm for the hierarchical torus network.

```
Routing HTN(s,d);
source node address:$s_\alpha, s_{\alpha-1}, s_{\alpha-2}, ..., s_1, s_0$
destination node address: $d_\alpha, d_{\alpha-1}, d_{\alpha-2}, ..., d_1, d_0$
tag: $t_\alpha, t_{\alpha-1}, t_{\alpha-2}, ..., t_1, t_0$
  for $i = \alpha : 3$
    if ($i/2 = 0$ and ($t_i > 0$ or $t_i = -(n-1)$)), routedir = North; endif;
    if ($i/2 = 0$ and ($t_i < 0$ or $t_i = \ (n-1)$)), routedir = South; endif;
    if ($i\%2 = 1$ and ($t_i > 0$ or $t_i = -(n-1)$)), routedir = East; endif;
    if ($i\%2 = 1$ and ($t_i < 0$ or $t_i = \ (n-1)$)), routedir = West; endif;
      while ($t_i \neq 0$) do
        $N_z = outlet_z(s, d, L, \text{routedir})$
        $N_y = outlet_y(s, d, L, \text{routedir})$
        $N_x = outlet_x(s, d, L, \text{routedir})$
        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;
        if ($t_i > 0$), $t_i = t_i - 1$; endif;
        if ($t_i < 0$), $t_i = t_i + 1$; endif;
      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 $t_i \leq \frac{m}{2}$) or ($t_i < 0$ and $t_i = -(m-1)$), movedir = positive; endif;
    if ($t_i > 0$ and $t_i = (m-1)$) or ($t_i < 0$ and $t_i \geq -\frac{m}{2}$), movedir = negative; endif;
    if (movedir = positive and $t_i > 0$), distance = $t_i$; endif;
    if (movedir = positive and $t_i < 0$), distance = $m + t_i$; endif;
    if (movedir = negative and $t_i < 0$), distance = $t_i$; endif;
    if (movedir = negative and $t_i > 0$), distance = $-m + t_i$; 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 = negative), 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 = negative), 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 = negative), move packet to $-x$ node; distance$_0$ = distance$_0 + 1$; endif;
  endwhile;
end
```

Figure 3.5: Routing algorithm of the 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 commonly used to evaluate and compare interconnection networks. These include symmetry and regularity, distance measures, connectivity and reliability, and scalability and expandability.

In this section, we discuss the static network performance that characterize the cost and performance of an interconnection network. Desirable properties of an interconnection network include low degree, low diameter, symmetry, low congestion, high connectivity, high fault tolerance, and efficient routing algorithms. We evaluate the node degree, diameter, cost ($= degree \times diameter$), average distance, bisection width, and connectivity to show the superiority of the HTN over various conventional and hierarchical interconnection networks for massively parallel computers.

3.3.1 Node Degree

The *Node Degree* (ND), simply degree, is defined as the number of physical channels emanating from a node. The degree of a network is the maximum of all node degrees in the network. This attribute is a measure of the I/O complexity of a node. In an interconnection network, the degree of a node can be constant or a function of the number of nodes in the network. For example, in a 2D-torus each node has degree 4, and in an nD-hypercube the degree of each node is n.

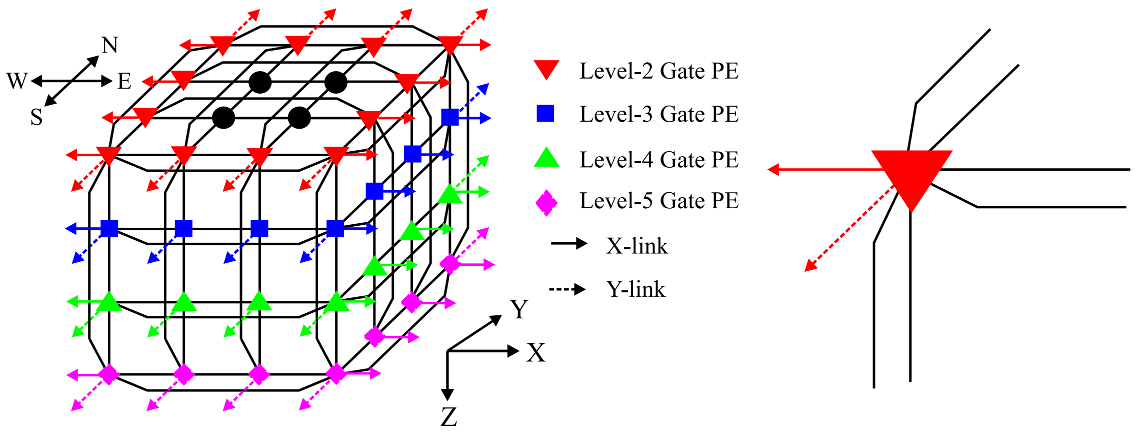


Figure 3.6: Illustration of degree of HTN

The degree relates the network topology to its hardware cost. The node degree should kept constant and as small as possible. Constant degree means that the cost of the network interface of a node remains unchanged with increasing size of the network. Small degree allows simple and low cost routers which amortizes the design cost. Constant degree networks are easy to expand and are often suitable for efficient VLSI implementation. On the other hand, small node degree implies less links and lower connectivity, which in turns implies larger distances. A high degree has high theoretical power but a low degree is more practical. For HTN, the node degree is constant. Since each node has eight links, therefore the degree of a node is 8. This is illustrated in Figure 3.6.

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 *network diameter*, and 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. Message delay is proportional to the number of links traversed. In fact, the diameter, sometimes (but not always), sets the lower bound for the running time of an algorithm performed on the network. Although diameter does not completely characterize the performance of an interconnection network with respect to their power to perform certain operations. For example, 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 networks with low diameter has received considerable attention in the literature. In this section, we evaluate the diameter of the HTN and compare it with other networks. 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 in the BM to the gate node 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 nodes 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

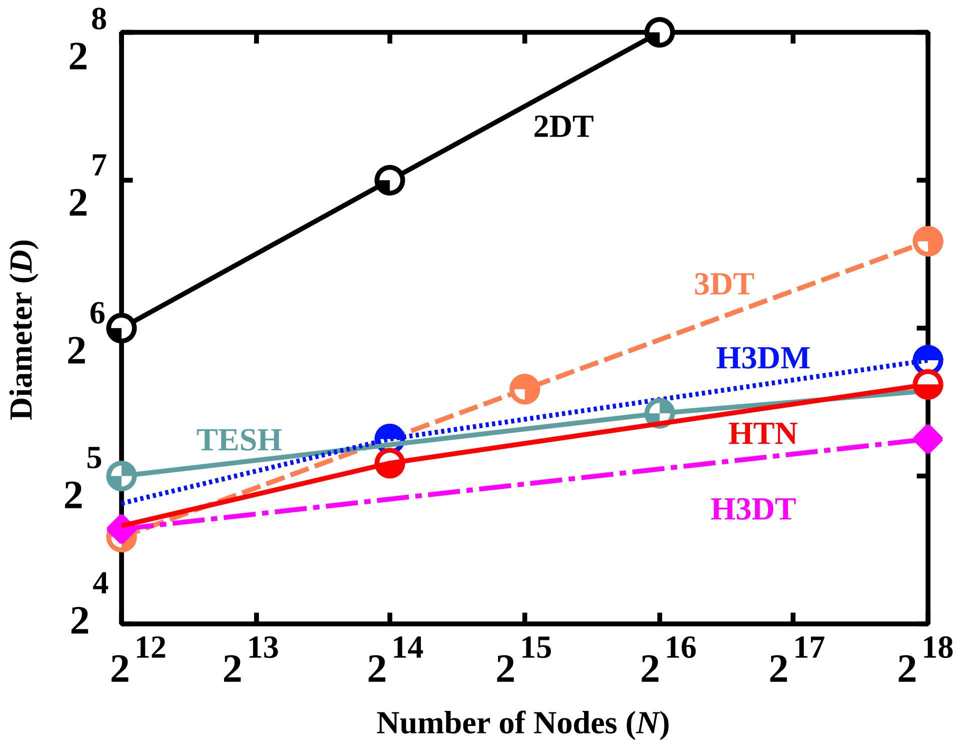


Figure 3.7: Diameter of networks as a function of number of nodes *(N)*

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

$$\begin{aligned} D &= D_{BM}^{to\ y-gate-L} + (L-1) \times D_{2D-torus} + (2L-3) \times\ D_{BM}^{axis\ move} + \\ &\quad (L-2) \times D_{BM}^{Level\ move} + D_{BM}^{to\ x-gate-2} \end{aligned} \tag{3.3}$$

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 PEs of the HTN with Level-L is:

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

$$D \approx O(\log N) \quad (3.4)$$

Therefore, 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 size. Each curve stands for a particular family of networks. It is shown that HTN has much smaller diameter than that of conventional k-ary 2-cube [64] and H3D-mesh network [59] and also smaller than that of TESH network[47–50] for a medium-sized network.

3.3.3 Average Distance

Although the diameter is used to compare the network performance of various interconnection networks, 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 an average, shorter paths 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 *mean inter-node distance or average distance* which is defined 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 simplest message distribution used to compare the average distance is the uniform distribution where each node sends message to every other nodes with equal probability.

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 3.8 shows a comparison of network average distance between the HTN and several other networks. Figure 3.9 also shows a comparison of average distance of various networks with 4096 nodes. It is seen that HTN has the smaller average distance than that of TESH [47–50], H3D-mesh [59, 60], and conventional k-ary n-cube [64] 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.

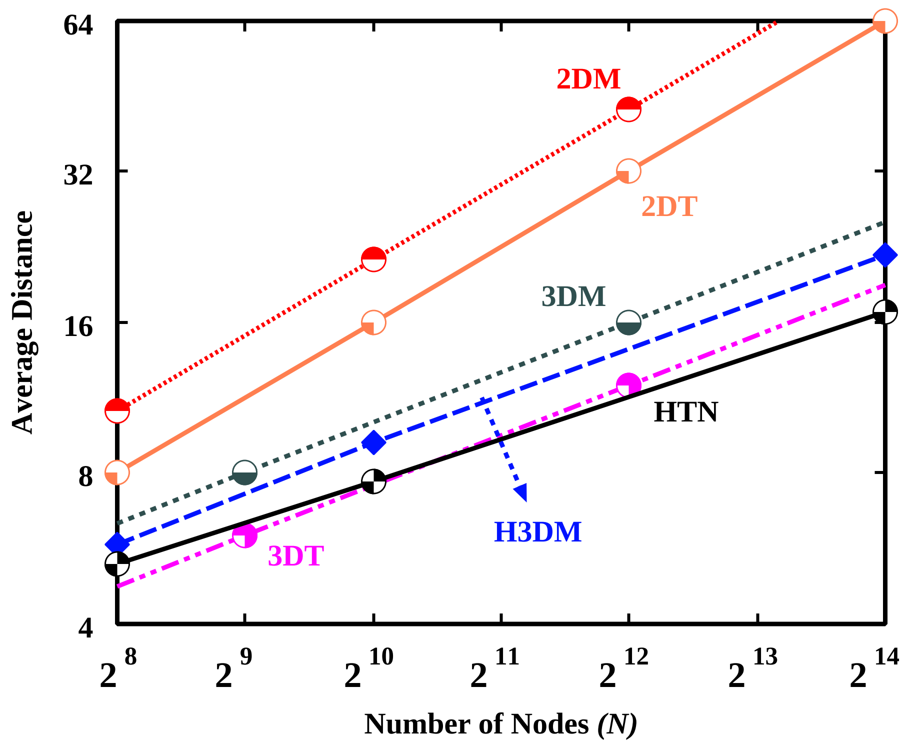


Figure 3.8: Average distance of networks as a function of number of nodes (N)

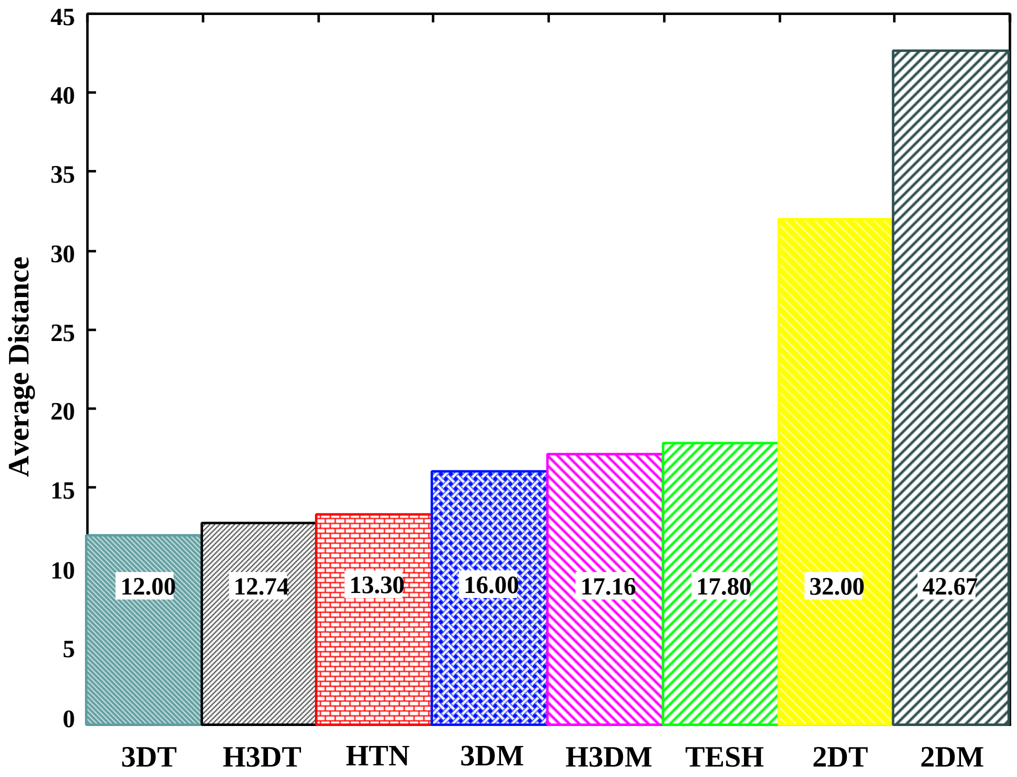


Figure 3.9: Average distance of various networks with 4096 nodes.

3.3.4 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 [55, 58]. 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 3.10 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 [64] and H3D-mesh network [59].

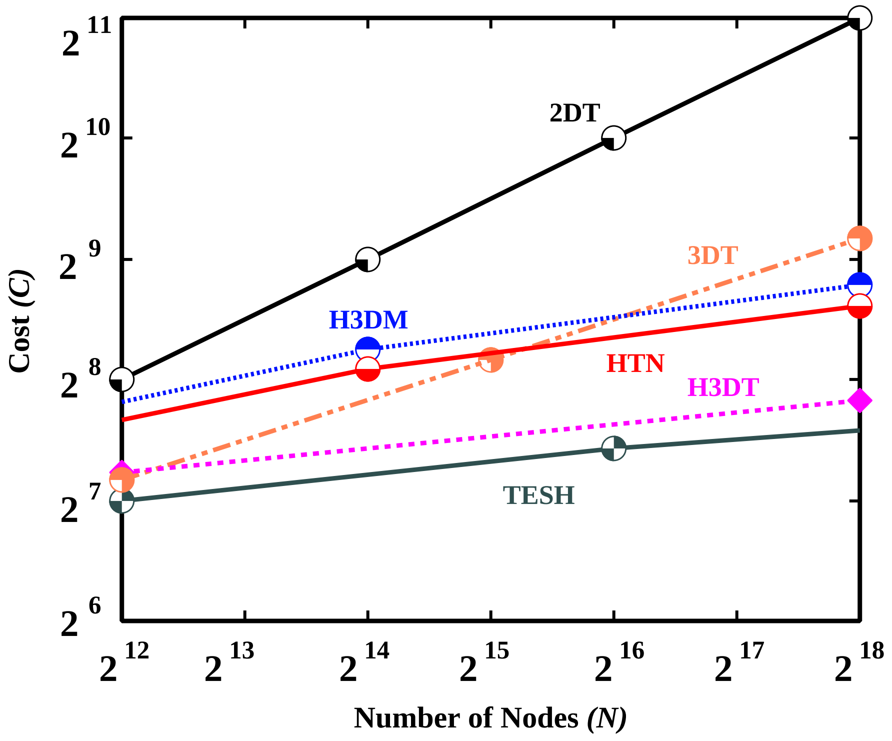


Figure 3.10: Cost of different networks as a function of number of nodes *(N)*

3.3.5 Connectivity

The *arc connectivity*, simply *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 links whose removal causes the network to be disconnected into two or more components. For symmetric networks, the connectivity is usually the same as the degree of nodes. For example, a 2D-torus network has connectivity 4 and a hypercube with N nodes has connectivity $\log N$. Connectivity cannot exceed the degree, since the network can be partitioned by removing all neighbors of a specific node. High connectivity implies a large number of node-disjoint paths between pairs of nodes; and it improves performance during normal operation by avoiding congested links. Figure 3.11 portrayed that removal of 2 links disconnects the 2D-mesh network into two parts. This is the minimum value of links to disconnect the 2D-mesh network. Thus, the connectivity of 2D-mesh network is 2. The connectivity of various networks including the HTN is shown in Table 3.2. It is shown that the connectivity of the HTN is higher than that of mesh, TESH, and H3D-torus networks and equal to that of k-ary n-cube and H3D-mesh networks.

A network is said to be maximally fault-tolerant if its connectivity is equal to the degree of the network. High connectivity of a network improves its fault tolerance. Connectivity is considered to be a qualitative measure of fault tolerance. Thus, quantitative measure is required for certain level of fault-tolerance of a network. As shown in Table 3.2, in comparison with degree and connectivity, the fault-tolerance performance of HTN is better than that of mesh, TESH, and H3D-torus networks and equal to that of H3D-mesh network and worse than that of k-ary n-cube network.

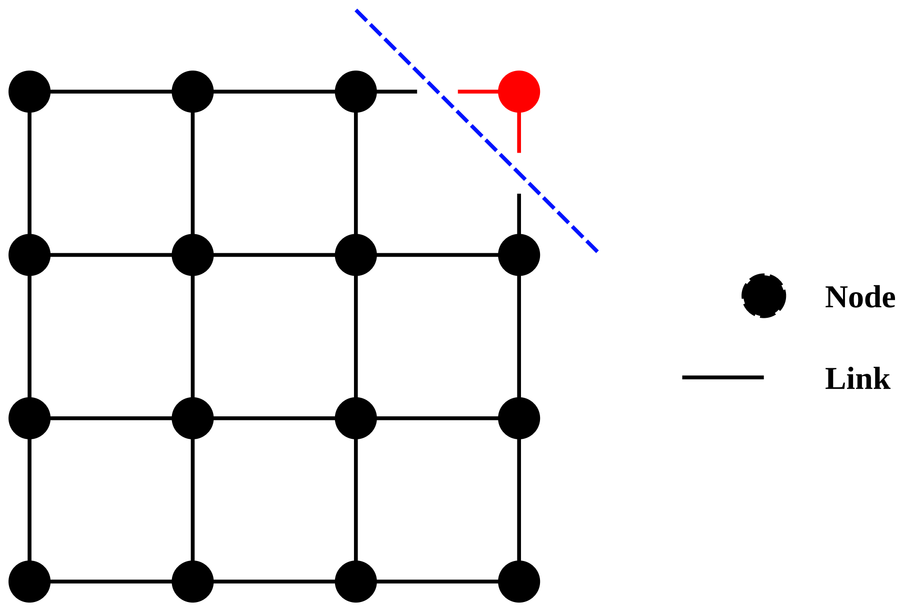


Figure 3.11: Illustration of connectivity for 2D-mesh network.

Table 3.2: Comparison of degree and connectivity for various networks

	2DM	2DT	3DM	3DT	TESH	H3DM	HTN	H3DT
Degree	4	4	6	6	6	8	8	6
Connectivity	2	4	3	6	2	6	6	3

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 links that have to be removed to partition the network into two equal halves. A network with odd number of nodes, one half will have one more node than the other after the bisection. 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 2D-mesh with N nodes is $\sqrt{N}$ and that of a n-dimensional hypercube with $N = 2^n$ nodes is $\frac{N}{2}$. The bisection width of the HTN is calculated by:

$$BW\ (HTN) \quad = \quad 2^{q+1} \times (m \times n) \tag{3.5}$$

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 $2m^2$.

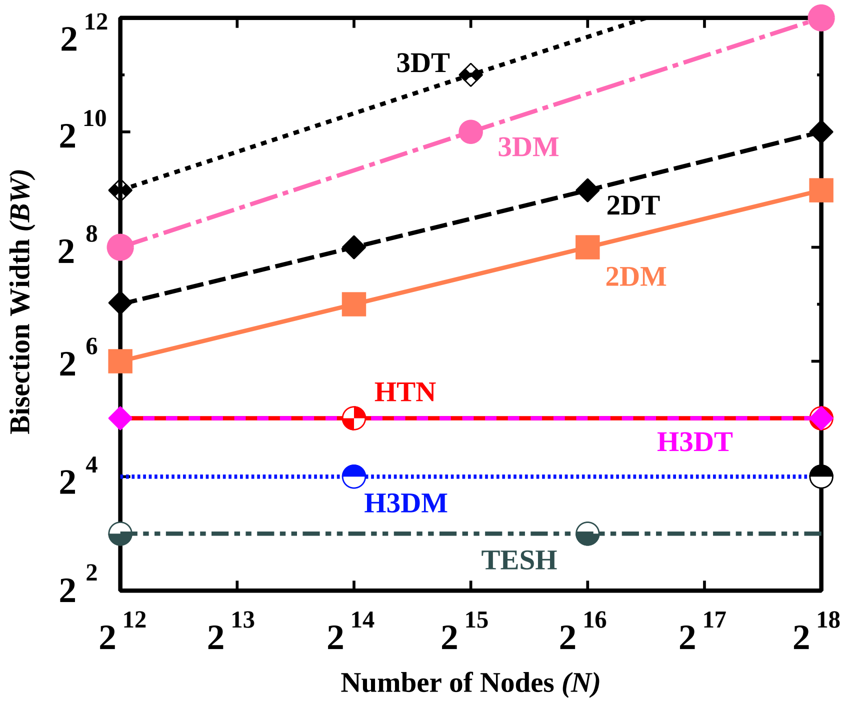


Figure 3.12: Bisection width of networks as a function of number of nodes *(N)*

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. A small bisection width implies low bandwidth between both halves and it can slowdown the final merging phase. A large bisection width enables faster data and information exchange and implies high degree of fault tolerance. Thus, in one hand, a large bisection width is preferable. A large bisection width requires a larger layout area for VLSI implementation of an interconnection network. Thus, on the other hand, a small bisection width is preferable for efficient VLSI realization. Therefore, a network with moderate bisection width is necessary for both faster data exchange and efficient VLSI implementation.

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 that of TESH [47–50] and H3D-mesh [59]networks, equal to that of H3D-torus network, and smaller than that of conventional k-ary n-cube [64] networks. Therefore, in context of bisection width, HTN is superior to other networks.

3.4 Wafer Stacked Implementation

3.4.1 3D Stacked Implementation

M. Little *et.al.* [43] 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 [44]. 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 part of the network is mapped to a silicon plane and all parts of the network are interconnected by vertical links between silicon planes. The vertical links between adjacent silicon planes are implemented by wafer feedthroughs and microbridges [43, 44]. The structure of microbridge and feedthrough are shown in Figure 3.14.

Vertical links (composed of feedthroughs and microbridges) realize the shortest pass between planes. However, the area required for vertical links is about hundred of μm^2. 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 adjacent silicon planes . The word *peak* here refers to the bulge due to the crowding of wires running between various planes.

As shown in Figure 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:

$$N = h \times M \tag{3.6}$$

Figure 3.16 illustrates the 3D stacked-implementation of a 2D-mesh network with 16 PEs on 4 stacked-planes. According to Eq. 3.6, here, $N = 16$, $M = 4$, and $h = 4$.

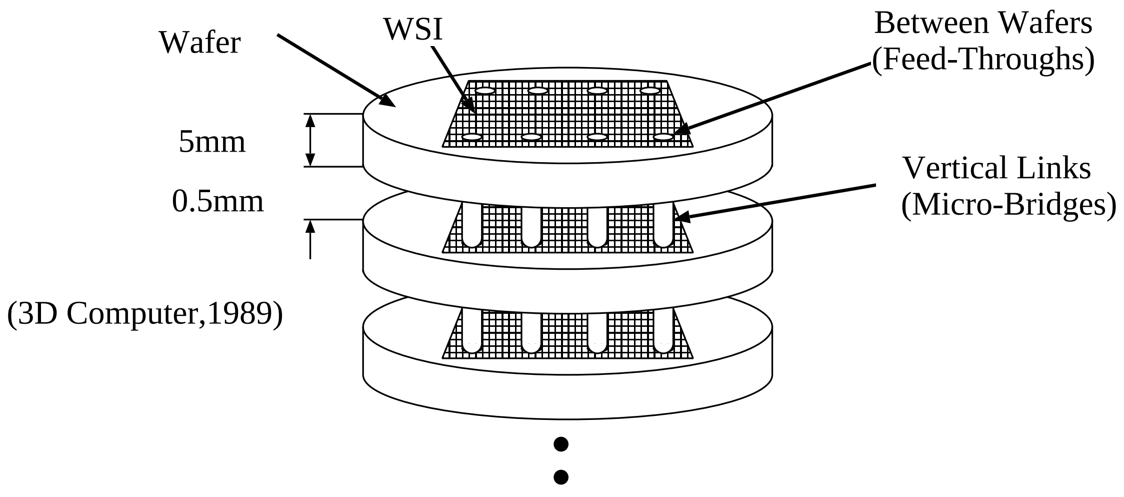


Figure 3.13: Structure of 3D stacked implementation

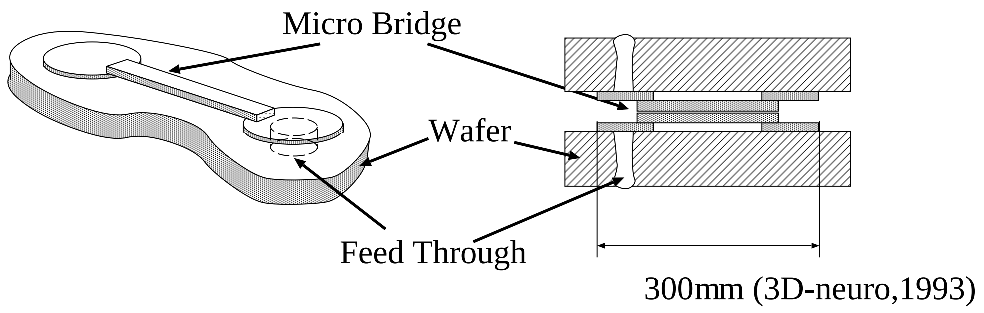


Figure 3.14: Structure of microbridge and feedthrough

Peak Number of Vertical Links for 2D-Torus Network

2D-torus network provides additional wrap-around links for end-to-end node connections to that of 2D-mesh network. In this section, we discuss the peak number of vertical links of 2d-torus network in 3D stacked implementation. The simplest mapping scheme is the row (or column) major scheme. For the row (or column) major scheme, the peak number of vertical links is given by:

$$C_{peak} = \begin{cases} 2\sqrt{N}+2 & N > 4 \\ 4 & N = 4 \end{cases} \tag{3.7}$$

Figure 3.17 illustrated the vertical links for a 2D-torus network in 3D stacked implementation by row major scheme with 16 nodes on four stacked planes. The row (or column) major scheme is difficult to realize 2D-torus in 3D stacked-implementation in context of the longest links for massively parallel computers. Figure 3.18(a) illustrates the row major scheme and the interconnection of vertical links of 2D-torus network. The link between $Node(0)$ and $Node(12)$ is one of the longest links of 2D-torus network in 3D stacked implementation.

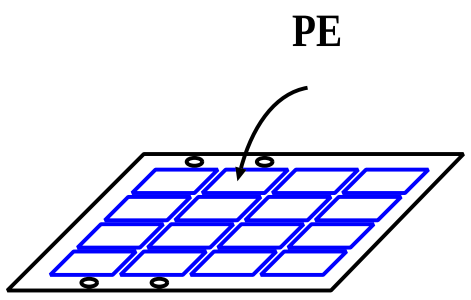


Figure 3.15: PE array in a silicon plane for wafer stacked-implementation

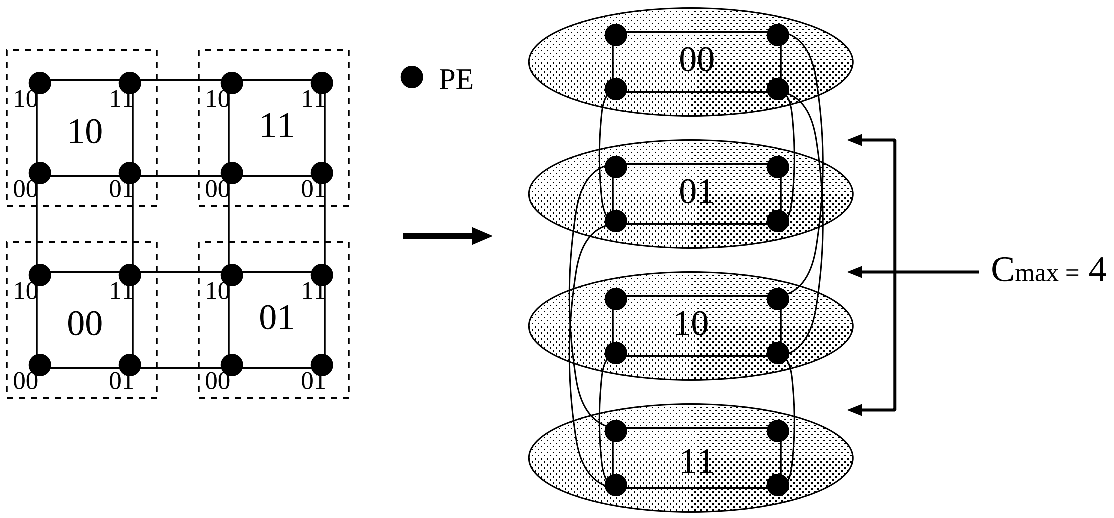


Figure 3.16: Vertical links of 2D-mesh network in 3D wafer stacked-implementation

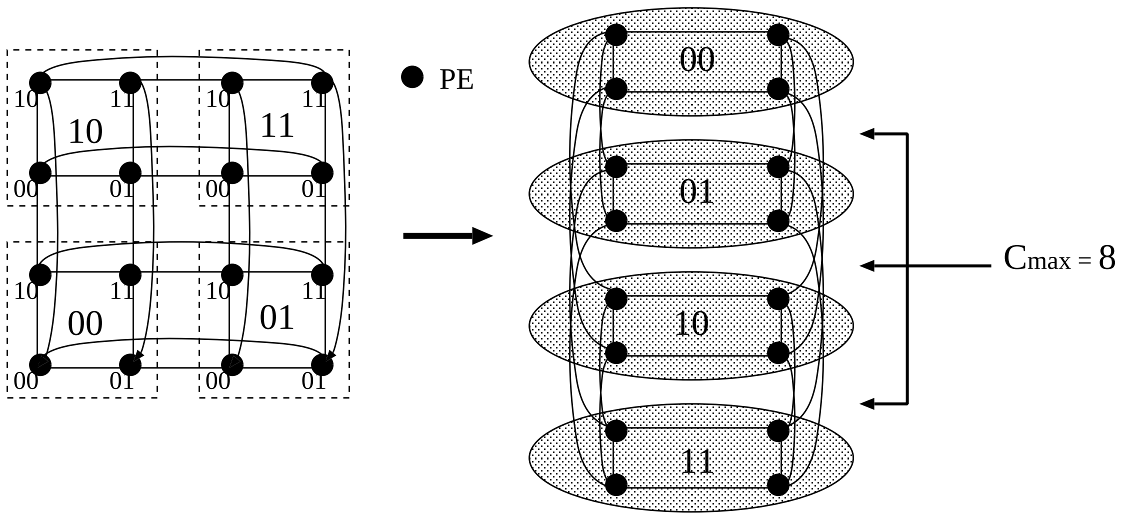


Figure 3.17: Vertical links of 2D-torus network in 3D wafer stacked-implementation

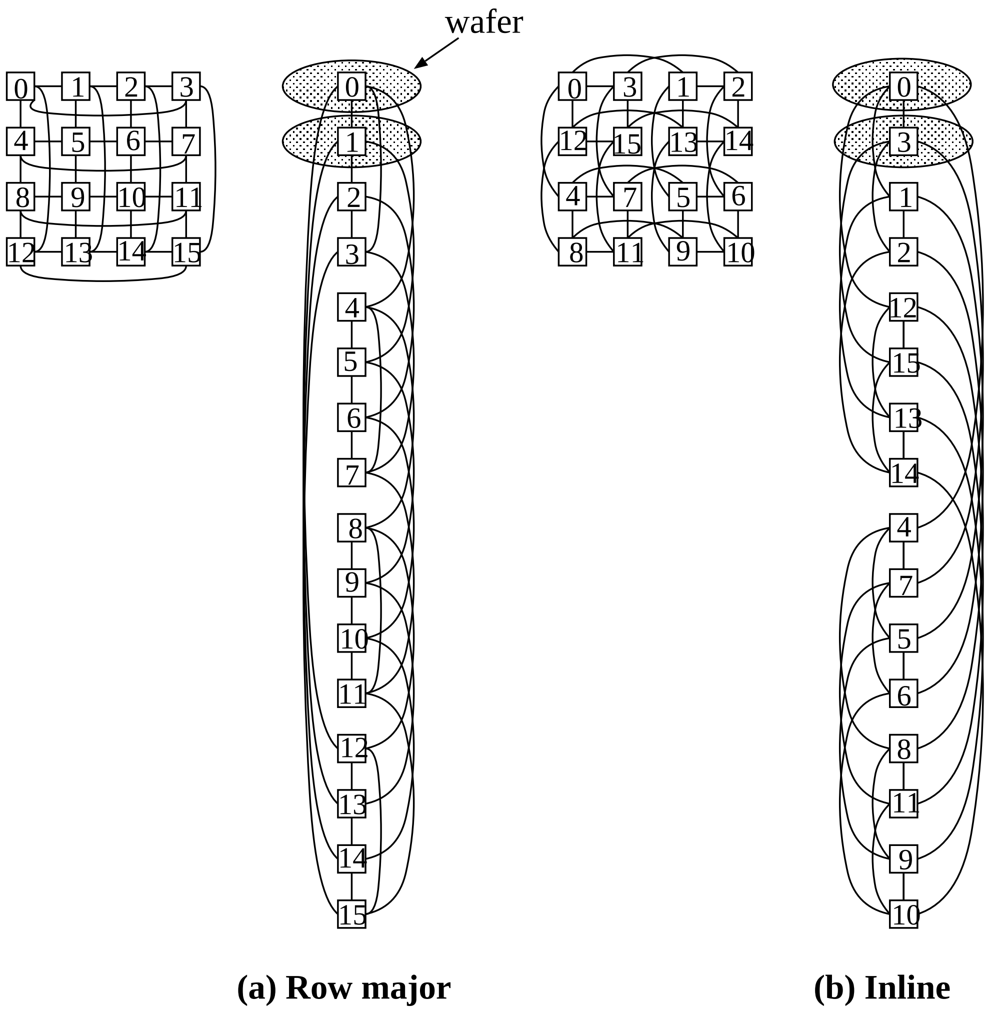


Figure 3.18: Interconnection scheme of 2D-torus in 3D stacked implementation

Figure 3.18(b) illustrates the in-line scheme [59]for 2D-torus network in 3D stacked implementation. The longest link in in-line scheme is shorter than that of row major scheme. The in-line scheme for a 2D-torus network is given by

$$I(N,i) \quad = \quad \begin{cases} \frac{i}{2} & i : even \\ N - \lceil \frac{i}{2} \rceil & i : odd \end{cases} \tag{3.8}$$

$$PE(k,j) = PE(I(n,k), I(n,j))$$

The peak number of vertical links in the in-line scheme for a 2D-torus network with bidirectional links is given by:

$$C_{peak} \quad = \quad \begin{cases} 2\sqrt{N} + 2\sqrt{m} & h = \frac{N}{m} \neq 4 \\ 4\sqrt{m} & h = \frac{N}{m} = 4 \end{cases} \tag{3.9}$$

If we consider $N = 16$, $M = 4$, and $h = 4$, then $m = 4$. The peak number of vertical links for a 2D-torus network is 8.

Peak Number of Vertical Links for 2D-Mesh Network

The peak number of vertical links for a 2D-mesh network with bidirectional links is given by:

$$C_{peak} = \begin{cases} \sqrt{N} + \sqrt{m} & h = \frac{N}{m} > 4 \\ \sqrt{N} & h = \frac{N}{m} = 4 \end{cases} \tag{3.10}$$

If we consider $N = 16$, $M = 4$, and $h = 4$, then $m = 4$. The peak number of vertical links for a 2D-mesh network is 4.

In mesh network, the peak number of vertical links is just half to that of 2D-torus network, because of the absence of wraparound links. Thus, the peak number of vertical links of an interconnection network depends upon the architecture of the network. The illustrations shown here uses bi-directional links. The peak number of vertical links using unidirectional links becomes twice as much as that using bidirectional links.

Peak Number of Vertical Links for HTN

In this section, we evaluate the peak number of vertical links for our proposed 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 sub-networks at the Level-$(i-1)$, where $i \leq L$. Let $L_d = \left(1 + \left\lfloor \log_g \frac{M}{n} \right\rfloor\right)$ be the highest level on a plane, and $s = \left(m \bmod ng^{L_d-1}\right)$ 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 as follows:

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 in-line 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. Therefore, 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:

$$\begin{aligned} C_{peak} &= C_{peak}\,(\text{2D} - \text{torus,16}, S_{Li}) \\ &= \begin{cases} 10\left(16^{L_i-2}\right) & S_{Li} = 1 \\ 12\left(16^{L_i-2}\right) & S_{Li} = 4 \end{cases} \end{aligned} \tag{3.11}$$

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:

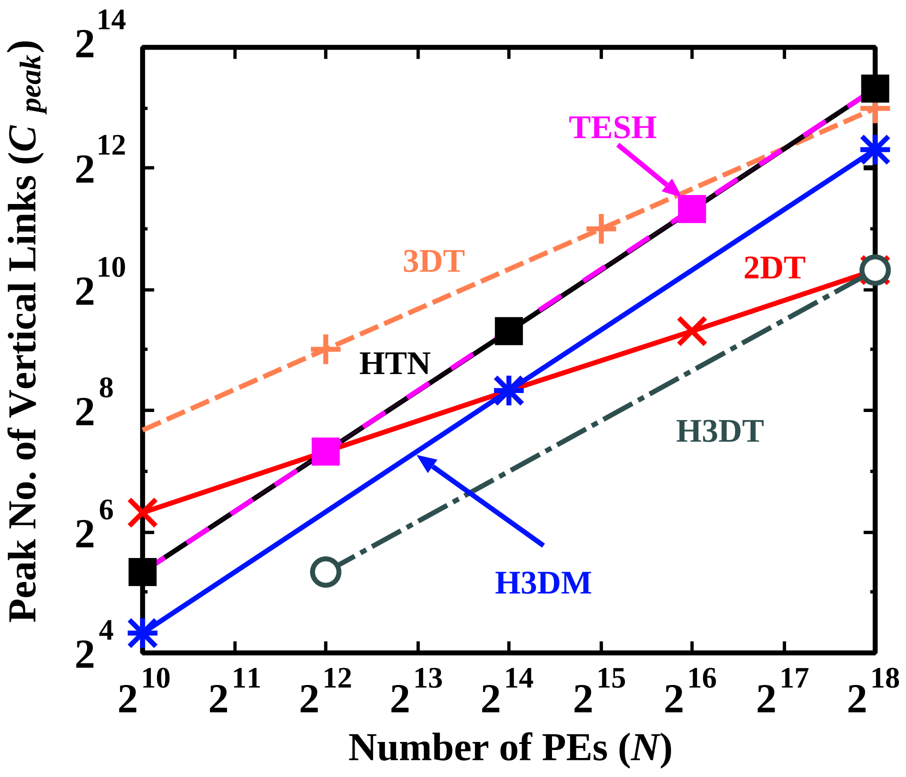


Figure 3.19: A comparison of peak number of vertical links of HTN with other networks

$$\begin{aligned} C_{peak} &= C_{peak}\left(\text{2D} - \text{torus,16}, s\right) 16^{L_d-2} + \sum_{L_i=L_d+1}^{L} C_{peak}\left(\text{2D} - \text{torus,16}, 1\right) 16^{L_i-2} \\ &= C_{peak}\left(\text{2D} - \text{torus,16}, s\right) 16^{L_d-2} + \sum_{L_i=L_d+1}^{L} 5\left(16^{L_i-2}\right) \qquad (3.12) \end{aligned}$$

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 3.19 shows that the peak number of vertical links C_{peak} for an HTN is less than that of the k-ary n-cube network and exactly same as that of the TESH network. It is more than that of H3D-mesh and H3D-torus networks; the wrap-around links for a 2D-torus network in the higher dimensional HTN results these additional vertical links. HTN is suitable for medium-sized networks.

A more quantitative way to compare different network families would be to use a figure of merit such as $f = N/\left(C_{peak} \times D\right)$ [47, 50], 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.

3.4.3 Layout Area

To discuss a suitable network for 3D stacked-implementation, layout area of network has to be evaluated. The following defines formulations of layout area for interconnection 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 3.20 illustrates the layout area of a 2D-torus for the case of $N = 16$, $L = 4$, and $p = 1$.

$\sqrt{m} \times \sqrt{m}$ PEs are implemented on a silicon plane, where the total number of PEs is N. The total area on a silicon plane is $\sqrt{m}W_{PE} \times \sqrt{m}W_{PE}$. For easy layout, $\sqrt{\frac{m}{L}} \times \sqrt{\frac{m}{L}}$ PEs are allocated in $\sqrt{L} \times \sqrt{L}$ blocks on a silicon planes. From the viewpoint of mechanical reliability and thermal cycling of 3D stacked-implementation, vertical links between planes should be uniformly distributed. To implement microbridge in each block, wiring space to connect microbridges of vertical links in blocks. Let $S_{DMB,x}$ and $S_{DMB,y}$ be wiring space to connect for interconnection of microbridge on L blocks in row and in column of PE array.

$$S_{DMB,x} = C_{peak} p W_{line} \tag{3.13}$$

$$S_{DMB,y} = C_{peak} p W_{line} \tag{3.14}$$

For regular layout on a silicon plane, wiring space of interconnection links between PEs is same in PEs array and does not depend on the location in a block.

Fukuda and Horiguchi [72] 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:

$$S_{TL,x} = \sqrt{m} t_x p W_{line} \tag{3.15}$$

$$S_{TL,y} = \sqrt{m} t_y p W_{line} \tag{3.16}$$

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

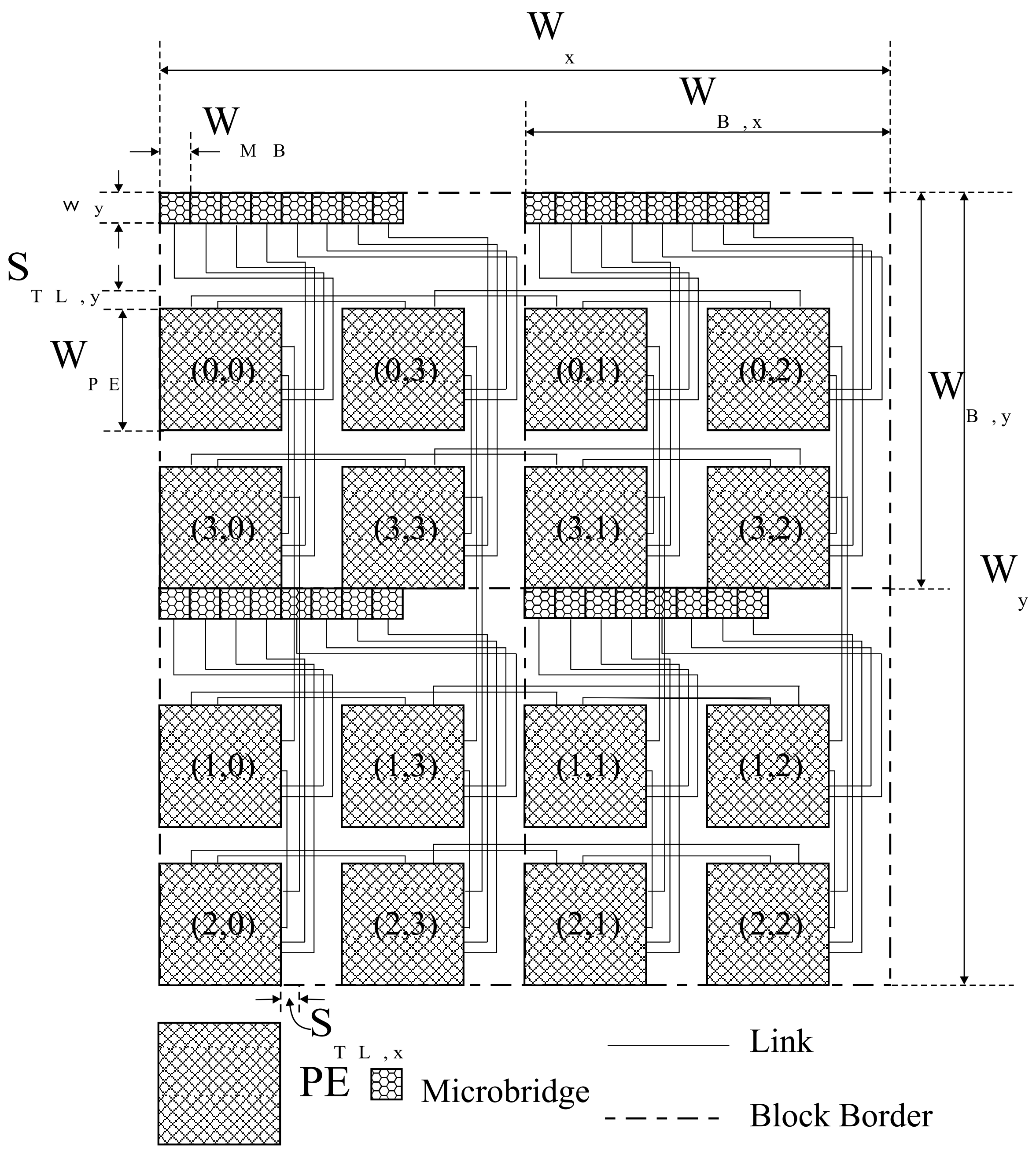


Figure 3.20: Layout area of 2D-torus for $N = 16$, $L = 4$ and $p = 1$.

$$L_e = \left\lceil \log_{16} \frac{m}{64} \right\rceil \tag{3.17}$$

$$t_x = t_y = \begin{cases} 8 & L_e \leq 2 \\ 8 + 2 \sum_{i=L_e}^{L} \left(4^{i-2}\right) & L_e > 2 \end{cases} \tag{3.18}$$

All the parameters needed to calculate the layout area 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 [46]. 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
- 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 $500\mu m \times 500\mu m$.

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 3.21 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 layout area of the HTN is almost equal to that of H3D-mesh [59] network. The peak number of vertical links for an HTN is exactly same as the TESH network [47–50], but the layout area is around 17% less.

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 between PEs	$S_{TL,x} = \sqrt{m} t_x p W_{line}$, $S_{TL,x} = \sqrt{m} t_x p W_{line}$
Wiring space of links between micro	$S_{DMB,x} = C_{peak} p W_{line}$, $S_{DMB,y} = C_{peak} p W_{line}$
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	$A = W_x \times W_y$

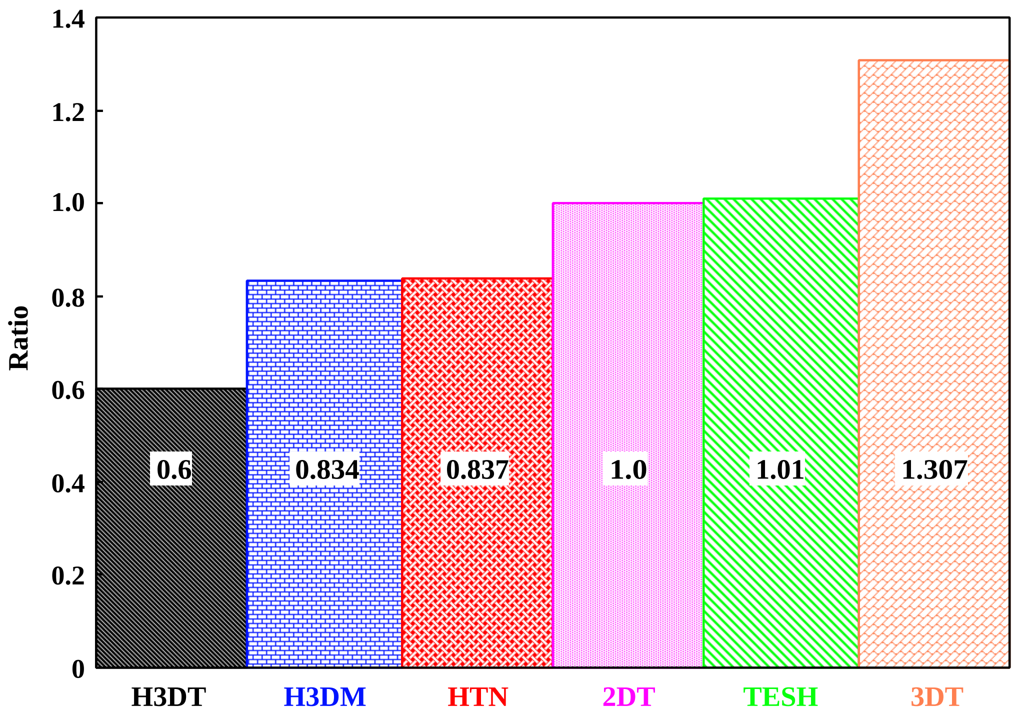


Figure 3.21: Normalized layout area

3.4.4 Maximum Wire Length

The cost of VLSI system 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 [59] 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 networks 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 [64] of k-ary n-cubes is:

$$LEN_{MAX}(k,n) \;=\; k^{\frac{n}{2}-1} \tag{3.19}$$

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:

$$LEN_{MAX}(k,n) \;=\; \sqrt{N} - \frac{\sqrt{N}}{k} = k^{(\frac{n}{2}-1)}(k-1) \tag{3.20}$$

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

$$LEN_{MAX}(HTN) \;=\; \left(n^{L-1}\right)(m-1) + \left(n^{L-1}-1\right) \tag{3.21}$$

The formula shown in Eq. 3.21 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 Figure 3.22. Our intuition is that a 3D wafer stacked implementation, can be considered, for brevity, by the equation shown in 3.21.

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

An important advantage of a 3D implementation is the reduction of the length of the interconnects. The longest wire in a planar Level-3 network are the wraparound wire which interconnect the physically farthest Level-2 subnetwork, and it is 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, 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. The thickness of the wafer is about $0.02\ inch = 0.051\ cm$ and the length of the microbridge is $0.002\ inch = 0.005\ cm$ [44].

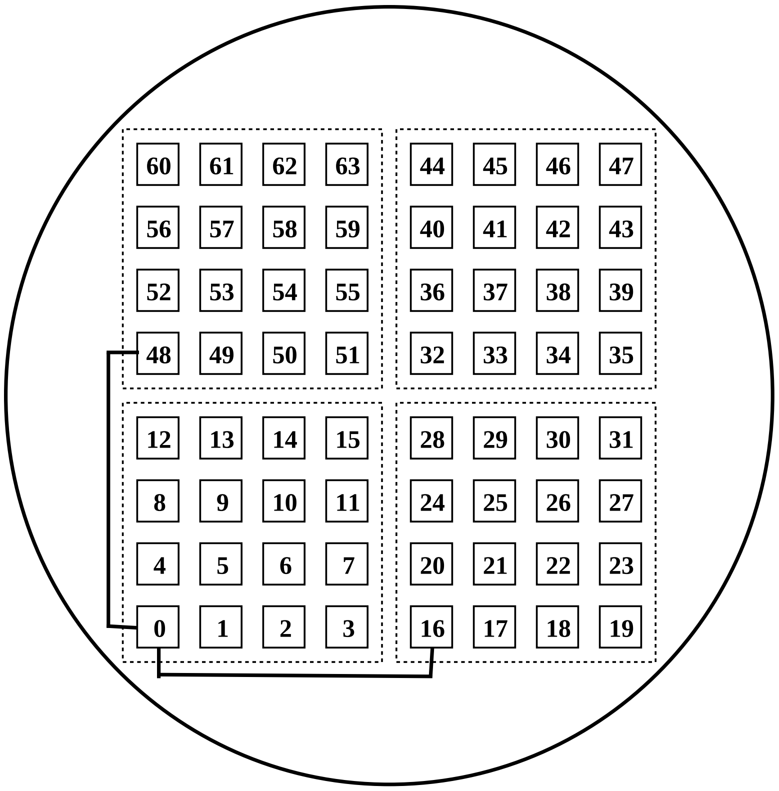


Figure 3.22: 2D-planner realization of 3D-torus network.

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

Thus, t is about 0.056 cm and the longest vertical wire has a length of 0.90 cm. Clearly, in 3D-stacked implementation the longest wires are the horizontal wires within the Level-2 subnetworks. For the HTN, the longest wire has a length of $15 \times 3.0\ mm = 45\ mm = 4.5\ cm$ which gives rise to a factor of 4.20 improvement over the planar implementation.

Using the HTN, millions of nodes can be interconnected together while maintaining good static network performance.

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 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. These include constant node degree, small diameter, small average distance, high connectivity, and better bisection width. Using the HTN, millions of nodes can be interconnected together while maintaining good static network performance.

The network is well suited for 3D stacked implementations. It is shown that the peak number of vertical links in 3D-stacked implementation is quite low for HTN as compared to other conventional and hierarchical networks. Thus, HTN permits efficient VLSI/ULSI/WSI realization. The layout area of HTN in 3D stacked-implementation 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. 3D-WSI implementation reduces the longest wire length 4.20 times over the planar implementation.

Chapter 4

"Science is nothing but developed perception, interpreted intent, common sense rounded out and minutely articulated."
– George Santayana (1863–1952)

Dynamic Communication Performance of the HTN

4.1 Introduction

The design of an interconnection network for a multiprocessor or multicomputer inevitably involves trade-offs 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 techniques are also needed to compare two or more networks, evaluate design trade-offs, determine the optimal value of design parameters.

Performance evaluation broadly includes *modeling* and *measurements*. Modeling can be performed analytically or by computer simulation. Analytical modeling involves development of a mathematical model to describe the behavior of the system to the desired degree of detail, and solving the model to obtain its performance. A simulation model captures the behavior of the system in a computer program. Simulation models can accommodate details that are difficult to model analytically, but are time consuming to develop and difficult to validate. In the case of interconnection networks, computer simulations are often used to evaluate their performance and verify the validity of analytical models.

Measurements are performed for *benchmarking*, that is to determine the performance of the system with respect to a standard workload. Unfortunately, there is no widely accepted workloads or programs existing for benchmarking the performance of interconnection networks; in addition, the performance of an interconnection network can be expected to vary significantly with the communication behavior of the workload. For these reasons, measurement-based studies for interconnection networks are relatively rare.

Performance evaluation of an interconnection networks includes: hardware cost, average message latency, network throughput, potential for fault tolerance, embedding capability, and partitioning capability. In an interconnection network, the hardware cost is expressed in terms of number of links and node degree. The details of hardware cost is addressed in Chapter 3. This chapter deals with the dynamic communication performance issue of Hierarchical Torus Network (HTN).

The basic function of an interconnection network is to transfer information among the nodes of a multicomputer in an efficient manner. *Routing Algorithm* determines the path from a source node to a destination node in a particular network. The dynamic communication performance of a specific network is dependent on a routing algorithm. 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 message latency, freedom from deadlocks, livelocks, and starvation [36].

This chapter is organized as follows: Some preliminaries of routing algorithms and its freedom from deadlock are present in Section 4.2. The deadlock free routing for HTN and investigation of minimum number of virtual channels to make the HTN deadlock free using dimension order routing algorithm are described in Section 4.3. Section 4.4 discusses the dynamic communication performance of HTN as well as its applicability to several commonly used networks for parallel computers. A suite of low-cost adaptive routing algorithms is proposed in Section 4.5. The router cost and speed of these routing algorithms are presented in Section 4.6. Section 4.7 discusses the dynamic communication performance of HTN using the proposed adaptive routing algorithms; provides their superiority over dimension order routing algorithm. Finally, some concluding remarks of this chapter are given in Section 4.8.

4.2 Routing Algorithm

An interconnection network must allow every node to send message to every other node. In the absence of complete topology, routing determines the path selected by a message to reach its destination. Routing is the act of transferring 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.

In a multicomputer network, routing can be accomplished through *centralized* or *distributed* control. In centralized routing, the designated node or controller makes the routing decision from knowledge of the status of all nodes in the network. It is possible for every node to have such global knowledge, in which case the source node can compute the best possible route. Such centralized routing can be used if a host computer performs routing decisions. The main problems with centralized routing are the need for a powerful central node to handle routing requests from all nodes and its vulnerability to failures. A centralized routing scheme is possible for very small network; for large network, it is impractical. This is why, most of the interconnection networks use distributed routing. In distributed routing, the path is determined in a distributed manner while the message travels across the network. Messages move from node to node, and routing decisions are made by intermediate nodes as messages move through these nodes. The distributed algorithm works with only local information about neighboring nodes. 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.
- *Deadlock and livelock freedom:* Ability to guarantee that packets will not block or wander across the network forever.
- *Adaptivity:* Ability to route packets through alternative paths in the presence of contention or faulty components.
- *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.

4.2.1 Resources and Allocation Units

To traverse an interconnection network, a message must be allocated resources: control state, channel bandwidth, and buffer capacity. When a packet arrives at a node, it must first be allocated some control state. The control state tracks the resources allocated to the packet within the node and the state of the packet's traversal across the node. To advance to the next node, the packet must be allocated bandwidth on an output channel of the node. Finally, a packet arrives at a node, it is temporarily held in buffer while awaiting channel bandwidth. Figure 4.1 shows the units in which network resources are allocated.

A *message* is a logically contiguous group of bits that are delivered from a source node to a destination node. Because messages may be arbitrarily long, resources are not directly allocated to messages. Instead, messages are divided into one or more packets that have fixed length. A packet is the smallest unit of data with complete routing information. Packets consist of one or more header flits and a number of data flits. The packet header includes routing information (RI) and, if needed, a sequence number (SN). The routing information is used to determine the route taken by the packet from source to destination. The sequence number is needed to record the packets of a message if they may get out of order in transit. This may occur, for example, if different packets follow different paths between the source and destination. If packet can be guaranteed to remain in order, the

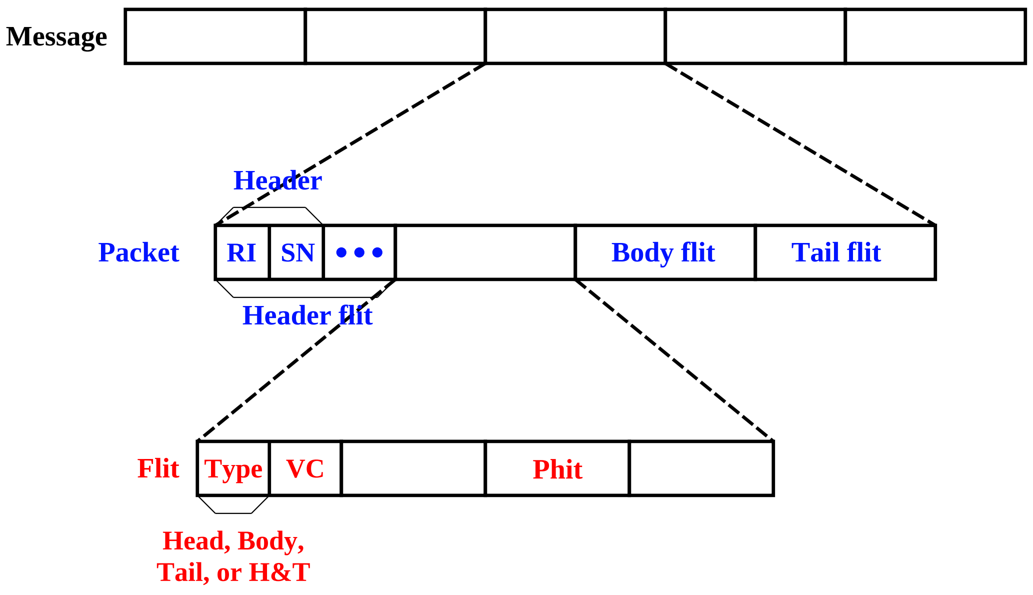


Figure 4.1: Units of resource allocation.

sequence number is not needed. A packet is further divided into **fl**ow control dig**it**s or *flits*. Flit is the smallest unit of data that a link can accept for transmission, i.e., a unit of bandwidth and storage allocation. Flits carry no routing and sequencing information and thus must follow the same path and remain in order. However, flits may contain a virtual channel identifier to identify which packets the flit belongs to in system where multiple packets may be in transit over a single physical link at the same time. A flit is itself subdivided in one or more **ph**ysiacl transfer dig**it**s or *phits*. A phit is the unit of information that can be transferred across a physical link in a single clock cycle. It is the number of bits that can be transferred in parallel in a single cycle.

There is no hard and fast rules about sizes. However, phits are typically between 1 bit and 64 bits in size, with 8 bits being typical. Flits usually range 16 bits (2 bytes) to 512 bits (64 bytes), with 64 bits (8 bytes) being typical. Finally, packets usually range from 128 bits (16 bytes) to 512 Kbits (64) Kbytes, with 1 Kbit (128 bytes) being typical.

The relationship between the size of phits, flits, and packets differs across machines. Many machines have the phit size equivalent to the flit size. In the IBM SP2 switch [11], a flit is 1 byte and is equivalent to a phit. Alternatively, in the CRAY T3D [30], a flit is comprised of eight 16-bit phits. The specific choices reflect trade-offs in performance, reliability, and implementation complexity.

4.2.2 Taxonomy of Routing Algorithm

Definition 4.1 ***(Routing Algorithm)*** *A routing algorithm determines the path a packet takes as it travels through the network from its source to its destination.*

Routing algorithms can be classified according to several criteria. Such as, adaptivity, progressiveness, minimality, number of paths, and so on.

Definition 4.2 ***(Deterministic Routing)*** *A routing algorithm is deterministic if the path taken by a packet is determined only by the source and destination of the packet. For a given pair of source and destination, the same path is always used even if there are multiple paths. It is also called oblivious routing.*

Definition 4.3 ***(Adaptive Routing)*** *A routing algorithm is adaptive if the path taken by a packet is determined by dynamic network condition (e.g., the presence of other packets in the network and presence of congested & faulty channels).*

Definition 4.4 ***(Progressive Routing)*** *A routing algorithm is progressive if the header flit moves forward, reserving a new channel at each routing operation.*

Definition 4.5 ***(Backtracking Routing)*** *A routing algorithm is backtracking if the header flit releases the previously reserved channels. Backtracking algorithms are mainly used for fault-tolerant routing*

Definition 4.6 ***(Minimal Routing)*** *A routing algorithm is minimal if the packets are routed along paths of minimal distance to their destination. Packets never move away from their destination.*

Definition 4.7 ***(Nonminimal Routing)*** *A routing algorithm is nonminimal if the packets temporarily move away from their destination (misrouting), but eventually arrive at their destination. Due to misrouting, the distance a packet travels may not be minimal.*

Definition 4.8 ***(Fully Adaptive Routing)*** *A routing algorithm is fully adaptive if all possible paths between source and destination are in potential use during routing message. It allows a packet to select a path from all of the possible paths to its destination.*

Definition 4.9 ***(Partially Adaptive Routing)*** *A routing algorithm is partially adaptive if a subset of all possible paths between source and destination are in use during routing message.*

4.2.3 Primitive Considerations

Wormhole Routing

Wormhole routing[1] [4, 37, 38] has become the dominant switching technique used in contemporary multicomputers. This is because it has low buffering requirements, and more importantly, it makes latency independent of the message distance. Implementations of wormhole routing typically divide each message into packets, which are then divided into flits. 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

[1]It is a flow control method and has nothing to do with routing.

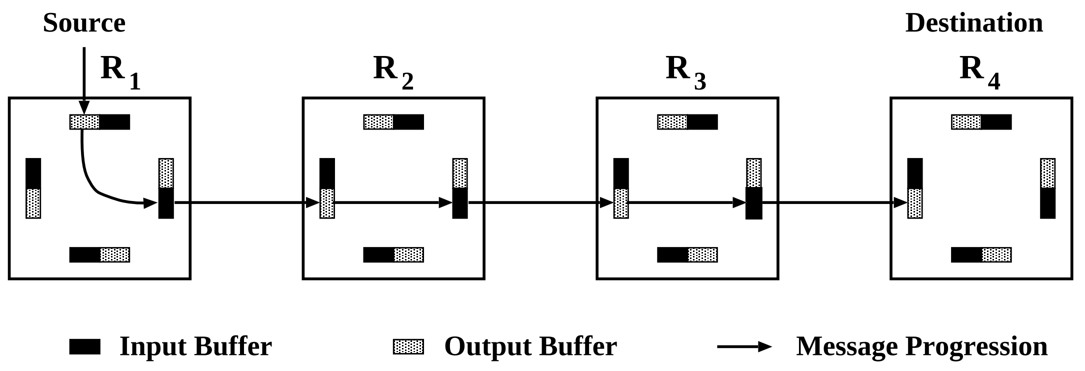


Figure 4.2: Wormhole routing

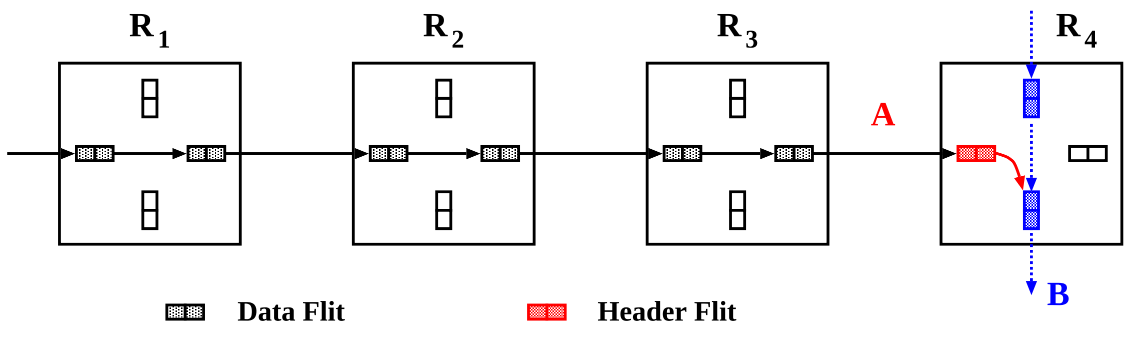


Figure 4.3: An example of the blocked wormhole-routed message

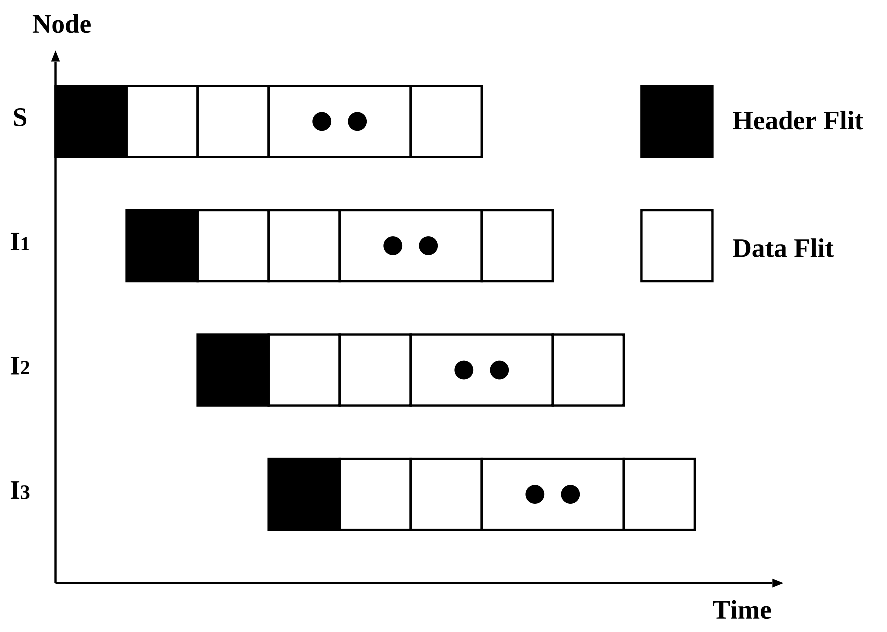


Figure 4.4: Time-space diagram of a wormhole-routed message

illustrated in Figure 4.2. 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 wormhole routing, message latency is insensitive of path length. When the header arrives at an intermediate router, the router immediately forwards the 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, Figure 4.3 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 Figure 4.4. 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, I_1, I_2, and I_3. 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.

Deadlock, Livelock, and Starvation

The nodes of an interconnection network send and receive packets through the router. In an interconnection network, packets usually travel across several intermediate nodes before reaching to its 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* occurs in an interconnection network when no message can advance towards its destination because of occupied channels and buffers [38, 113]. Many studies [85, 114–118] addressed the deadlock-free routing in multicomputer networks. Figure 4.5 shows an example of deadlock in wormhole routing involving four packets. Every packet is requesting resources held by other packet(s) while holding resources requested by other packet(s). All the packets involved in a deadlocked configuration are blocked forever. This situation is analogous to blocking a car that wants to continue straight behind a car that is waiting for a break in traffic to make a left turn.

Deadlock is a catastrophic to a network. After a few resources are occupied by deadlocked packets, other packets block on these resources, paralyzing network operation. There are three strategies to prevent this situation: *deadlock prevention*, *deadlock avoid-*

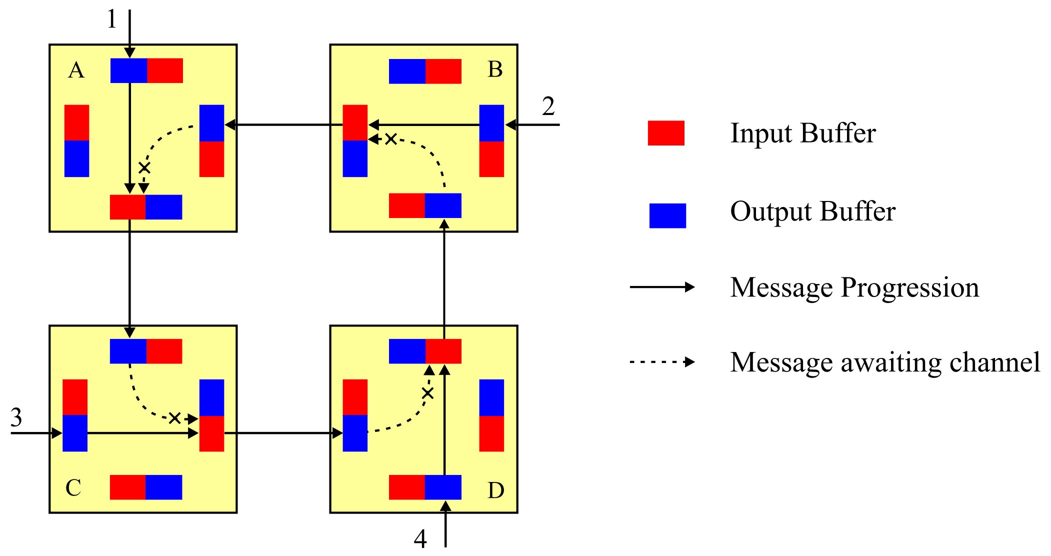


Figure 4.5: An example of deadlock involving four packets

ance, 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. Almost all modern networks use deadlock avoidance.

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* [5] . In livelock, packets continue to move through the network, but they do not make progress toward their destination. 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 routing 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. Even when the nonminimal paths are used, livelock can be prevented by limiting the number of misroutes of a packet away from its destination.

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* [5] 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.

Virtual Channel

Since wormhole routing relies on a blocking mechanism for flow control, deadlock can occur because of cyclic dependencies over network resources during message routing. *Virtual channel (VC)* [38, 39] was originally introduced to solve the problem of deadlock in wormhole-routed networks. Continuing our roadmap analogy, while the conflict situation is the deadlock, is solution is to add a lane in the left allowing the blocked car to make progress without waiting. 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.

A virtual channel consists of a buffer together with associated state information capable of holding one or more flits of a message [39]. They 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 [38]. Figure 4.6 shows the schematic diagram of virtual channels. Each virtual channel is 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.

In [5, 39], it has been shown that virtual channels can also be used to improve network performance and latency by relieving contention. It also increased the network throughput 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, Figure 4.7 shows two messages crossing the physical channel between router R_2 and R_3. Only one physical channel is used here. With no virtual channels packet A will prevent packet 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 packet 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 [119]. As illustrated in Figure 4.8, there are two virtual channels multiplexed over a 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

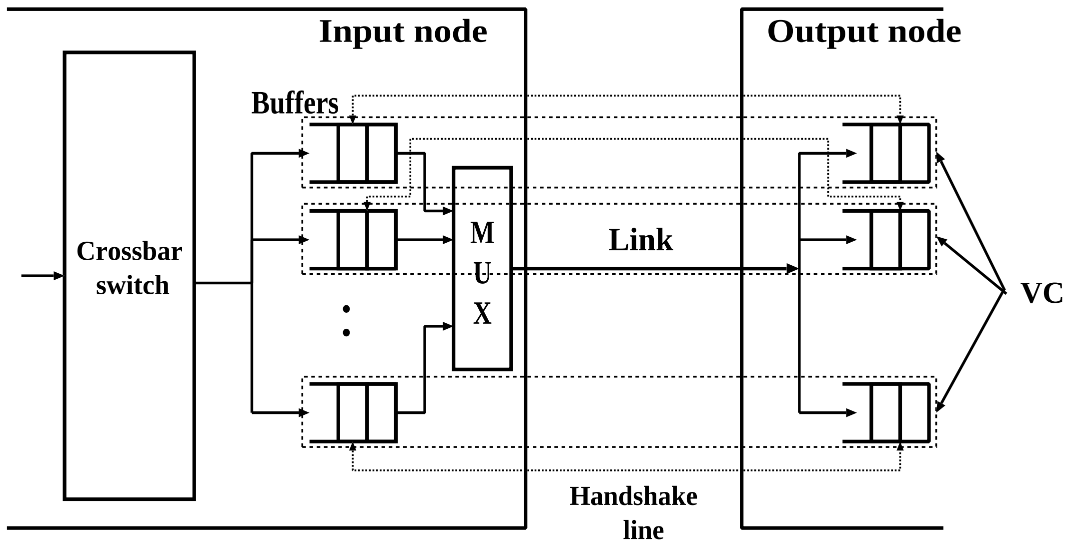


Figure 4.6: Virtual channel

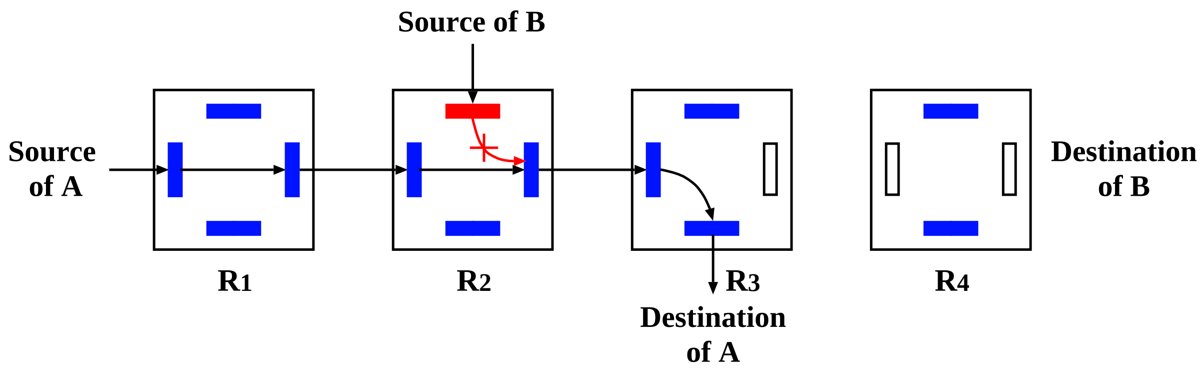


Figure 4.7: Message blocking while physical channels remain idle

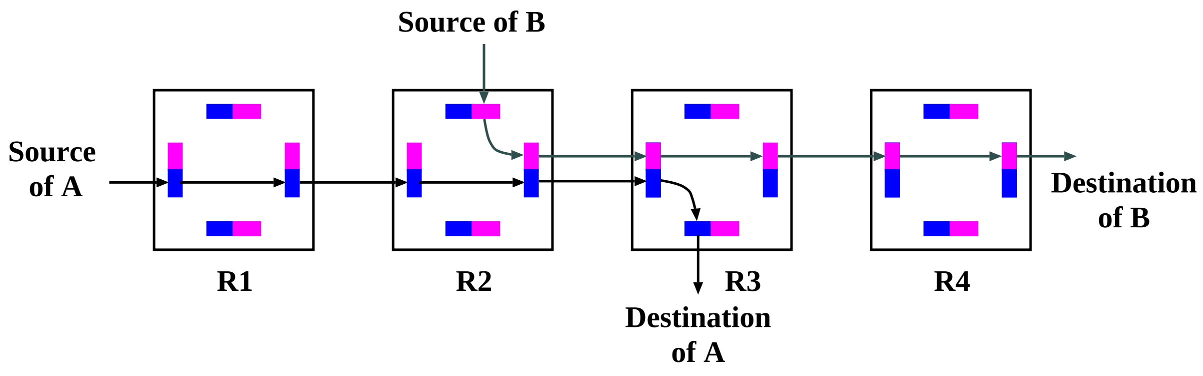


Figure 4.8: Virtual channel allows to pass blocked message

message latency. The network throughput will also be increased due to increased physical channel utilization.

It is often observed that increasing the number of virtual channels will increase the network performance. The advantages and disadvantages of using virtual channels have been thoroughly investigated [40, 41, 120–123]. Performance is dependent on network parameters and there is an optimal number of virtual channels where the network performance is maximized [5, 39].

4.2.4 Channel Dependency Graph

Wormhole routing is particularly susceptible to deadlocks as a blocked communication request can be holding several communication channels, potentially causing cyclic waits for channels. The resources which cause deadlocks in wormhole routing are the communication channels. The widely-used approach to deadlock avoidance in wormhole routing consists in splitting the physical channels into virtual channels. Deadlocks are prevented by imposing constraints on the allocation of the virtual channels to communication requests.

The theoretical model of deadlock avoidance presented relies on the concept of channel dependency graph [38, 67]. When a packet is holding a channel, and if 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-routed network.

The channel dependency graph is constructed as follows: Each virtual channel in the network is represented by a vertex. A directed edge is introduced from vertex i to vertex j if and only if the routing algorithm allows a packet arriving on virtual channel i to be forwarded on virtual channel j. Dally and Setz [38] showed that deadlocks can occur only if the channel dependence graph of the channel allocation scheme has a directed cycle. They presented some definitions and a theorem to make the routing algorithm deadlock free. Here, we borrowed those definitions and theorem for clarity.

Definition 4.10 *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 edges (C) of the graph represent the set of communication channels.*

Definition 4.11 *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 4.12 *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 4.13 *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 4.14 *A deadlock configuration for a routing function R is a nonempty legal configuration of channel queues $\ni$*

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

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 $\boldsymbol{R}$ (deterministic) for an interconnection network $\boldsymbol{I}$ is deadlock free if and only if there are no cycles in the channel dependency graph $\boldsymbol{D}$.*

A cycle in the channel dependence graph indicates that it is possible for a deadlock to occur; it is necessary but not sufficient condition for deadlock. A common strategy to avoid deadlock is to remove all cycles from the channel dependency graph. 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 4.9(a) illustrates the phenomena of channel dependency graph and breaking the cycle by using virtual channels in a four-node 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 Figure 4.9(a) is shown in Figure 4.9(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 Figure 4.9(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 Figure 4.9(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

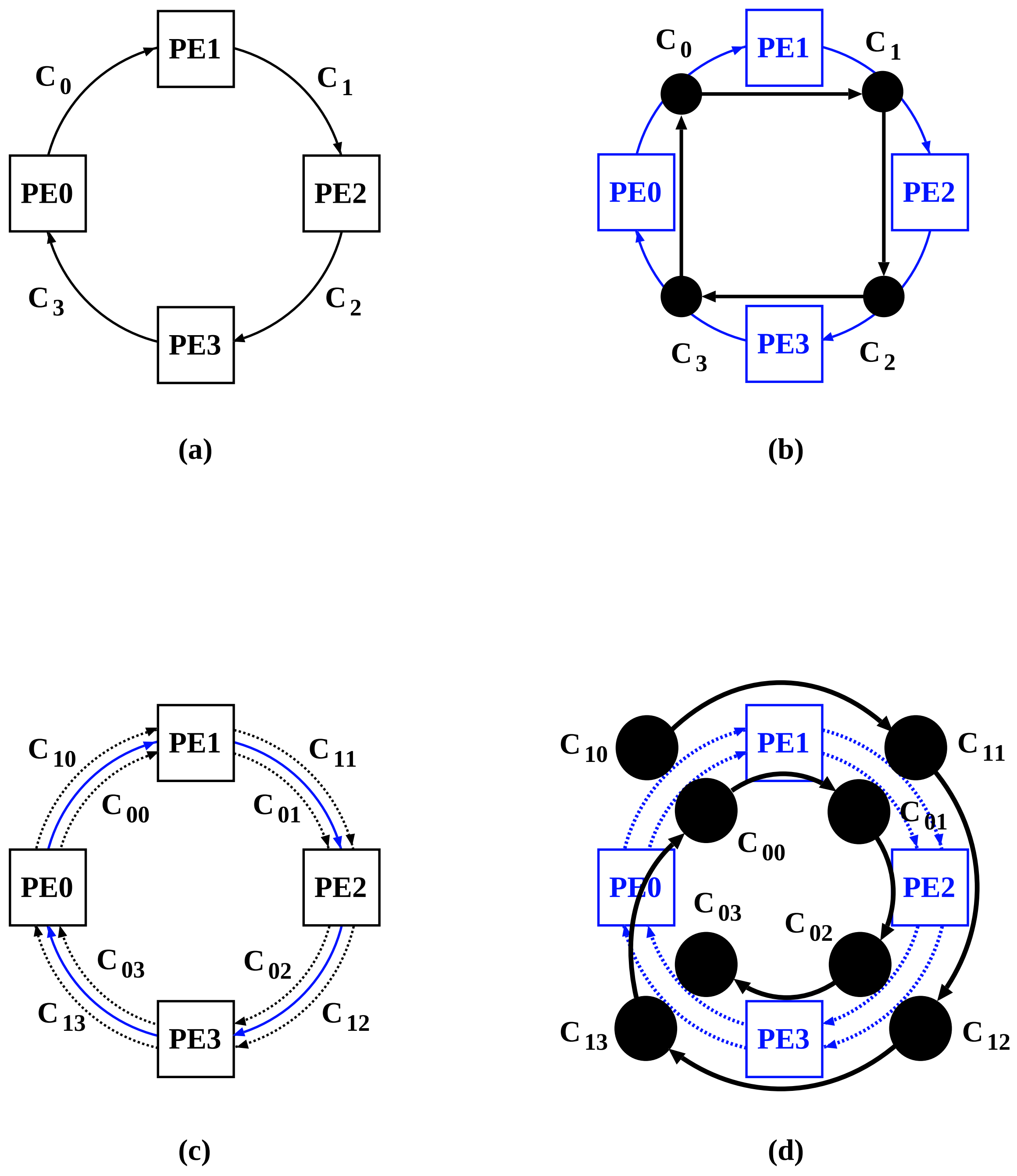


Figure 4.9: (a) A ring network with unidirectional channels. (b) The associated channel dependency graph contains a cycle. (c) Each physical channel is logically split into two virtual channels. (d) A modified channel dependency graph without cycles.

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 4.9.(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.

Although the channel dependency graph is elegantly detect the deadlocks in wormhole-routed networks, it is sometimes tedious to construct the graph and check for cycles.

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 [113]. All the possible turns a message can make are shown in Figure 4.10. The deadlock situation in mesh network are prevented by proper routing algorithms.

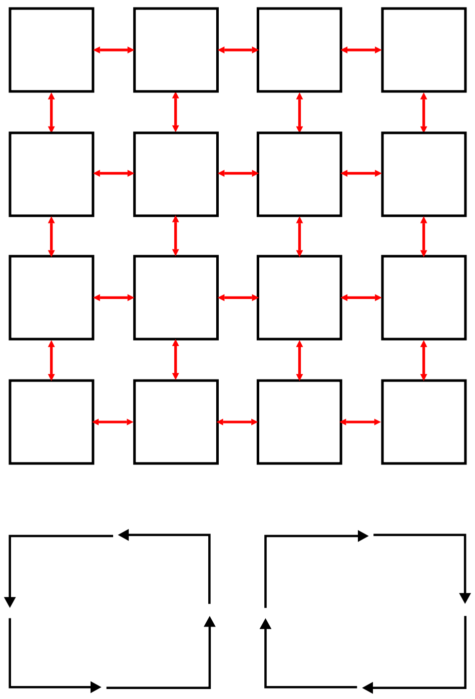


Figure 4.10: Deadlock configuration in mesh network

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 each dimension [113]. The possible deadlock configurations for a torus network are shown in Figure 4.11. Additional virtual channel is required to break the wrap-around dependencies of the torus network.

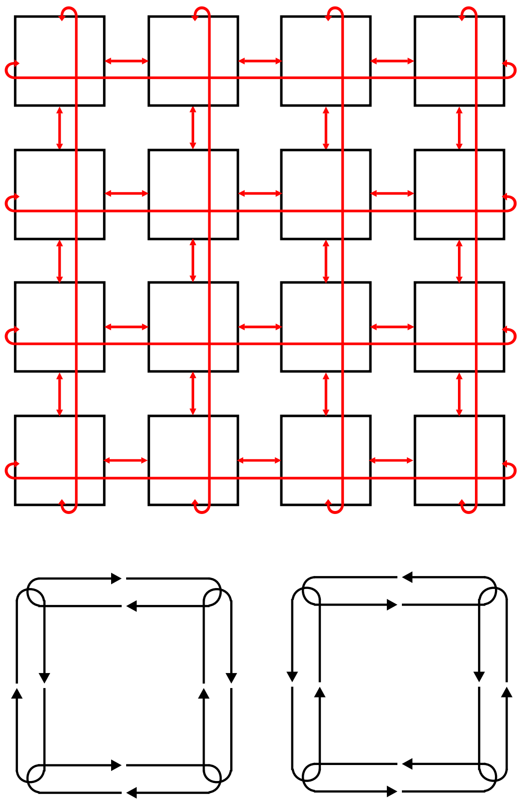


Figure 4.11: Deadlock configuration in torus network

Necessity of Deadlock-Free Routing

Routing algorithms for interconnection network aim to minimize message blocking by efficiently utilizing network virtual and physical channels while ensuring freedom from deadlock. Deadlock in an interconnection network is the situation in which some packets can not advance forever because of blocking by other packets. If a deadlock occurs, packet delivery is delayed indefinitely. In addition to this packet delivery rate is also reduced. In short, once a deadlock has occurred, the dynamic communication performance is drastically reduced, which is undesirable. A good routing algorithm for wormhole-routed network must reduce message latency and increase network throughput as much as possible with freedom from deadlock.

4.3 Dimension-Order Routing (DOR) for HTN

Dimension order routing algorithm is very popular and receives several names, like *XY* routing [4] (for 2D-mesh network) or *e-cube* [38] (for Hypercube network). In the dimension order routing, the header flit 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, routes a packet successively in each dimension until the distance in that dimension is zero, then proceeds to the next dimension. The dimension order routing algorithm determines the only route for the source and destination pair. Figure 4.12 illustrates the mechanism of dimension order routing in a 2D-mesh network.

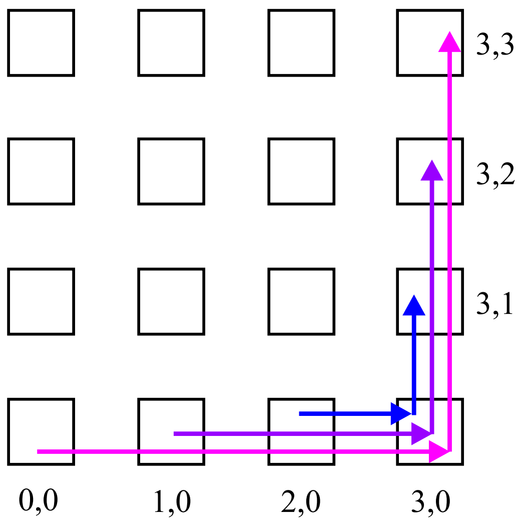


Figure 4.12: A set of routing paths created by the dimension order routing in a 2D-mesh network

Deadlocks are mainly avoided by using a proper routing algorithm within the network. One approach to designing a deadlock-free routing for a wormhole-routed network is to ensure that cycles are avoided in the channel dependency graph [38]. This can be achieved by assigning each channel a unique number and allocating channels to a 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 splitted 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.

Dimension-order routing has been popular in multicomputers because it has minimal hardware requirements and allows the design of simple and fast routers. Additionally, switches can be decomposed into smaller and faster switches, thus increasing speed. It is well suited for uniform traffic pattern. Although there are numerous paths between any source and destination, dimension-order routing defines a single path from source to destination; it cannot respond to dynamic network condition.

4.3.1 Routing Algorithm for HTN

Wormhole routing is used for switching. In this section, we have considered popular deterministic, dimension order routing algorithm for routing messages. We recall the routing algorithm stated in Section 3.2.3 . Routing of messages in the HTN is performed from top level to bottom level as in TESH network [47, 124, 125]. That is, it is first done at the highest level network; then, after the packet reaches its highest level sub-destination, routing continues 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's destination is 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. As mentioned earlier, We have considered a simple dimension-order routing algorithm. Routing at the higher level is performed first in the y-direction and then in the x-direction. In a BM, the routing order is initially in the z-direction, next in the y-direction, and finally in the x-direction.

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.
- *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 4.13 shows the routing algorithm (R_1) 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 Figure 4.14.

4.3.2 Deadlock-free Routing

The proposed routing algorithm enforces some routing restrictions to avoid deadlocks [38, 67]. Since dimension-order routing is used in HTN, routing at the higher level is performed first in the y-direction and then in the x-direction. In a BM, the routing order is initially in the z-direction, then in the y-direction, and finally in the x-direction.

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

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. The channels are allocated according to Eq. 4.6 for a 3D-torus network. Initially, messages are routed over virtual channel 0. Then, messages are routed over virtual channel 1 if the message is going to use a wrap-around channel. If the messages routing is enforced and virtual channels are used according to the above mentioned phenomena, then cyclic dependency will not occur. Therefore, freedom from is proved.

```
Routing HTN(s,d);
source node address:$s_\alpha, s_{\alpha-1}, s_{\alpha-2}, ..., s_1, s_0$
destination node address: $d_\alpha, d_{\alpha-1}, d_{\alpha-2}, ..., d_1, d_0$
tag: $t_\alpha, t_{\alpha-1}, t_{\alpha-2}, ..., t_1, t_0$
  for $i = \alpha : 3$
    if $(i/2 = 0$ and $(t_i > 0$ or $t_i = -(n-1)))$, routedir = North; endif;
    if $(i/2 = 0$ and $(t_i < 0$ or $t_i = \ \ (n-1)))$, routedir = South; endif;
    if $(i\%2 = 1$ and $(t_i > 0$ or $t_i = -(n-1)))$, routedir = East; endif;
    if $(i\%2 = 1$ and $(t_i < 0$ or $t_i = \ \ (n-1)))$, 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;
        if $(t_i > 0)$, $t_i = t_i - 1$; endif;
        if $(t_i < 0)$, $t_i = t_i + 1$; endif;
      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 $t_i \leq \frac{m}{2})$ or $(t_i < 0$ and $t_i = -(m-1))$, movedir = positive; endif;
    if $(t_i > 0$ and $t_i = (m-1))$ or $(t_i < 0$ and $t_i \geq -\frac{m}{2})$, movedir = negative; endif;
    if (movedir = positive and $t_i > 0$), distance = $t_i$; endif;
    if (movedir = positive and $t_i < 0$), distance = $m + t_i$; endif;
    if (movedir = negative and $t_i < 0$), distance = $t_i$; endif;
    if (movedir = negative and $t_i > 0$), distance = $-m + t_i$; 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 = negative), 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 = negative), 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 = negative), move packet to $-x$ node; distance$_0$ = distance$_0 + 1$; endif;
  endwhile;
end
```

Figure 4.13: Dimension-order routing algorithm for HTN

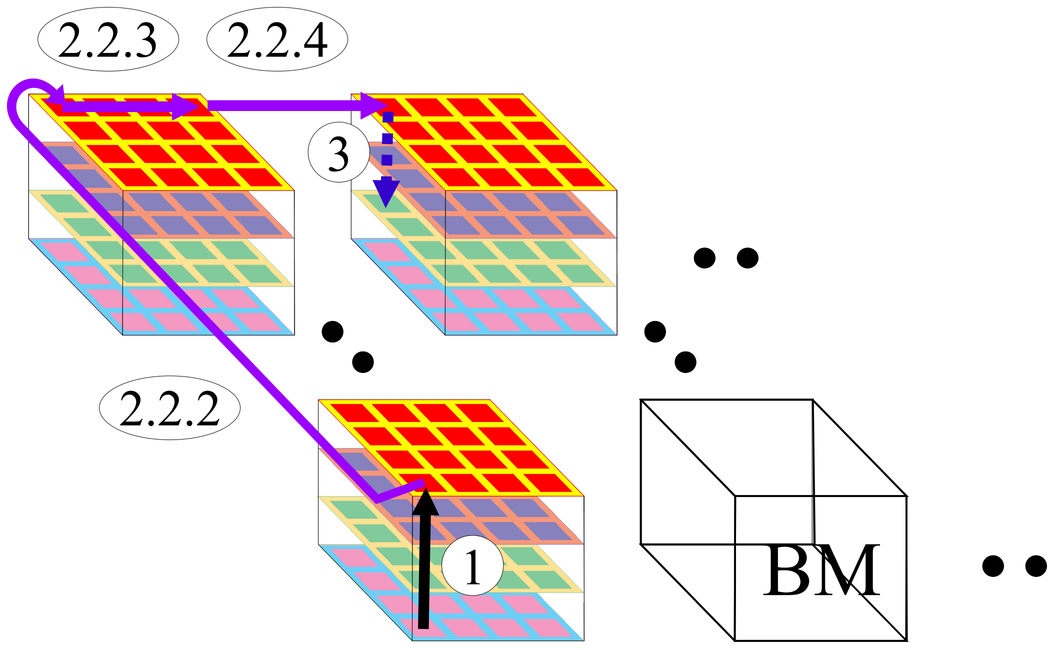


Figure 4.14: An example of message routing in HTN

$$C = \begin{cases} (l, vc, a_2), & z+ \text{ channel}, \\ (l, vc, m - a_2), & z- \text{ channel}, \\ (l, vc, a_1), & y+ \text{ channel}, \\ (l, vc, m - a_1), & y- \text{ channel}, \\ (l, vc, a_0), & x+ \text{ channel}, \\ (l, vc, m - a_0), & x- \text{ channel} \end{cases} \tag{4.6}$$

Here, $l = \{l_0, l_1, l_2, l_3, l_4, l_5\}$ are the links used in the BM, $l = \{l_0, l_1\}$, $l = \{l_2, l_3\}$, and $l = \{l_4, l_5\}$ are the links used in the z–direction, y–direction, and x–direction interconnection, respectively. $vc = \{VC_0, VC_1\}$ are the virtual channels, m is the size of the BM, and a_0, a_1, and a_2 are the node 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 deadlock freeness is proved.

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

Here, $l = \{l_6, l_7\}$ are the links used for higher-level interconnection, l_6 is used in the interconnection of the east and west directions and l_7 is used in the interconnection of the north and south directions. $vc = \{VC_0, VC_1\}$ are the virtual channels, n is the size of the higher level networks, and a_{2L} and a_{2L-1} are the node addresses in the higher level, where L is the level number.

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 phase-1 and phase-3 routing is 2. Intra-BM links between inter-BM communication on the xy-plane of the BM are used in sub-phases $2.i.1$ and $2.i.3$. These sub-phases utilize channels over intra-BM links, sharing either the channels of phase-1 or phase-3. The exterior links at the contours of the xy-plane 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 phase-2 routing is 2.

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

4.3.3 Minimum Number of Virtual Channels

The most expensive part of an interconnection network is the wire that forms the physical links; for a particular topology, the physical link cost is constant. The second most expensive elements are the buffers and switches. Since the networks we consider are wormhole-routed, the main factor in buffer expense is the number of virtual channels. Virtual channels reduce the effect of blocking; they are used widely in parallel computer systems, to improve dynamic communication performance by relieving contention and to design deadlock-free routing algorithms.

The use of virtual channels requires a careful analyses in order to maximize the benefits and minimize the drawbacks. In [39], Dally showed that the performance of a dimension-order routing under uniform traffic load improves significantly as virtual channels are initially added. The benefits then diminish as more channels are added. Aoyama and Chien [120] compare the cost of adding virtual channels for wormhole-routed 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 switch controller speed, which is determined by the routing 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 channels 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. Since the hardware cost increases as the number of virtual channels increases, the unconstrained use of virtual channels is not cost-effective in parallel computers [122, 123]. 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.

HTN is a hierarchical network, where both the BM and the higher level networks are torus connected. In torus network, 2 virtual channels are necessary to make the routing algorithm deadlock free. By sharing the virtual channels between different *phase* of routing, we can reduce the number of virtual channels. The routing of the message in source-BM and destination-BM is carried out separately. The virtual channels required in *phase*-1 and *phase*-3 can share each other. Therefore, it reduces 2 virtual channels. The mesh connection of the higher level 2D-torus network shares the virtual channel of either *sub-phase*-2.i.1 or *sub-phase*-2.i.3. Again, it reduces one more virtual channel. The channel required for the wrap-around connection of the higher level 2D-torus network can also be shared with that of either *phase*-1 or *phase*-3. This phenomena are described in the subsequent theorem.

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 routing is 2. The routing of the message in source-BM and destination-BM is carried out separately. Thus, 2 virtual channels are shared in phase-1 and phase-3 routing. In phase-2 routing, packets are routed in the higher level networks. The higher level networks of the HTN is a 2D-torus network. According to Corollary 4.1, the number of necessary virtual channels for phase-2 routing is 2. Intra-BM links between inter-BM communication on the xy-plane of the BM are used in sub-phases 2.i.1 and 2.i.3. These sub-phases utilize channels over intra-BM links, sharing either the channels of phase-1 or phase-3. The exterior links at the contours of the xy-plane of the BM are used in sub-phase 2.i.2 and sub-phase 2.i.4. The required 2 virtual channels of these two sub-phases are also cleverly shared with either phase-1 or phase-3 routing. The main idea for this sharing is that initially 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.

The minimum number of virtual channels that required to make the routing algorithm deadlock free for the HTN using dimension order routing is 2. However, by the use of the well known Up*/Down* routing algorithm [126], only one channel (i.e., no virtual channel) is sufficient to avoid deadlocks. This kind of routing algorithm is suitable for tree topology. Routing of messages using the Up*/Down* routing algorithm is beyond the scope of this paper.

4.4 Dynamic Communication Performance using DOR

The overall performance of a multicomputer system is affected by the performance of the interconnection network as well as by the performance of the node. Continuing advances in VLSI/WSI technology promise to deliver more power to the individual node. On the other hand, low performance of the communication network will severely limit the speed of the entire multicomputer system [127]. Therefore, the success of massively parallel computers is highly dependent on the efficiency of their underlying interconnection networks.

4.4.1 Performance of Interconnection Networks

Performance Metrics

The dynamic communication performance of a multicomputer is characterized by message latency and network throughput. Message latency is the time required for a packet to traverse the network from source to destination. It refers to the time elapsed from the instant when the first flit is injected to the network from the source to the instant when the last flit of the message is received at the destination. Message latency in wormhole routing is the average value of the time elapsed between injection of the header flit into the network at the source and reception of the last unit of the data flit at the destination. Network throughput is the rate at which packets are delivered by the network for a particular traffic pattern. It refers to the maximum amount of information delivered per unit of time through the network. It also can be defined as the maximum traffic accepted by the network. For the network to have good performance, low latency and high throughput must be achieved.

Latency is measured in time units. However, when comparing several design choices, the absolute value is not important; because the comparison is performed by computer simulation, latency is measured in simulator clock cycles. Throughput depends on message length and network size. Therefore, throughput is usually normalized, dividing it by message length and network size. When throughput is compared by computer simulation and wormhole routing is used for switching, throughput can be measured in flits per node and clock cycle.

Zero-load latency gives a lower bound on the average latency of a packet through the network. The zero-load assumption is that a packet never contends for network resources with other packets. Under this assumption, the average latency of a packet is its serialization latency plus its hop latency. Throughput is a property of the entire network and depends on routing and flow control as much as on the topology. A resource is in saturation when the demands being placed on it are beyond its capacity for servicing those demands. A channel become saturated when the amount of data that wants to be routed over the channel exceeds the bandwidth of the channel. The saturation throughput of a network is the smallest rate of traffic for which some channel in the network becomes saturated. If no channels are saturated, the network can carry more traffic. We also called this saturation throughput as maximum throughput .

Performance Measures

Performance of an interconnection network is described by a latency vs. throughput curve. This curve shows the average latency of a packet as a function of network throughput. To draw a particular latency vs. throughput curve, the traffic pattern must also be specified. Latency vs. throughput graphs share a distinctive shape that starts at the horizontal asymptote of zero-load latency and slopes upward to the vertical asymptote of maximum throughput. At low throughput, latency approaches zero-load latency. As load increases, increased contention causes latency to increase as packets must wait for resources (buffers and channels). Eventually, the latency curve approaches a vertical asymptote as the throughput approaches the saturation throughput of the network.

4.4.2 Simulation Environment

To evaluate dynamic communication performance, we have developed a wormhole routing simulator. In our simulator, the parameters affecting simulation results are as follows:

1. **Network Topology**
   Dynamic communication performance is evaluated for the HTN, H3D-mesh, H3D-torus, TESH, 2D-mesh, and 2D-torus networks.

2. **Network Size**
   We have considered 256, 512, and 1024 node networks for simulation.

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

4. **Routing Strategy**
   We have chosen dimension-order routing algorithm. The dimension order routing algorithm provides the only route for the source and destination pair. In the next section, we will consider adaptive routing for performance evaluation.

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

6. **Arbitration of Virtual Channels**
   The virtual channels are arbitrated by a round-robin algorithm. A round-robin arbiter operates on the principle that a request that was just served should have the lowest priority on the next round of arbitration.

7. **Message Length**
   A short message is equal to 16 flits. Similarly, medium and long message are equal to 64 flits and 256 flits, respectively [5]. Simulation studies were carried out for short, medium, and long messages. We considered 3 different message lengths to show the effect of message lengths; two flits were used as the header flit. To evaluate dynamic communication performance, flocks of messages were sent in the network to compete for the output channels. For each simulation run, we considered that message generation rate was constant and the same for all the nodes.

8. **Traffic Pattern**
   We have considered both uniform and non-uniform traffic pattern. In the subsequent section, we will discuss details about traffic patterns.

9. **Simulation Time**
   Flits were transmitted at $20,000$ cycles. In each clock cycle, one flit was transferred from the input buffer to the output buffer or from output to input if the corresponding buffer in the next node was empty. Therefore, transferring data between two nodes took 2 clock cycles.

4.4.3 Traffic Patterns

One of the most important factors influencing network performance is the traffic pattern generated by the applications being executed on a machine. Traffic patterns are pairs of nodes that communicate. In an interconnection network, sources and destinations for messages form the traffic pattern. The traffic model is basically defined by three parameters: message injection time, message length and message destination address distribution [128]. Message destination distributions vary a great deal depending on the network topology and the application's mapping onto different nodes. The most frequently used, simplest and most elegant pattern is the uniform traffic where the source and the destination are randomly selected. However, depending on the characteristics of the application, some nodes may communicate with each other more frequently than with others. Also, traffic patterns are critically dependent on the placement of tasks on the networks. Consequently, non-uniform traffic patterns are frequent and cause uneven usage of traffic resources, significantly degrading the dynamic communication performance of the network. We have evaluated the dynamic communication performance of the HTN under the uniform traffic pattern. To observe the effect of non-uniformity, we also interested in how non-uniform traffic patterns affect the dynamic communication performance of the HTN. Throughout the performance evaluation, we use five different traffic patterns: uniform [129], hot-spot [130], dimension-reversal [70, 71], bit-reversal [131], complement [132], bit-flip [133], and perfect shuffle [134] .

- **Uniform** – In the uniform traffic pattern, every node sends messages to every other node with equal probability in the network. That is, source and destination are randomly selected.

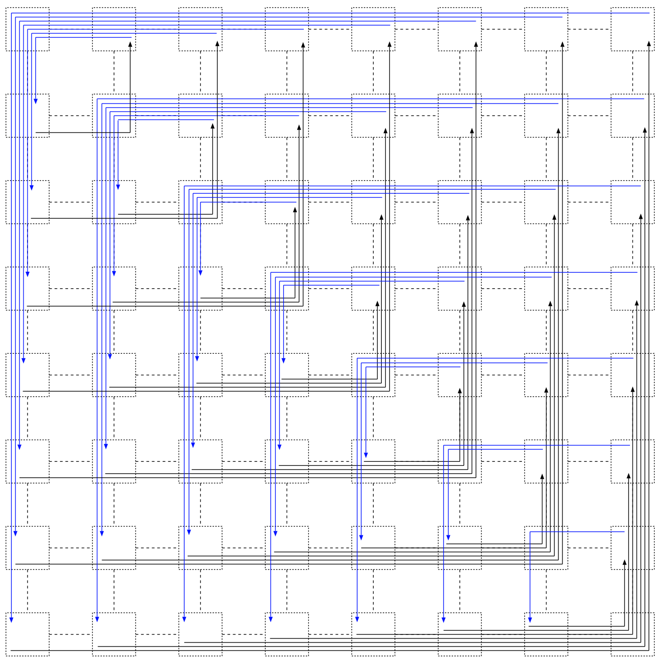
- **Hot-Spot** – A hot spot is a node that is accessed more frequently than other nodes in the uniform traffic distribution. In hot-spot traffic pattern, each node generates a random number. If that number is less than a threshold, the message is sent to the hot spot node. Otherwise it is sent to other nodes with a uniform distribution.

- **Dimension reversal** – In dimension reversal traffic, each node sends messages to a node with an address of the reversed dimension index. In 2-dimensional networks, node(x, y) communicates with node(y, x). This gives the same traffic as a matrix-transpose traffic pattern. For 3-dimensional networks, we use an analogous traffic pattern, in which node(x, y, z) sends messages to node $\left(y, x, \sqrt[3]{N} - (z+1)\right)$. Here, N is the total number of nodes in the network. In 4-dimensional networks, node(x, y, z, w) communicates with node(y, x, w, z).

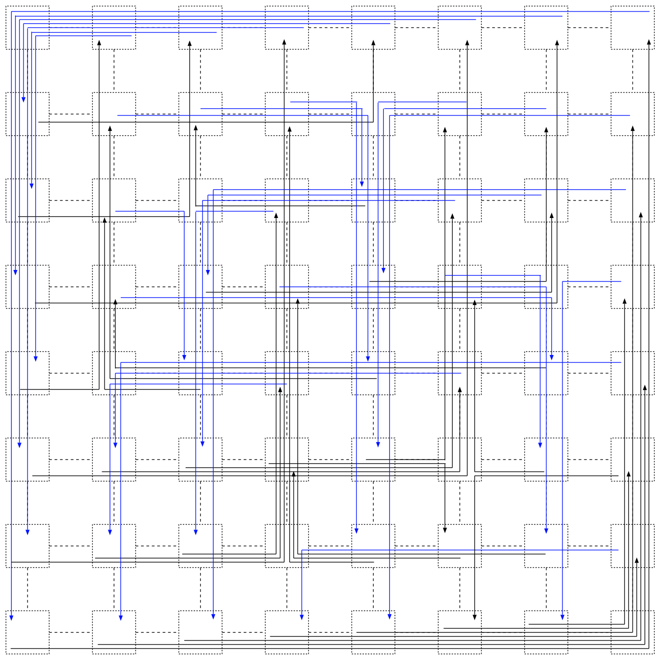
- **Bit-reversal** – The binary representation of the node address is $a_{\beta-1}, a_{\beta-2}$ ... ... a_1, a_0. In bit-reversal traffic, the node $(a_{\beta-1}, a_{\beta-2} \dots \dots a_1, a_0)$ communicates with the node $(a_0, a_1, a_2 \dots \dots a_{\beta-2}, a_{\beta-1})$.

- **Complement** – The binary representation of the node address is $a_{\beta-1}, a_{\beta-2}$ ... ... a_1, a_0. In complement traffic, the node $(a_{\beta-1}, a_{\beta-2} \dots \dots a_1, a_0)$ communicates with the node $(\overline{a_{\beta-1}}, \overline{a_{\beta-2}}, \dots \dots \overline{a_2}, \overline{a_1}, \overline{a_0})$.

- **Bit-flip** – The node with binary coordinates $a_{\beta-1}, a_{\beta-2} \dots \dots a_1, a_0$ communicates with the node $(\overline{a_0}, \overline{a_1}, \dots \dots \overline{a_{\beta-2}}, \overline{a_{\beta-1}})$. That is, complement of bit-reversal traffic.



(a)



(b)

Figure 4.15: Nonuniform traffic patterns on a 8 × 8 mesh networks: (a) dimension-reversal traffic and (b) bit-reversal traffic

- **Perfect shuffle** – The node with binary coordinates $a_{\beta-1}, a_{\beta-2}$ ... ... a_1, a_0 communicates with the node $(a_{\beta-2}, a_{\beta-3},$ ... ... $a_1, a_0, a_{\beta-1})$. That is, rotate left 1 bit.

The uniform traffic pattern has been used to evaluate the network's communication performance in many previous studies and therefore, provides an important point of reference. When a hot spot occurs, a particular communication link experiences a much greater number of requests than the rest of the links – more than it can service. In a remarkably short period of time, the entire network may become congested resulting in serious performance degradation due to saturation. Hot spots are particularly troublesome because they may result from the cumulative effects of very small traffic imbalances. Hot spots often occur because of the bursty nature of program communication and data requirements and, therefore can provide a benchmark for interconnection networks. Dimension order routing has only one route for a source-destination pair. And in dimension reversal and bit reversal traffic patterns, the source node sends messages to a predefined destination. They both create significant congestion under dimension order routing in the network, and when congestion occurs, the network throughput decreases precipitously. Dimension reversal permutations is often generated by numerical programs and are considered as interesting benchmarks for the interconnection networks. Bit Permutation and Computation (BPC) [135, 137] is a class of non-uniform traffic pattern, which are very common in scientific applications. These include bit-reversal, bit-flip, and perfect shuffle, etc. BPC communication patterns take into account the permutations that are usually performed in parallel numerical algorithms [134, 138]. These distributions achieve the maximum degree of temporal locality and are also considered as benchmarks for interconnection networks. Permutation traffic is exhibited in many parallel applications such as computing multidimensional FFT, matrix problems, finite elements and fault-tolerant routing [137]. While these traffic patterns are by no means exhaustive, they represent some interesting extremes of the space of possible patterns.

As an example, Figure 4.15 shows the dimension-reversal and bit-reversal traffic patterns on 8×8 mesh networks. The paths shown in the figure follow dimension-order routing. While the two nonuniform traffic generate a similar number of channel conflicts, they concentrate the conflicts in different places. In dimension-reversal traffic in Figure 4.15(a), most conflicts arise along the diagonal line connecting upper-left and lower-right corners. However, as shown in Figure 4.15(b), bit-reversal traffic scatters the locations of the conflicts in the center. Consequently, various nonuniform traffic patterns behave differently under different routing algorithms.

4.4.4 Dynamic Communication Performance Evaluation

Most of the hierarchical interconnection networks proposed in the literature are based on hypercube network; a few of them are based on k-ary n-cube network. Available k-ary n-cube based hierarchical networks are TESH, H3D-mesh, H3D-torus, SRT [96], and RDT [95]. From Table 4.1, it is seen that the number of links of the SRT and RDT is higher than that of the HTN. Again, the presence of huge number of diagonal links results higher peak number of vertical links for 3D-WSI than that of HTN. This is why, we did not consider SRT and RDT for performance evaluation. We have consider H3D-mesh,H3D-torus, and TESH networks along with original k-ary n-cube networks for performance evaluation.

Table 4.1: The total number of links of various networks with 1024 node

Network	# of links
2D-mesh	1984
2D-torus	2048
H3D-mesh	3168
HTN	3200
SRT	3904
RDT	4096

Comparing dynamic communication performance among different hierarchical interconnection networks such as HTN, H3D-mesh [59], H3D-torus [61], and TESH [47] is not an easy task, because each network has a different interconnection architecture, which makes it difficult to match the total number of nodes. 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, and Level-2 H3D-mesh with $n = 2$ [59], and 16×16 mesh and torus networks also have 256 nodes. If $m = 4$, $n = 4$, and $L = 2$, then the total number of nodes in the HTN is 1024. Again, Level-2 H3D-mesh with $n = 4$, and 32×32 mesh and torus networks also have 1024 nodes. If the size of the basic module is $(4 \times 4 \times 4)$ and the higher level network is $(2 \times 2 \times 2)$, then the total number of nodes of a Level-2 H3D-torus network is 512. We have considered a rectangular HTN to match the total number of nodes with H3D-torus network. In this rectangular HTN, the size of the BM is $(4 \times 4 \times 4)$ and the higher level network is (2×4); hence the total number of nodes of a Level-2 rectangular HTN is also 512. In this paper, 256-node, 512-node, and 1024-node networks are referred to as small, medium, and large-size networks, respectively. According to the structure of the TESH network [47], it is not possible to construct a 1024-node TESH network.

In this section, we have shown some simulation results of algorithm proposed in Section 4.3.

Uniform Traffic Pattern

In a uniform traffic pattern , each node sends with equal probability to all other nodes in the networks. Figure 4.16 shows a comparison of dynamic communication performance of HTN with H3D-mesh network and mesh network under uniform-random traffic. The figures show the average transfer time as a function of network throughput for different networks. Each curve stands for a particular network. The inter-level connectivity is $q = 0$. 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, H3D-mesh, and HTN, respectively. Thus, a fair comparison of the communication performance of these families of interconnection networks is not shown in Figure 4.16.

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 com-

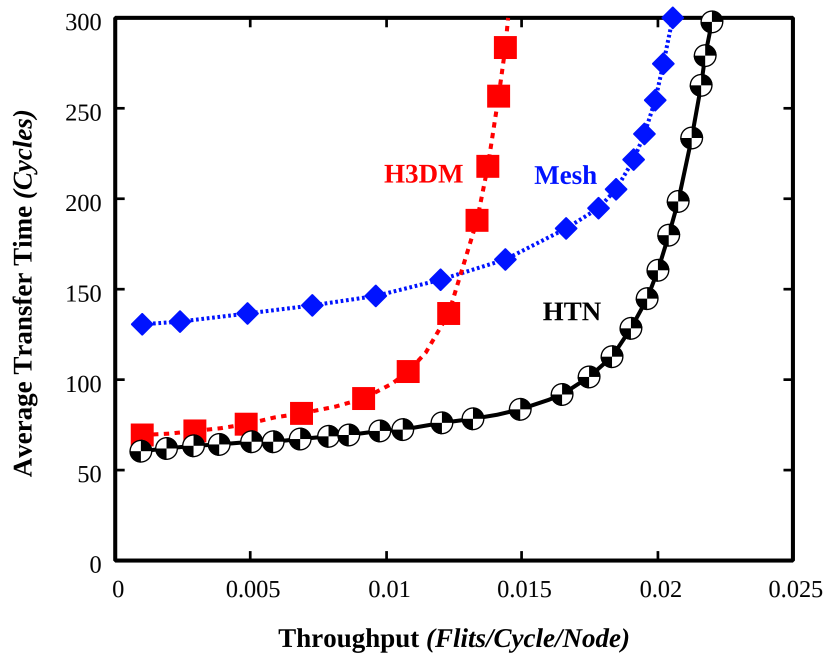


Figure 4.16: Dynamic communication performance of dimension-order routing with uniform traffic pattern on various networks: 1024 nodes, different virtual channels, short message, and $q = 0$

.

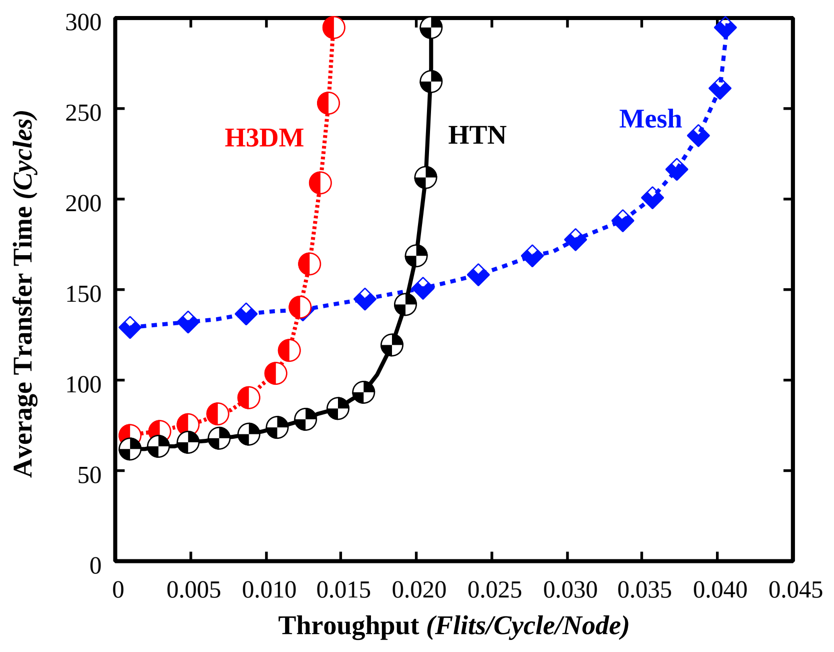


Figure 4.17: Dynamic communication performance of dimension-order routing with uniform traffic pattern on various networks: 1024 nodes, 3 virtual channels, short message, and $q = 0$.

munication performance between various networks. Figure 4.17 shows a comparison of dynamic communication performance between HTN, H3D-mesh network, and mesh network with 3 virtual channels. Figure 4.17 shows that the performance (both the average transfer time and throughput) of HTN is better than that of H3D-mesh network. The average transfer time of the HTN is less than that of mesh network; but the throughput of HTN is less than mesh network. To improve the throughput of HTN, inter-level connectivity (q) of HTN is increased from $q = 0$ to $q = 1$. With this condition, we have evaluated the dynamic communication performance of HTN, H3D-mesh, H3D-torus, TESH, and mesh networks and presented in Figure 4.18.

Figure 4.18 depicts the result of simulations under uniform traffic pattern for the various network models. As shown in Figure 4.18(a), for small-size networks, the average transfer time of the HTN is lower than that of the H3D-mesh, TESH, and mesh networks, and the maximum throughput of the HTN is higher than that of the H3D-mesh, TESH, and mesh networks. As shown in Figure 4.18(b), for medium-size networks, the average transfer time of the HTN is lower than that of the H3D-torus [61] network with 1 link and 4 links, and its maximum throughput is far higher than that of the H3D-torus network with 1 link and 4 links. As shown in Figure 4.18(c), for large-size networks, the average transfer time of the HTN is lower than that of the H3D-mesh and mesh networks, and its maximum throughput is higher than that of the H3D-mesh and mesh networks. As shown in Figure 4.18(d), for medium-length message large-size networks, the average transfer time of the HTN is lower than that of the H3D-mesh and mesh networks, and its maximum throughput is higher than that of the H3D-mesh and mesh networks. As shown in Figure 4.18(e), for long message large-size networks, the average transfer time of the HTN is lower than that of the H3D-mesh and mesh networks, and its maximum throughput is higher than that of the H3D-mesh and mesh networks. From Figure 4.18, it is seen that the average transfer time of the HTN is remarkably lower than that of the mesh, TESH, and H3D-torus network; it even is lower than the H3D-mesh network but this difference is not impressive. The maximum throughput of the HTN is higher than that of those networks. Therefore, HTN achieves better dynamic communication performance than that of the other hierarchical networks and conventional mesh network under a uniform traffic pattern.

The hardware cost of an interconnection network is increased along with the increase of number of virtual channels. This is why, we have investigated the minimum number of virtual channels for deadlock free routing of HTN. 2 virtual channels per physical channel is the minimum number of virtual channels required to make the routing algorithm deadlock-free. We have simulated the dynamic communication performance of various networks with 2 virtual channels.

Figures 4.19 depicts the result of simulations under uniform traffic pattern for the various network models of small and large-size networks with 2 virtual channels. As shown in Figures 4.19(a), 4.19(b), and 4.19(c) for small-size networks, the average transfer time of the HTN is lower than that of the H3D-mesh, TESH, mesh, and torus networks, and the maximum throughput of the HTN is higher than that of the H3D-mesh, TESH, mesh, and torus networks, for short, medium-length, and long message respectively. For simplicity, we have considered only hierarchical networks for medium and long messages on small-size network. As shown in Figures 4.19(d), 4.19(e), and 4.19(f) for large-size networks, the average transfer time of the HTN is lower than that of the H3D-mesh, mesh, and torus

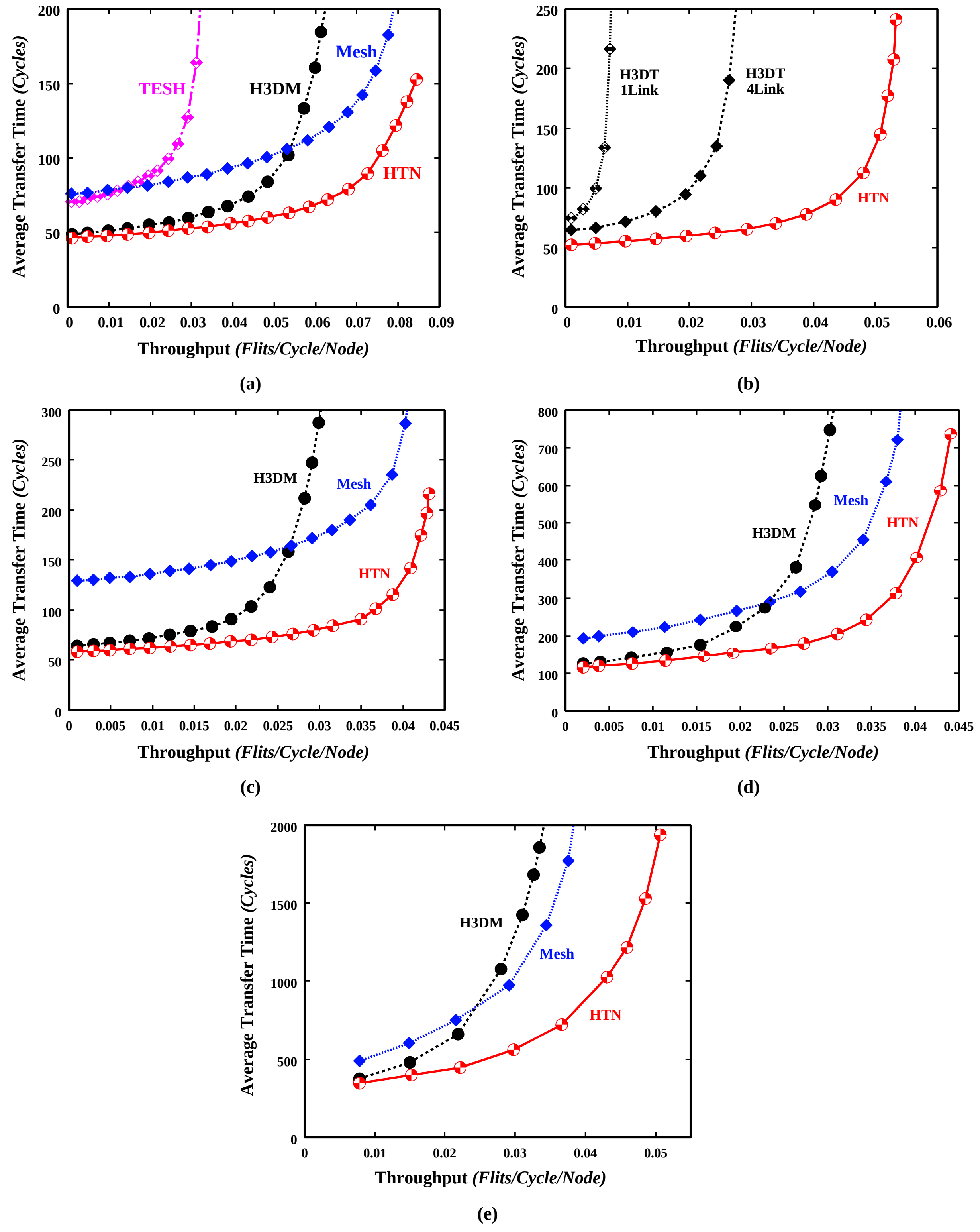


Figure 4.18: Dynamic communication performance of dimension-order routing with uniform traffic pattern on various networks: (a) 256 nodes, 3 virtual channels, short message, and $q = 1$ (b) 512 nodes, 3 virtual channels, short message, and $q = 1$, (c) 1024 nodes, 3 virtual channels, short message, and $q = 1$, (d) 1024 nodes, 3 virtual channels, medium-length message, and $q = 1$, (e) 1024 nodes, 3 virtual channels, long message, and $q = 1$,.

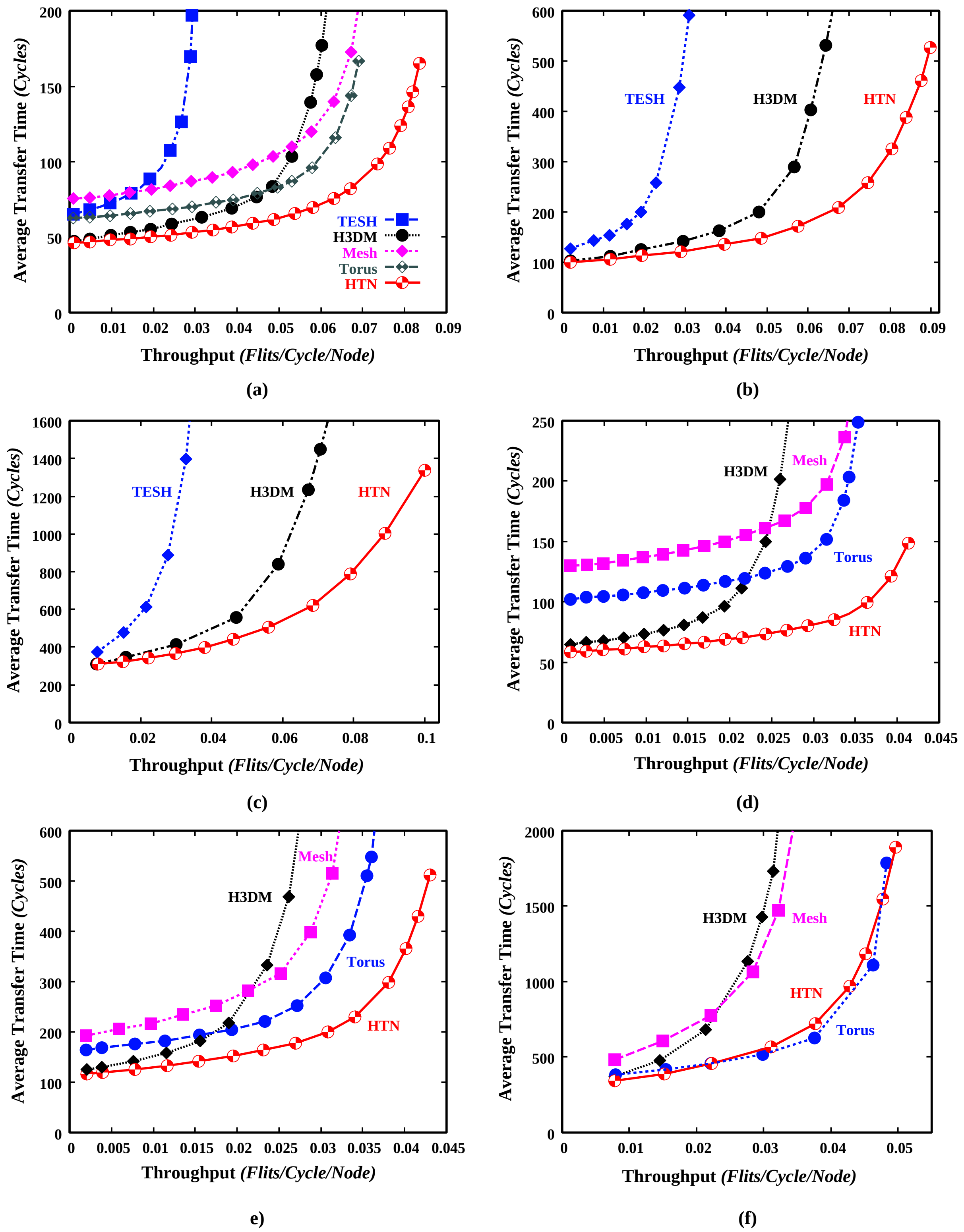


Figure 4.19: Dynamic communication performance of dimension-order routing with uniform traffic pattern on various networks: (a) 256 nodes, 2 virtual channels, short message, and $q = 1$ (b) 256 nodes, 2 virtual channels, medium-length message, and $q = 1$, (c) 256 nodes, 2 virtual channels, long message, and $q = 1$, (d) 1024 nodes, 2 virtual channels, short message, and $q = 1$, (e) 1024 nodes, 2 virtual channels, medium-length message, and $q = 1$, and (f) 1024 nodes, 2 virtual channels, long message, and $q = 1$.

networks, and its maximum throughput is higher than that of the H3D-mesh, mesh, and torus networks, for short, medium-length, and long message respectively.

From Figure 4.19, it is seen that the average transfer time of the HTN is remarkably lower than that of the TESH, mesh, and torus networks; it even is lower than the H3D-mesh network but this difference is not impressive. The maximum throughput of the HTN is higher than that of those networks. Therefore, HTN achieves better dynamic communication performance than that of the other hierarchical and conventional networks under the uniform traffic pattern. In the uniform traffic pattern, the traffic distribution is randomly chosen. If the source and the destination are in the same BM, then the routing is performed within that BM only. This is why the dynamic communication performance of the HTN is better than that of other networks.

Nonuniform Traffic Pattern

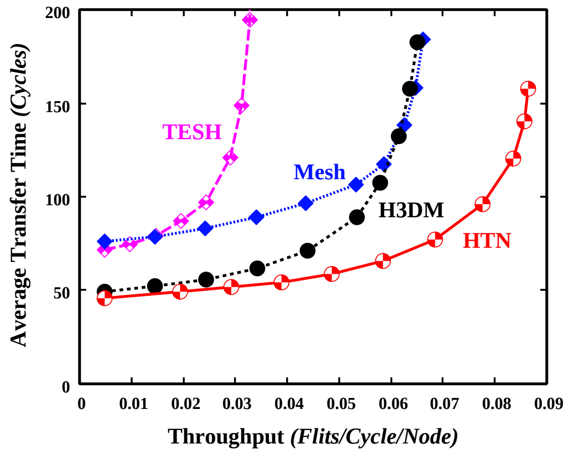
Hot-Spot Traffic

For dynamic communication performance evaluation under nonuniform traffic pattern, we have considered 3 virtual channels. The reason will be explained in the next subsection.

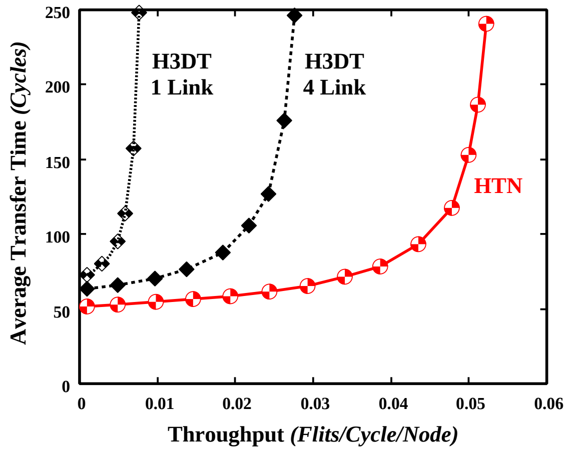
For generating hot-spot traffic we used a model proposed by Pfister and Norton [130]. According to this model, each node first generates a random number. If that number is less than a predefined threshold, it will send the message to the hot spot node. Otherwise the message will be sent to other nodes with a uniform distribution. Here, in uniform distribution, message destinations are chosen randomly with equal probability between the nodes in the networks. For performance evaluation, the hot spot percentages are assumed to be 2%, 5%, and 10%. The hot spot nodes are assumed to be centered 4 nodes. In hierarchical networks, the centered 4 nodes that connect the sub-network module are considered as hot spot nodes.

Figure 4.20 shows the message latency versus network throughput curve for hot spot traffic. This figure compares the performance of various networks under 5% hot spot traffic. Figure 4.20 (a) compares the performance of small-size networks. The average transfer time of the HTN is lower than that of the H3D-mesh, TESH, and mesh networks, and its maximum throughput is higher than that of the H3D-mesh, TESH, and mesh networks. Figure 4.20 (b) compares the performance of medium-size networks, HTN and H3D-torus networks. It shows that the average transfer time of the HTN is lower than that of the H3D-torus network with 1 link and 4 links, and its maximum throughput is far higher than that of the H3D-torus network with 1 link and 4 links. Figure 4.20 (c) compares the performance of large-size networks. Here again, the average transfer time of the HTN is lower than that of the H3D-mesh and mesh networks, and the maximum throughput is higher than that of the H3D-mesh and mesh networks. It is seen that under 5% hot spot traffic, the dynamic communication performance of the HTN is better than that of the other hierarchical and conventional mesh networks.

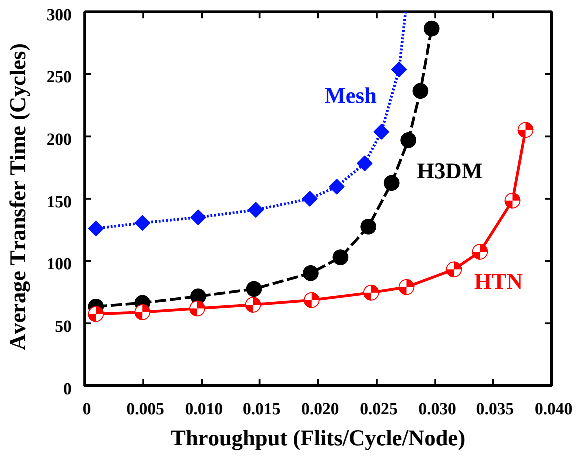
Figure 4.21 shows the message latency versus network throughput curves with hot 2% and 10% spot traffic for small and large-size networks. Figure 4.21(a), and 4.21(b) compares the performance of small-size networks under 2% and 10% hot spot traffic, respectively. The average transfer time of the HTN is lower than that of the H3D-mesh, TESH, and mesh networks, and its maximum throughput is higher than that of the H3D-mesh, TESH, and mesh networks. Figure 4.21(c), and 4.21(d) compares the



(a)



(b)



(c)

Figure 4.20: Dynamic communication performance of dimension-order routing with hot-spot traffic pattern on various networks: (a) 256 nodes, 3 virtual channels, 5% hot-spot traffic, short message, and $q = 1$ (b) 512 nodes, 3 virtual channels, 5% hot-spot traffic, short message, and $q = 1$, and (c) 1024 nodes, 3 virtual channels, 5% hot-spot traffic, short message, and $q = 1$

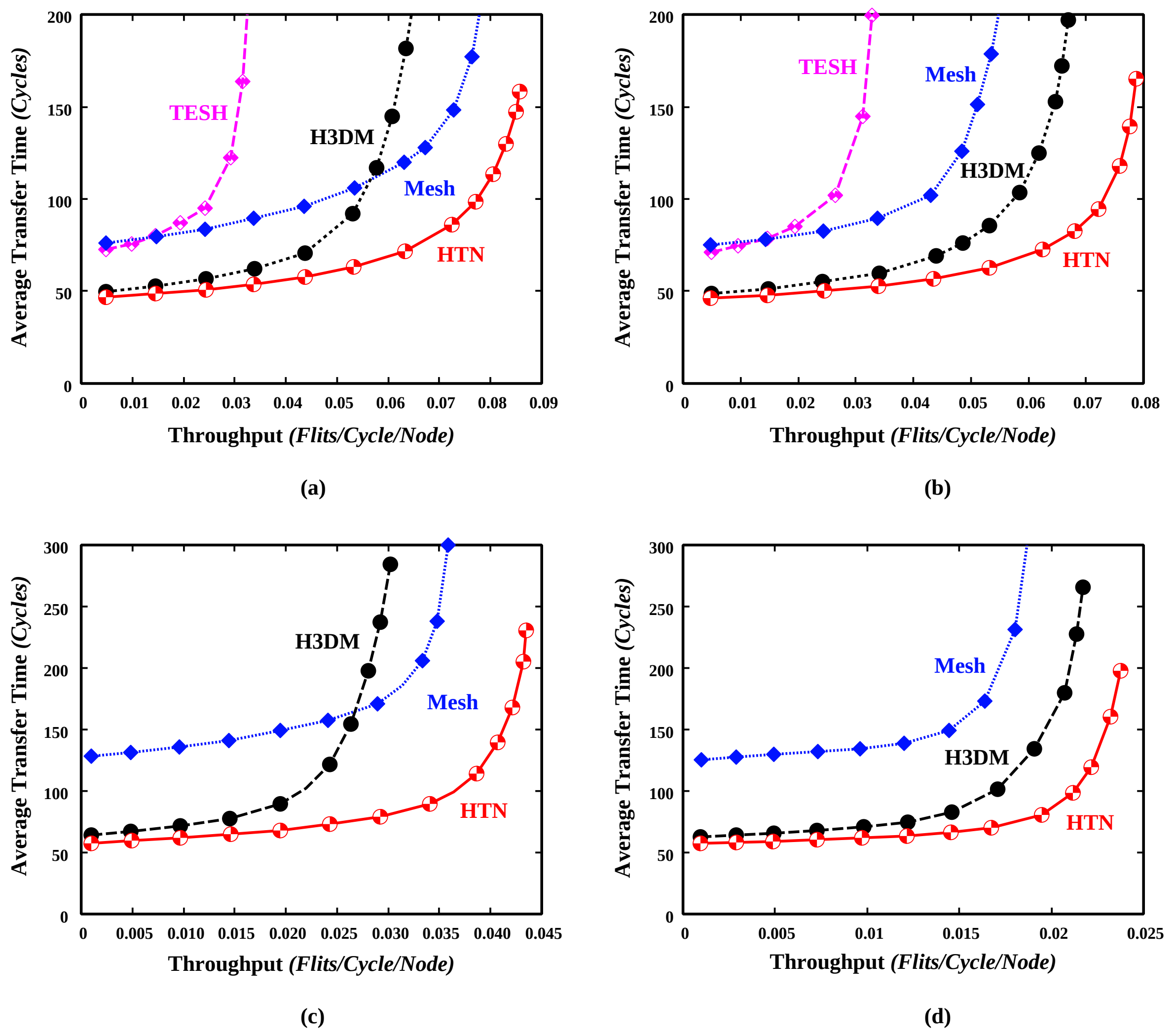


Figure 4.21: Dynamic communication performance of dimension-order routing with hot-spot traffic pattern on various networks: (a) 256 nodes, 3 virtual channels, 2% hot-spot traffic, short message, and $q = 1$ (b) 256 nodes, 3 virtual channels, 10% hot-spot traffic, short message, and $q = 1$, (c) 1024 nodes, 3 virtual channels, 2% hot-spot traffic, short message, and $q = 1$, and (d) 1024 nodes, 3 virtual channels, 10% hot-spot traffic, short message, and $q = 1$.

performance of large-size networks under 2% and 10% hot spot traffic, respectively. Here again, the average transfer time of the HTN is lower than that of the H3D-mesh and mesh networks, and the maximum throughput is higher than that of the H3D-mesh and mesh networks. It is seen that under 2% and 10% hot spot traffic, the dynamic communication performance of the HTN is better than that of the other hierarchical and conventional mesh networks. Therefore, with a hot spot traffic pattern, HTN also achieves better dynamic communication performance than that of the other hierarchical and conventional mesh networks.

The higher the hot spot traffic, the lower the performance. But the performance deviation in the HTN is lower than that of mesh network. One interesting point is that the difference in maximum throughput between HTN and mesh is greater for the hot spot traffic pattern than for other traffic patterns, including the uniform traffic pattern. Moreover, the maximum throughput of the H3D-mesh network is higher than that of the mesh network with a hot spot traffic pattern, but usually less than that of the mesh network with other traffic patterns.

Dimension Reversal Traffic

HTN is a hierarchical network, where the BM is a 3D-torus and the higher level network is a 2D-torus network. In our simulation, we use a 2-dimensional reversal traffic and a 3-dimensional reversal traffic pattern. In 2-dimensional reversal traffic, in the BM, node (x, y) sends messages to node (y, x) and the z-axis remains fixed. At higher levels, the x-coordinates and y-coordinates of the subnet modules at a certain level are transposed. For instance, in the Level-2 HTN, BM (x, y) sends messages to BM (y, x). In 3-dimensional reversal traffic, in the BM, node (x, y, z) sends messages to node $\Big(y, x, \sqrt[3]{N_{BM}} - (z + 1)\Big)$. Here, N_{BM} is the total number of nodes in the BM. As the higher level network is a 2D-torus, the traffic pattern is like a 2-dimensional reversal traffic pattern for higher level networks.

The dynamic communication performance of various networks under the 2-dimensional reversal traffic pattern is shown in Figures 4.22(a) and (b), for small and large-size networks. The figure shows the average transfer time as a function of network throughput for different networks. Each curve stands for a particular network. In small-size networks, as shown in Figure 4.22(a), the average transfer time of the HTN is lower than that of the H3D-mesh, TESH, and mesh networks, and the maximum throughput of the HTN is higher than that of the H3D-mesh, TESH, and mesh networks. In large-size networks, Figure 4.22(b), the average transfer time of the HTN is lower than that of the H3D-mesh and mesh networks, and the maximum throughput of the HTN is higher than that of the H3D-mesh and mesh networks. Therefore, HTN achieves better dynamic communication performance than that of the other hierarchical networks and the conventional mesh network.

A 3-dimensional reversal of traffic is impossible in a 2D-mesh network or in TESH, which is also a 2D-hierarchical network. Therefore, for a 3-dimensional reversal traffic we have evaluated the dynamic communication performance of only small and large-size HTN and H3D-mesh networks; this is portrayed in in Figure 4.22(c) and (d), for small and large-size networks. Each curve stands for a particular network. As Figure 4.22 shows, the dynamic communication performance of the HTN is better than that of the

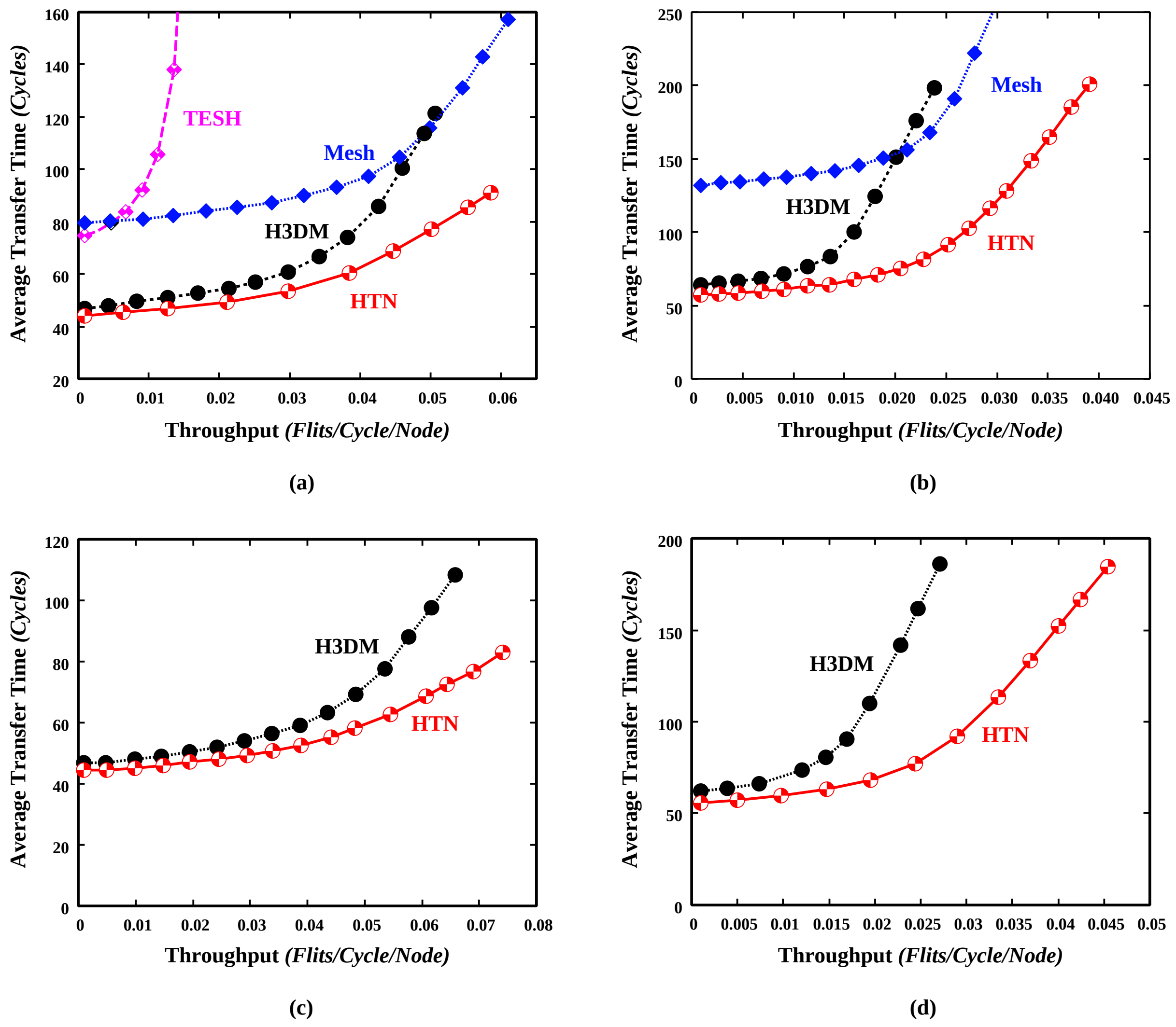


Figure 4.22: Dynamic communication performance of dimension-order routing with dimension reversal traffic pattern on various networks: (a) 256 nodes, 3 virtual channels, 2-dimensional reversal traffic, short message, and $q = 1$ (b) 1024 node, 3 virtual channels, 2-dimensional reversal traffic, short message, ands $q = 1$, (c) 256 nodes, 3 virtual channels, 3-dimensional reversal traffic, short message, and $q = 1$, (d) 1024 nodes, 3 virtual channels, 3-dimensional reversal traffic, short message, and $q = 1$.

H3D-mesh network.

Dimension reversal traffic patterns (both 2D and 3D) are applicable to the H3D-torus network. However, neither is applicable to the HTN, because we have considered a rectangular HTN for the medium-size network to match the total number of nodes. This is why, dynamic communication performance for dimension reversal traffic patterns is presented only for small and large-size networks.

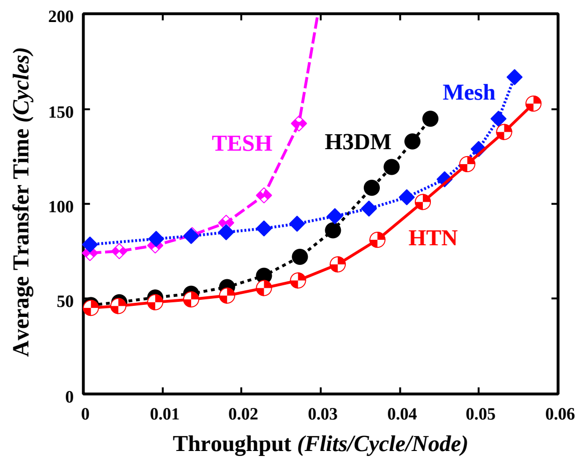
Bit Reversal Traffic

In a bit reversal traffic pattern, a node with address Node $(a_{\beta-1}, a_{\beta-2} \dots \dots a_1, a_0)$ sends messages to Node $(a_0, a_1, \dots \dots a_{\beta-2}, a_{\beta-1})$. Figure 4.23 depicts the result of simulations under bit reversal traffic pattern for the various network models. As shown in Figure 4.23(a), for small-size networks, the average transfer time of the HTN is lower than that of the H3D-mesh, TESH, and mesh networks, and the maximum throughput of the HTN is higher than that of the H3D-mesh, TESH, and mesh networks. As shown in Figure 4.23(b), for medium-size networks, the average transfer time of the HTN is lower than that of the H3D-torus network with 1 link and 4 links, and its maximum throughput is far higher than that of the H3D-torus network with 1 link and 4 links. As shown in Figures 4.23(c), (d), and (e), for large-size networks with short, medium, and long message, the average transfer time of the HTN is lower than that of the H3D-mesh and mesh networks, and its maximum throughput is higher than that of the H3D-mesh and mesh networks. In Figure 4.23(e), for long message, it is seen that the maximum throughput of the H3D-mesh network is higher than that of the mesh network with a bit-reversal traffic pattern.

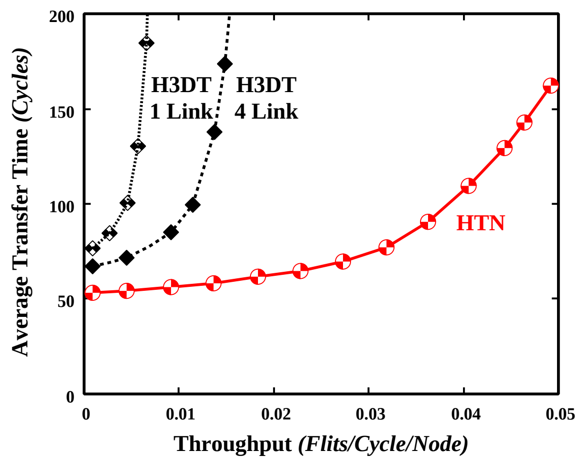
From Figure 4.23, it is seen that the average transfer time of the HTN is remarkably lower than that of the mesh, TESH, and H3D-torus network; it even is lower than the H3D-mesh network but this difference is not impressive. The maximum throughput of the HTN is higher than that of those networks. Therefore, HTN achieves better dynamic communication performance than that of the other hierarchical networks and conventional mesh network under a bit reversal traffic pattern.

Complement Traffic

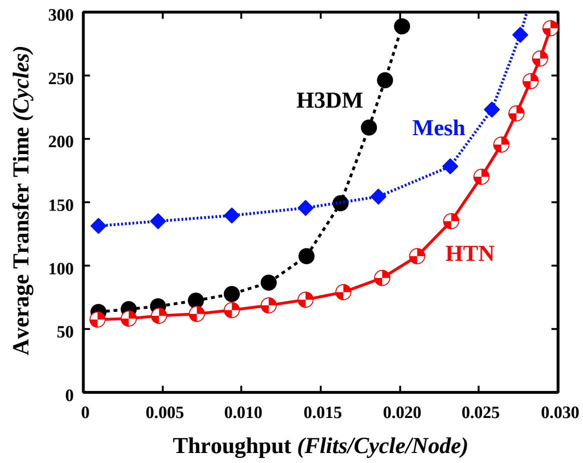
In a complement traffic pattern, a node with address Node $(a_{\beta-1}, a_{\beta-2} \dots \dots a_1, a_0)$ sends messages to Node $(\overline{a_{\beta-1}}, \overline{a_{\beta-2}}, \dots \dots \overline{a_2}, \overline{a_1}, \overline{a_0})$. The complement is a particularly difficult traffic pattern, since it requires all packets to cross the network bisection. Figure 4.24 shows the result of simulations under complement traffic pattern for the various network models. As shown in Figure 4.24(a), for small-size networks, the average transfer time of the HTN is lower than that of the H3D-mesh, TESH, and mesh networks, and the maximum throughput of the HTN is higher than that of the H3D-mesh, TESH, and mesh networks. As shown in Figure 4.24(b), for medium-size networks, the average transfer time of the HTN is lower than that of the H3D-torus network with 1 link and 4 links, and its maximum throughput is far higher than that of the H3D-torus network with 1 link and 4 links. As shown in Figure 4.24(c), (d), and (e), for large-size networks with short, medium, and long message, the average transfer time of the HTN is lower than that of the H3D-mesh and mesh networks, and its maximum throughput is higher than that of the H3D-mesh and mesh networks.



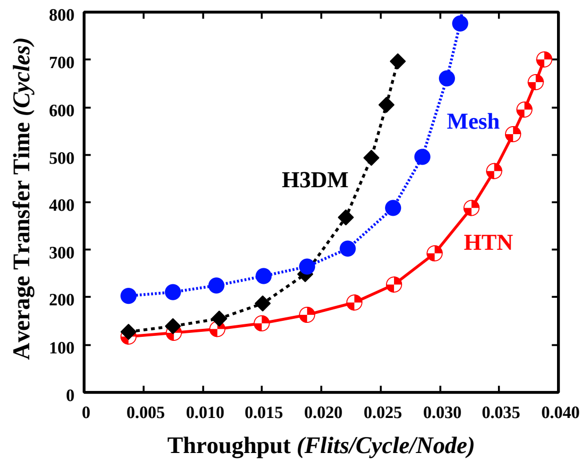
(a)



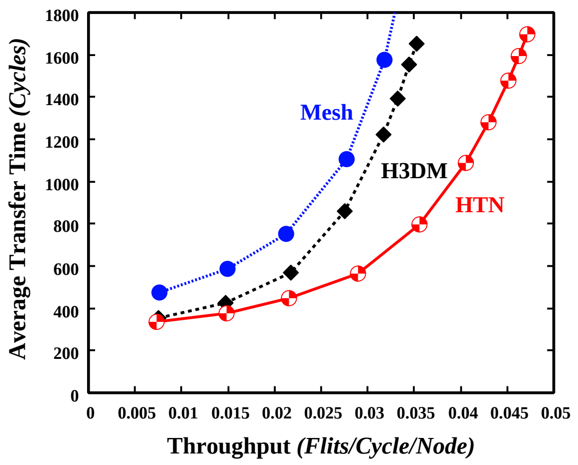
(b)



(c)



(d)



(e)

Figure 4.23: Dynamic communication performance of dimension-order routing with bit-reversal traffic pattern on various networks: (a) 256 nodes, 3 virtual channels, short message, and $q = 1$ (b) 512 nodes, 3 virtual channels, short message, and $q = 1$, (c) 1024 nodes, 3 virtual channels, short message, and $q = 1$, (d) 1024 nodes, 3 virtual channels, medium-length message, and $q = 1$, and (e) 1024 nodes, 3 virtual channels, long message, and $q = 1$.

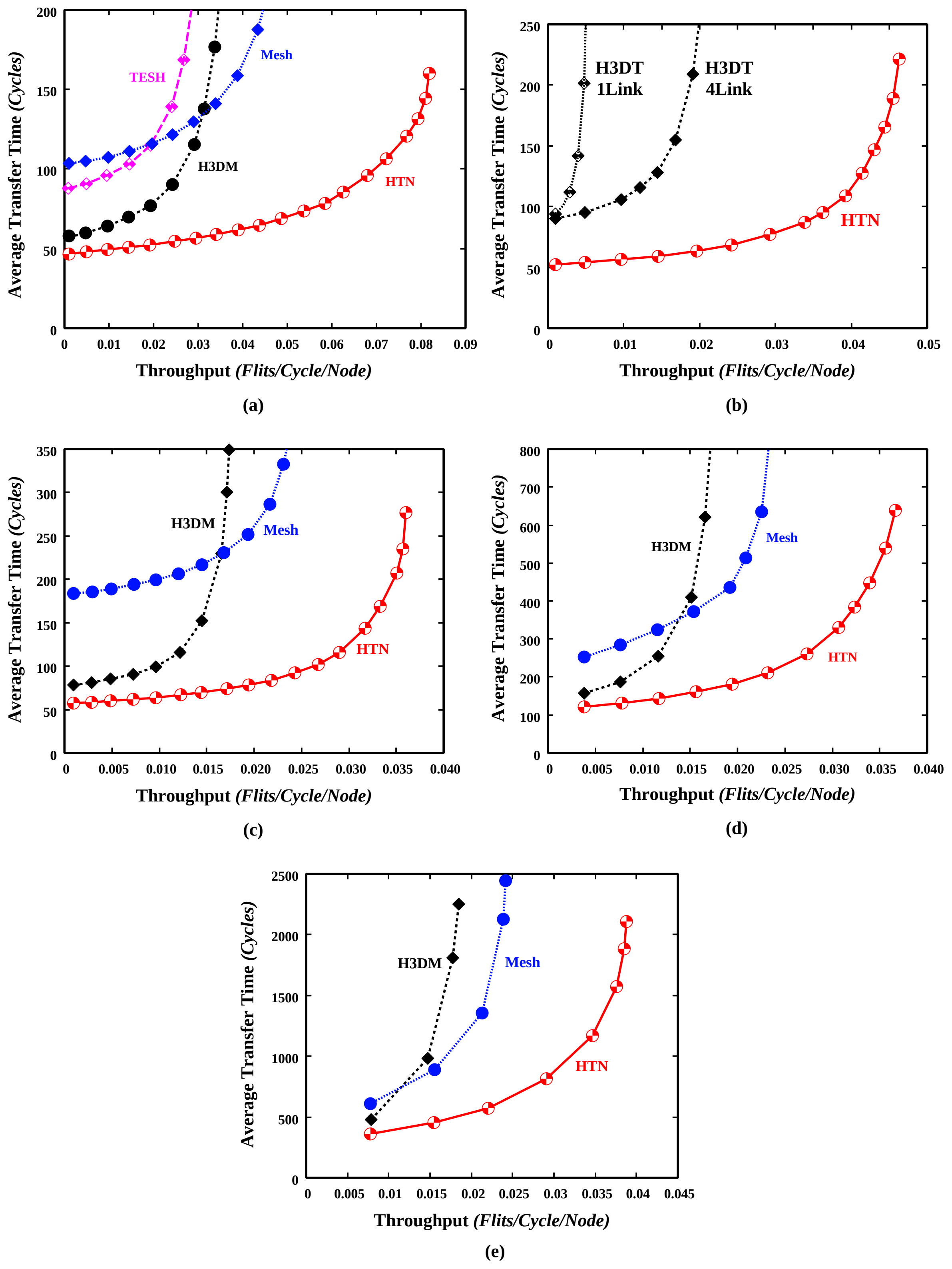


Figure 4.24: Dynamic communication performance of dimension-order routing with complement traffic pattern on various networks: (a) 256 nodes, 3 virtual channels, short message, and $q = 1$, (b) 512 nodes, 3 virtual channels, short message, and $q = 1$, (c) 1024 nodes, 3 virtual channels, short message, and $q = 1$ (d) 1024 nodes, 3 virtual channels, medium-length message, and $q = 1$, and (e) 1024 nodes, 3 virtual channels, long message, and $q = 1$

From Figure 4.24, it is seen that the average transfer time of the HTN is remarkably lower than that of the mesh, H3D-mesh, and H3D-torus networks. The maximum throughput of the HTN is higher than that of those networks. Usually the average transfer time of the HTN less than that of the H3D-mesh networks but this difference is not impressive for other traffic patterns. However, in complement traffic, it is remarkably lower than that of the H3D-mesh network. Therefore, HTN also achieves better dynamic communication performance than that of the other hierarchical networks and conventional mesh network under a complement traffic pattern.

Bit Flip Traffic

In a bit flip traffic pattern, a node with address Node $(a_{\beta-1}, a_{\beta-2} \ldots \ldots a_1, a_0)$ sends messages to node $(\overline{a_0}, \overline{a_1}, \ \ldots \ldots \ \overline{a_{\beta-2}}, \overline{a_{\beta-1}})$. Figure 4.25 depicts the result of simulations under bit flip traffic pattern for the various network models. As shown in Figure 4.25(a), for small-size networks, the average transfer time of the HTN is lower than that of the mesh network with 1 and 2 virtual channels, H3D-mesh, TESH, and torus networks, and the maximum throughput of the HTN is higher than that of the mesh, H3D-mesh, TESH, and torus networks. As shown in Figures 4.23(b), (c), and (d) , for large-size networks with short, medium, and long message, the average transfer time of the HTN is lower than that of the mesh network with 1 and 2 virtual channels,H3D-mesh, and torus networks, and its maximum throughput is higher than that of the mesh, H3D-mesh, and torus networks. In bit-flip traffic pattern, the maximum throughput of the HTN is remarkably higher than that of mesh, H3D-mesh, TESH, and torus networks.

From Figure 4.25, it is seen that the average transfer time of the HTN is remarkably lower than that of the mesh, torus, and TESH network; it even is lower than the H3D-mesh network but this difference is not impressive. The maximum throughput of the HTN is higher than that of those networks. Therefore, HTN yields better dynamic communication performance than that of the other hierarchical and conventional networks under a bit flip traffic pattern. Under bit flip traffic pattern, the dynamic communication performance of the HTN is even better than that of the torus network.

The comparison of dynamic communication performance, in Figures 4.18, 4.19, 4.20, 4.21, 4.22, 4.23, 4.24, and 4.25, reveals that the HTN outperforms the H3D-mesh, H3D-torus, TESH, mesh, and torus networks because it yields low latency and high throughput, which are indispensable for high performance massively parallel computers.

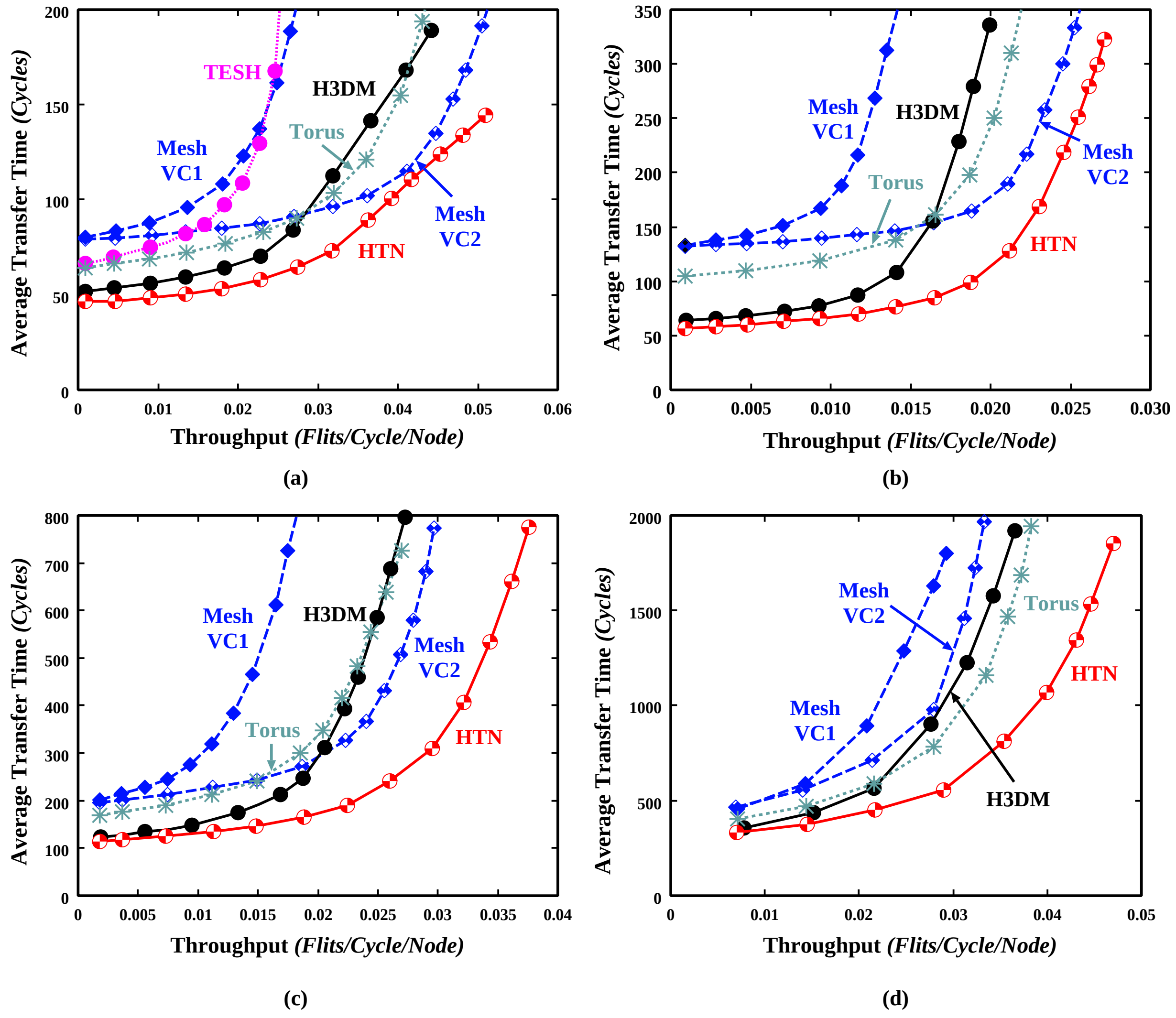


Figure 4.25: Dynamic communication performance of dimension-order routing with bit-flip traffic pattern on various networks: (a) 256 nodes, 2 virtual channels, short message, and $q = 1$ and (b) 1024 nodes, 2 virtual channels, short message, and $q = 1$.

Maximum Throughput of the HTN

Figure 4.26 shows the message latency versus network throughput curve for the large-sized HTN with several traffic patterns. At 137 cycles, the throughput of the HTN under the various traffic patterns is shown by the arrowhead. It is shown that the throughput delivered by the uniform and bit reversal traffic patterns is the maximum and minimum, respectively. The throughput with hot-spot and 3D-reversal traffic patterns is closest to that of the uniform pattern. Moreover, the maximum throughput with the 3D-reversal traffic pattern is higher than that of the uniform pattern.

The maximum throughput of the HTN under different traffic patterns is presented in Table 4.2. It is shown that, except with the bit reversal traffic pattern, the maximum throughput under non-uniform traffic patterns is close to that of uniform traffic pattern.

Although the maximum throughput of the HTN under the bit reversal traffic pattern is lower than that of other traffic patterns, it is still higher than that of the other network models.

The wrap-around connections in the torus network of the BM and in the higher level network make the HTN symmetric. This symmetry diminishes the uneven traffic distribution in the network, which in turn improves the dynamic communication performance.

The maximum throughput of the HTN is higher than that of other hierarchical interconnection networks and conventional mesh network; and under different non-uniform traffic patterns it is closer to that of a uniform traffic pattern. Therefore, HTN is a good interconnection network for the next generation of massively parallel computers.

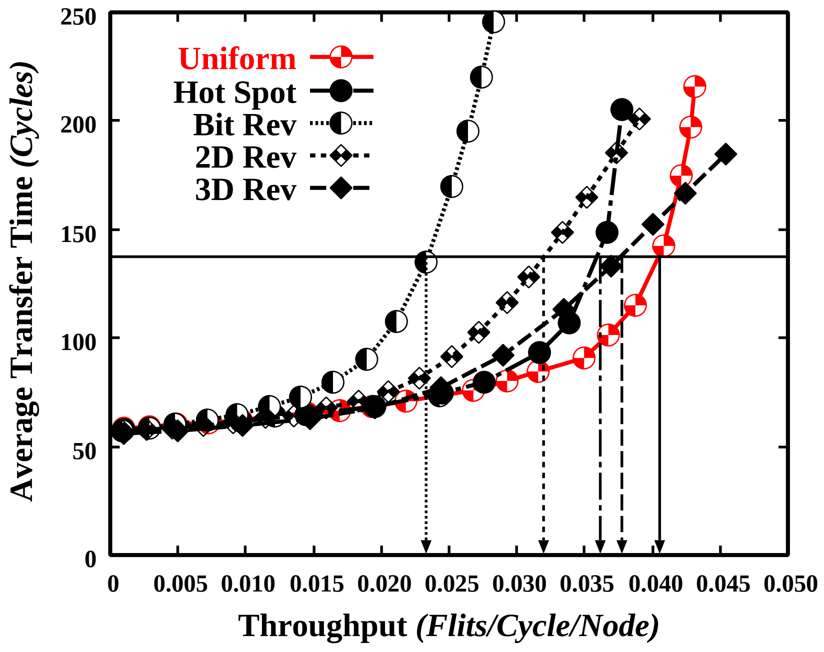


Figure 4.26: Dynamic communication performance of large-size HTN by dimension-order routing under various traffic patterns: 3 virtual channels, short message.

Table 4.2: Maximum throughput of the HTN *(Flits/Cycle/Node)*

Network Size	Bit Reversal	2D Reversal	3D Reversal	Hot Spot	Uniform
Small-size	0.056944	0.058534	0.074053	0.078721	0.084491
Medium-size	0.049228	x	x	0.052277	0.053364
Large-size	0.029518	0.039052	0.045408	0.037789	0.043173

4.4.5 Effect of Message Length

Figure 4.27 shows the effect of message length on HTN dynamic communication performance. It shows the average message latency divided by message length as a function of network throughput for the uniform traffic pattern. The average message latency is smaller for longer messages; this is because wormhole switching is used for message switching and thus the messages are pipelined in nature. Path setup time is amortized among more flits when messages are long. Moreover, data flits can advance faster than header flits because headers have to make routing decisions. Hence, headers have to wait for the routing control unit to compute the output channel, and possibly wait for the output channel to become free. Therefore, when the header reaches the destination node, data flits advance faster, thus favoring longer messages. Figure 4.27 illustrates that average transfer time decreases and maximum throughput increases in a Hierarchical Torus Network (HTN) as message length increases.

HTN is a combination of 2D-torus and 3D-torus networks. The diameter and average distance of the HTN are lower than those of the other conventional and hierarchical interconnection networks. This is why message latency of the HTN is lower than that of those networks. Due to wrap-around connections in the torus network, the message will find an alternative path to pass through. Thus, more messages can be delivered in the network.

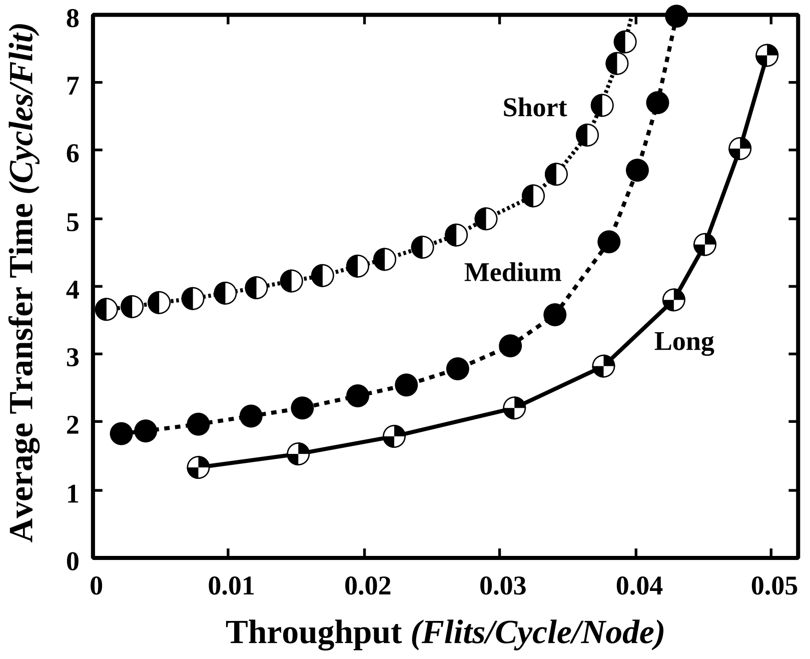


Figure 4.27: Average message latency divided by message length vs. network throughput of HTN: 1024 nodes, 2 VCs, and $q = 1$.

4.4.6 Effect of the Number of Virtual Channels

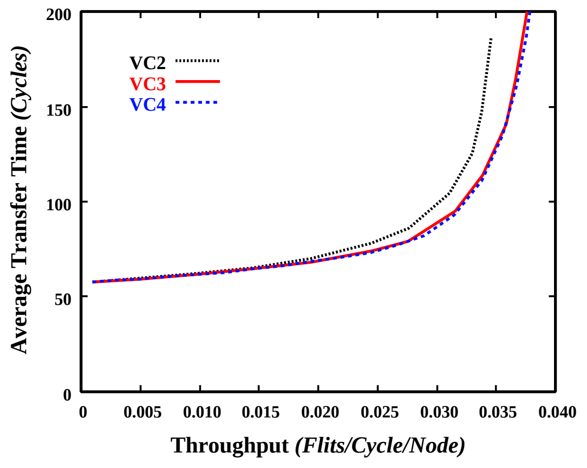
Splitting each physical channel into several virtual channels increases the number of routing choices, allowing messages to pass blocked messages. On the other hand, flits from several messages are multiplexed onto the same physical channel, slowing down both messages. Therefore, it is better to use as much virtual channel as it yields optimum performance. However, that optimal number depends on several parameters, including network traffic.

A useful parallel computer must be both efficient and reliable. A key component of reliability is freedom from deadlock. To reduce the router cost, eventually the total hardware cost, deadlock-free routing algorithm for an arbitrary interconnection network with a minimum number of virtual channels is preferred. However, there is a trade-off between dynamic communication performance and the number of virtual channels. One design alternative that can be considered is to implement dimension-ordered routing with extra virtual channels instead of adaptive routing, since virtual channels provide substantial performance improvement [5, 39] and are relatively inexpensive compared to the logic involved in implementing adaptive routing. In [39], Dally showed that the performance of dimension-order routing algorithm under uniform traffic load improves significantly as virtual channels are initially added. The benefits of added virtual channels then diminish as more channels are added.

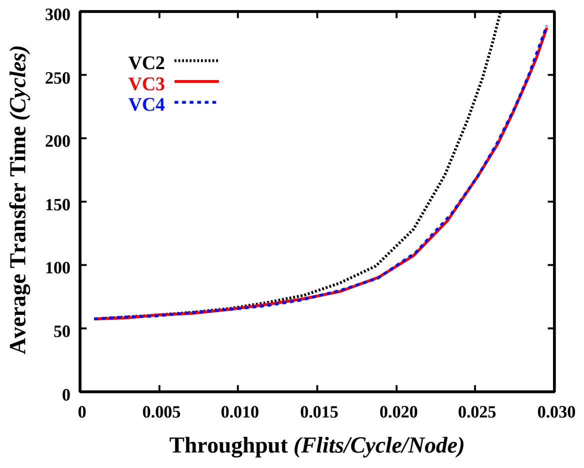
In this section, we have investigated the effect of adding extra virtual channels on the HTN for dimension-order routing. Figure 4.28 depicts the average transfer time as a function of network throughput under various nonuniform traffic patterns for different virtual channels. In Section 4.3.3, we have shown that the minimum number of virtual channels for deadlock-free routing of HTN is 2. Adding 1 extra virtual channel, that is, using 3 virtual channels, substantially improves the dynamic communication performance of the HTN. We have also evaluated the dynamic communication performance of the HTN using 4 virtual channels. Figure 4.28 shows that the maximum throughput of the HTN using 3 virtual channels is far higher than that using 2 virtual channels and almost same as that of 4 virtual channels for all traffic patterns. This striking difference of throughput shows that we can significantly improve the dynamic communication performance of the HTN by adding 1 extra virtual channel to the minimum number of virtual channels.

4.5 Adaptive Routing

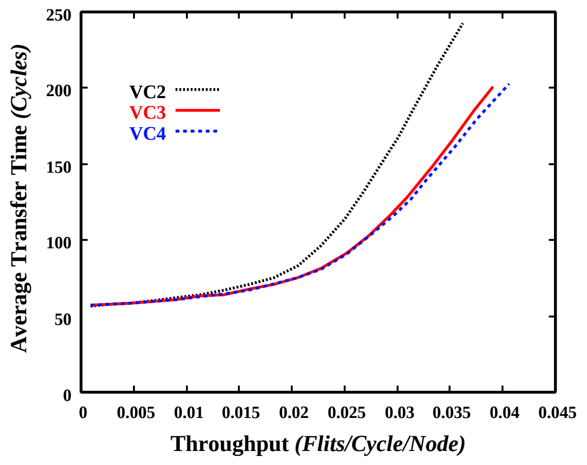
A routing algorithm specifies how a message selects its path to cross from source to destination in the network. An efficient routing is critical to the performance of an interconnection network. In a practical router design, the routing decision process must be as fast as possible to reduce network latency. Deterministic, dimension-order routing has been popular in multicomputers because it has minimal hardware requirements and allows the design of simple and fast routers [38]. Although there are numerous paths between any source and destination, dimension-order routing defines a single path from source to destination. Thus, dimension-order routing does not make effective use of the network's physical links. A lack of routing flexibility in deterministic routing algorithm limits the network performance. If the selected channel is congested, the traffic between that source and destination is delayed, despite the presence of uncongested alternative



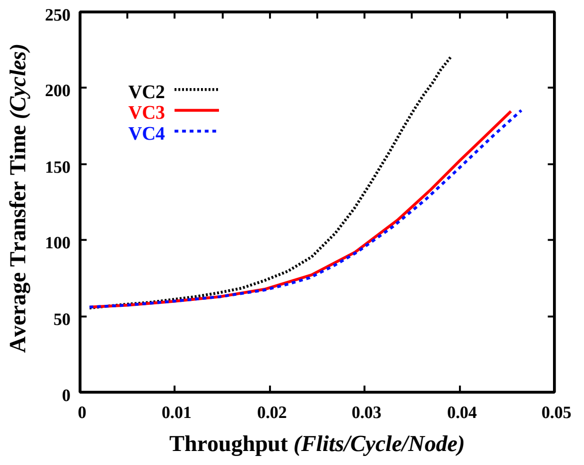
(a)



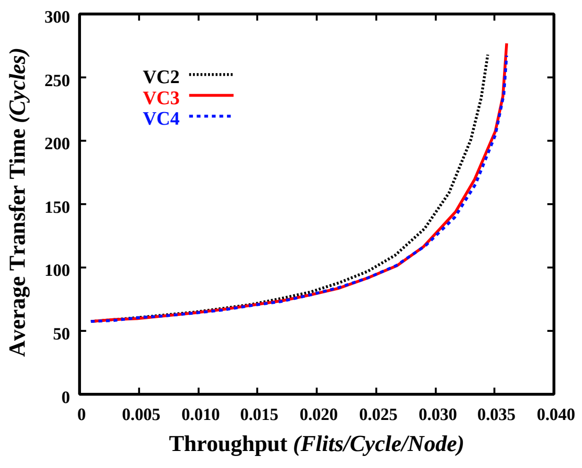
(b)



(c)



(d)



(a)

Figure 4.28: Dynamic communication performance of dimension-order routing with different virtual channels and short message on the large-size HTN: (a) hot spot traffic, (b) bit reversal traffic, (c) 2-dimension reversal, (d) 3-dimension reversal, and (e) complement traffic patterns.

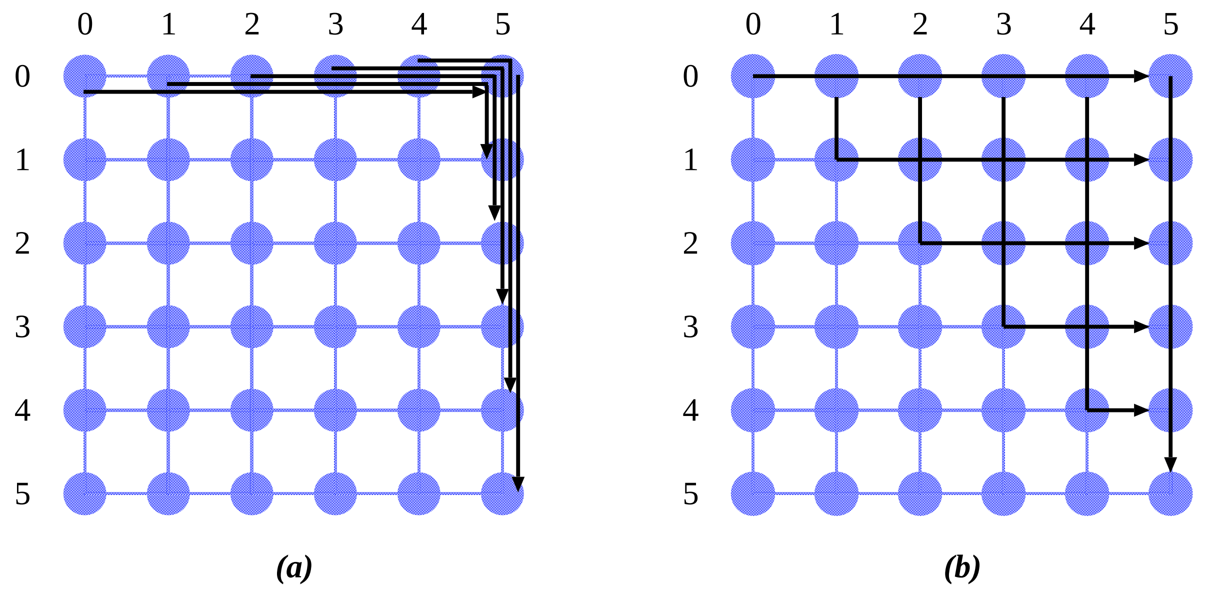


Figure 4.29: Routing messages in an 6x6 mesh from node $(0, i)$ to node $(i, 5)$ (for $0 \leq i \leq 5$); (a) Using dimension order routing, five messages must traverse the channel from $(0, 4)$ to $(0, 5)$, (b) Using adaptive routing, all messages proceed simultaneously.

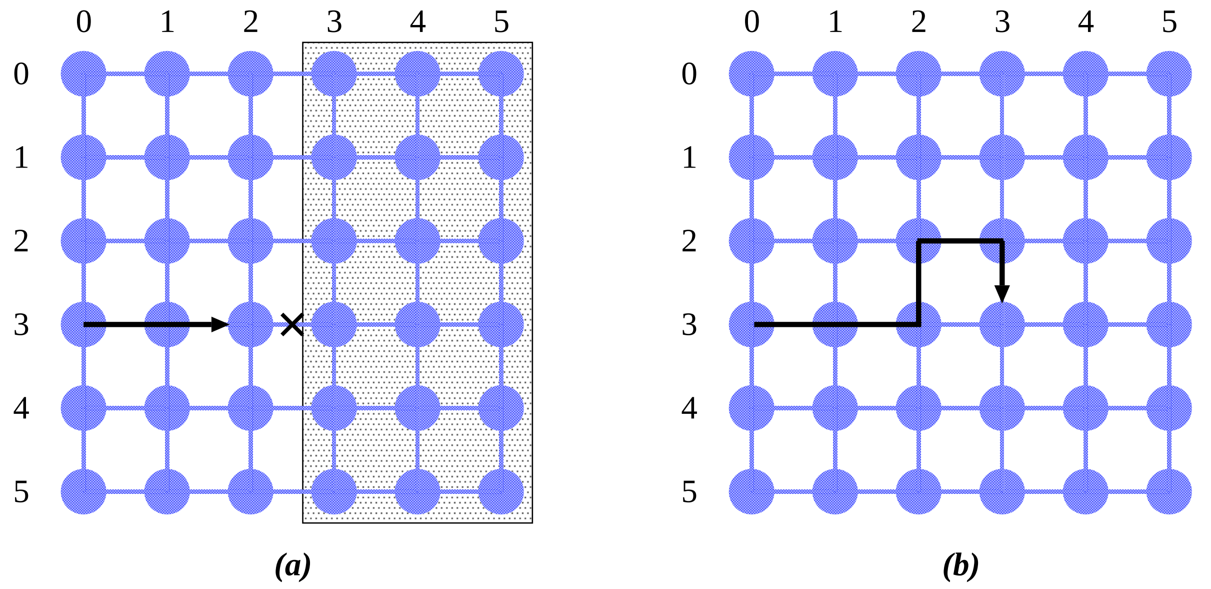


Figure 4.30: A 6×6 mesh with a faulty link from node $(3, 2)$ to node $(3, 3)$. (a) With dimension order routing messages from dark nodes to the shaded area cannot be delivered. (b) With adaptive routing, messages can be delivered between all pairs of nodes.

paths.

Adaptive routing allows paths to be chosen dynamically [67, 69, 70]: routes are used based on whether links are busy when a packet header arrives. If the desired link is busy, another link leading towards the destination may be chosen. Thus, adaptive routing offers the potential for making better use of network resources. A number of pairs of nodes may transmit packets simultaneously without blocking. Adaptive routing improves both the performance and fault tolerance of an interconnection network and, more importantly, it has the ability to provide performance which is less sensitive to the communication pattern [136].

Figure 4.29 shows a 6×6 mesh in which the node $(0, i)$ sends a message to the node $(i, 5)$ for $0 \leq i \leq 5$. With dimension order deterministic routing, as shown in Figure 4.29(a), five of the six messages must traverse the channel from $(0, 4)$ to $(0, 5)$. Thus only one of these five messages can proceed at a time. With adaptive routing, as shown in Figure 4.29(b), all of the messages can proceed simultaneously using alternate paths. Figure 4.30 shows the same network with a faulty channel from $(3, 2)$ to $(3, 3)$. With dimension-ordered routing messages from node $(3, i)$ to node (k, j) where $0 \leq i \leq 2 < j \leq 5$, $0 \leq k \leq 5$, cannot be delivered.

Although adaptive routing increases routing freedom, potentially improving performance, it also increases the cost of preventing deadlock. This cost can reduce the network clock speed, overwhelming the benefit of adaptive routing. The logic involved in implementing adaptive routing is complex. This is why most existing multicomputers, such as J-machine, Touchstone, Ametek 2010, and Cosmic cube, use deterministic routing instead of adaptive routing. If the logic for implementing adaptive routing could be as simple as for dimension-order routing, and the hardware cost exactly equal to that of dimension-order routing, then adaptive routing would be a good choice for the routing of messages in multicomputers, rather than the dimension-order routing. In this section, we present a suite of low cost adaptive routing algorithm for effective use of the physical links and virtual channels of the HTN while keeping the routing algorithm as simple as for dimension-order routing.

4.5.1 Link-Selection (LS) Algorithm

In each dimension of the wrap-around connected interconnection networks, such as k-ary n-cube network, some links are used for inter-node connection and another link is used for wrap-around connection of end to end nodes; the message will find an extra path to pass through. Dimension-order routing does not make effective use of these links. However, effective use of these links significantly improves dynamic communication performance. A Link-selection (LS) algorithm provides an efficient use of these physical links. HTN is a combination of 2D-torus and 3D-torus networks. Thus, the LS algorithm can be also used in the BM and higher-level networks of the HTN.

Figure 4.31 shows an example of routing a message in a ring network using the LS algorithm. As shown in Figure 4.31, the number of hops from source to destination in the clockwise direction and in the counter-clockwise direction is 2. Then, the message can follow either path-a (clockwise) or path-b (counter-clockwise), as shown in Figure 4.31. Therefore, if the distance from source to destination is equal in both the clockwise and counter-clockwise directions, then the packet can follow either of these two directions.

This is the principal idea of the LS algorithm. A 2D-torus consists of $x-y$ rings. Similarly, a 3D-torus consists of $x-y-z$ rings. Thus, LS algorithm is applicable in each direction of the HTN.

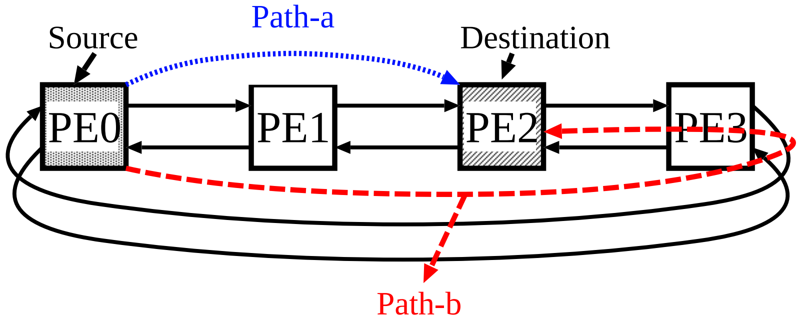


Figure 4.31: Selection of physical link by link-selection algorithm

If the following equation is satisfied, a packet can select from either a clockwise or counter-clockwise direction.

$$|s-d| = \begin{cases} \frac{m}{2} & \text{if } L=1\text{, i.e., BM} \\ \frac{n}{2} & \text{if } L \geq 2 \end{cases} \tag{4.8}$$

Here s and d denotes the source and destination node addresses, respectively. m and n are the size of the BM and higher level networks.

The deterministic routing algorithm also uses the wraparound links. If the source-to-destination distance using wraparound channels is less than that of not using wraparound links, then message will follow the wraparound links in dimension order routing. In our previous approach, described in Section 4.3, if the distance from source to destination is equal in both the clockwise and counter-clockwise directions, then the message will follow only the clockwise direction. In this current approach, however, the message can move in direction; first, it will attempt the clockwise direction and then, if the clockwise channel is busy, it will follow the counter-clockwise direction.

Routing in the HTN is strictly defined by the source node address and the destination node address as dimension order routing. Let a source node address be $s_\alpha, s_{\alpha-1}, s_{\alpha-2}, ..., s_1$, s_0, a destination node address be $d_\alpha, d_{\alpha-1}, d_{\alpha-2}, ..., d_1, d_0$, and a routing tag be $t_\alpha, t_{\alpha-1}$, $t_{\alpha-2}, ..., t_1, t_0$, where $t_i = d_i - s_i$. The source node address of HTN is expressed as $s = (s_{2L}, s_{2L-1}), (s_{2L-2}, s_{2L-3}), ..., (s_2, s_1, s_0)$. Similarly, the destination node address is expressed as $d = (d_{2L}, d_{2L-1}), (d_{2L-2}, d_{2L-3}), ..., (d_2, d_1, d_0)$. The proposed LS algorithm enforces some routing restrictions as dimension-order routing to make the routing algorithm simple and avoid deadlocks. Routing order is strictly followed to route packets. The routing algorithm that applies this link-selection principle is denoted as (R_2). Figure 4.32 shows the routing algorithm (R_2) for the HTN .

The proposed LS algorithm can only improve the performance of the network when the number of nodes in each direction is even. According to the architecture of the HTN, the preferable number of nodes in each direction is even. Again in the HTN, a BM with $m = 4$ and the higher levels with $n = 4$ is perhaps the most interesting network size because it has better granularity than the larger sizes. Therefore, the proposed LS algorithm is suitable for HTN.

```
Routing HTN(s,d);
source node address:$s_\alpha, s_{\alpha-1}, s_{\alpha-2}, ..., s_1, s_0$
destination node address: $d_\alpha, d_{\alpha-1}, d_{\alpha-2}, ..., d_1, d_0$
tag: $t_\alpha, t_{\alpha-1}, t_{\alpha-2}, ..., t_1, t_0$
  for $i = \alpha : 3$
    if ($i/2 = 0$ and ($t_i > 0$ or $t_i = -(n-1)$)), routedir = North; endif;
    if ($i/2 = 0$ and $t_i = \frac{n}{2}$ and routedir $\neq$ North); routedir = South; endif;
    if ($i/2 = 0$ and ($t_i < 0$ or $t_i = \quad (n-1)$)), routedir = South; endif;
    if ($i/2 = 0$ and $t_i = -\frac{n}{2}$ and routedir $\neq$ South); routedir = North; endif;
    if ($i\%2 = 1$ and ($t_i > 0$ or $t_i = -(n-1)$)), routedir = East; endif;
    if ($i\%2 = 1$ and $t_i = \frac{n}{2}$ and routedir $\neq$ East); routedir = West; endif;
    if ($i\%2 = 1$ and ($t_i < 0$ or $t_i = \quad (n-1)$)), routedir = West; endif;
    if ($i\%2 = 1$ and $t_i = -\frac{n}{2}$ and routedir $\neq$ West); routedir = Eest; endif;
      while ($t_i \neq 0$) do
        $N_z = outlet_z(s, d, L, \text{routedir})$
        $N_y = outlet_y(s, d, L, \text{routedir})$
        $N_x = outlet_x(s, d, L, \text{routedir})$
        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;
        if ($t_i > 0$), $t_i = t_i - 1$; endif;
        if ($t_i < 0$), $t_i = t_i + 1$; endif;
      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 $t_i \leq \frac{m}{2}$) or ($t_i < 0$ and $t_i = -(m-1)$), movedir = positive; endif;
    if ($t_i = \frac{m}{2}$ and movedir $\neq$ positive), movedir = negative; endif;
    if ($t_i > 0$ and $t_i = (m-1)$) or ($t_i < 0$ and $t_i \geq -\frac{m}{2}$), movedir = negetive; endif;
    if ($t_i = -\frac{m}{2}$ and movedir $\neq$ negative), movedir = positive; endif;
    if (movedir = positive and $t_i > 0$), distance = $t_i$; endif;
    if (movedir = positive and $t_i < 0$), distance = $m + t_i$; endif;
    if (movedir = negative and $t_i < 0$), distance = $t_i$; endif;
    if (movedir = negative and $t_i > 0$), distance = $-m + t_i$; 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 4.32: Link-selection algorithm for HTN

4.5.2 Channel-Selection (CS) Algorithm

A deadlock-free routing algorithm for a k-ary n-cube network using dimension order routing can be obtained by using two virtual channels. The first channel is used for routing inter-node routing and the message is switched to second channel if the wrap-around link is going to be used. Only one channel is used at a time and other remains idle in dimension order routing. Efficient use of these virtual channels improves dynamic communication performance. A Channel-Selection (CS) algorithm provides an efficient use of these virtual channels. HTN is a combination of 2D-torus and 3D-torus networks. Thus, the CS algorithm can be used in the BM and higher-level networks of the HTN.

If the first virtual channel is congested and the second virtual channel is not used in routing, then the message is switched over to the second channel, or the second channel is selected initially. This is the main idea of the channel-selection algorithms. Figure 4.33 portrays a 4-PE ring network with allocation of virtual channel number. Two virtual channels are numbered as VC_0 and VC_1. When the wrap-around channels are not used in routing, such as routing a message from PE_0 to PE_2, only VC_0 is used. In this case, because VC_1 is not used in dimension order routing, it is possible to move from VC_0 to VC_1 or use VC_1 initially. That is, if path-a is congested, then the message can initially follow either path-b or switched over to path-c, as shown in Figure 4.33. Similar scenario will happen for routing message from PE_2 to PE_0. Virtual channels are efficiently used by this phenomenon.

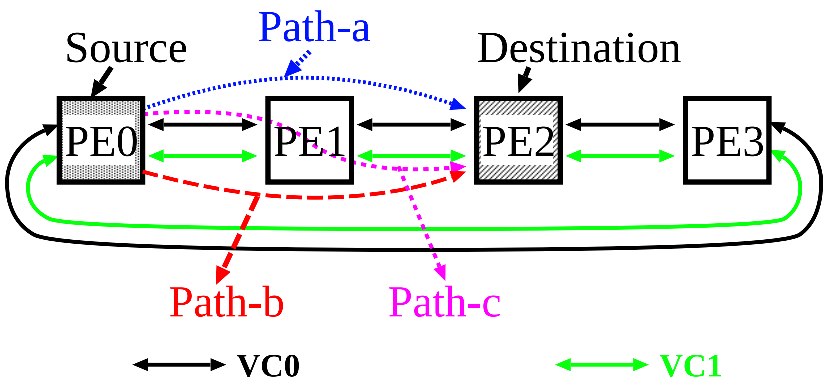


Figure 4.33: Selection of virtual channels by channel-selection algorithm

The proposed CS algorithm breaks the restriction of using virtual channels in dimension order routing and provides an efficient use of them. However, the routing order is strictly followed to route messages. The routing algorithm that applies this channel-selection principle is denoted as R_3.

4.5.3 Combination of LS and CS (LS+CS) Algorithm

The link-selection algorithm is used to select a physical link in a network and the channel-selection algorithm is used to select a virtual channel in a physical link. Therefore, both the link-selection algorithm and channel-selection algorithm can be applied at the same time. The routing algorithm that applies both the link-selection and channel-selection principles is denoted as R.

4.5.4 Deadlock-Free Routing

The most expensive part of an interconnection network is the wire that creates the physical links; for a particular topology, the physical link cost is constant. Efficient use of this physical link significantly improves dynamic communication performance. The second most expensive elements are the buffers and switches. Since we have considered wormhole-routed HTN, the main factor in buffer expense is the number of virtual channels. Virtual channels [39] reduce the effect of blocking; they are used widely in parallel computer systems, to improve dynamic communication performance by relieving contention in the multicomputer network and to design deadlock-free routing algorithms [38, 68]. Since the hardware costs increase as the number of virtual channels increases, the unconstrained use of virtual channels is not cost-effective in parallel computers. Again, efficient use of this virtual channel also improves dynamic communication performance.

In Section 4.4.6, We have shown that using three virtual channels in the HTN with dimension-order routing yields better dynamic communication performance than other approaches. In this section, we have also considered three virtual channels to make the LS and CS algorithms deadlock free. We have also considered 3 virtual channels to compare the dynamic communication performance of the HTN: the LS and CS algorithms in addition to the dimension order routing algorithm.

In Section 4.3, we have presented dimension-order routing. To prove that the dynamic routing algorithm is deadlock-free, we can use 3 virtual channels. In this section, we recall the dynamic routing algorithm and the proof of a deadlock-free routing for HTN using dimension-order routing and 3 virtual channels and present a proof for deadlock-free routing that applies the link-selection and channel-selection principles using 3 virtual channels. By using Lemma 4.1 and Corollary 4.1, we will prove that the proposed LS and CS algorithm for the HTN is deadlock-free using 3 virtual channels.

Theorem 4.4 *The routing algorithm (R_1) for the Hierarchical Torus Network (HTN) with* 3 *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-phases $2.i.1$ and $2.i.3$. These sub-phases utilize channels over intra-BM links, sharing either the channels of phase-1 or phase-3. PEs at the contours of the xy-plane are assigned to each high level as gate nodes. The exterior links 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, which 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 mesh connection of the higher-level 2D-torus network shares the virtual channel of either sub-phase $2.i.1$ or sub-phase $2.i.3$. The wrap-around connection of the higher-level 2D-torus networks requires 1 more virtual channel.

Therefore, the total number of necessary virtual channels for the whole network is 3.

As mentioned earlier, the routing of a message using the LS algorithm strictly follows the selection order. LS algorithms diversify the use of physical links in each direction of the network. Now, using theorem 4.4 and the following lemma, we will prove that the proposed LS algorithm (R_2) for the HTN is deadlock-free using 3 virtual channels.

Lemma 4.2 *In a k-ary n-cube network, if two virtual channels are used according to condition 1 or condition 2, and the links are used according to condition 3, then the network is deadlock free .*

Condition 1: *Initially use virtual channel 0.*
Condition 2: *When the packet is going to use wrap-around links, use virtual channel 1.*
Condition 3: *Packets can move either in the clockwise direction or the counter-clockwise direction if Eq. 4.8 is satisfied. Otherwise, move to a link nearer to the destination.*

Proof: The physical links and the virtual channels are allocated in the BM according to Lemma 4.1 and in the higher-level network according to Corollary 4.1. We are applying the LS algorithm in each direction of the network. Each direction of the network is a ring network. According to Lemma 4.1 and Corollary 4.1, no cyclic dependencies will occur in the BM and in the higher-level network, respectively. And according to Theorem 4.4, the whole network is deadlock free.

Theorem 4.5 *Suppose routing algorithm R_1 of the HTN is deadlock free. The routing algorithm R_2 which applies the LS algorithm is also deadlock free.*

Proof: If Eq. 4.8 (i.e., the condition for using LS algorithm) is not satisfied, then the routing of the message is carried out using routing algorithm R_1. According to Theorem 4.4, routing algorithm R_1 for the HTN is deadlock free. If the LS algorithm is used to route the message, then according to Lemma 4.2, the LS algorithm is also deadlock free. Therefore, the proposed LS algorithm R_2 is deadlock free.

As mentioned earlier, the CS algorithm effectively uses the virtual channels of a wrap-around connected interconnection networks. Now, using Theorem 4.4 and following lemma, we will prove that the proposed CS algorithm (R_3) for the HTN is deadlock-free using 3 virtual channels.

Lemma 4.3 *In a k-ary n-cube network, if two virtual channels are used according to condition 1 or condition 2, and the links are used according to condition 3, then the network is deadlock free .*

Condition 1: *Initially use virtual channel 0.*
Condition 2: *When the packet is going to use wrap-around links, use virtual channel 1.*
Condition 3: *If the wrap-around links are not used in routing and packet is in virtual channel 0, then the packet can select virtual channel 1.*

Proof: The channels are allocated according to Theorem 4.4. It is proven that channel circulation will not occur during message flow. Thus, the network is deadlock-free.

Theorem 4.6 *Suppose routing algorithm R_1 of the HTN is deadlock free. The routing algorithm R_3 which applies the CS algorithm, is also deadlock free.*

Proof: Each direction of the HTN is a ring network. Routing algorithm R_1 is deadlock-free with 3 virtual channels. According to the Lemma 4.3, the routing algorithm R_3 for the HTN is also deadlock-free with 3 virtual channels.

Theorem 4.7 *The routing algorithm R for the HTN is deadlock free with* 3 *virtual channels.*

Proof: According to Theorem 4.5, the routing algorithm R_2 for the HTN which applies the LS algorithm, is deadlock-free with 3 virtual channels. Similarly according to Theorem 4.6, the routing algorithm R_3 for the HTN which applies the CS algorithm, is deadlock free with 3 virtual channels. Therefore, the routing algorithm R for the HTN which applies both the CS and LS algorithms, is deadlock free with 3 virtual channels.

4.6 Router Cost and Speed

A wormhole router must perform several basic functions: switching, routing, flow control, multiplexing physical channels, inter-chip signaling, and clock recovery. A wormhole router architecture for HTN is shown in Figure 4.34. Data moves from left to right through the router and the complementary flow control signals move along parallel paths in the opposite direction. The essential components are the crossbar switch (CB), flow control unit (FC), address decoder (AD), routing decision unit (RD), and virtual channel controllers (VC).

Packets arriving at the router inputs are fed into the address decoders, which generate a set of requests for possible outputs. The RD combines the request from all the inputs and the router status, and matches inputs to appropriate outputs. The CB switch connects the router input to output. Our router architecture can connect all inputs to outputs in a single cycle. Once an appropriate output has been selected, the switch connection is maintained for the entire packet. Following the last flit, the switch connection is broken and the output freed for subsequent communications. The FC performs flow control between routers and buffers a message in place, if necessary. In this section, we characterize the hardware cost of our proposed router circuit in gates and its speed as latency. To evaluate the hardware cost and delay of our proposed router circuit, we employed the A.A. Chien model [139].

4.6.1 Router Gate Counts

Gate counts for router modules are shown in Table 4.3 according to the Chien Model [120, 139, 140]. Here P is the number of inputs or outputs for the CB switch. F is the routing freedom, the number of output choices an input can have, and is typically the same as P. V is the number of virtual channels that a virtual channel controller multiplexes onto the physical channel.

The total number of gates in the proposed routing network is evaluated based on the module gate count formulas in Table 4.3 and the number of modules required. In

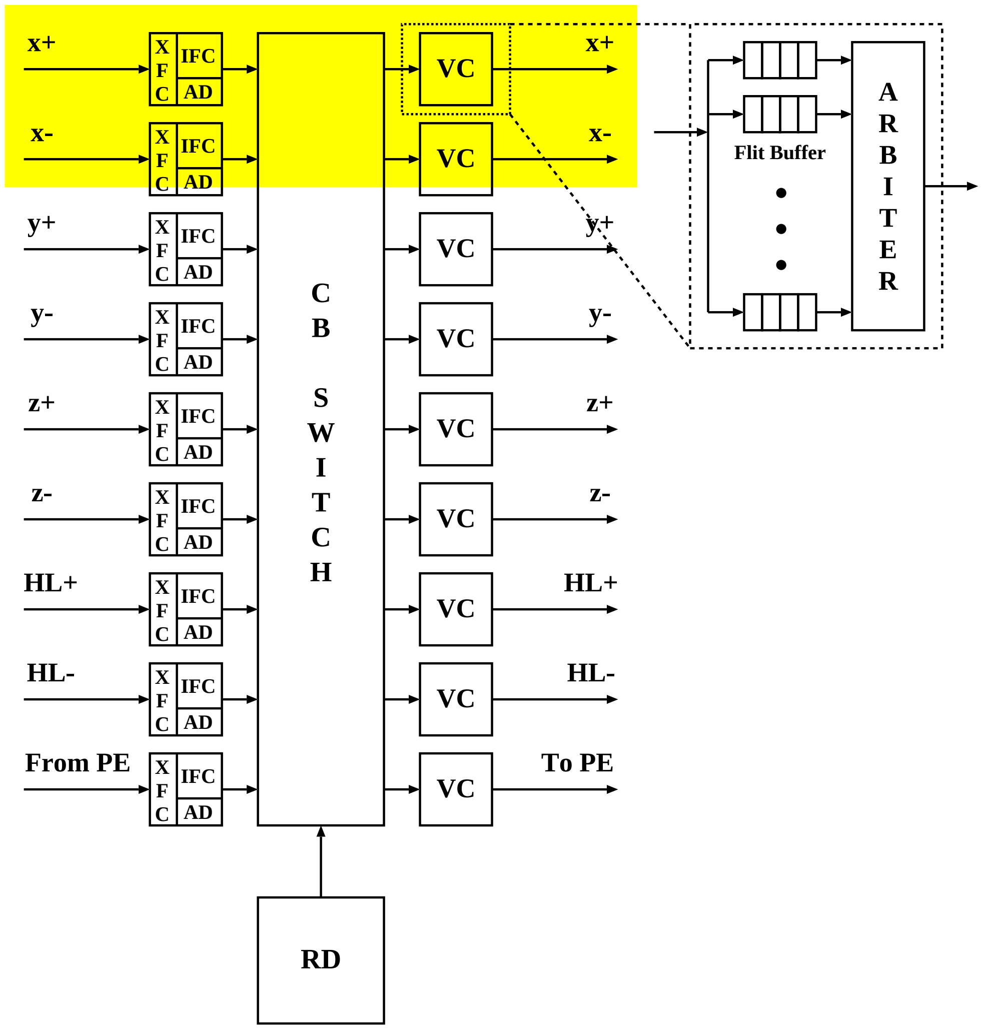


Figure 4.34: A block diagram of router architecture

our router architecture $P = 9$, $F = 9$, and $V = 3$. We need nine internal flow control units (IFC), nine address decoders, one crossbar switch, one routing decision unit, and eight virtual channel controllers. Gate counts of the proposed router architecture are shown in Table 4.4. All gate counts assumes routers with 16-bit datapaths and channels. This estimates are conservative as they do not include gates required for input/output buffering, pads, and clock synchronization.

The proposed LS and CS algorithms adapt the routing of messages within one direction only. In the whole network, the routing order is restricted as dimension order routing. This is why the hardware cost of the DOR, CS, LS, and LS+CS algorithms are almost equal. The CS algorithm allows packets a 2-way branch and the selection function is simple, so it requires little additional hardware. Like CS algorithm, the LS algorithm allows packet only 2-way branch and the selection function is also simple, thus, the hardware overhead is small. There is a similar scenario for LS+CS algorithm. According to this conservative study, the hardware cost of the proposed adaptive routing is almost equal to the dimension order routing.

Table 4.3: Gate counts for router modules

Module	Parameter	Gate Count	Complexity
Crossbar Switch	P	$29 \times P^2$	$O(P^2)$
XFC	n.a.	530	$O(1)$
IFC	n.a.	320	$O(1)$
Address Decoder	n.a.	100	$O(1)$
Routing Decision	F	$17 \times F^2$	$O(F^2)$
VC Controller	V	$126 \times V$	$O(V)$

Table 4.4: Gate counts for HTN routers

Block	# of gates	# of the blocks in the router	Total # of gates in this block type
FC	320	9	2880
AD	100	9	900
RD	1377	1	1377
CB	2349	1	2349
VC	378	8	3024
Total			10530

4.6.2 Router Speed

Routing network performance has two important attributes: routing latency and bandwidth. A router contributes to these times through its routing setup latency and its flow control latency, respectively. The per-node routing latency consists of two parts: the inter-router delay and the internal router latency. We focus on the internal router latency, the time needed to create a connection through the router. It can be decomposed as follows: (1) decode address and generate correct request (T_{AD}), (2) routing decision (T_{RD}), (3) updated header selection (T_{SEL}), (4) drive data through the crossbar (T_{CB}), and (5) virtual channel controller delay (T_{VC}). The speed of each of these operations directly affects the router latency. A router's channel bandwidth depends on the size of the flow control unit (flit) and the time to do a flow control operation. The internal flow control latency limits the flit rate on network channels in multicomputer routers. Flits are unit of resource multiplexing, so flow control latency determines the network's ability to share internal connections and external channels amongst different packets. Flow control latency can be decomposed into the following contributions: (1) FC delay (T_{FC}) (2) forward CB switch delay for data (T_{CB}), and (3) virtual channel controller delay (T_{VC}).

Formulas for the module delays are shown in Table 4.5 according to the A.A. Chien Model [120, 139, 140]. Here also, P is the number of inputs or outputs for the CB switch, F is the routing freedom, and V is the number of virtual channels. The timing estimates in Table 4.5 are based on nominal estimates of wiring capacitance, nominal processing, and nominal operating temperature. Using complete designs for each module, combined with gate-level timing estimates, gives the number in Table 4.6 for constants (c_j) used in the expressions of module delay. The numerical values presented in Table 4.6 are based on a 0.8 micron CMOS gate array process[2]. Using the values of (c_j) in Table 4.6 and the parameters of our router architecture ($P = 9$, $F = 9$, and $V = 3$), we tabulate the delay for each module in Table 4.7.

Table 4.5: Delays for the router module

Module	Delay
Crossbar	$c_0 + c_1 \times \log P$
Flow Control Unit	c_2
Address Decoder	c_3
Routing Decision	$c_4 + c_5 \times \log F$
Header Selection	$c_6 + c_7 \times \log F$
VC Controllers	$c_8 + c_9 \times \log V$

Table 4.6: Module delay constants for a 0.8 micron CMOS process.

Constant	Value (nanoseconds)
c_0	0.4
c_1	0.6
c_2	2.2
c_3	2.7
c_4	0.6
c_5	0.6
c_6	1.4
c_7	0.6
c_8	1.24
c_9	0.6

The setup delay in a dimension-order router involves decoding the address, a trivial routing decision as to whether to continue in the current dimension or proceed to the next, connecting the crossbar, sending data through the crossbar, and multiplexing physical links into virtual channels. The equation for setup delay can be written as Eq. 4.9, which, for the 0.8 micron CMOS gate array, gives a setup delay of 9.69ns. Performing the flow control operation in a dimension-order router requires delay through the crossbar, flow controller, and virtual channel controller. The flow control delay can be written as shown in Eq. 4.10, which yields 6.69ns.

$$\begin{aligned} T_{DOR} &= T_{AD} + T_{RD} + T_{CB} + T_{VC} \qquad (4.9) \\ &= 2.70 + 2.50 + 2.30 + 2.19 \end{aligned}$$

[2]Numerical figure presented are based on Mitsubishi Electronics M60007x and M6008x series 0.8 micron gate array family.

Table 4.7: Module delay for a 0.8 micron CMOS process

Module	Delay (nanoseconds)
T_{CB}	$(0.4 + 0.6 \times \log_2 9) = 2.30$
T_{FC}	2.20
T_{AD}	2.70
T_{RD}	$(0.6 + 0.6 \times \log_2 9) = 2.50$
T_{SEL}	$(1.40 + 0.60 \times \log_2 9) = 3.30$
T_{VC}	$(1.24 + 0.6 \times \log_2 3) = 2.19$

$$\begin{aligned} &= 9.69 \text{ ns} \\ T_{fc-DOR} &= T_{CB} + T_{FC} + T_{VC} \qquad (4.10) \\ &= 2.30 + 2.20 + 2.19 \\ &= 6.69 \text{ ns} \end{aligned}$$

$$\begin{aligned} T_{*S} &= T_{AD} + T_{RD} + T_{SEL} + T_{CB} + T_{VC} \qquad (4.11) \\ &= 2.70 + 2.50 + 3.30 + 2.30 + 2.19 \\ &= 12.99 \text{ ns} \\ T_{fc-*S} &= T_{CB} + T_{FC} + T_{VC} \\ &= 2.30 + 2.20 + 2.19 \qquad (4.12) \\ &= 6.69 \text{ ns} \end{aligned}$$

As mentioned earlier, the CS, LS, and LS+CS algorithms adapt the routing of messages in one direction at a time. And in the whole network, the routing order is like dimension order routing. The critical path setup in the CS, LS, and LS+CS algorithms is similar to that of the dimension order routing, with one exception. Adaptive routers make routing decisions based on router state. As the routing decision unit uses both the incoming packet and the current router state to make an output assignment, the header selection is required to select the appropriate output channel. The setup delay of CS, LS, and LS+CS algorithms consists of address decoding, routing decision, header selection, crossbar delay, and virtual channel controller delay. The router setup delay for the CS, LS, and LS+CS algorithms is calculated Eq. 4.12, which, for the 0.8 micron CMOS gate array, gives 12.99ns. The flow control paths in the CS, LS, and LS+CS algorithms are similar to that of the dimension order router. Thus, the flow control delay is shown by Eq. 4.13, which gives a flow control delay of 6.69ns.

In this section, we have presented a conservative estimate of hardware cost and setup and flow control delay of our proposed router to show the simplicity of the proposed adaptive routing algorithms. Many well known adaptive wormhole routing approaches are available in the literature. However, they are significantly more complex, and require significant hardware and additional latency beyond that needed for a dimension-order router. This is why we keep our routing algorithms (CS, LS, and LS+CS) very close to the DOR.

4.7 Dynamic Communication Performance using Adaptive Routing

Simulation Environment

We have developed a wormhole routing simulator to evaluate dynamic communication performance. In our simulation, we use dimension-order routing (DOR), link-selection (LS), channel-selection (CS), and a combination of link-selection and channel-selection (LS+CS) algorithms. The dimension-order routing algorithm, which is exceedingly simple, provides the only route for the source-destination pair. The routing restriction of the LS, CS, and LS+CS algorithms is similar to the dimension-order routing, and it provides efficient use of the network's physical links. Extensive simulations for the HTN have been carried out for five different traffic patterns: uniform [129], hot spot [130], bit-reversal [131], bit-flip [133], and perfect shuffle [134]. These traffic patterns are described in Section 4.15. Three virtual channels per physical channel are simulated, and the virtual channels are arbitrated by a round-robin algorithm. For all of the simulation results, the packet size is 16 and 256 flits; 2 flits are used as header flits. In the evaluation of dynamic communication performance, flocks of messages are sent in the network to compete for the output channels. For each simulation run, we have considered that the message generation rate is constant and the same for all nodes. Flits are transmitted at $20,000$ cycles. In each clock cycle, one flit is transferred from the input buffer to the output buffer, or from output to input if the corresponding buffer in the next node is empty. Therefore, transferring data between two nodes takes 2 clock cycles.

Dynamic Communication Performance Evaluation

In a uniform traffic pattern, message destinations are chosen randomly with equal probability between the nodes in the networks. Figure 4.35 depicts the result of simulations under uniform traffic for the HTN using the DOR, LS, CS, and LS+CS algorithms. The figure shows the average transfer time as a function of network throughput. Each curve stands for a particular algorithm. From Figure 4.35, it is seen that the maximum throughput of HTN using LS, CS, and LS+CS algorithms are higher than when the DOR algorithm is used. Also the maximum throughput of HTN using the LS+CS algorithm is higher than when the LS and CS algorithms are individually used. The average transfer time of HTN using the LS+CS algorithm is lower than when the DOR, LS, and CS algorithms are used, but the difference among them is trivial. Therefore, with the uniform traffic pattern, the LS, CS, and LS+CS algorithms achieve better dynamic communication performance than the dimension order routing algorithm. And the LS+CS algorithm yields better dynamic communication performance than individual use of the LS and CS algorithms.

For generating hot spot traffic, we used a model proposed by Pfister and Norton [130]. According to this model, each node first generates a random number. If that number is less than a predefined threshold, the message will be sent to the hot spot node. Otherwise, the message will be sent to other nodes with a uniform distribution. Here, in uniform distribution, the source and the destination are randomly selected. The hot-spot percentage is assumed to be 5%. In the HTN, the centered 4 nodes that connect the sub-network module are considered as hot spot nodes.

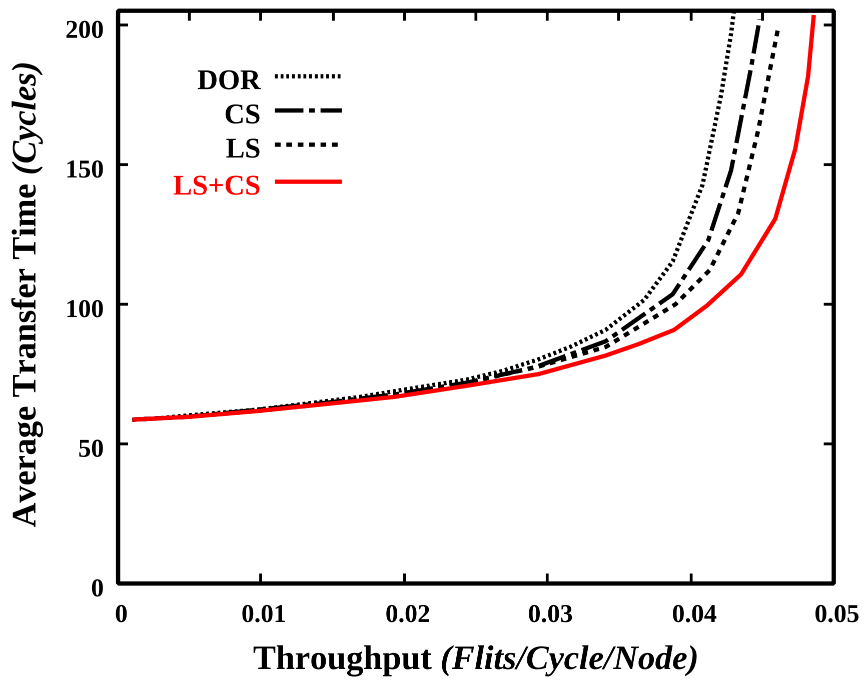
Figure 4.36 shows the message latency versus network throughput curve for hot spot traffic. From Figure 4.36, it is also seen that the maximum throughput of HTN using the LS, CS, and LS+CS algorithms is higher than when the DOR algorithm is used. Also the maximum throughput of HTN using the LS+CS algorithm is higher than when the LS and CS algorithms are individually used. The average transfer time of HTN using the LS+CS algorithm is lower than when the DOR, LS, and CS algorithms are used, but the difference among them is trivial. Therefore, with the hot-spot traffic pattern, the LS, CS, and LS+CS algorithms achieve better dynamic communication performance than the dimension order routing algorithm. And the LS+CS algorithm achieves better dynamic communication performance than either LS or CS algorithm.

In the bit-reversal traffic pattern, a node with address Node $(a_{\beta-1}, a_{\beta-2} \ldots a_1, a_0)$ sends messages to Node $(a_0, a_1, \ldots a_{\beta-2}, a_{\beta-1})$. Figure 4.37 depicts the result of simulations under bit-reversal traffic for the HTN. From Figure 4.37, it is seen that the maximum throughput of HTN using the LS, CS, and LS+CS algorithms are higher than when the DOR algorithm is used. Also the maximum throughput of HTN using the LS+CS algorithm is higher than when the LS and CS algorithms are individually used. The average transfer time of HTN using the LS+CS algorithm is lower than when the DOR, LS, and CS algorithms are used, but the difference among them is trivial. Therefore, with the bit-reversal traffic pattern, the LS, CS, and LS+CS algorithms achieve better dynamic communication performance than the dimension order routing algorithm. And the LS+CS algorithm yields better dynamic communication performance than the LS and CS algorithms used individually.

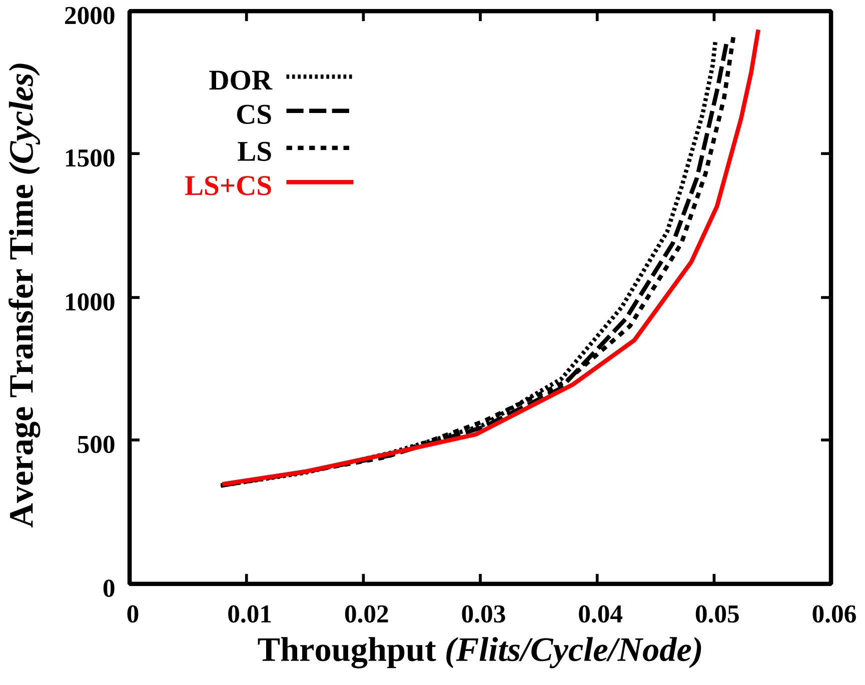
Figure 4.38 shows the message latency versus network throughput curve for bit-flip traffic. From Figure 4.38, it is seen that maximum throughput of HTN using the LS, CS, and LS+CS algorithms is higher than when the DOR algorithm is used. Also, the maximum throughput of HTN using the LS+CS algorithm is higher than when the LS and CS algorithms are individually used. The average transfer time of HTN using the LS+CS algorithm is lower than when the DOR, LS, and CS algorithms are used. Therefore, with the bit-flip traffic pattern, the LS, CS, and LS+CS algorithms achieve better dynamic communication performance than the dimension order routing algorithm. And the LS+CS algorithm yields better dynamic communication performance than the individual use of the LS and CS algorithms.

Figure 4.39 shows the message latency versus network throughput curve for perfect shuffle traffic. From Figure 4.39, it is seen that maximum throughput of HTN using the LS, CS, and LS+CS algorithms is higher than when the DOR algorithm is used. Also, the maximum throughput of HTN using the LS+CS algorithm is higher than when the LS and CS algorithms are individually used. The average transfer time of HTN using the LS+CS algorithm is lower than when the DOR, LS, and CS algorithms are used. Therefore, with the perfect shuffle traffic pattern, the LS, CS, and LS+CS algorithms achieve better dynamic communication performance than the dimension order routing algorithm. And the LS+CS algorithm yields better dynamic communication performance than the individual use of the LS and CS algorithms.

From these results, as shown in Figures 4.35, 4.36, 4.37, 4.38, and 4.39, it is clear that the selection algorithms (LS, CS, and LS+CS) outperform the dimension order routing algorithm, especially in terms of network throughput. Using the LS+CS algorithm on



(a) Short message



(b) Long message

Figure 4.35: Comparison of dynamic communication performance of the HTN between DOR, LS, CS, and LS+CS algorithms with uniform traffic pattern: 1024 nodes, 3 virtual channels, and $q = 1$.

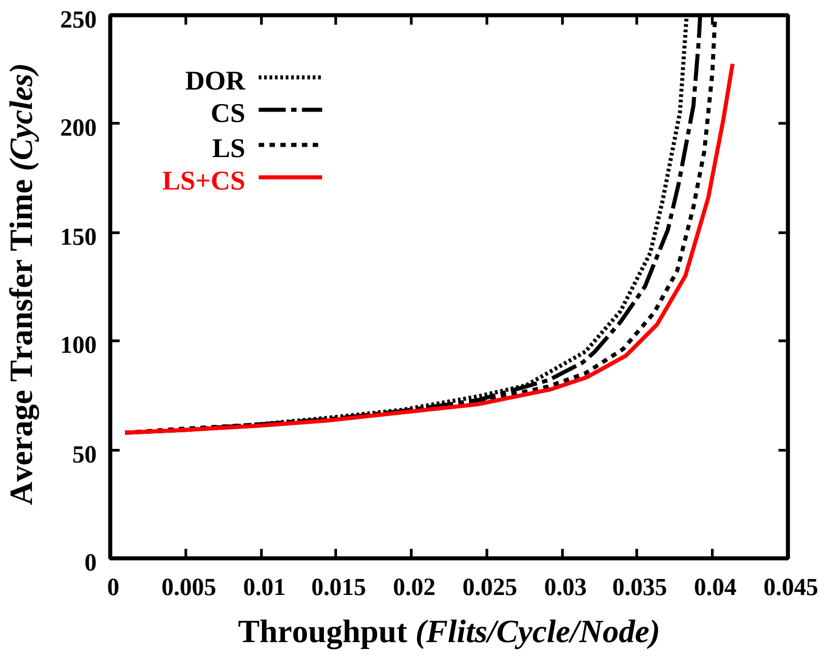


Figure 4.36: Comparison of dynamic communication performance of the HTN between DOR, LS, CS, and LS+CS algorithms with 5% hot-spot traffic pattern: 1024 nodes, 3 virtual channels, short message, and $q = 1$.

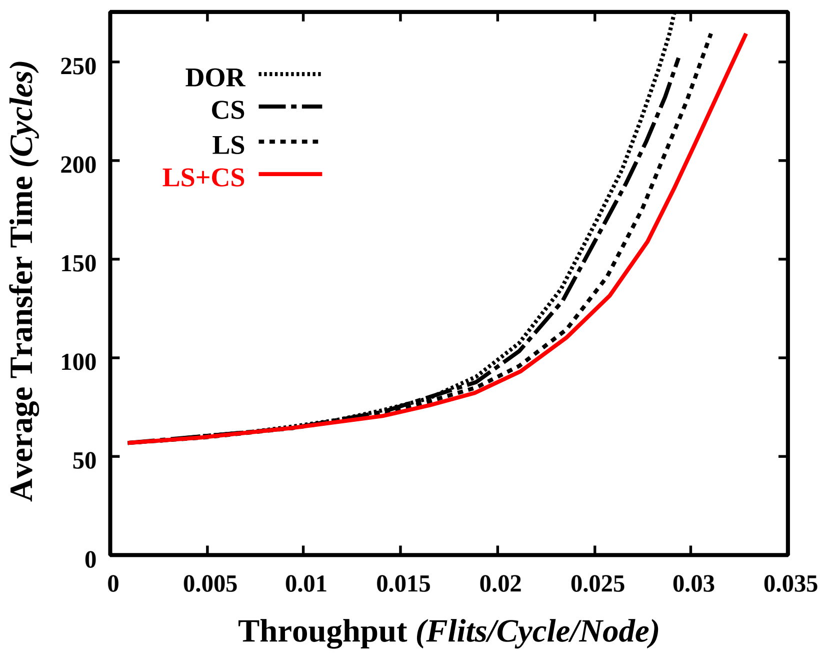


Figure 4.37: Comparison of dynamic communication performance of the HTN between DOR, LS, CS, and LS+CS algorithms with bit-reversal traffic pattern: 1024 nodes, 3 virtual channels, 16 flits, and $q = 1$.

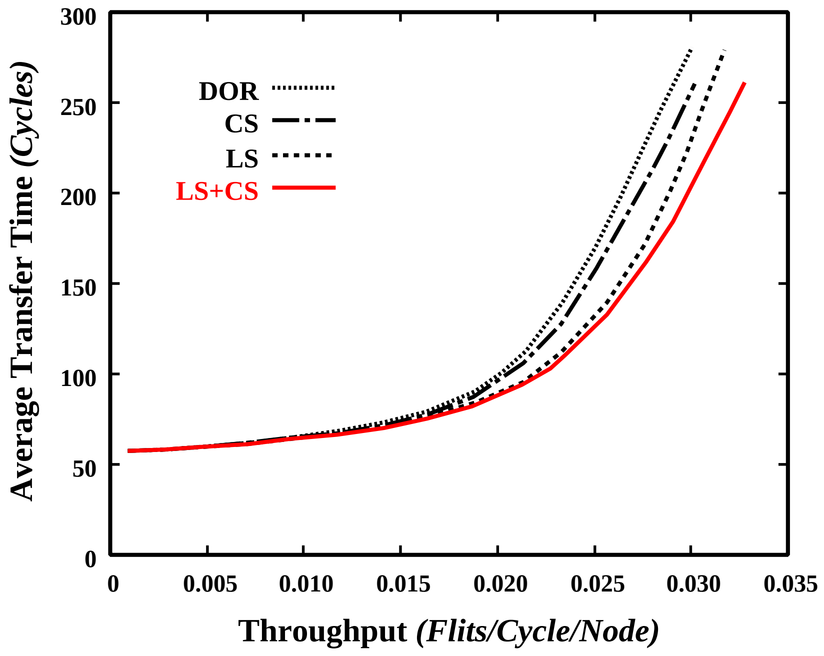


Figure 4.38: Comparison of dynamic communication performance of the HTN between DOR, LS, CS, and LS+CS algorithms with bit-flip traffic pattern: 1024 nodes, 3 virtual channels, short message, and $q = 1$.

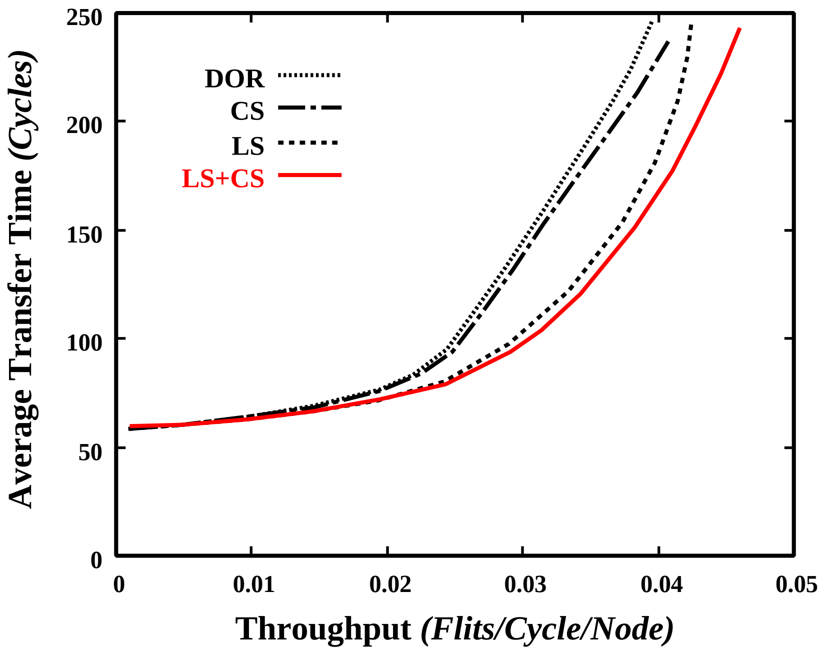


Figure 4.39: Comparison of dynamic communication performance of the HTN between DOR, LS, CS, and LS+CS algorithms with perfect shuffle traffic pattern: 1024 nodes, 3 virtual channels, short message, and $q = 1$.

HTN achieves better network throughput than the individual use of either the LS or the CS algorithms.

Performance Improvement

The maximum throughput and the average transfer time to achieve this maximum throughput of the HTN under different traffic patterns using DOR, LS, CS, and LS+CS algorithms is presented in Figure 4.40 and their corresponding numerical values are plotted in Table 4.8. The enhancement of maximum throughput and reduction of message latency are shown in percentage. It is shown that, in all traffic patterns the maximum throughput is increased. Also the average transfer time to achieve the maximum throughput is reduced. Therefore, LS+CS algorithm significantly improves the dynamic communication performance of the HTN over the dimension-order routing algorithm is used.

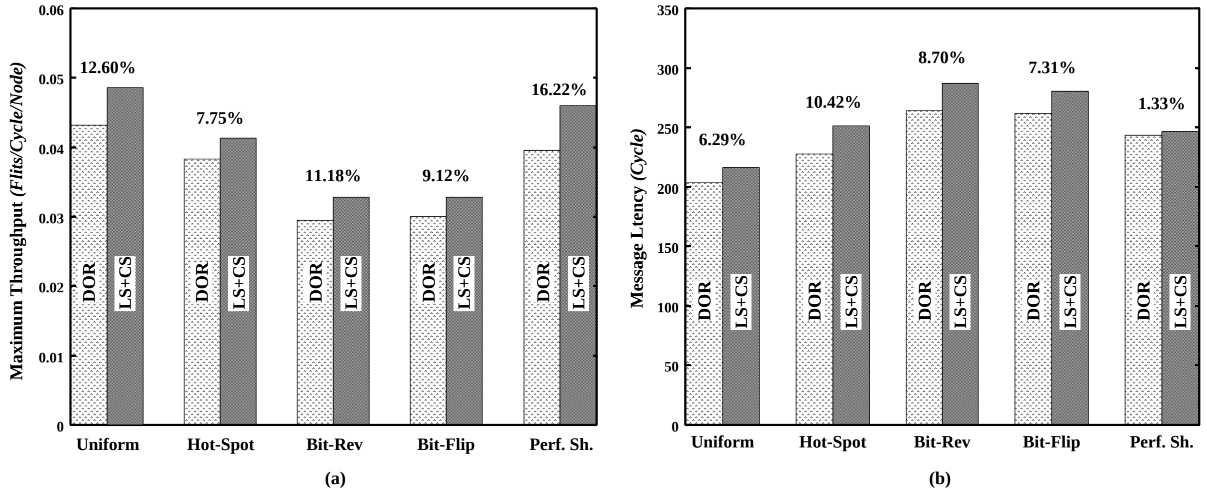


Figure 4.40: Dynamic communication performance improvement by LS+CS algorithm over DOR algorithm (a) Maximum throughput enhancement and (b) Message latency reduction.

Table 4.8: Performance Improvement using selection algorithm over dimension-order routing

Traffic Patterns	Maximum Throughput		Message Latency		Throughput Enhancement	Latency Reduction
	DOR	LS+CS	DOR	LS+CS	%	%
Uniform	0.04317	0.04861	216.01	203.23	12.60	6.29
Hot-Spot	0.03833	0.04130	251.11	227.41	7.75	10.42
Bit-Reversal	0.02951	0.03281	287.09	264.11	11.18	8.70
Bit-Flip	0.03005	0.03279	280.49	261.38	9.12	7.31
Perfect Shuffle	0.03958	0.04600	246.42	243.19	16.22	1.33

4.8 Conclusions

In this chapter we have proposed a deadlock free routing algorithm using dimension order routing with a minimum number of virtual channels for the HTN. It has been proven that 2 virtual channels per physical channel are sufficient for the deadlock free routing algorithm of the HTN; 2 is also the minimum number of virtual channels for dimension order routing. By using dimension order routing and various traffic patterns, we have evaluated the dynamic communication performance of the HTN as well as that of several other commonly used networks and hierarchical interconnection networks. The average transfer time of HTN is lower than that of the H3D-mesh, H3D-torus, TESH, mesh, and torus networks. Maximum throughput of the HTN is also higher than that of those networks. A comparison of dynamic communication performance reveals that the HTN outperforms the H3D-mesh, H3D-torus, TESH, mesh, and torus networks because it yields low latency and high throughput, which are indispensable for high performance massively parallel computers.

We have analyzed the effect of the number of virtual channels and select the best choice for the number of virtual channels, which yields optimum performance. It is shown that 3 virtual channels per physical channels is the best choice to achieve optimum performance. Three channels improve the performance substantially. We have investigated the impact of non-uniform traffic patterns on the HTN using dimension order routing with 3 virtual channels. Under various nonuniform traffic patterns, the dynamic communication performance of the HTN is always better than that of the H3D-mesh, H3D-torus, TESH, and mesh networks. It is also shown that the impact of non-uniform traffic patterns on the HTN is less than on the other networks.

In this chapter, we have described a suite of low cost adaptive routers, LS, CS, and LS+CS, with dimension order routing, analyzed their cost, and evaluate the dynamic communication performance for the HTN. The hardware cost for the LS, CS, and LS+CS algorithms is exactly equal to dimension order routing. Based on a 0.8 micron gate array technology, we characterized the speed of those adaptive routers including the popular dimension order router. The inter-node router delay of path setup and data through are 9.69 ns & 6.69 ns for DOR and 12.99 ns & 6.69 ns for for the LS, CS, and LS+CS algorithms respectively. The only overhead imposed is router delay for header selection.

The proposed adaptive routing algorithms – CS, LS, and LS+CS – are simple and efficient for using the physical links and virtual channels of an HTN to improve dynamic communication performance. The freedom from deadlock of the proposed CS, LS, and LS+CS algorithms using 3 virtual channels has been proved. Using the routing algorithms described in this paper and several traffic patterns, we have evaluated the dynamic communication performance of the HTN. The average transfer time of the HTN using the CS, LS, and LS+CS algorithms is lower than when the dimension order routing is used, but the differences are not impressive. On the other hand, maximum throughput using the CS, LS, and LS+CS algorithms is higher than when the dimension order routing algorithm is used. Efficient use of physical links and virtual channels improves the dynamic communication performance significantly. A comparison of dynamic communication performance reveals that the LS+CS algorithm outperforms all other algorithms; an HTN using the LS+CS algorithm yields low latency and high throughput.

Chapter 5

"The things that will destroy us are: politics without principle; pleasure without conscience; wealth without work; knowledge without character; business without morality; science without humanity; and worship without sacrifice. "
– Mahatma Gandhi (1869–1948)

Reconfiguration Artchitecture and Application Mappings

5.1 Introduction

Massively parallel computer systems usually have stringent requirements for reliability [33] because of the large investments in such systems as well as the nature of the applications for which they are likely to be used. A fault-tolerant [141] network has the ability to route information with the presence of certain faults in the network. In hardware based fault tolerant technique, network is enhanced by additional hardware and providing enough redundancy in the original network design to tolerate a certain number of faults [42, 48, 49, 72, 142]. In software based fault tolerant technique, a certain amount of faults can be tolerated by clever routing algorithm. Despite dramatic improvements in defect density in recent years, it is still necessary to provide redundancy and defect circumvention to achieve acceptable system-level yields.

In parallel computers, the problem is broken into smaller parts, which are solved simultaneously, each by a different node. Mapping an algorithm to an interconnection network involves mapping computations to the nodes of that network so that the algorithm runs efficiently. Thus, the mapping of different applications, especially those used with regularity for computations in the designed network, is a desirable attribute when designing an interconnection network for massively parallel computers [72]. Frequently used advanced applications in computation such as sorting, FFT, and finding the maximum, are mapped to an interconnection network to observe its suitability.

The remainder of this chapter is organized as follows: Section 5.2 describes the hierarchical redundant scheme for reconfiguration and the critical issue of fault tolerance performance – estimation of system yield by redundancy. Mapping of various primitive applications is discussed in Section 5.3. Finally, Section 5.4 contains a summary of this chapter.

5.2 Reconfiguration Architeccture of the HTN

5.2.1 Reconfiguration Scheme

Despite continuing improvements in wafer processing, defect densities still significantly affect production yields. Since the performance of the HTN depends critically on the underlying topology, a fault tolerant HTN must be preserved by offering spare nodes and the means for plugging them in place of the faulty nodes. The goal of such a reconfiguration is to offer fully functional HTN in spite of internal faults. Spare modules are provided within each level of the HTN to support the fault tolerance.

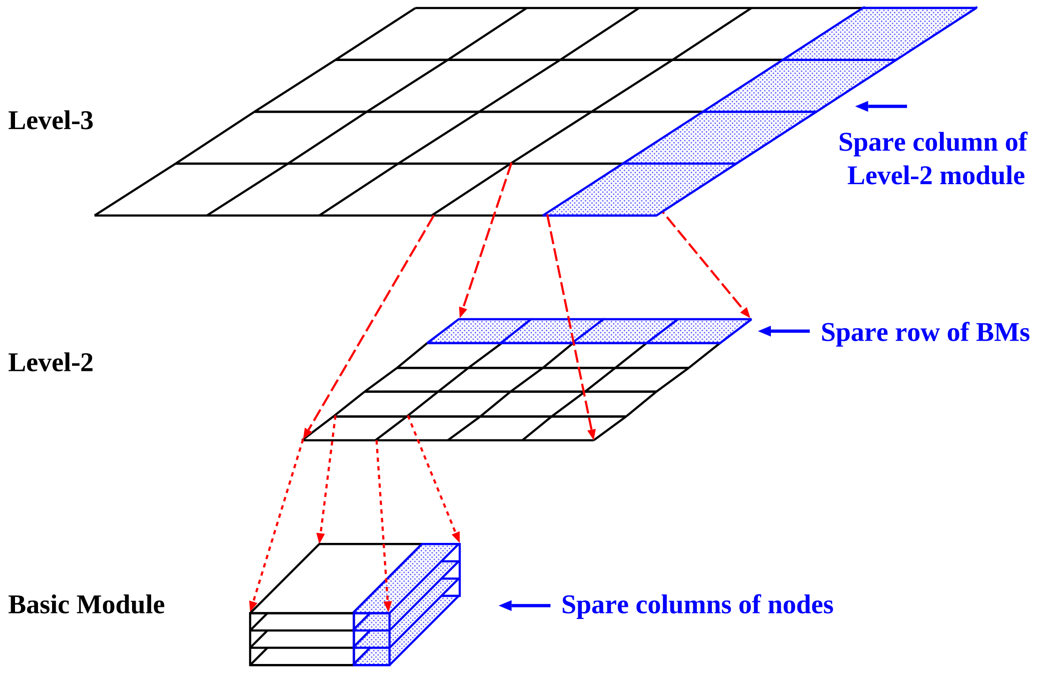


Figure 5.1: Hierarchical redundancy of the HTN

The HTN is implemented with redundancy at each level of hierarchy, i.e., BMs 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 Figure 5.1.

The soft switches used for reconfiguration around the defective nodes (or the defective BMs) are the usual 12-transistor CMOS switch [141]. 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, are portrayed in Figure 5.2.

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 switches to reconfigure the module using the healthy nodes. We need a large number of 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 xy-plane of the BM includes a column of spare nodes to replace the faulty nodes of

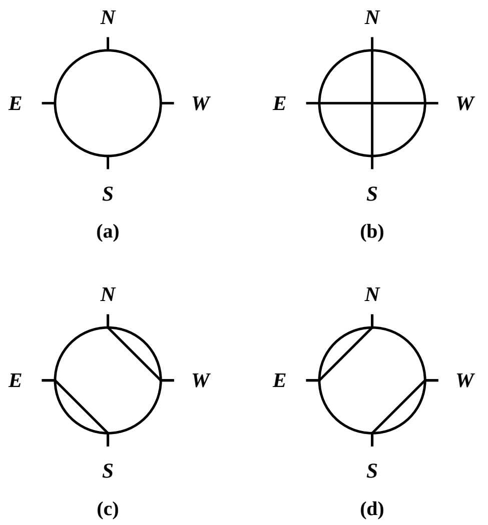


Figure 5.2: Different switch states for reconfiguration: (a) no connect, (b) north-to-south and east-to-west, (c) north-to-west and south-to-east, and (d) north-to-east and south-to-west connects.

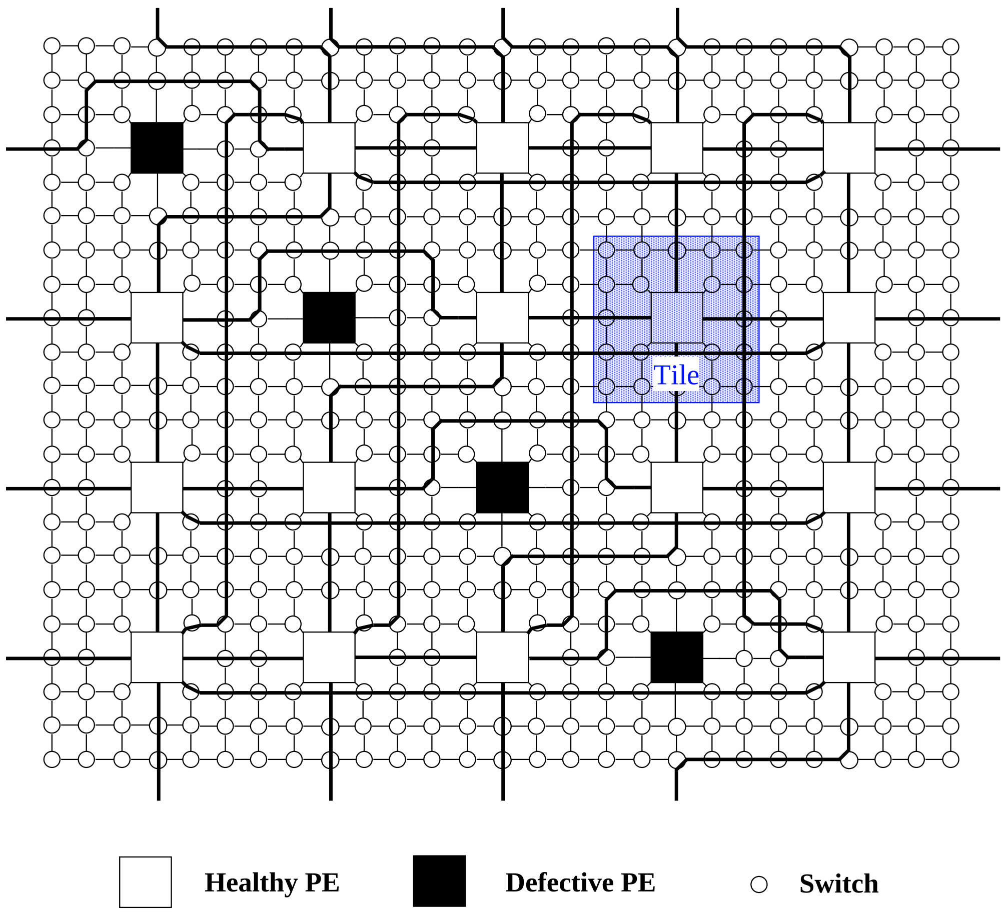


Figure 5.3: Reconfiguration of a plane for the BM in the presence of 4 faulty PEs: Diagonal

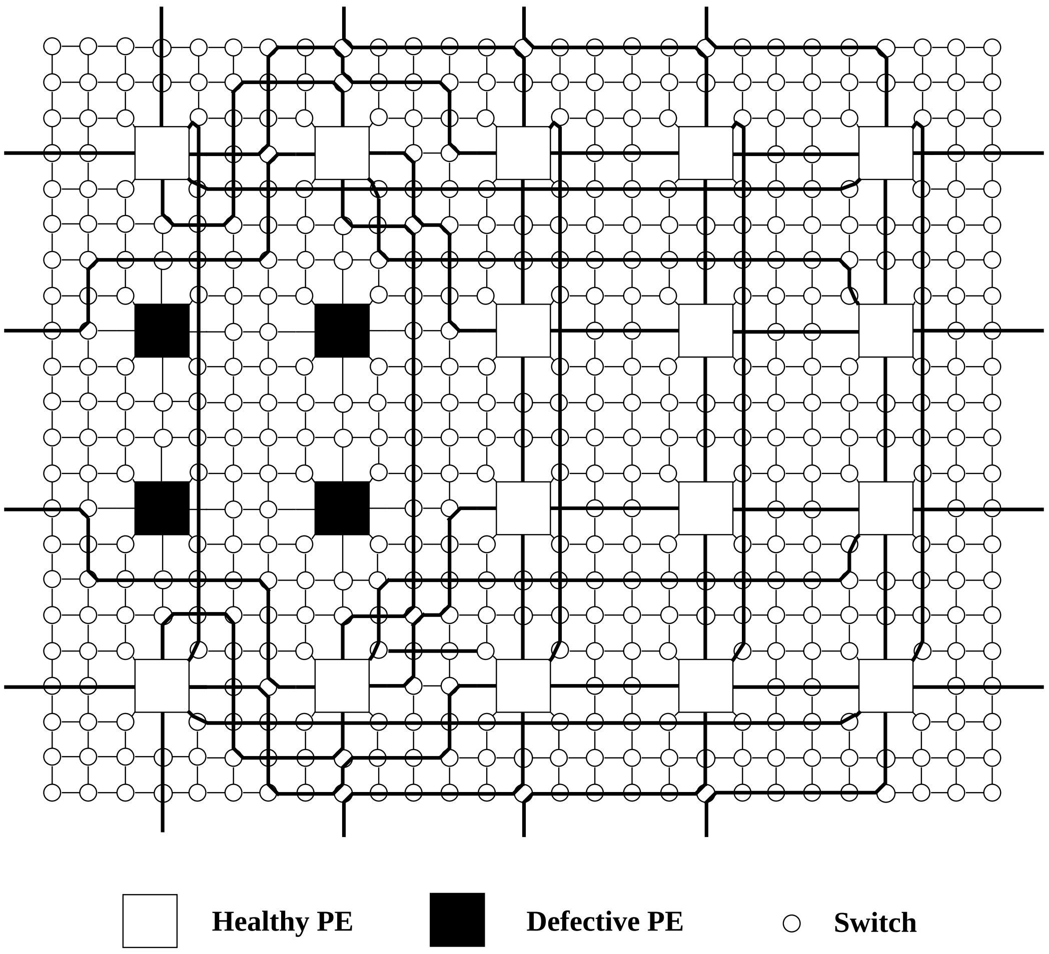


Figure 5.4: Reconfiguration of a plane for the BM in the presence of 4 faulty PEs: Square

that plane using the healthy nodes of the spare column. Each xy-plane of the BM is a 2D-torus network.

For example, a $(4 \times 4 \times 4)$ redundant basic module with the 4 columns of spare nodes has $5 \times 4 \times 4 = 80$ 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 by three examples in Figures 5.3, 5.4, and 5.5. In Figure 5.3, four faulty PEs are placed diagonally. In Figure 5.4, four faulty PEs are placed in square at the left side. In Figure 5.5, four faulty PEs are placed in concatenated L-shape and inverse L-shape in the center. We have also tested for several other cases. For simplicity, we have presented three examples here. In all the cases, the plane of the BM is successfully reconfigured using the spare PEs. Therefore, a maximum of 4 faulty PEs per plane can be tolerated through replacement by the spare PEs. Needless to say that more than 4 PEs per plane could turn out to be defective or faulty in which case the BM is declared as non-reconfigurable.

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 BM and

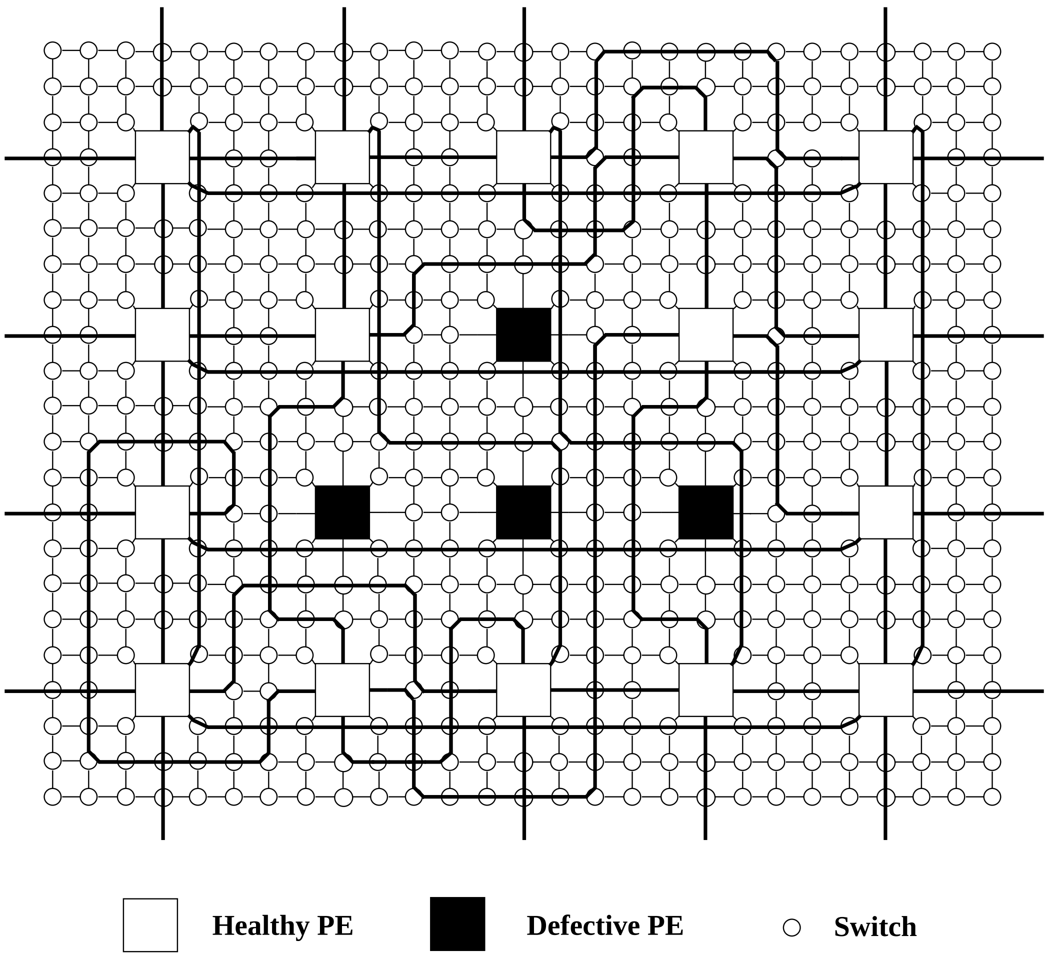


Figure 5.5: Reconfiguration of a plane for the BM in the presence of 4 faulty PEs: Concatenated L-shape and inverse L-shape

adjacent BMs. This can enhance the harvesting even further.

5.2.2 System Yield of the HTN

To examine the reconfigurability of the HTN against redundancy, in this section, we 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 BM 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 to be used to construct a BM and (b) the probability of all links and switches being healthy. This is a conservative estimation to evaluate the system yield by redundancy.

To enhance the yields, defect tolerant chips can be constructed using the concept of spare modules. Suppose a large chip has been implemented with P modules but only Q functional modules are needed for correct functionality. As this is a conservative estimation and each plane of the redundant BM has 4 extra PEs. Therefore, in a $(4 \times 4 \times 4)$ BM, a maximum of 4 faulty PEs can be tolerated in each plane of the BM. That is, a maximum of 16 faulty PEs can be tolerated in each BM. The yield of the PEs per plane of the BM can be estimated by Eq. 5.2. Four planes are available in that BM. Thus, the yield of the PEs per BM can be estimated by Eq. 5.2. Here, Y_{node} is the probability that a PE is fault free.

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

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

Consider that the size of the PE is 2×2 mm^2 in 0.25 micron CMOS[1] technology [49], then $Y_{node} = e^{-0.04D}$, where D denotes the fault density per cm^2. Assuming a channel width of 1.0 mm between PEs, the tile area is 9.00 mm^2. The tile is shown in Figure 5.3. Since the PE area is 4 mm^2, the total channel area per tile becomes 5.00 mm^2. It is reasonable to assume that the area occupied by the switches and links is about 30% of the channel area [48, 49]; further, we assume that 50% of this is critical area [48, 49]. Therefore, the critical area used by the switches and links per tile is 0.75 mm^2. As stated above, each redundant plane of the BM consists of 20 PEs or titles. Thus, the total critical area of switches and links per plane, denoted as *CA_of_SL*, is $20 \times 0.75 = 15.00$ $mm^2 = 0.15$ cm^2. Therefore the yield of the switches and links is computed as:

$$\begin{aligned} Y_{switches,\ links} &= \left[e^{-CA\_of\_SL \times D}\right]^4 \\ &= \left[e^{-0.15D}\right]^4 \end{aligned} \tag{5.4}$$

For the second level, as mentioned above, a row of spare BMs is provided. Therefore, the second level network yield becomes as follows:

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

For different fault densities without and with spare elements the yield is shown in Figure 5.6 and Figure 5.7, respectively. It is shown that the use of an additional column

[1]A processing element with significant local memory and processing power would of course be much larger. However, we are considering medium grain PEs so that the 2×2 mm square is a reasonable size for this case example. Other sizes would lead to somewhat different parameters, and corresponding changes in the discussion.

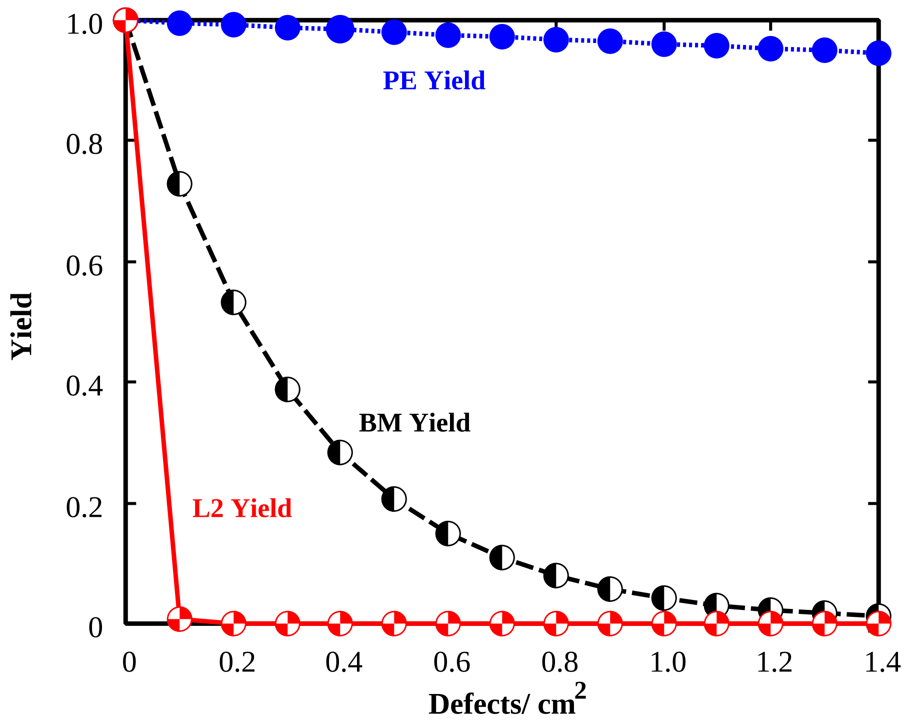


Figure 5.6: Yield for BM and Level-2 network vs. fault density without spare node

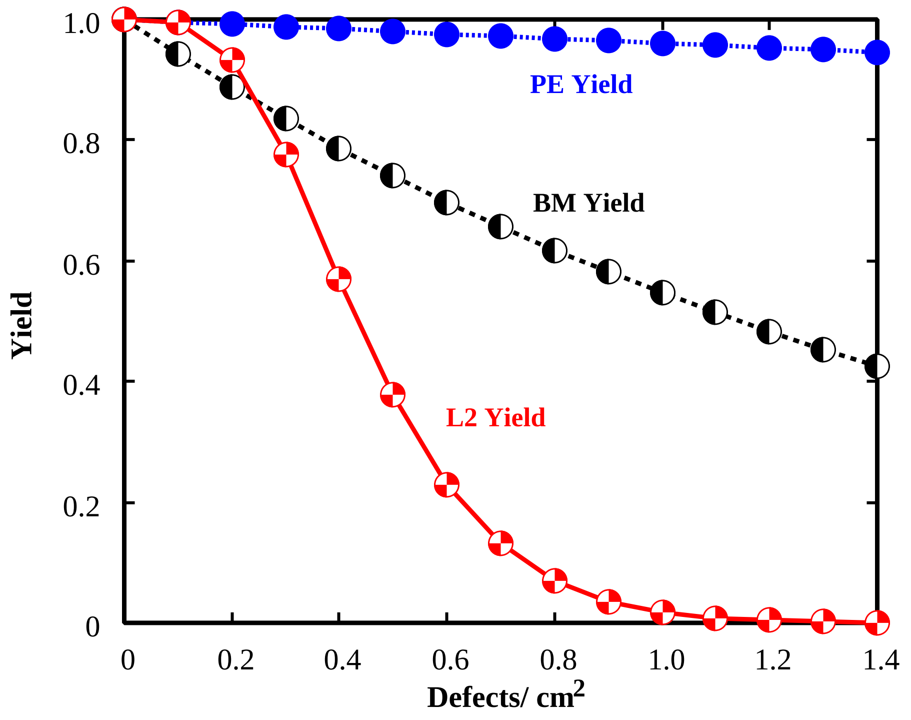


Figure 5.7: Yield for BM and Level-2 network vs. fault density with spare node

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 $faults/cm^2$ to 0.50 $faults/cm^2$, pointing to a satisfactory network yield.

5.3 Application Mappings on HTN

The interconnection network used in a massively parallel computer systems plays a key role in determining how fast applications can be executed on the system. Any non-trivial parallel program running on a multicomputer requires some amount of information exchange among the nodes; in many cases, the time for such communication is a significant part of the overall running time of the program. Therefore, it is important to design algorithms and applications to use the interconnection network efficiently.

There are two cases of mapping algorithms to an interconnection network. Mapping an algorithm to an architecture involves mapping computations to processors so that the algorithm runs efficiently. In the first, a known parallel algorithm for solving the problem is mapped to the interconnection network to achieve maximum parallelism. This does not modify the computation behavior of the algorithm. In the second case, 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 advantage of the underlying network. Thus, mapping also includes the design of parallel algorithms for a specific interconnection network. When computations are mapped to nodes, nodes that communicate frequently should be able to do so through the interconnection network efficiently. With a particular algorithm and a fixed interconnection network, different mapping may require different amounts of communications; the goal is to chose a mapping that minimizes the total communication.

Most parallel algorithms are designed so that they have some regular types of data movement among nodes. These data movement may involve frequently-used permutations such as shuffle or matrix transpose, global communication operations such as broadcasting, or may be expressed in terms of more elementary operations such as finding the maximum value or sorting. How fast such common operations can be executed on an interconnection network determines the performance of the parallel algorithm on that multicomputer. Therefore, the suitability of a given interconnection network for certain applications can often be estimated by studying how efficiently these common operations such as sorting, finding the maximum value, and broadcasts can be performed on the given network. In this section, we will discuss the mapping of known advanced applications, namely bitonic merge, Fast Fourier Transform (FFT), and finding the maximum on the HTN to observe the suitability of the HTN .

5.3.1 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 functions to the interconnection networks [47].

Let N be an integer number, where $N = 2^k$, the value of data be $d[w]$, where $(w = 0, 1, 2, \ldots \ldots \ldots N-1)$. The 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, the CONVERGE function executes an operation between $2^{k-1}, 2^{k-2}, \ldots \ldots \ldots, 2^1, 2^0$. The CONVERGE and DIVERGE functions are defined as follows:

```
 CONVERGE();
  for $j = k-1 : -1 : 0$
    for $0 \le w \le N-1$ do in parallel
      if $a_j = 0$,**OPERATION**$(w, w+2^j)$; endif;
    endfor;
  endfor;
end;

 DIVERGE();
  for $j = 0 : k-1$
    for $0 \le w \le N-1$ do in parallel
      if $a_j = 0$,**OPERATION**$(w, w+2^j)$; endif;
    endfor;
  endfor;
end;
```

Where a_j is the j-th bit of the binary representation of w, **OPERATION**$(w, w+2^j)$ is an operation between $d[w]$ and $d[w+2^j]$. Figure 5.8 illustrates the execution of converge on a 4×4 2D-mesh.

5.3.2 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 [143, 144] 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 $a_1, a_2, a_3, \ldots\ldots\ldots, a_{2n}$ is said to be bitonic if either*

1. *there is an integer $1 \le j \le 2n$ such that $a_1 \le a_2 \le a_3 \le \ldots\ldots \le a_j \ge a_{j+1} \ge a_{j+2} \ge \ldots\ldots \ge a_{2n}$, or*

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 $\{2, 5, 7, 8, 6, 4, 3, 1\}$.

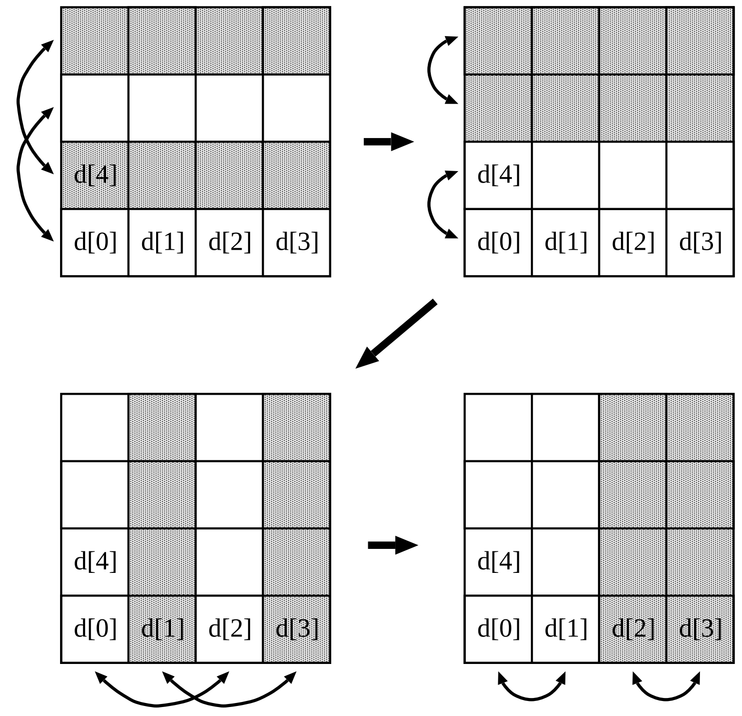


Figure 5.8: CONVERGE on a 4 × 4 2D-mesh

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* $1 \le i \le n$*, then*

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

The bitonic merge 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($w, w + 2^j$)
{
  move $R_1(w + 2^j)$ to $R_2(w)$;
  if $a_{j+1}$=0,
    [$R_1(w), R_2(w)$]=[min{$R_1(w), R_2(w)$},max{$R_1(w), R_2(w)$}];
  else
    [$R_1(w), R_2(w)$]=[max{$R_1(w), R_2(w)$}, min{$R_1(w), R_2(w)$}];
  endif;
  move $R_2(w)$ to $R_1(w + 2^j)$;
}
```

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

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

```
    move $R_2(w)$ to $R_1(w + 2^j)$;
  }
```

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

5.3.3 Fast Fourier Transform (FFT)

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

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

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

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

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

Since the two input data of a butterfly operation in the w-th stage are 2^{k-w} 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($w, w + 2^j$)
 {
   move $R_1(w + 2^j)$ to $R_2(w)$;
   $Temp(w) = R_1(w) + R_2(w)$;
   $R_2(w) = (R_1(w) - R_2(w)) * W_N{}^r$;
   $R_1(w) = Temp(w)$;
   move $R_2(w)$ to $R_1(w + 2^j)$;
 }
```

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 the time required to perform the operations in the middle 3 lines of the above pseudo-code.

5.3.4 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 $0 \leq w \leq \frac{N}{2}^{k-j-1} - 1$ do in parallel
      if $a_j = 0$, **OPERATION**$(w, w+2^j)$; endif;
    endfor;
  endfor;
end;
```

Thus, the operation to find the maximum is given by:

```
**OPERATION**$(w, w+2^j)$
  {
    move $R_1(w+2^j)$ to $R_2(w)$;
    $R_1(w) = \max[R_1(w), R_2(w)]$;
  }
```

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**$(w, w+2^j)$;endif;
        CONVERGE_network-$j$();
      endif;
    endfor;
  }
```

5.3.5 Processing Time

To estimate the processing time of an application mapping on an 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. The execution time depends on the system clock cycle. For a particular system, the execution time is constant. Consideration of execution time is beyond the scope of this dissertation. CONVERGE and DIVERGE functions require $\log N$ steps. Thus, the total time required for an application is as follows:

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

where,

$$\mu = \text{Total number of communication steps.}$$
$$T_{move} = \text{Transfer time between adjacent nodes.}$$
$$T_{OPER} = \text{Execution time for OPERATION.}$$

Application mapping performance 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

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

$$S^{2D-Torus}_{max} = 2\left(\sqrt{N}-1\right) \qquad (5.10)$$

3D-Torus

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

$$S^{3D-Torus}_{max} = 3\left(N^{\frac{1}{3}}-1\right) \qquad (5.12)$$

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

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

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

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 $(4\times4\times4)$ and the size of the higher level network is $(4\times4\times4)$, then

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

$$S^{H3D-Torus}_{max} = 18L \qquad (5.16)$$

H3D-Mesh

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

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

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

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

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

HTN

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

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

where,

$$\begin{aligned} L &= \text{Level number.} \\ S^{BM}_{bitonic} &= \text{No. of communication steps in the BM.} \end{aligned}$$

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

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

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

The total number of communication steps for bitonic merge on various interconnection networks is shown in Figures 5.9 and 5.10. The total number of nodes of various networks is different due to the difference of their architecture. To fit various networks and make a versatile comparison, we plotted the number of communication steps for bitonic merge in two graphs. The figures show the number of communication steps as a function of network size for different networks. Each curve stands for a particular network. Since bitonic merge requires a large communications between all nodes at each step, the total number of

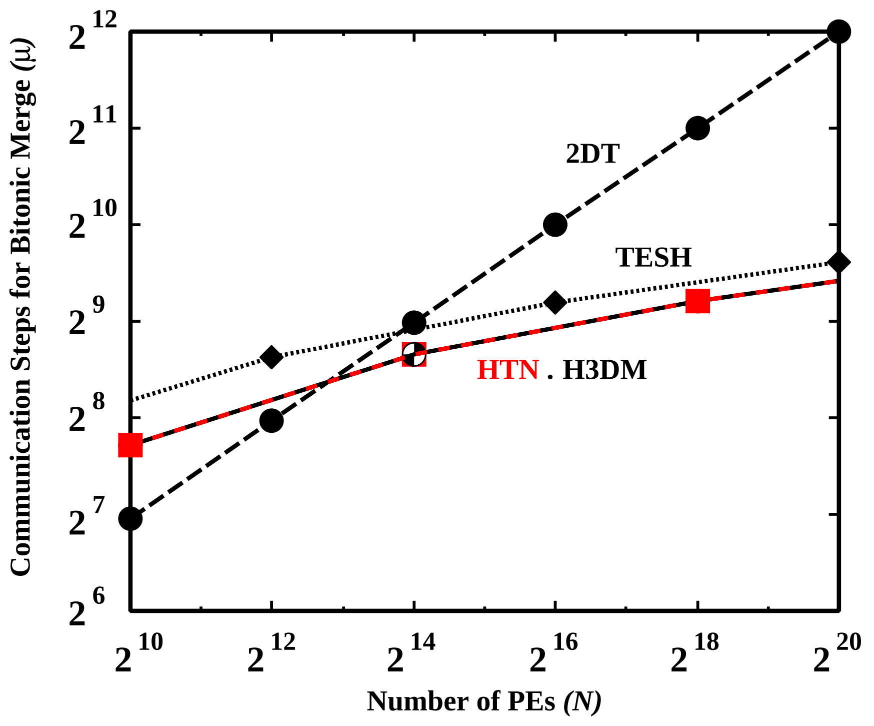


Figure 5.9: The total number of communication steps of the bitonic merge in different networks

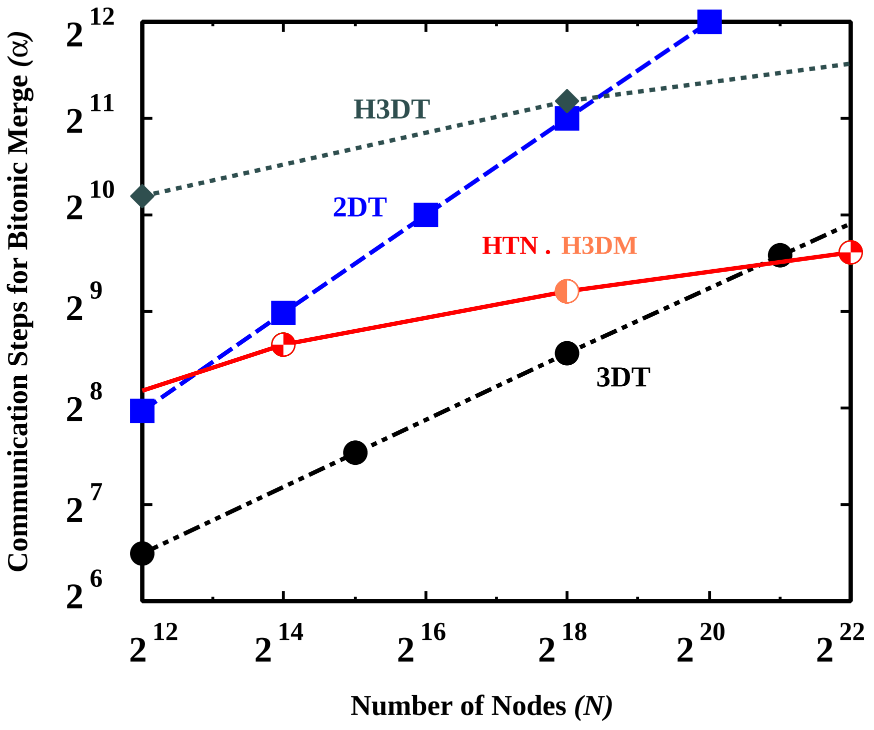


Figure 5.10: The total number of communication steps of the bitonic merge in different networks

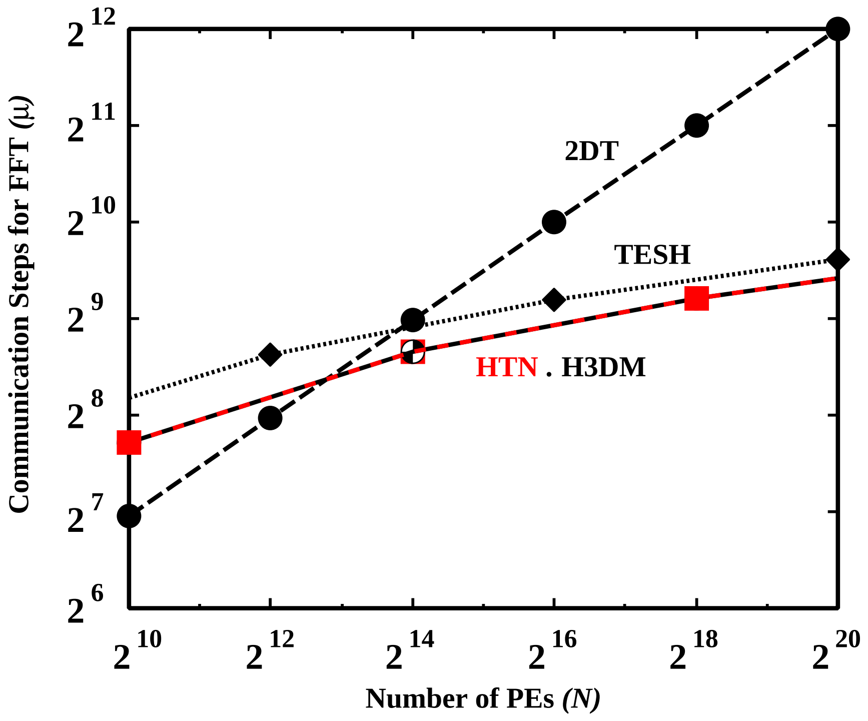


Figure 5.11: The total number of communication steps of the FFT in different networks

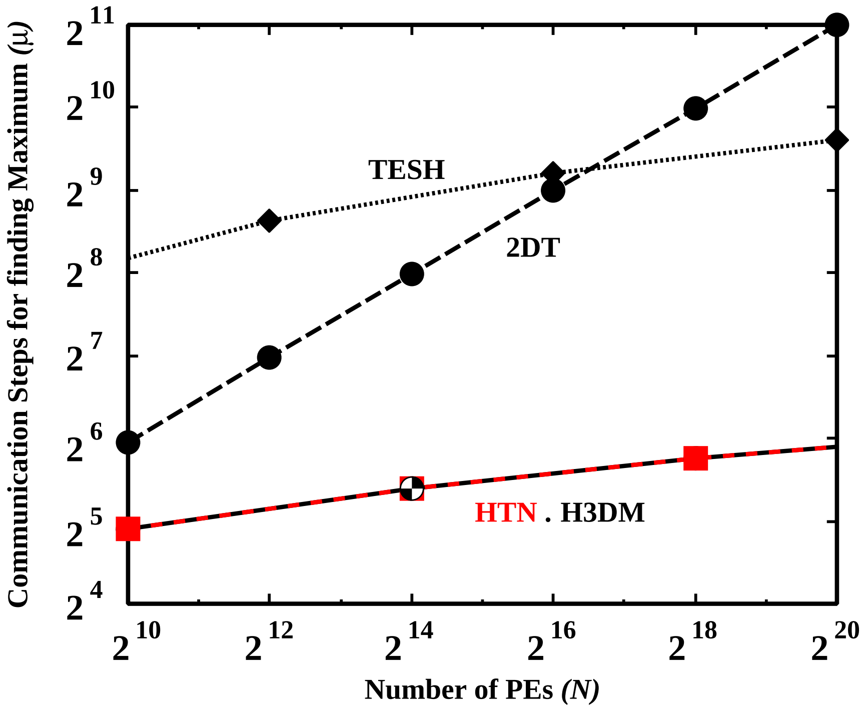


Figure 5.12: The total number of communication steps for finding the maximum in different networks

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	$192(L-1)+18$	$192(L-1)+18$	$12(L-1)+18$

communication steps in the HTN is larger than that in the 2D-torus network for thousands of nodes and equal to that of H3D-mesh network. However, for tens of thousands or millions of nodes, when more hierarchy is used, HTN shows better performance than that of the 2D-torus network. Even for millions of nodes, HTN shows better performance than that of the 3D-torus network. Also, HTN shows better performance than that of the H3D-torus and TESH networks. Therefore, HTN achieves better mapping performance for bitonic merge than do the other conventional and hierarchical networks.

The total number of communication steps for FFT on various interconnection networks is shown in Figure 5.11. The figure shows the number of communication steps as a function of network size for different networks. Each curve stands for a particular network. As mentioned earlier, the number of communication steps required to perform the FFT on N pieces of data is the same as the time required to sort N pieces of data with the bitonic merge algorithm. Let us consider Figure 5.9 for performance comparison of FFT. From Figures 5.9 and 5.11, it is seen that the total number of communication steps for FFT on the HTN is smaller than that on the 2D-torus, 3D-torus, H3D-torus, and TESH networks, but equal to the H3D-mesh network. Therefore, in the FFT mapping, HTN also achieves better mapping performance than do the other conventional and hierarchical interconnection networks.

The total number of communication steps for finding the maximum on various interconnection networks is shown in Figure 5.12. The figure shows the number of communication steps as a function of network size for the different networks. It is shown that the number of communication steps needed to find the maximum on the HTN is far smaller than on the TESH and 2D-torus networks, but equal to the H3D-mesh network. Therefore, in finding the maximum, HTN also achieves better mapping performance than do the other conventional and hierarchical interconnection networks.

Out comparison of the total number of communication steps for various advanced applications mapping such as bitonic merge, FFT, and finding the maximum, on various hierarchical interconnection networks such as HTN, H3D-mesh, H3D-torus, TESH, and conventional k-ary n-cube networks, has shown that the HTN outperforms those networks.

5.4 Conclusions

In this chapter, for the HTN, reconfiguration of faulty nodes by redundant nodes is presented. A hierarchical redundancy approach is explored, in which redundancy is provided at each level of the network. Expressions for yield are presented considering the redundant circuit. We have evaluated the yield of the HTN. The results indicate that, with a 25% redundancy at each level, the system yield at the basic module and second level are satisfactory. In short, we conclude that we have done the reconfiguration by redundancy and estimated the yield of the HTN successfully. We also conclude that a fault tolerant network is very essential for the reliability of massively parallel computer systems. Because, a single node failure in a massively parallel computer system may make the computer out of order or the computer will produce erroneous computational result.

Mappings of some commonly used advanced applications, such as bitonic merge, FFT, and finding the maximum, have also been presented in this chapter. The processing time of an application mapping in an interconnection network depends on the number of communication steps. The versatility of the HTN in various application mappings is investigated by evaluating the total number of communication steps. It is shown that the number of communication steps for various advanced applications mapping on the HTN is lower than for conventional and other hierarchical interconnection networks. A comparison of the total number of communication steps reveals that the HTN outperforms the H3D-mesh, H3D-torus, TESH, and k-ary n-cube networks.

Chapter 6

"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 (1903–1957)

Pruned Hierarchical Torus Network

6.1 Introduction

Pruning is the process of removing links from a basis network with the goal of simplifying its implementation (scalability, modularity, VLSI layout, etc.) and overcoming the bandwidth limitations at various levels of the packaging hierarchy, while maintaining the smaller diameter and average inter-node distance associated with higher degrees of connectivity. A number of useful interconnection networks were derived by pruning suitably chosen networks or were subsequently shown to be pruned versions of other networks. Examples include pruned 3-D torus [145], incomplete k-ary n-cube [147], periodically regular chordal rings [151], and binary Gaussian cubes [152].

One advantage of a pruned network is that it may be capable of emulating the original basis network efficiently, given their structural similarities. However, if pruning is not done carefully, the routing algorithm may become so complicated and/or message traffic so unbalanced as to hurt performance. Therefore, symmetric pruned network is preferable over the asymmetric one.

Whereas pruning leads to reduced node degree, Cartesian or cross-product networks work in the opposite direction [153] . The k-ary n-cube [64] is simply the cross-product of n rings of size k: Pruning a crossproduct network may be viewed as an attempt to offset the complexity introduced by using Cartesian products, while maintaining some of the benefits gained. Pruned versions of k-ary n-cubes have been shown to be quite effective [154, 155].

The links of a richly connected network can be removed in a periodic fashion for reduced complexity and hence increased performance. Such incomplete networks derived by pruning the original networks have been shown to be quite effective [150, 155]. The peak number of vertical links between silicon planes for 3D-WSI, maximum length of the longest links, and the number of longest wires of our proposed Hierarchical Torus Network (HTN) are lower than that of the torus network. Numerous basic modules of 3D-torus network are connected by 2D-torus network. The wrap around links of each individual basic module results a large number of physical links than that of 2D-torus network. To reduce the wiring complexity of the HTN, we will apply the pruning technique on it and the resulting network is called Pruned HTN.

The remainder of this chapter is organized as follows: Section 6.2 describes the pruning technique and its application on HTN. The main issue of wiring complexity and the 3D-WSI implementation are present in Section 6.3. Finally, Section 7.4 contains a summary of this chapter.

6.2 Pruned Network

6.2.1 Pruned Torus Network

A 3-dimensional, radix-k torus, also called k-ary 3-cube, consists of $N = k^3$ nodes arranged in an 3-dimensional cube with k nodes along each dimension. Each node is assigned an 3-digit radix-k address as (x, y, z), where $0 \leq x, y, z \leq k-1$. Each node is connected to 6 neighbors, such as, $[(x \pm 1) \bmod k, y, z]$, $[x, (y \pm 1) \bmod k, z]$, and $[x, y, (z \pm 1) \bmod k]$. Figure 6.1 shows a 3D-torus network of size $(4 \times 4 \times 4)$. Henceforth, it will be understood that all node-index expressions are calculated modulo k. We assume that k is an even number to ensure that the pruned torus is regular and of degree 4.

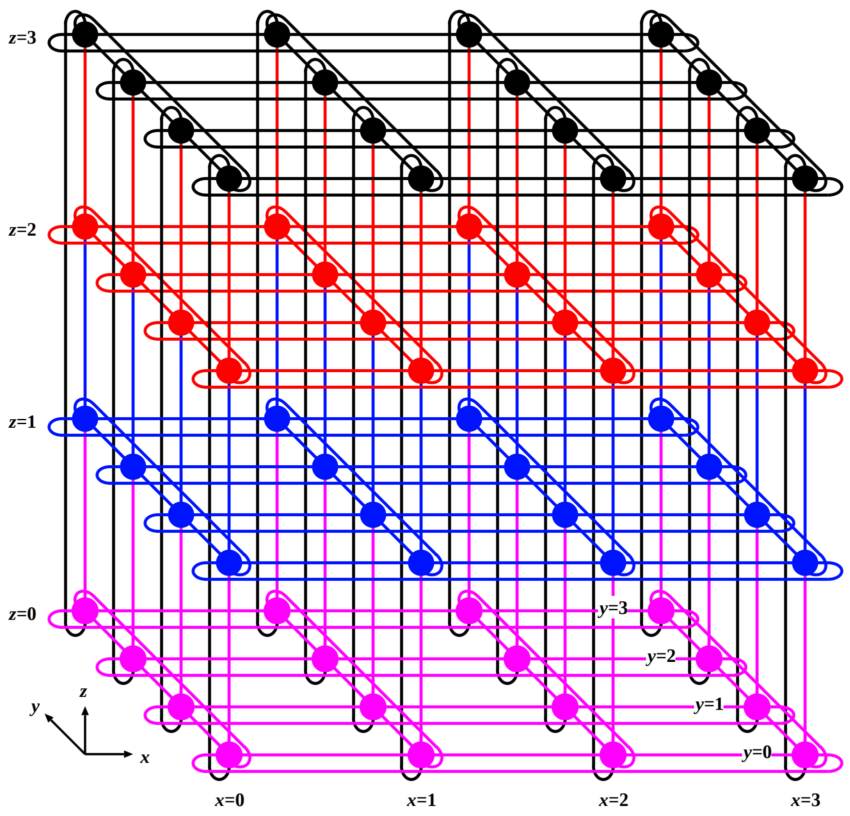


Figure 6.1: A $(4 \times 4 \times 4)$ torus network

The links of 3D-torus network is removed in a periodic fashion to form pruned torus network. Pruning technique can be applied in one direction and three direction. Pruning along the z-direction, resulting network is denoted as (T_1), node (x, y, z) is connected to its neighbor as follows. Each node (x, y, z) is connected to its two neighbors $(x, y, z \pm 1)$ along dimension z and other four neighbors are connected according to Eq. 6.1. Figure 6.2 shows an example of pruned $4 \times 4 \times 4$ torus network T_1, where the nodes are shaded if their positions along the pruning direction are odd-numbered.

$$\begin{cases} (x \pm 1, y, z) & \text{if } z = \text{even} \\ (x, y \pm 1, z) & \text{if } z = \text{odd} \end{cases} \tag{6.1}$$

Pruning along $(x+y+z)$-direction, resulting network is denoted as (T_2), node (x, y, z) is connected to its neighbor as follows. Each node (x, y, z) is connected to its two neighbors $(x, y, z \pm 1)$ along the dimension z and other four neighbors are connected according to Eq. 6.2. Figure 6.3 shows an example of pruned $4 \times 4 \times 4$ torus network T_2, where the nodes are shaded if their positions along the pruning direction are odd-numbered.

$$\begin{cases} (x + 1, y, z) \text{ and } (x, y + 1, z) & \text{if } x + y + z = \text{even} \\ (x - 1, y, z) \text{ and } (x, y - 1, z) & \text{if } x + y + z = \text{odd} \end{cases} \tag{6.2}$$

The conditions for the X and Y connections in T_1 and T_2 define the pruning directions as z and $x+y+z$, respectively. It is easy to verify that both pruned networks are regular and of degree four. Furthermore, both are Hamiltonian, meaning that they contain a ring, encompassing all the nodes, as a subgraph.

Here it is noted that there does not exist a pruning direction $f(x, y, z)$ that combines two out of the three directions, since such selections leave the resulting networks unconnected. Figure 6.4 shows an example that pruning along $x + y$ makes the shaded and non-shaded sub-networks disjoint. An exhaustive check through all possibilities confirms that permuting X, Y, and Z dimensions leads to networks that are isomorphic to either T_1 or T_2. The edge symmetry makes all links look alike. This property distinguishes T_1 from T_2. T_1 is edge symmetric, while T_2 is not edge symmetric. In [145], it is shown that the architecture T_1 is better than T_2, in terms of both regularity and performance.

Like 3D-torus network, we can apply the pruning technique on 2D-torus network. Pruned 2D-torus network is also known as Honeycomb Rectangular Torus (HReT) [146]. It is not possible to prune the 2D-torus along one direction. Because, if one direction keeps intact, then only one direction is remaining. Obviously, It is not possible to alternate the pruning direction. This is why, we apply the pruning along both direction. Pruning along $(x+y)$-direction, resulting network is called pruned 2D-torus, node (x, y) is connected to its neighbor as follows. Each node(x, y) is connected to its two neighbors $(x \pm 1, y)$ along x-direction and the other two neighbors are connected according to Eq. 6.3. Figure 6.5 shows an example of pruned 4×4 torus network.

$$\begin{cases} (x, y + 1) & \text{if } x + y = \text{even} \\ (x, y - 1) & \text{if } x + y = \text{odd} \end{cases} \tag{6.3}$$

6.2.2 Pruned HTN

We have proposed a new hierarchical interconnection networks called HTN for large-scale 3D multicomputers. The HTN consists of multiple basic modules which are 3D-torus of

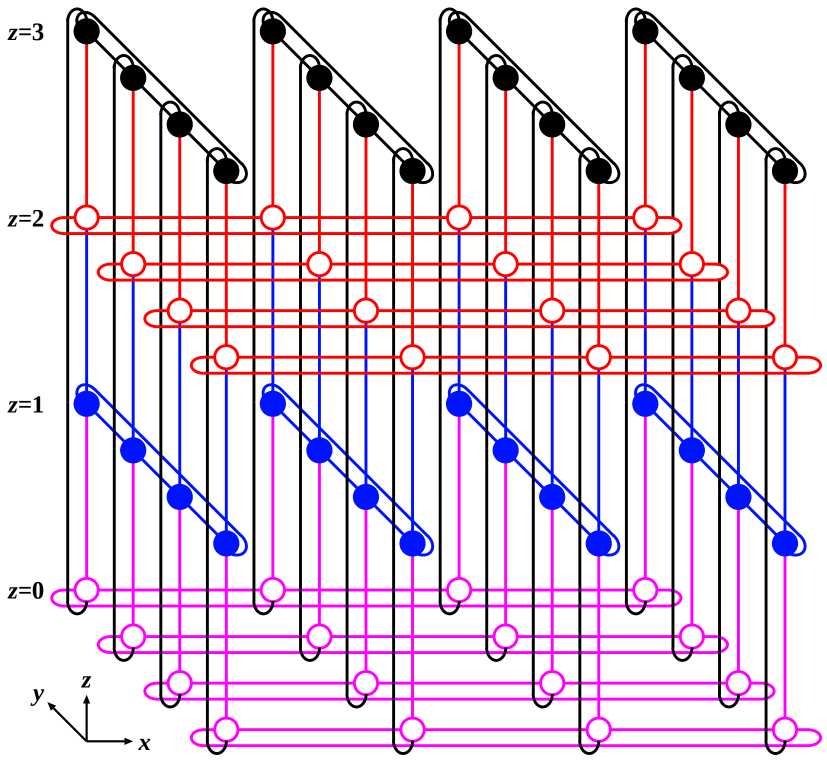


Figure 6.2: A $(4 \times 4 \times 4)$ pruned torus obtained by pruning along the z direction.

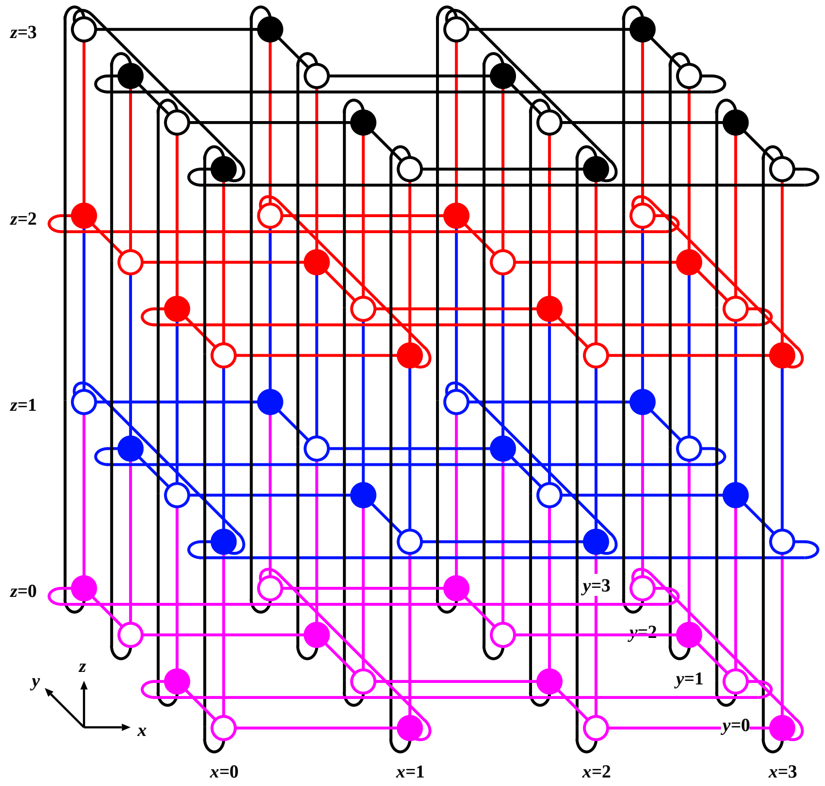


Figure 6.3: A $(4 \times 4 \times 4)$ pruned torus obtained by pruning along the $x + y + z$ direction.

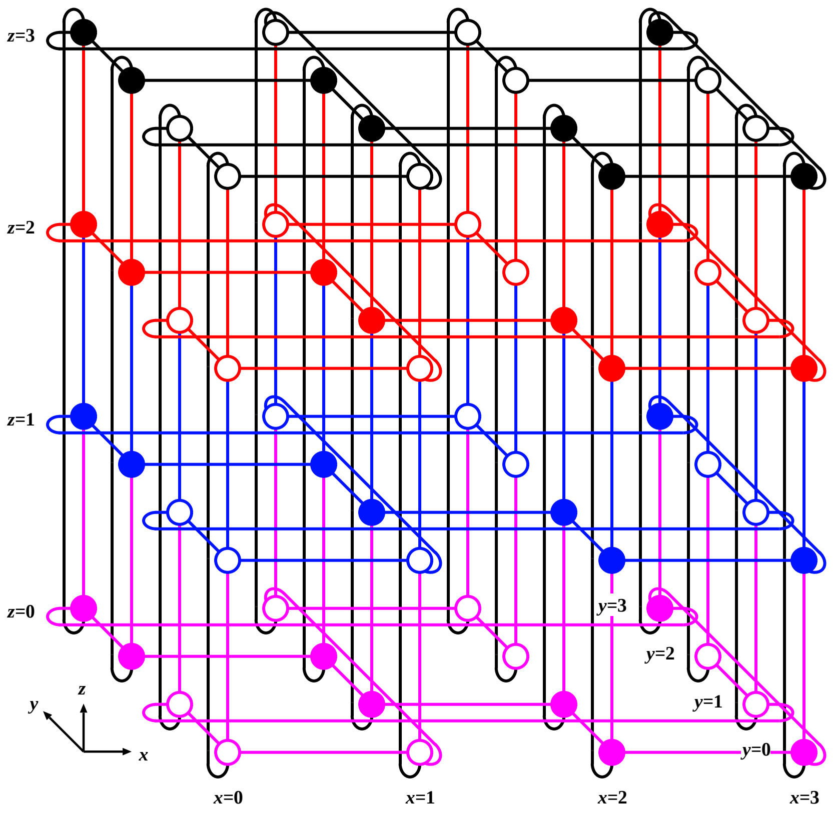


Figure 6.4: A $(4 \times 4 \times 4)$ pruned torus obtained by pruning along the $x + y$ direction.

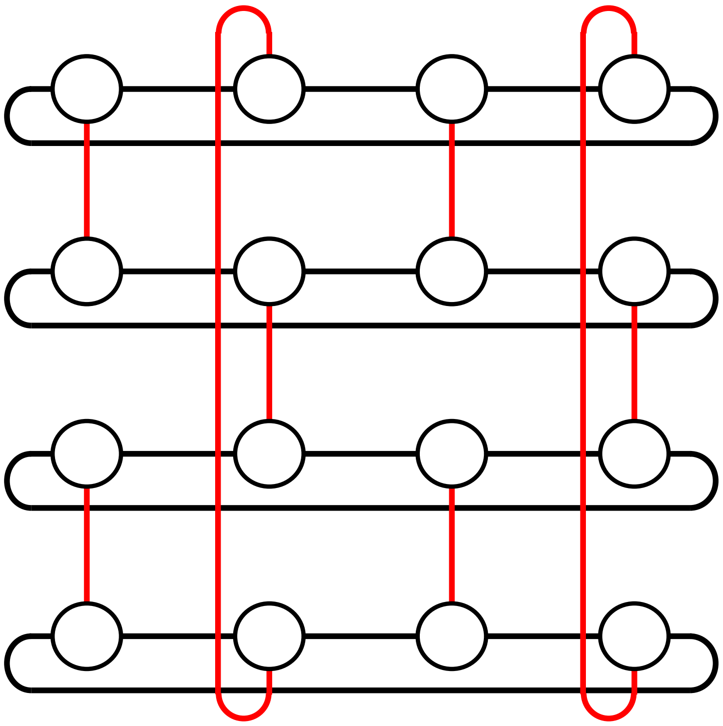


Figure 6.5: A (4×4) pruned torus obtained by pruning along the $x + y$ direction.

size $(m \times m \times m)$. The basic modules are hierarchically interconnected by a 2D-torus of size $(n \times n)$ to build higher level networks. The wrap around links of each individual basic module results a large number of physical links for higher level HTN. We can reduce the number of physical links using pruning technique. Pruning technique is applied both in the basic module and in higher level networks of the HTN. We just replace the torus network in the HTN by pruned torus network. Using pruned 3D tori T_1 and T_2 and pruned 2D-torus, we can build three pruned variants of the HTN called HTN$_1$, HTN$_2$, and HTN$_3$.

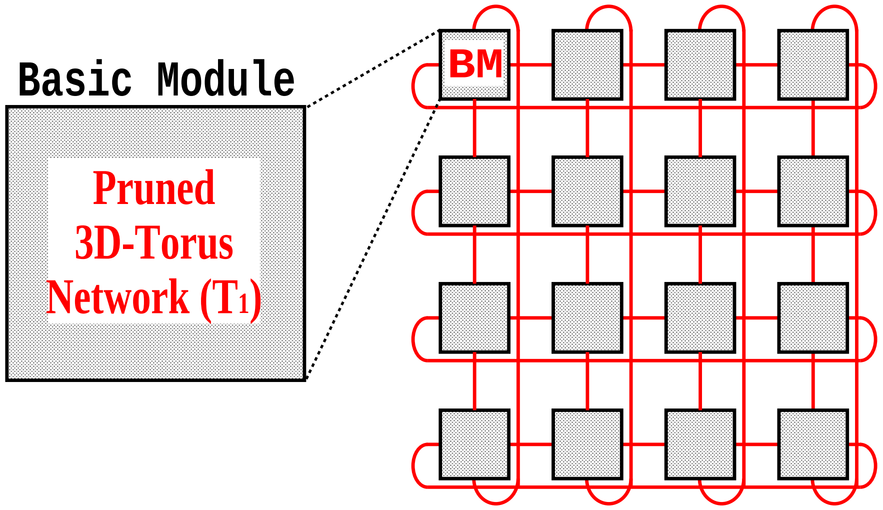


Figure 6.6: HTN$_1$ $(m = 4, n = 4)$

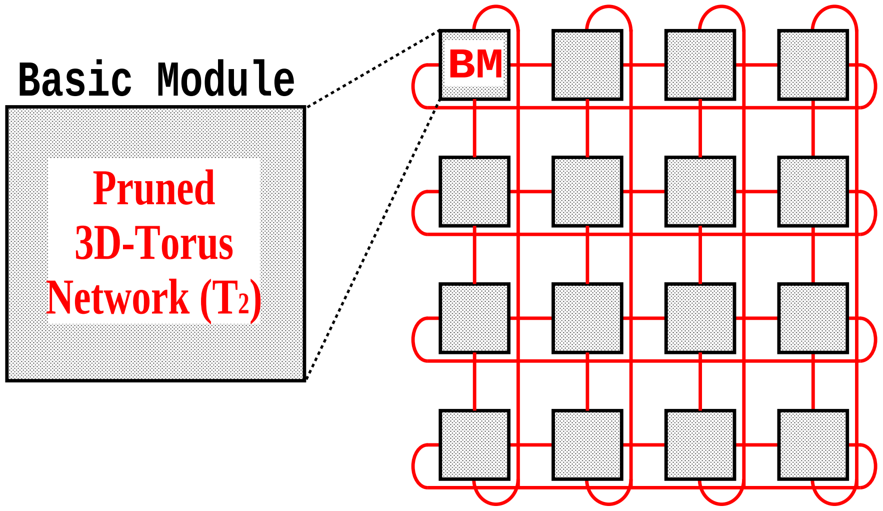


Figure 6.7: HTN$_2$ $(m = 4, n = 4)$

The HTN$_1$ consists of a basic module which is a pruned 3D-torus T_1 $(m \times m \times m)$. The basic modules are hierarchically interconnected by 2D-torus $(n \times n)$. Figure 6.6 shows an example of the HTN$_1$, where $m = 4$ and $n = 4$. The HTN$_2$ consists of a basic module which is a pruned 3D-torus T_2 $(m \times m \times m)$. The basic modules are hierarchically interconnected by 2D-torus ($n \times n$). Figure 6.7 shows an example of the HTN$_2$, where $m = 4$ and $n = 4$. The HTN$_3$ consists of a basic module which is a pruned 3D-torus T_2 ($m \times m \times m$). The basic modules are hierarchically interconnected by pruned 2D-torus ($n \times n$). Figure 6.8 shows an example of the HTN$_3$, where $m = 4$ and $n = 4$. Figure 6.9 illustrates a Level-2 HTN$_3$. To avoid clutter, we represents the basic module (pruned 3D-torus T_2) as a cube. The higher level network is a pruned 2D-torus of size (4×4).

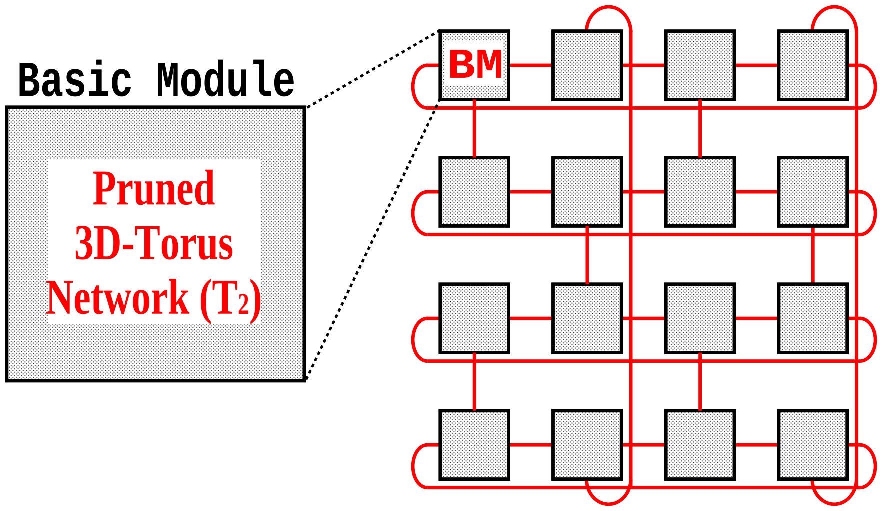


Figure 6.8: HTN$_3$ ($m = 4$, $n = 4$)

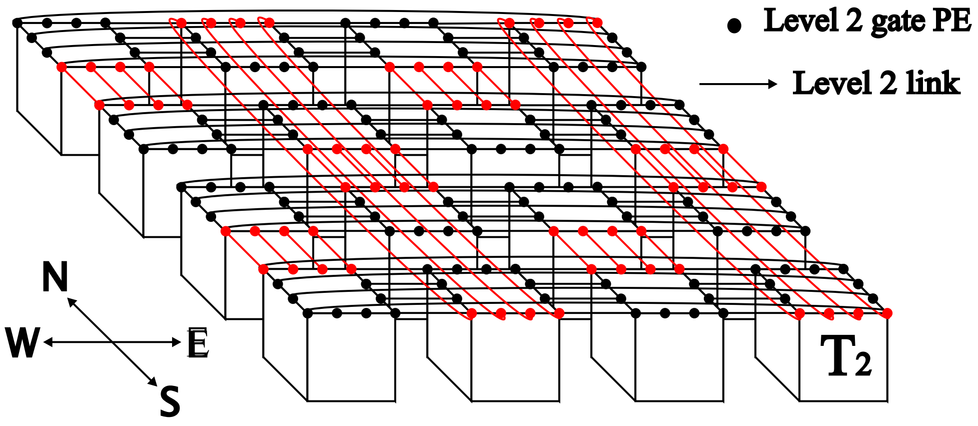


Figure 6.9: An illustration of Level-2 HTN$_3$

For the pruned HTN, the node degree is independent of network size. Maximum number of links for a single node is 6. The degree of the pruned HTN is 6, while the degree of non-pruned HTN is 8. Thus, pruning technique reduces the degree, which in turn reduces the I/O interface cost. Because I/O interface cost is related to the degree. Moreover, 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.

The wiring complexity of an interconnection network refers to the number of links required to be connected to a node as a network. The wiring complexity depends on the node degree and it has a direct correlation to hardware cost and complexity. Pruning technique reduces the number of physical links, which in turn reduces the hardware cost. We have evaluated the number of links for a Level-2 pruned HTN along with its non-pruned counterpart and tabulated in Table 6.1. It is seen that HTN_1 and HTN_2 need equal number of links because in these cases pruning technique is applied only in the basic module; they are less than that of HTN_0, i.e., non-pruned HTN. The number of links required for HTN_3 is less than that of HTN_1 and HTN_2 because here pruning technique is applied both in the basic module and Level-2 interconnection.

Table 6.1: Comparison of wiring complexity of various Level-2 HTN

Network	# of links
HTN_0	3200
HTN_1	2176
HTN_2	2176
HTN_3	2144

6.3 3D-WSI Implementation of the Pruned HTN

6.3.1 Peak Number of Vertical Links

The processing elements are placed on a square array on each silicon plane. A number of such silicon planes are connected together by vertical links. In this section, we evaluate the peak number of vertical links for various pruned HTN and compare them with that of original HTN.

$C_{max}(network, N, m)$ expresses the peak number of vertical links in 3D stacked implementation. If h is the number of silicon planes, M is the number of PEs on each plane, and N is the total number of PEs, then the relation between the h, M, N is $N = h \times M$. First, we evaluate the peak number of vertical links of Level-2 pruned HTN by physical realization. For our evaluation, we have considered the number of planes $h = n \times n$ and the number of PEs in each plane $M = m \times m \times m$. If $m = 4$ and $n = 4$, the peak number of vertical links of Level-2 pruned HTN is 28. Then, the peak number of vertical links of Level-L pruned HTN is calculated according to Eq. 6.4.

$$C_{max} = \left\{C_{max}\left(PrHTN - L2, (n \times n), (m \times m \times m)\right)(n \times n)^{L-2}\right\} \tag{6.4}$$

Figure 6.10 shows the peak number of vertical links of various pruned HTN. The ordinate indicates the peak number of vertical links and the abscissa indicates the number of PEs. It is seen that the peak number of vertical links C_{peak} of the HTN_3 is less than that of HTN_0, HTN_1, and HTN_2. Also C_{peak} of the HTN_1 and HTN_2 are exactly same as that of the HTN_0. Here it is noted that HTN_0 is the non-pruned HTN as described in Section 3.2. For more than 2^{13} PEs, C_{peak} of the HTN_3 is lower than that of HTN_0. Therefore, bigger sized suitable network can be realized by pruning technique.

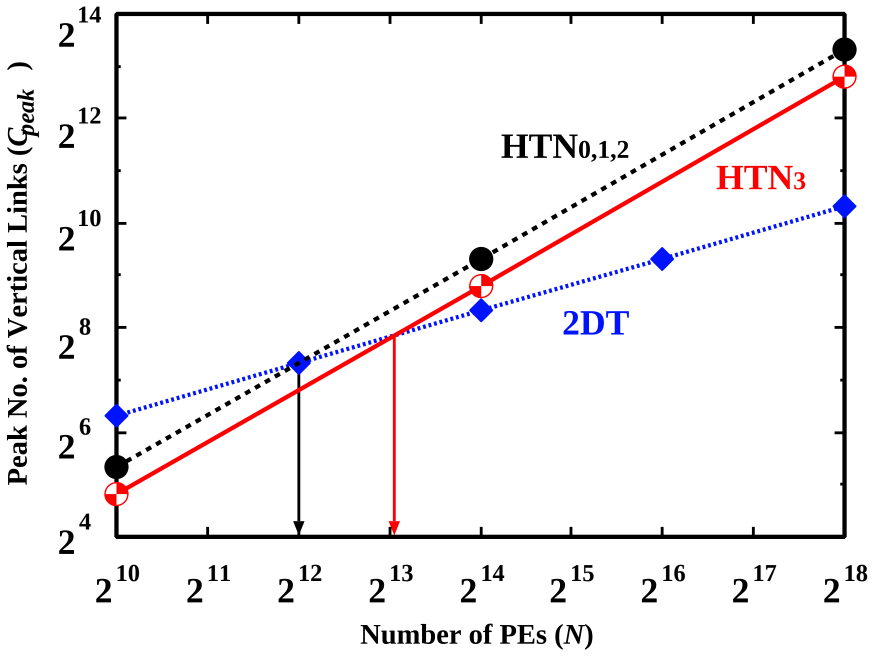


Figure 6.10: A comparison of peak number of vertical links of various HTN.

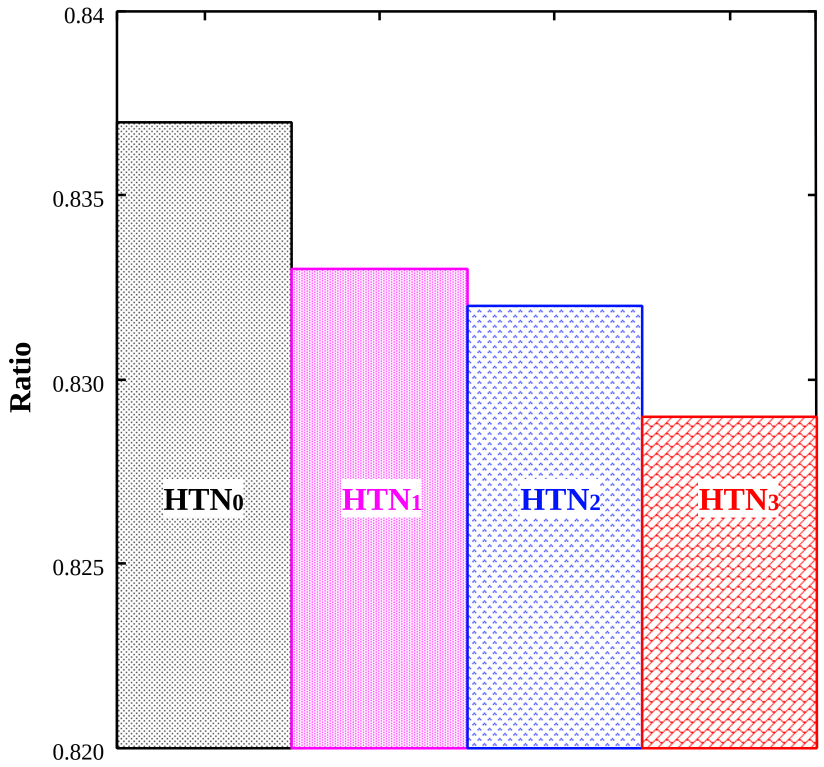


Figure 6.11: Normalized layout area (1024 PEs, 16 Wafers, and 64 PEs/Wafer)

6.3.2 Layout Area

To show the suitability of an interconnection network for 3D wafer stacked-implementation, layout area has to be evaluated. In this section, we evaluate the layout area for various pruned HTN. We have considered the parameter for evaluation and evaluated the layout area of pruned HTN according to the formulas of Section 3.4.3. We have evaluated the layout area of a Level-2 HTN and its pruned counterpart. If $m = 4$ and $n = 4$, then $M = 64$, $h = 16$ and $N = 1024$. The layout area is normalized by 2D-torus network and the normalized values are plotted in Figure 6.11. It is shown that pruned HTN can be implemented on smaller area. The layout area of pruned HTN in 3D stacked implementation is more acquiescent than that of non-pruned HTN for 3D implementation.

It is seen that the layout area of HTN_1 is less than that of HTN_0. Similarly, the layout area of HTN_2 is less than that of HTN_1 and the layout area of HTN_3 is less than that of HTN_2. In HTN_1 andHTN_2, pruning technique is applied on the basic module only. In HTN_3, pruning technique is applied both in the basic module and the Level-2 network. Thus, we can conclude that higher the pruning technique applied, the lower the layout area for 3D-WSI.

6.4 Conclusion

We have presented pruning technique on a new hierarchical interconnection network, called Hierarchical Torus Network (HTN). The architecture of the pruned HTN and 3D-Integration issue were discussed in detail. Pruned HTN reduces both the I/O interface cost and hardware cost and complexity. The pruned HTN is well suited for 3D stacked-implementations. It is shown that the peak number of vertical links in 3D stacked-implementation is low for pruned HTN as compared to its non-pruned counterpart. The layout area of pruned HTN in 3D-WSI is less than that of non-pruned HTN. Therefore, HTN permits efficient VLSI/ULSI/WSI realization.

Chapter 7

"No one believes an hypothesis except its originator but everyone believes an experiment except the experimenter."
– W.I.B. Beveridge

Modification of other Hierarchical Networks based on Torus-Torus Interconnectiuon

7.1 Introduction

It has already been shown that a torus network has better dynamic communication performance than a mesh network. Due to presence of wrap-around links between end to end nodes in torus network it provides an extra path for message to pass through. Thus, more messages can be delivered in the network, which in turn increases the network throughput. However, the length of the wrap-around links is a limiting factor for massively parallel computers with thousands or millions of nodes. The long wire decreases the clock-speed of the network. Higher dimensional torus reduces the wire length. On the contrary, higher dimensional torus results huge vertical links between 3D-WSI. Therefore, hierarchical interconnection network consists of torus-torus network is a plausible alternative way for massively parallel computers. There are some hierarchical network which consist of torus and mesh networks. By replacing the mesh by torus network, we can improve the dynamic communication performance of those networks. In this chapter, we will apply our main idea "torus-torus" hierarchical network to other networks.

This chapter is organized as follows: Section 7.2 describes the basic structure of the Modified Hierarchical 3D-Torus (MH3DT) network, including addressing and routing, static network performance, and dynamic communication performance. Section 7.3 describes the basic structure of the Modified TESH network, including addressing and routing, static network performance, and dynamic communication performance. Finally, some concluding remarks are given in Section 7.4.

7.2 Modified Hierarchical 3D-Torus Network

An H3D-torus network [61, 62] has been put forward as a new interconnection network for large-scale 3D multicomputers. The H3D-torus network consists of multiple basic modules (BMs) which are 3D-mesh of size $(m \times m \times m)$. The BMs are hierarchically interconnected by a 3D-torus of size $(n \times n \times n)$ to build higher level networks. The restricted use of physical links between basic modules in the higher level networks reduces the dynamic communication performance of this network. Even when the inter-BM links are increased, the network throughput of the H3D-torus network is still lower than that of the conventional mesh network. It has already been shown that a torus network has better dynamic communication performance than a mesh network [64]. This is the key motivation that led us to replace the 3D-mesh network by a 3D-torus network.

The modified hierarchical 3D-torus (MH3DT) network consists of BMs which are themselves 3D-tori $(m \times m \times m)$, hierarchically interconnected in a 3D-torus $(n \times n \times n)$ networks . In the MH3DT network, both the BMs and the interconnection of higher levels have toroidal interconnections.

7.2.1 Interconnection of the MH3DT Network

The MH3DT network consists of BMs that are hierarchically interconnected to form higher level networks. The BM of the MH3DT network is a 3D-torus network of size $(m \times m \times m)$, where m is a positive integer. The BM of $(4 \times 4 \times 4)$ torus, which is shown in Figure 7.1, has some free ports at the corners of the xy-plane. These free ports are used for higher level interconnection. All ports of the interior Processing Elements (PEs) are used for intra-BM connections. All free ports of the exterior PEs are used for inter-BM connections to form higher level networks. In this dissertation, unless specified otherwise, BM refers to a Level-1 network.

Processing Elements (PEs) in the BM are addressed by three base-m digits, the first representing the x-direction, the second representing the y-direction, and the last representing the z-direction. The address of a PE in the BM is expressed by

$$A^{BM} = (a_z)\,(a_y)\,(a_x)\,, \quad (0 \leq a_z, a_y, a_x \leq m-1) \tag{7.1}$$

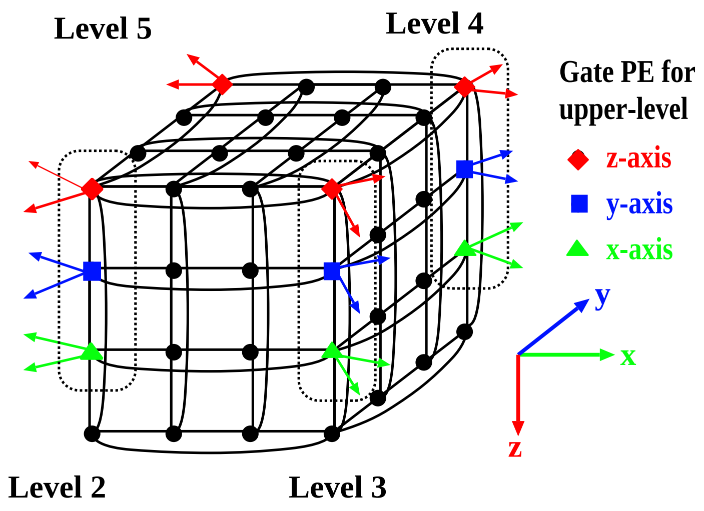


Figure 7.1: Basic module of the MH3DT network

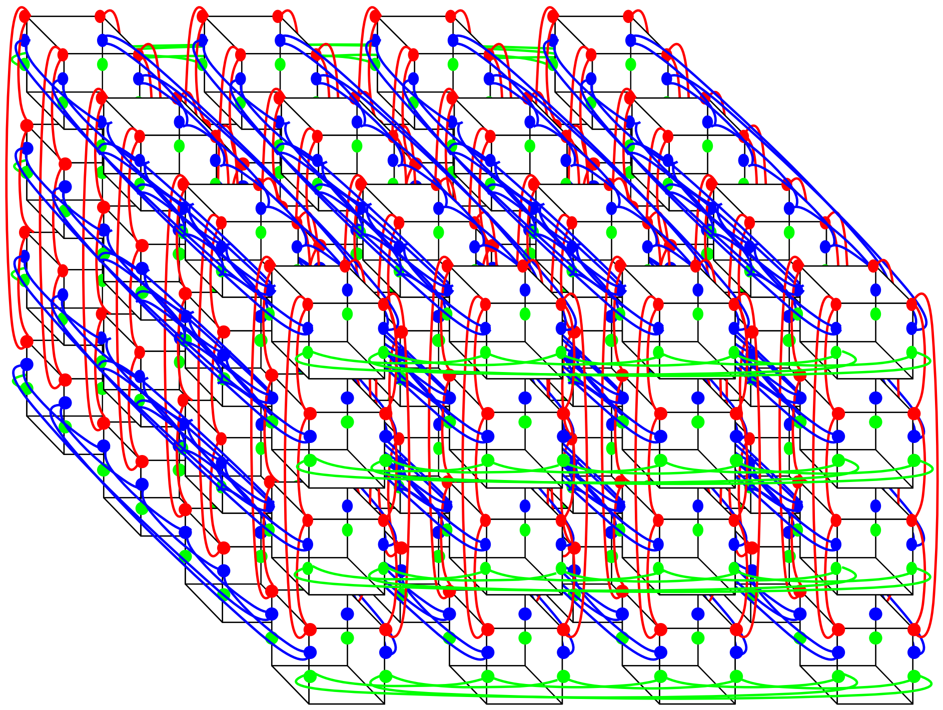


Figure 7.2: Interconnection of a Level-2 MH3DT network

All ports of the interior PEs are used for intra-BM connections. Three PEs ($0 \leq a_z \leq 2$) have two free ports, as shown in Figure 7.1, which are used for inter-BM connections to form higher level networks. Let $a_z = 0$ be the z-direction link, $a_z = 1$ be the y-direction link, and $a_z = 2$ be the x-direction link. We define a gate node as a PE that has free links to interconnect with PEs at the higher level. Thus each *gate* node has two links and is hierarchically interconnected with the PEs at the higher level by a 3D-torus network.

Successively higher level networks are built by recursively interconnecting lower level subnetworks in a 3D-torus of size $(n \times n \times n)$, where n is also a positive integer. As illustrated in Figure 7.2, a Level-2 MH3DT network, for example, can be formed by interconnecting 64 BMs as a $(4 \times 4 \times 4)$ 3D-torus network. Each BM is connected to its logically adjacent BMs. 2^q gate nodes are used higher level interconnection, where q is the inter-level connectivity. As each xy-plane of the BM has 4 gate nodes, $q \in \{0, 1, 2\}$. $q = 0$ leads to minimal inter-level connectivity, while $q = 2$ leads to maximum inter-level connectivity. By using the parameters m, n, L, and q, we can define the MH3DT network as MH3DT(m, n, L, q).

Let N be the total number of nodes in an MH3DT network. Then an MH3DT network with level L has $N = \left[m^3 \times n^{3(L-1)}\right]$. With $q = 0$, for example, Level-5 is the highest possible level to which a $(4 \times 4 \times 4)$ BM can be interconnected. The total number of nodes in a network having $(4 \times 4 \times 4)$ BMs and a $(4 \times 4 \times 4)$ Level-5 network is $N = 2^{30}$, i.e, more than one billion. A PE in the Level-L is addressed by three base-n numbers as follows:

$$A^L = \left(a_z^L\right)\left(a_y^L\right)\left(a_x^L\right), \quad \left(0 \leq a_z^L, a_y^L, a_x^L \leq n-1\right) \tag{7.2}$$

The address of a PE at Level-L is represented by:

$$A^L = (a_z^L)(a_y^L)(a_x^L) \qquad L \text{ is the level number.} \tag{7.3}$$

More generally, in a Level-L MH3DT network, the node address is represented by:

$$\begin{aligned} A &= A^L A^{L-1} A^{L-2} \dots \dots A^2 A^1 \\ &= a_\alpha \; a_{\alpha-1} \; a_{\alpha-2} \; a_{\alpha-3} \dots \dots a_3 \; a_2 \; a_1 \; a_0 \\ &= a_{3L-1} \; a_{3L-2} \; a_{3L-3} \; a_{3L-4} \dots \dots a_3 \; a_2 \; a_1 \; a_0 \\ &= (a_{3L-1} \; a_{3L-2} \; a_{3L-3}) \dots \dots (a_2 \; a_1 \; a_0) \end{aligned} \tag{7.4}$$

Here, the total number of digits is $\alpha = 3L$, where L is the level number. Groups of digits run from group number 1 for Level-1 (i.e. the BM), to group number L for the L-th level. In particular, i-th group $(a_{3i-1} \; a_{3i-2} \; a_{3i-3})$ indicates the location of a Level-$(i-1)$ subnetwork within the i-th group to which the node belongs; $2 \leq i \leq L$. In a two-level network, for example, the address becomes $A = (a_5 \; a_4 \; a_3) \; (a_2 \; a_1 \; a_0)$. The last group of digits $(a_5 \; a_4 \; a_3)$ identifies the BM to which the node belongs, and the first group of digits $(a_2 \; a_1 \; a_0)$ identifies the node within that basic module.

7.2.2 Routing Algorithm

Routing of messages in the MH3DT network is performed from top to bottom. That is, it is first done at the highest level network; then, after the packet reaches its highest level sub-destination, routing continues 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's destination is 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. We have considered a simple deterministic, dimension-order routing algorithm. Routing of messages in the network is performed initially in the z-direction, next in the y-direction, and finally in the x-direction.

Routing in the MH3DT network is strictly defined by the source node address and the destination node address. Let a source node address be $s_\alpha, s_{\alpha-1}, s_{\alpha-2}, ..., s_1, s_0$, a destination node address be $d_\alpha, d_{\alpha-1}, d_{\alpha-2}, ..., d_1, d_0$, and a routing tag be $t_\alpha, t_{\alpha-1}, t_{\alpha-2}, ..., t_1, t_0$, where $t_i = d_i - s_i$. The source node address of the MH3DT network is expressed as $s = (s_{3L-1}, s_{3L-2}, s_{3L-3}), ..., \; ..., (s_2, s_1, s_0)$. Similarly, the destination node address is expressed as $d = (d_{3L-1}, d_{3L-2}, d_{3L-3}), ..., \; ..., (d_2, d_1, d_0)$. Figure 7.3 shows the routing algorithm for the MH3DT network.

As an example, a routing between $PE_{(123)(211)}$ and $PE_{(333)(111)}$ in the MH3DT network is given. First the packet is moved to the gate node in the z-axis at Level-2, $PE_{(123)(000)}$. Then the packet at $PE_{(123)(000)}$ is moved to the node with the same address in the z-axis, $PE_{(323)(000)}$. Next, routing in the y-axis is applied in the same manner. The packet at $PE_{(323)(000)}$ is moved to the gate node $PE_{(323)(100)}$ in the y-axis. The packet at $PE_{(323)(100)}$ is moved to $PE_{(333)(100)}$ with the same address in the y-axis. Finally, the routing is applied within the BM and terminated.

```
Routing MH3DT(s,d);
source node address:$s_\alpha, s_{\alpha-1}, s_{\alpha-2}, ..., s_1, s_0$
destination node address: $d_\alpha, d_{\alpha-1}, d_{\alpha-2}, ..., d_1, d_0$
tag: $t_\alpha, t_{\alpha-1}, t_{\alpha-2}, ..., t_1, t_0$
  for $i = \alpha : 3$
    if ($t_i > 0$ and $t_i \le \frac{n}{2}$) or ($t_i < 0$ and $t_i = -(n-1)$), movedir = positive; endif;
    if ($t_i > 0$ and $t_i = (n-1)$) or ($t_i < 0$ and $t_i \ge -\frac{n}{2}$), movedir = negative; endif;
    if (movedir = positive and $t_i > 0$), distance = $t_i$; endif;
    if (movedir = positive and $t_i < 0$), distance = $n + t_i$; endif;
    if (movedir = negative and $t_i < 0$), distance = $t_i$; endif;
    if (movedir = negative and $t_i > 0$), distance = $-n + t_i$; endif;
      $j = i$ mod 3
      while($t_i \neq 0$ or distance $\neq 0$) do
        if ($j = 2$), gate-node = $z$-axis gate-node of Level-$\lceil \frac{i}{3} \rceil$; endif
        if ($j = 1$), gate-node = $y$-axis gate-node of Level-$\lceil \frac{i}{3} \rceil$; endif
        if ($j = 0$), gate-node = $x$-axis gate-node of Level-$\frac{i}{3} + 1$; endif
        if (routedir = positive), move packet to next BM; endif;
        if (routedir = negative), move packet to previous BM; endif;
        if ($t_i > 0$), $t_i = t_i - 1$; endif;
        if ($t_i < 0$), $t_i = t_i + 1$; endif;
     endwhile;
  endfor;
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 $t_i \le \frac{m}{2}$) or ($t_i < 0$ and $t_i = -(m-1)$), movedir = positive; endif;
    if ($t_i > 0$ and $t_i = (m-1)$) or ($t_i < 0$ and $t_i \ge -\frac{m}{2}$), movedir = negative; endif;
    if (movedir = positive and $t_i > 0$), distance = $t_i$; endif;
    if (movedir = positive and $t_i < 0$), distance = $m + t_i$; endif;
    if (movedir = negative and $t_i < 0$), distance = $t_i$; endif;
    if (movedir = negative and $t_i > 0$), distance = $-m + t_i$; 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 = negative), 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 = negative), 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 = negative), move packet to $-x$ node; distance$_0$ = distance$_0 + 1$; endif;
  endwhile;
end
```

Figure 7.3: Routing algorithm of the MH3DT network

7.2.3 Deadlock-Free Routing

The most expensive part of an interconnection network is the wire that forms the physical channels; for a particular topology, the physical channel cost is constant. The second most expensive elements are the buffers and switches. Since the networks we consider are wormhole-routed, the main factor in buffer expense is the number of virtual channels. Virtual channels [39] reduce the effect of blocking; they are used widely in parallel computer systems, to improve dynamic communication performance by relieving contention in the multicomputer network and to design deadlock-free routing algorithms. Since the hardware cost increases as the number of virtual channels increases, the unconstrained use of virtual channels is not cost-effective in parallel computers. Therefore, a deadlock-free routing algorithm for an arbitrary interconnection network with a minimum number of virtual channels is preferred. In this section, we discuss the minimum number of virtual channels for deadlock-free routing of the MH3DT network. We also present a proof that the MH3DT network is deadlock free.

To prove the proposed routing algorithm for the MH3DT network is deadlock free, we divide the routing path into three phases, as follows:

- *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 z-link.

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

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

  sub-phase $2.i.4$: Inter-BM transfer of Level $(L-i)$ through the y-link.

  sub-phase $2.i.5$: Intra-BM transfer to the outlet PE of Level $(L-i)$ through the x-link.

  sub-phase $2.i.6$: Inter-BM transfer of Level $(L-i)$ through the x-link.
- *Phase 3:* Intra-BM transfer path from the outlet of the inter-BM transfer path to the destination PE.

The proposed routing algorithm enforces some routing restrictions to avoid deadlocks [38]. Since dimension-order routing is used in the MH3DT network, messages in the network are routed first in the z-direction then in the y-direction, and finally in the x-direction. The interconnection of the BM and the higher levels of the MH3DT network is a toroidal connection. The number of virtual channels required to make the routing algorithm deadlock-free for the MH3DT network is determined using Lemma 4.1.

Theorem 7.1 *An MH3DT network with* 2 *virtual channels is deadlock free.*

Proof: Both the BM and the higher levels of the MH3DT network 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 MH3DT network 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 are used in sub-phases $2.i.1$, $2.i.3$, and $2.i.5$. Thus, sub-phases $2.i.1$, $2.i.3$, and $2.i.5$ utilize channels over intra-BM links, sharing the channels of either phase-1 or phase-3. The gate nodes, as mentioned earlier, are used for higher level interconnection. The free links in these gate nodes are used in sub-phases $2.i.2$, $2.i.4$, and $2.i.6$, and these links form a 3D-torus network for the higher level network. According to Lemma 4.1, the number of necessary virtual channels for this 3D-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.

7.2.4 Static Network Performance

In Section 3.3, we have discussed details about the static network performance of various networks. In this section, we discuss some of the properties and performance metrics that characterize the cost and performance of an interconnection network. The static network performance of various networks with 4096 nodes, along with that of a CCC network with 4608 nodes, is tabulated in Table 7.1. The static network performance of the 4608-node CCC network can not be compared with the other 4096-node networks. However, its performance is included in Table 7.1 to show its topological properties.

Node Degree

For the MH3DT network, the node degree is independent of network size. Since each node has eight channels, its degree is 8. 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.

Diameter

The *diameter* of a network is the maximum inter-node distance, i.e., the maximum number of links that must be traversed to send a message to any node along a shortest path. Networks with small diameters are preferable. Table 7.1 shows a comparison of the MH3DT network diameter with several other networks. Clearly, the MH3DT network has a much smaller diameter than the conventional mesh, torus, TESH [47, 49], and H3D-torus [61, 62] networks but larger than that of the hypercube network.

Cost

The product (*diameter* $\times$ *node degree*) is defined as the cost of a network in technology independent manner. Table 7.1 shows that the cost of the MH3DT network is smaller than that of the mesh, torus, and H3D-torus [61, 62] networks, equal to that of the hypercube network, and a bit higher than the TESH [47, 49] network.

Average Distance

The *average distance* is the mean distance between all distinct pairs of nodes in a network. We have evaluated the average distance for different conventional topologies by the corresponding formula and of different hierarchical networks by simulation. As we see in Table 7.1, the MH3DT network has a smaller average distance than the conventional mesh & torus network and hierarchical TESH and H3D-torus networks. However, the average distance of the MH3DT network is higher than that of the hypercube network.

Arc Connectivity

Arc connectivity is the minimum number of links that must removed in order to break the network into two disjoint parts. A network is maximally fault-tolerant if its connectivity is equal to the degree of the network. The arc connectivity of several networks is shown in Table 7.1. Clearly, the arc connectivity of the MH3DT network is higher than a conventional mesh, TESH, and H3D-torus networks and it is closer to the node degree. The arc connectivity of an interconnection networks is independent of its total number of nodes. The arc connectivity of the torus and CCC network are exactly equal to the node degree. Thus, torus and CCC are more fault tolerant than the MH3DT network.

Bisection Width

The *Bisection Width (BW)* of a network is defined as the minimum number of links that must removed to partition the network into two equal halves. The bisection width of the MH3DT network is calculated by:

$$BW_{(MH3DT)} \quad = \quad 2^{q+1} \times (m \times n) \tag{7.5}$$

BW 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 the interconnection of basic modules here. The basic module is simply a 3D-torus network so its bisection width is $2m^2$. Table 7.1 shows that the bisection width of the MH3DT network is higher than that of the conventional mesh and TESH networks and equal to that of the torus and H3D-torus networks.

Wiring Complexity

The *wiring complexity* of an interconnection network refers to the number of links needed to be connected to a node as the network is scaled up. The wiring complexity depends on the node degree, which is the number of channels incident to a node. The wiring complexity has a direct correlation to hardware cost and complexity. An n-dimensional torus network has $n \times N$ links, where N is the total number of nodes. The wiring complexity of a Level-L MH3DT network is represented as shown in Eq. 7.6.

$$\left[3m^3 \times n^{3(L-1)} + \sum_{i=2}^{L} 3(2^q)\left\{n^{3(L-1)}\right\}\right] \tag{7.6}$$

Table 7.1 compares the wiring complexity for a MH3DT network with several other networks. The total number of physical links in the MH3DT network is higher than that in the mesh, torus, TESH, and H3D-torus networks; therefore, the cost of physical links is higher for the MH3DT network. However, it is almost half the wiring complexity of the hypercube network.

Table 7.1: Comparison of static network performance of various network with 4096 node

	Node Degree	Diameter	Cost	Average Distance	Arc Connectivity	Bisection Width	Wiring Complexity
Hypercube	12	12	144	6	12	2048	24576
2D-Mesh	4	126	504	42.67	2	64	8064
2D-Torus	4	64	256	32	4	128	8192
CCC[1]	3	22	66	12.75	3	256	6912
H3D-torus (4,4,2,0)	6	25	150	12.74	3	32	9408
MH3DT (4,4,2,0)	8	20	160	10.36	6	32	12480
H3D-torus (4,4,2,2)	6	21	126	10.77	3	128	9792
MH3DT (4,4,2,2)	8	18	144	9.37	6	128	12864
TESH Network	4	32	128	17.80	2	8	8192

†CCC network with 4608 nodes

7.2.5 Dynamic Communication Performance

The dynamic communication performance of a multicomputer is characterized by message latency and network throughput. A network with low latency and high throughput is preferable for massively parallel computers. To evaluate dynamic communication performance, we have developed a wormhole routing simulator. In our simulation, we use a dimension-order routing algorithm. The dimension-order routing algorithm, which is exceedingly simple, provides the only route for the source-destination pair. Extensive simulations for several networks have been carried out under the uniform traffic pattern, in which each node sends messages to every other node with equal probability. Two virtual channels per physical channel are simulated, and the virtual channels are arbitrated by a round robin algorithm. For all of the simulation results, the packet size is 16 flits. Two flits are used as the header flit. Flits are transmitted at $20,000$ cycles; in each clock cycle, one flit is transferred from the input buffer to the output buffer, or from output to input if the corresponding buffer in the next node is empty. Therefore, transferring data between two nodes takes 2 clock cycles.

Dynamic Communication Performance Evaluation

We have evaluated the dynamic communication performance of several 4096 node networks under uniform traffic patterns. As mentioned earlier that it is not possible to construct a CCC network with 4096 nodes. Thus, we can not make a fair comparison between the dynamic communication performance of the 4608-node CCC network and the other, 4096-node networks.

Under the assumption of constant bisection width, low-dimensional networks with wide channels provide lower latency, less contention, and higher throughput than higher-dimensional networks with narrow channels [64]. However, if the bisection width is high the network throughput will be high, and vice-versa. The bisection width of hypercube networks is very high. Thus, their throughput is high, with the accompanying cost of large latency [64]. Moreover, it has already been shown that the number of vertical links between silicon wafers in the 3D-WSI implementation for the hypercube network is very large; thus it is not suitable for 3D-WSI implementation [47, 49, 59]. This is why we have considered conventional mesh networks rather than hypercubes for comparing the dynamic communication performance. We have compared the dynamic communication performance of various interconnection networks, including hierarchical interconnection networks, with low bisection width.

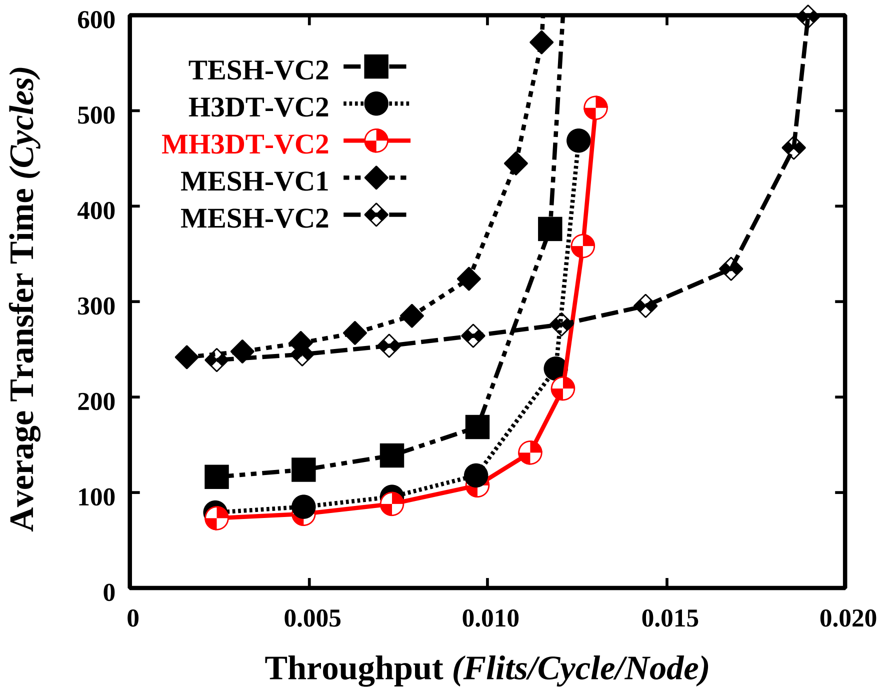
To evaluate the dynamic communication performance of the H3D-torus and MH3DT networks, we have used maximum inter-level connectivity, i.e., $q = 2$. For a fair comparison, we allocated 2 virtual channels to the router for performance evaluation. Figure 7.4(a) depicts the results of simulation under uniform traffic patterns for the various network models. This figure presents the average transfer time as a function of network throughput. Each curve stands for a particular network. As shown in Figure 7.4(a), the average transfer time of the MH3DT network is lower than that of the H3D-torus, TESH, and mesh networks; its maximum throughput is higher than that of the H3D-torus network, TESH network, and mesh network with 1 virtual channel, but lower than that of the mesh network with 2 virtual channels.

Effect of Buffer Size

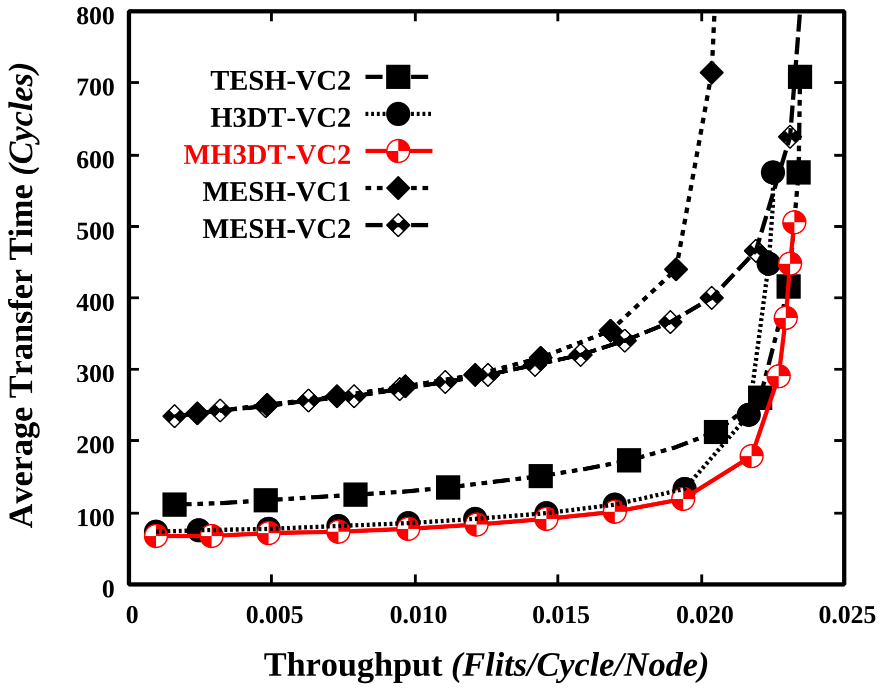
In a large buffer , more flits can be stored in the one channel buffer; therefore, frequency of packet blocking and deadlocks are decreased and the dynamic communication performance of the interconnection network is increased. We have evaluated the dynamic communication performance of various network models using a channel buffer size of 20 flits and the result is portrayed in Figure 7.4(b). As shown in Figure 7.4(b), the average transfer time of the MH3DT network is lower than that of the H3D-torus, TESH, and mesh networks; the maximum throughput of the MH3DT network is higher than that of the H3D-torus network and mesh network with 1 & 2 virtual channels and equal to that of the TESH network.

Effect of Message Length

In this section, we analyzed the effect of message length on dynamic communication performance. We have evaluated the dynamic communication performance of the MH3DT network for short, medium, and long messages to show the effect of message length. Figure



(a) Buffer size, 2 flits



(b) Buffer size, 20 flits

Figure 7.4: Dynamic communication performance of dimension-order routing with uniform traffic pattern on various networks: 4096 nodes, 2 VCs, 16 flits

7.5 shows the average transfer time divided by message length for the uniform traffic pattern. It is shown that in the MH3DT network, average transfer time decreases and maximum throughput increases with an increase in message length.

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

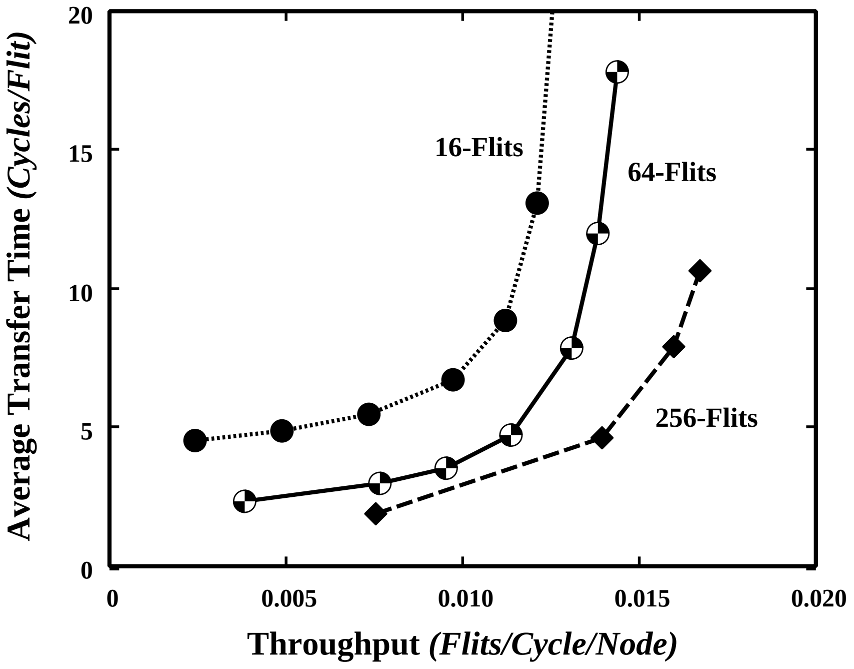


Figure 7.5: Average transfer time divided by message length versus network throughput of MH3DT network: 4096 nodes, 2 VCs, 16 flits, Buffer Size 2 flits.

Effect of Virtual Channels

We investigated the effect of adding extra virtual channels on the MH3DT network for dimension-order routing. Figure 7.6 depicts the average transfer time as a function of network throughput under the uniform traffic pattern for different virtual channels. The minimum number of virtual channels for deadlock-free routing is 2. Adding 1 extra virtual channel, for a total of 3, substantially improves the dynamic communication performance of the MH3DT network. We have also evaluated the dynamic communication performance of the MH3DT network using 4 virtual channels. Figure 7.6 shows that the maximum throughput of the MH3DT network using 3 virtual channels is higher than that using 2 and almost same as that of 4 virtual channels. This striking difference of throughput shows that we can significantly improve the dynamic communication performance of the MH3DT network by adding 1 extra virtual channel over the minimum number.

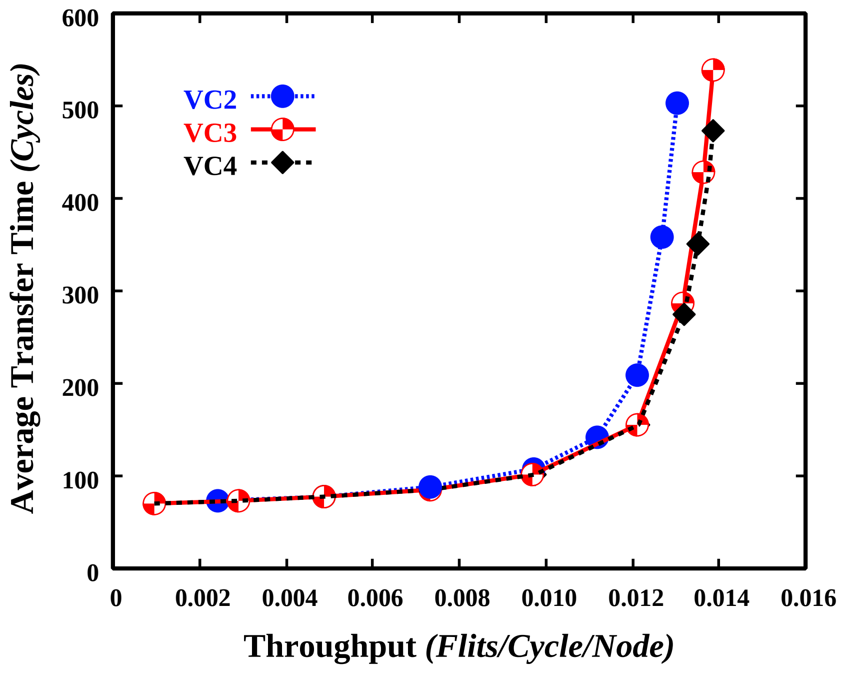


Figure 7.6: Dynamic communication performance of dimension order routing with uniform traffic pattern on the MH3DT network: 4096 nodes, various virtual channels,, 16 flits, Buffer Size 2 flits.

7.2.6 Summary

We have modified the H3D-torus network and presented a new hierarchical interconnection network, called the MH3DT network for massively parallel computers. The architecture of the MH3DT network, routing of messages, and static network performance were discussed in detail. A deadlock-free routing algorithm with a minimum number of virtual channels has been proposed for the MH3DT network. It has been proven that 2 virtual channels per physical channel are sufficient for the deadlock-free routing algorithm of the MH3DT network – 2 is also the minimum required number of virtual channels.

From the static network performance, it has been shown that the MH3DT network possesses several attractive features including constant node degree, small diameter, small cost, high connectivity, small average distance, and better bisection width. By using the routing algorithm described in this paper and the uniform traffic pattern, we have evaluated the dynamic communication performance of the MH3DT network as well as that of several other interconnection networks. The average transfer time of the MH3DT network is lower than that of the H3D-torus, TESH, and mesh networks. Maximum throughput of the MH3DT network is also higher than that of those networks. A comparison of dynamic communication performance reveals that the MH3DT network achieves better results than the H3D-torus, TESH, and mesh networks. The MH3DT network yields low latency and high throughput, which are very essential for high-performance massively parallel computers.

7.3 Modified TESH Network

A **T**ori connected m**ESH** (TESH) network [42, 47–50, 69, 124, 142] is a new interconnection network for large-scale 3D multicomputers. The TESH network consists of multiple basic modules (BMs) which are 2D-mesh networks. The BMs are hierarchically interconnected by a 2D-torus to build higher level networks. A TESH is a suitable network for massively parallel computers. But the restricted use of physical links between basic modules in the higher level networks reduces the dynamic communication performance of this network. With the increase of inter-level connectivity, the communication performance of the TESH network is better than that of mesh network. But with the increase of inter-level connectivity, the vertical links between silicon planes will also be increased. Thus, increasing the inter-level connectivity is not a suitable solution. It has already been shown that a torus network has better dynamic communication performance than a mesh network [64]. This is the key motivation that led us to replace the 2D-mesh of a TESH network by a 2D-torus network.

The modified TESH network consists of BMs which are themselves 2D-tori, hierarchically interconnected in a 2D-torus networks. Analogous to the TESH network, we called it as **T**ori connected **T**orus **N**etwork (TTN) .

7.3.1 Interconnection of the TTN

The TTN consists of BMs that are hierarchically interconnected to form higher level networks. The BM of the TTN is a 2D-torus network of size $(2^m \times 2^m)$, where m is a positive integer. However, in this section, we focus attention on (4×4) BM's, for which $m = 2$. The BM has some free ports at the contours for higher level interconnection. A BM of (4×4) torus is shown in Figure 7.7. All ports of the interior Processing Elements (PEs) are used for intra-BM connections. All free ports of the exterior PEs, either one or two, are used for inter-BM connections to form higher level networks. BM refers to a Level-1 network.

Successive higher level networks are built by recursively interconnecting immediate lower level subnetworks in a 2D-torus network. A higher-level network having a $(2^m \times 2^m)$ BM is built through a 2D-toroidal connection of (2^{2m}) immediate lower level subnetworks. For example, considering $(m = 2)$ a second-level subnetwork, can be formed by interconnecting 16 BMs. Similarly, a third-level network can be formed by interconnecting 16 second-level subnetworks, and so on. As illustrated in Figure 7.8, a Level-2 TTN, for example, can be formed by interconnecting 16 BMs as a (4×4) 2D-torus network. Each BM is connected to its logically adjacent BMs.

A TTN(m, L, q) is built using $(2^m \times 2^m)$ BMs and has L levels in its hierarchy. Here, q is the inter-level connectivity. A $(2^m \times 2^m)$ BM has 2^{m+2} free ports at the contours for higher level interconnection. It is useful to note that for each higher level interconnection, a BM must use $4(2^q)$ of its free links. $2(2^q)$ free links for vertical interconnections and $2(2^q)$ free links for horizontal interconnections. Here, $q \in \{0, 1, ..., m\}$, is the inter-level connectivity. $q = 0$ leads to minimal inter-level connectivity, while $q = m$ leads to maximum inter-level connectivity. As shown in Figure 7.7, for example, the (4×4) BM has 16 free ports. If we chose $q = 0$, then 4 of the free ports and their associated links are used for each higher level interconnection. The highest level network which can be

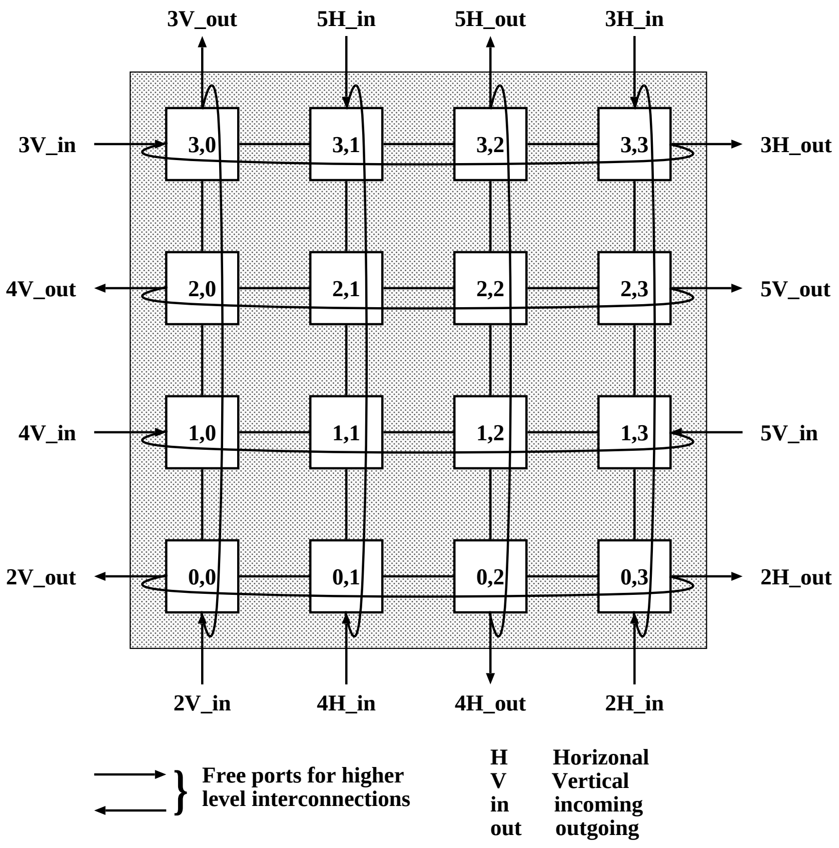


Figure 7.7: Basic module of the TTN

built from $(2^m \times 2^m)$ BM is $L_{max} = 2^{p-q} + 1$. With $q = 0$, $L_{max} = 2^{2-0} + 1 = 5$. Level-5 is the highest possible level that (4×4) BM can be interconnected. The total number of nodes in a TTN having $(2^m \times 2^m)$ BMs is $N = 2^{2mL}$. $L_{max} = 2^{p-q} + 1$ denotes maximum level of hierarchy. Thus, the maximum number of nodes which can be interconnected by a TTN(m, L, q) is $N = 2^{2m(2^{p-q})}$.

Base-4 numbers are used for convenience of address representation. As seen in Figure 7.7, nodes in the BM are addressed by two digits, the first representing the row index and the next representing the column index. More generally, in a Level-L TTN, the node address is represented by:

$$\begin{aligned} A &= A^L A^{L-1} A^{L-2} \ldots \ldots \ldots A^2 A^1 \\ &= a_{n-1}a_{n-2}\ a_{n-3}a_{n-4} \ldots \ldots \ldots a_3 a_2\ a_1 a_0 \\ &= a_{2L-1}a_{2L-2}\ a_{2L-3}a_{2L-4} \ldots \ldots \ldots a_3 a_2\ a_1 a_0 \\ &= (a_{2L-1}\ a_{2L-2})\ (a_{2L-3}\ a_{2L-4}) \ldots \ldots \ldots \ldots \ldots (a_3\ a_2)\,(a_1\ a_0) \end{aligned} \tag{7.7}$$

Here, the total number of digits is $n = 2L$, where L is the level number. Pairs of digits run from group number 1 for Level-1, i.e., the BM, to group number L for the L-th level. Specifically, i-th group $(a_{2i-1}\ a_{2i-2})$ indicates the location of a Level-$(i-1)$ subnetwork within the i-th group to which the node belongs; $2 \leq i \leq L$. In a two-level network,

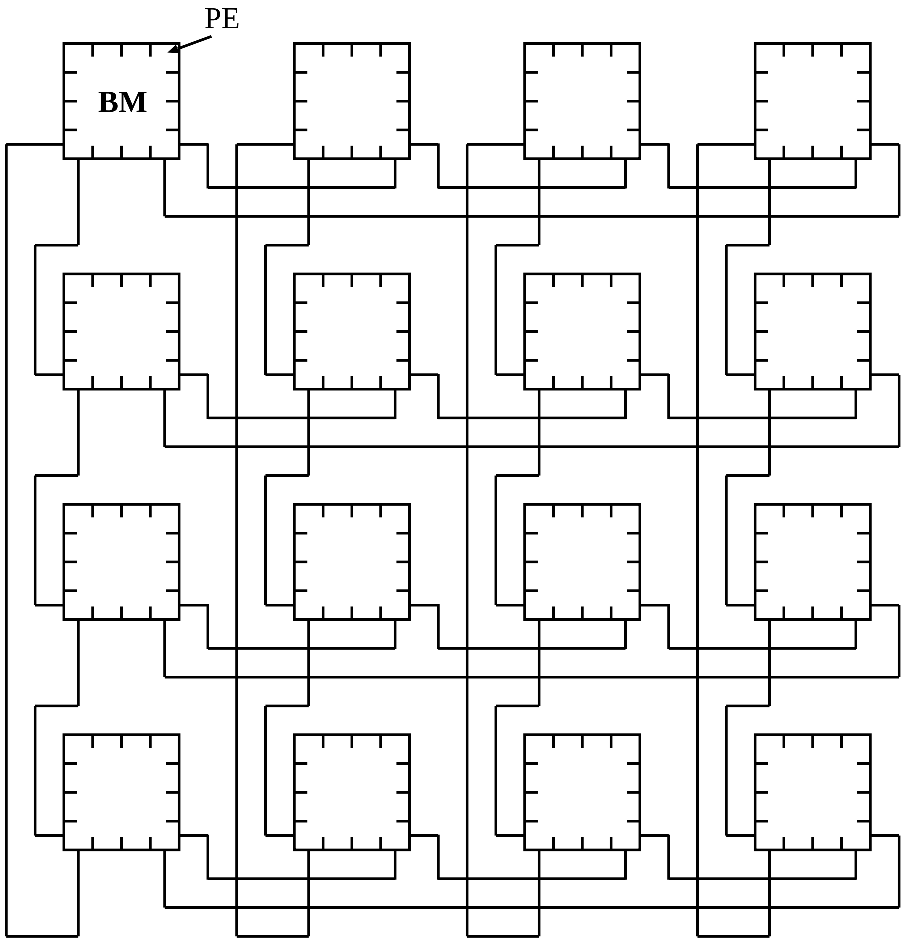


Figure 7.8: Interconnection of a Level-2 TTN

for example, the address becomes $A = (a_4\ a_3)\ (a_1\ a_0)$. The last group of digits $(a_4\ a_3)$ identifies the BM to which the node belongs, and the first group of digits $(a_1\ a_0)$ identifies the node within that BM.

7.3.2 Routing Algorithm

Routing of messages in the TTN is performed from top to bottom. That is, it is first done at the highest level network; then, after the packet reaches its highest level sub-destination, routing continues 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's destination is 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.

Deterministic, dimension order routing algorithm is used by most existing multicomputers due to its simplicity. We have also considered the dimension order routing algorithm for the TTN. We use the following strategy: at each level, vertical routing is performed first. Once the packet reaches the correct row, then horizontal routing is performed. Routing in the TTN is strictly defined by the source node address and the destination node address. Let a source node address be $s = (s_{2L-1}, s_{2L-2}), (s_{2L-3}, s_{2L-4}), ...,$ $(s_3, s_2), (s_1, s_0)$, a destination node address be $d = (d_{2L-1}, d_{2L-2}), (d_{2L-3}, d_{2L-4}), ..., (d_3, d_2),$ (d_1, d_0), and a routing tag be $t = (t_{2L-1}, t_{2L-2}), (t_{2L-3}, t_{2L-4}), ..., (t_3, t_2), (t_1, t_0)$, where $t_i = d_i - s_i$. Figure 7.9 shows the routing algorithm for the TTN.

In the case of $q \geq 1$, there are multiple links for same level and direction. Logically, in this case, each node selects the nearest link. For example, to transfer a packet from Node-$(2, 1)$ in a BM to Level-2 vertical link, the packet forwarded to Node$(2, 0)$ since the link of Node-$(2, 0)$ is nearest among the nodes with the same level link. The function *get_group_number* gets a group number. Arguments of this function are s, d, and routing direction. Each free-link is labeled as (g, l, d, δ), where, $2 \leq l \leq L$ is the level, $d \in \{V, H\}$ is the dimension, and $\delta \in \{+, -\}$ is the direction. The function *outlet_x* and *outlet_y* results the outlet node of the BM for higher level.

7.3.3 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 TTN. We also present a proof that the TTN is deadlock-free by these number of virtual channels.

To prove the proposed routing algorithm for the TTN is deadlock free, we divide the routing path into three phases, as follows:

- *Phase 1:* Intra-BM transfer path from source PE to the outlet node 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.

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

The proposed routing algorithm enforces some routing restrictions to avoid deadlocks [38]. Since dimension-order routing is used in the TTN, routing of message first performed in the vertical direction and then in the horizontal direction. The interconnection of the BM and the higher levels of the TTN is a toroidal connection. The number of virtual channels required to make the routing algorithm deadlock-free for the TTN is determined using Corollary 4.1.

Theorem 7.2 *A TTN with* 4 *virtual channels is deadlock free.*

```
Routing TTN(s,d);
source node address:$s_{2L-1}, s_{2L-2}, s_{2L-3}, ..., s_1, s_0$
destination node address: $d_{2L-1}, d_{2L-2}, d_{2L-3}, ..., d_1, d_0$
tag: $t_{2L-1}, t_{2L-2}, t_{2L-3}, ..., t_1, t_0$
  for $i = 2L - 1 : 2$
    if $\{(d_i - s_i + 2^m) \bmod 2^m\} \leq \frac{2^m}{2}$ then routedir = positive;
         $t_i = \{(d_i - s_i + 2^m) \bmod 2^m\}$;
    else routedir = negative;
         $t_i = \{2^m - (d_i - s_i + 2^m) \bmod 2^m\}$; endif;
    $g$ = get_group_number($s, d$, routedir);
    while ($t_i \neq 0$) do
       if ($i$ mod 2) = 0, then
            outlet_node$_x$ = outlet_x($g, \lfloor \frac{i}{2} + 1 \rfloor, H$, routedir);
            outlet_node$_y$ = outlet_y($g, \lfloor \frac{i}{2} + 1 \rfloor, H$, routedir); endif;
       if ($i$ mod 2) = 1, then
            outlet_node$_x$ = outlet_x($g, \lfloor \frac{i}{2} + 1 \rfloor, V$, routedir);
            outlet_node$_y$ = outlet_y($g, \lfloor \frac{i}{2} + 1 \rfloor, V$, routedir); endif;
        BM_Routing(outlet_node$_x$, outlet_node$_y$)
        if (routedir = positive), move packet to next BM; endif;
        if (routedir = negative), move packet to previous BM; endif;
        if ($t_i > 0$), $t_i = t_i - 1$; endif;
        if ($t_i < 0$), $t_i = t_i + 1$; endif;
     endwhile;
  endfor;
    BM_Routing($t_y, t_x$)
end
BM_Routing ($t_1, t_0$);
BM_tag $t_1, t_0$ = receiving node address ($r_1, r_0$) − destination ($d_1, d_0$)
  for $i = 1 : 0$
    if ($t_i > 0$ and $t_i \leq 2^{m-1}$) or ($t_i < 0$ and $t_i = -(2^m - 1)$), movedir = positive; endif;
    if ($t_i > 0$ and $t_i = (2^m - 1)$) or ($t_i < 0$ and $t_i \geq -2^{m-1}$), movedir = negative; endif;
    if (movedir = positive and $t_i > 0$), distance = $t_i$; endif;
    if (movedir = positive and $t_i < 0$), distance = $2^m + t_i$; endif;
    if (movedir = negative and $t_i < 0$), distance = $t_i$; endif;
    if (movedir = negative and $t_i > 0$), distance = $-2^m + t_i$; endif;
 endfor
  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 = negative), 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 = negative), move packet to $-x$ node; distance$_0$ = distance$_0 + 1$; endif;
  endwhile;
end
```

Figure 7.9: Routing algorithm of the TTN

Proof: Both the BM and the higher levels of the TTN 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 TTN is a 2D-torus network. According to Corollary 4.1, the number of necessary virtual channels for phase-1 and phase-3 is 2. The routing of the message in source-BM and destination-BM is carried out separately. The virtual channels required in *phase*-1 and *phase*-3 can share each other. Intra-BM links between inter-BM links are used in sub-phases 2.i.1 and 2.i.3. Thus, sub-phases 2.i.1 and 2.i.3 utilize channels over intra-BM links, sharing the channels of either phase-1 or phase-3. The free links of the BM are used in inter-BM routing, i.e., sub-phases 2.i.2 and 2.i.4, and these links form a 2D-torus network for the higher level network. According to Corollary 4.1, the number of necessary virtual channels for this 2D-torus network is also 2.

Therefore, the total number of necessary virtual channels for the whole network is 4.

7.3.4 Static Network Performance

In this section, we discuss some of the properties and performance metrics that characterize the cost and performance of an interconnection network. The static network performance of various networks with 4096 nodes including the proposed TTN is tabulated in Table 7.2. We did not consider hypercube because of it very high wiring complexity. The performance of MH3DT is better than that of H3D-torus networks

Node Degree

For the TTN, the node degree is independent of network size. Since each node has six channels, its degree is 6. 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. The I/O interface cost of a particular node is proportional to its degree. The degree of the TTN is higher than that of its counterpart TESH network but lower than that of MH3DT network.

Diameter

The *diameter* of a network is the maximum inter-node distance, i.e., the maximum number of links that must be traversed to send a message to any node along a shortest path. Table 7.2 shows a comparison of the TTN diameter with several other networks. Clearly, the TTN has a much smaller diameter than the conventional mesh, torus, and TESH networks. Although MH3DT provides a comparable diameter with the TTN, it requires more links.

Cost

The product (*diameter* $\times$ *node degree*) is defined as the cost of a network in technology independent manner. Table 7.2 shows that the cost of the TTN is smaller than that of the mesh, torus, TESH(2,3,0) and MH3DT networks. From the performance of TESH(2,3,0) and TESH(2,3,1), it can be believed that the cost of TTN(2,3,1) is also lower than that of the TESH(2,3,1).

Average Distance

The *average distance* is the mean distance between all distinct pairs of nodes in a network. We have evaluated the average distance for different conventional topologies by the corresponding formula and of different hierarchical networks by simulation. As shown in Table 7.2, the TTN has a smaller average distance than the conventional mesh & torus network and hierarchical TESH network. However, the average distance of the TTN is higher than that of the MH3DT network.

Arc Connectivity

Arc connectivity is the minimum number of links that must removed in order to break the network into two disjoint parts. A network is maximally fault-tolerant if its connectivity is equal to the degree of the network. The arc connectivity of various networks is shown in Table 7.2. Clearly, the arc connectivity of the TTN is higher than a conventional mesh and TESH networks and equal to that of the torus network. However, the arc connectivity of the torus network is exactly equal to its degree. Thus, torus is more fault tolerant than the TTN.

Table 7.2: Comparison of static network performance of various network with 4096 node

	Node Degree	Diameter	Cost	Average Distance	Arc Connectivity	Bisection Width	Wiring Complexity
2D-Mesh	4	126	504	42.67	2	64	8064
2D-Torus	4	64	256	32	4	128	8192
MH3DT (4,4,2,2)	8	18	144	9.37	6	128	12864
TESH (2,3,0)	4	32	128	17.80	2	8	8192
TESH (2,3,1)	4	28	112	14.53	2	16	10240
TTN (2,3,0)	6	20	120	10.59	4	8	10240

Bisection Width

The *Bisection Width (BW)* of a network is defined as the minimum number of links that must removed to partition the network into two equal halves. Table 7.2 shows that the bisection width of the TTN is lower than that of the mesh, torus, and MH3DT networks and equal to that of the TESH network.

Wiring Complexity

The *wiring complexity* of an interconnection network refers to its total number of links. The wiring complexity has a direct correlation to hardware cost and complexity. The wiring complexity of a Level-L TTN is represented as shown in Eq. 7.8.

$$\left[k^{2(L-1)} \times \left\{2k^2 + 4(2^q)(L-1)\right\}\right] \tag{7.8}$$

Table 7.1 compares the wiring complexity for a TTN with several other networks. The total number of physical links in the TTN is higher than that in the mesh, torus, and TESH, networks; therefore, the cost of physical links is higher for the TTN. But it is lower than that of MH3DT network.

7.3.5 Dynamic Communication Performance

The dynamic communication performance of a multicomputer is characterized by message latency and network throughput. A network with low latency and high throughput is preferable for massively parallel computers. To evaluate dynamic communication performance, we have developed a wormhole routing simulator. In our simulation, we use a dimension-order routing algorithm. The dimension-order routing algorithm, which is exceedingly simple, provides the only route for the source-destination pair. Extensive simulations for several networks have been carried out under the uniform traffic pattern, in which each node sends messages to every other node with equal probability. We did not consider H3D-torus network in this section, rather we consider MH3DT network. Four virtual channels per physical channel are simulated, and the virtual channels are arbitrated by a round robin algorithm. For all of the simulation results, the packet size is 16 flits. Two flits are used as the header flit. Flits are transmitted at 20,000 cycles; in each clock cycle, one flit is transferred from the input buffer to the output buffer, or from output to input if the corresponding buffer in the next node is empty. Therefore, transferring data between two nodes takes 2 clock cycles.

Dynamic Communication Performance Evaluation

We have evaluated the dynamic communication performance of several 4096 node networks under uniform traffic patterns. It is already shown that the dynamic communication performance of the MH3DT network is better than H3D-torus network. This is why, in this section, for comparison, we did not consider H3D-torus network. We also exclude hypercube for its high cost. We have also consider conventional mesh and torus networks.

To evaluate the dynamic communication performance of the TESH and TTN networks, we have used minimum inter-level connectivity, i.e., $q = 0$. For a fair comparison, we allocated 4 virtual channels to the router for performance evaluation. Figure 7.10 shows the results of simulation under uniform traffic patterns for the various network models. This figure presents the average transfer time as a function of network throughput. Each curve stands for a particular network. As shown in Figure 7.10, the average transfer time of the TTN is far lower than that of the mesh network, remarkably lower than that of TESH network, and a bit higher than that of MH3DT network. This benefit of latency for MH3DT network is achieved with the cost of extra links. The maximum throughput of the TTN is higher than that of the MH3DT, TESH, and mesh networks. Therefore, the dynamic communication performance of the TTN with minimum inter-level connectivity is better than that of mesh, TESH, and MH3DT networks.

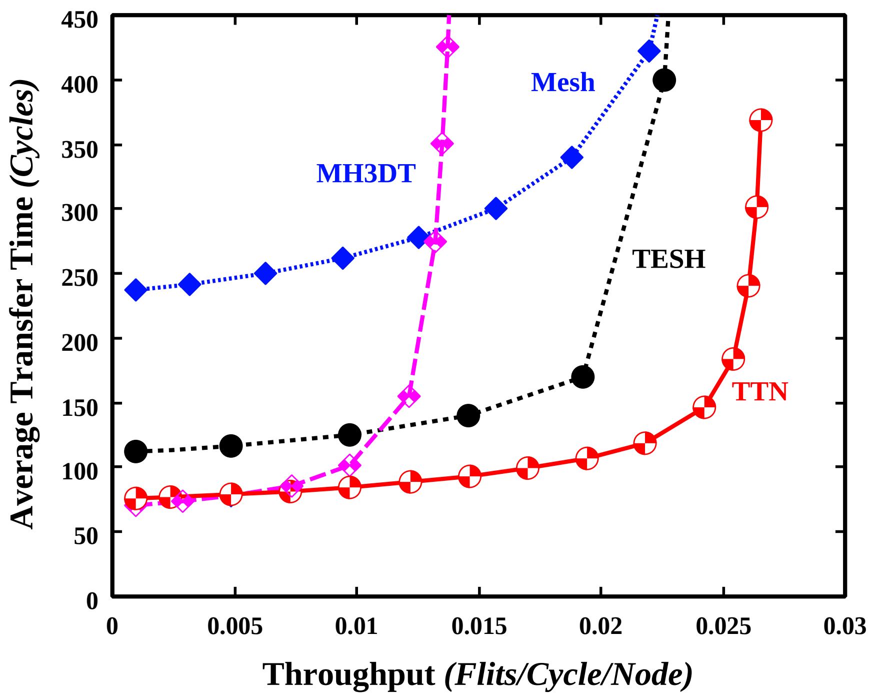


Figure 7.10: Dynamic communication performance of dimension-order routing with uniform traffic pattern on various networks: 4096 nodes, 4 VCs, 16 flits, and $q = 0$.

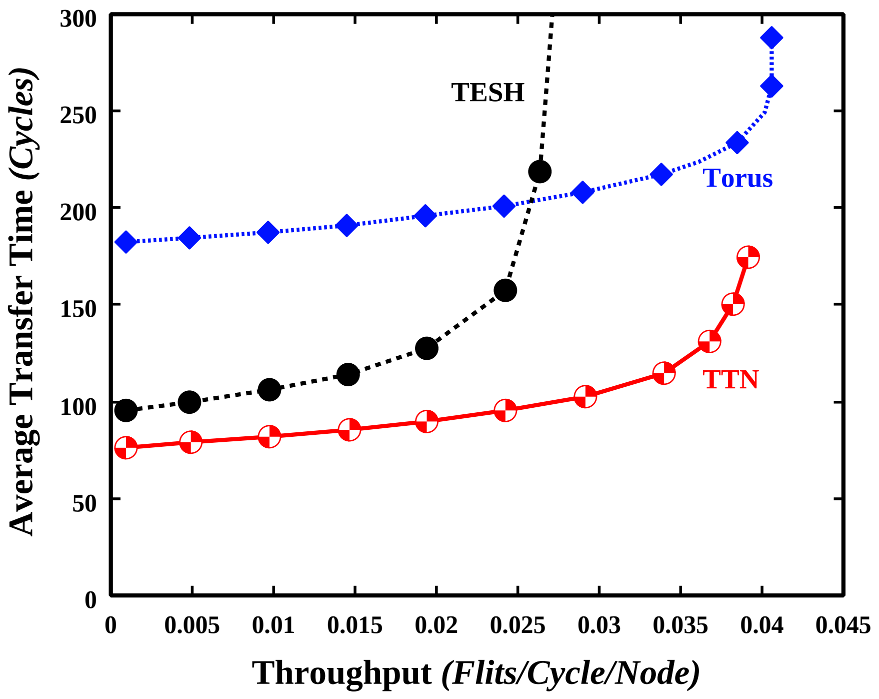


Figure 7.11: Dynamic communication performance of dimension-order routing with uniform traffic pattern on various networks: 4096 nodes, 4 VCs, 16 flits, and $q = 1$.

To show the superiority of the TTN over torus network, we have increased the inter-level connectivity from $q = 0$ to $q = 1$. Figure 7.11 shows the results of simulation under uniform traffic patterns for the TTN along with TESH and torus networks. Each curve stands for a particular network. As shown in Figure 7.11, the average transfer time of the TTN is far lower than that of the torus network and remarkably lower than that of TESH network. The maximum throughput of the TTN is higher than that of the TESH network. However, the maximum throughput of the TTN is a little bit higher than that of torus, with the accompanying cost of large latency. Therefore, we can conclude that the the dynamic communication performance of the TTN is better than that of torus and TESH networks.

In [124, 125], it is shown that the dynamic communication performance, especially network throughput, of the TESH network is lower than that of mesh network with $q = 0$ and higher than that of mesh network with $q = 1$. But, the benefits 3D-WSI implementation of TESH network is diminishing with the increase of inter-level connectivity (q). Because the the vertical links between wafer planes will increase with q. As shown in Figure 7.10, it is shown that the dynamic communication performance of the TTN is better than that of mesh network with $q = 0$, i.e. minimum inter-level connectivity. As shown in Table 7.2, it is shown that the total number of physical links of TESH(231) and TTN(231) is equal. Therefore, with the same physical link cost, TTN yields better performance than that of TESH network.

Torus is a suitable network for parallel computers due to its symmetry and regularity. However, the length of the longest wire is a limiting factor for a network with thousands or millions of nodes. The operating speed of a network is limited by the physical length of links. With the cost of some additional short length links, the dynamic communication performance of the TTN is better than that of torus network.

To show the versatility of the TTN, it is required do more experiments with various point of view such as deadlock-free routing with minimum number of virtual channels, dynamic communication performance under various traffic patterns, effect of message length, and effect of virtual channels.

7.3.6 Summary

We have modified the TESH network and presented a new hierarchical interconnection network, called TTN for massively parallel computers. The architecture of the TTN, routing of messages, and static network performance were discussed in detail. A deadlock-free routing algorithm with 4 virtual channels has been proposed for the TTN.

From the static network performance, it has been shown that the TTN possesses several attractive features including constant node degree, small diameter, small cost, high connectivity, small average distance, and better bisection width. By using the routing algorithm described in this paper and the uniform traffic pattern, we have evaluated the dynamic communication performance of the TTN as well as that of several other interconnection networks. The average transfer time of the TTN is lower than that of the mesh, torus, TESH, and MH3DT networks. Maximum throughput of the MH3DT network is also higher than that of those networks. A comparison of dynamic communication performance reveals that the TTN achieves better results than the mesh, torus, TESH, and MH3DT networks. The TTN yields low latency and high throughput with reasonable cost. Therefore, TTN is a good choice would be a good interconnection network for future generation massively parallel computers.

7.4 Conclusion

In this chapter, we have modified two hierarchical (H3D-torus and TESH) networks using torus-torus networks. We called these networks as Modified Hierarchical 3D-Torus (MH3DT) Network and Tori connected Torus Network (TTN). The architecture of these networks, routing of messages, static network performance, and dynamic communication performance were discussed in detail. Form the performance evaluation, the modification using torus-torus networks gives better result than that of its original one as well as other networks.

Chapter 8

"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 (1642–1727)

Conclusions

8.1 Introduction

The interconnection network is a crucial component in massively parallel computer systems because any interaction between the processing elements ultimately depends on its effectiveness [156]. Although many network architectures have been studied, and indeed deployed, none has proved clearly superior in all aspects, since the communication requirements of different applications vary widely. Communication bottlenecks in parallel computers arise from the fact that richly connected networks are impractical for large number of nodes. The k-ary n-cube has undoubtedly been the most popular interconnection network used in practice because it has the most desirable properties. However, it is impractical for 3D-WSI implementation with several thousands or millions of nodes. The research for this dissertation began with the search for a superior interconnection network for massively parallel computer systems.

8.2 Conclusions

Parallel computers are generally built from processing elements which are interconnected in a network. In this dissertation, we have proposed a new hierarchical interconnection network called **Hierarchical Torus Network (HTN)** to overcome the drawbacks of conventional networks. Using the proposed HTN, millions of nodes can be connected together while retaining good network properties. The design of an interconnection network is a trade-off among attributes. We studied various aspects of HTN to demonstrate its superiority over other networks.

The network architecture of the HTN, routing of messages in the network, static network performance, and 3D-WSI issues have been discussed in detail. From the static network performance, it can be seen that the HTN possesses several attractive features.

These include constant node degree, small diameter, small average distance, and high connectivity. A large bisection width implies a high degree of fault tolerance; however, it requires a larger layout area for VLSI implementation. A small bisection width network is wire efficient for VLSI implementation; however, it slows down the data and information exchange between the two halves of the network, making a network with moderate bisection width preferable. The bisection width of the HTN is higher than that of TESH and H3D-mesh networks, equal to that of a H3D-torus network, and smaller than that of conventional mesh and torus networks. Finally, we can conclude that the HTN has better bisection width. Using the HTN, millions of nodes can be interconnected while maintaining good static network performance.

The HTN is well suited for 3D stacked-implementation. It is shown that the peak number of vertical links in 3D stacked-implementation is quite low for HTN as compared to similar networks. Thus, HTN permits efficient VLSI/ULSI/WSI realization. The layout area of HTN in 3D stacked-implementation is amenable to 3D implementation. In part, this is due to the lower number of vertical wires needed than in almost all other multi-computer networks. 3D-WSI implementation reduces the longest wire length 4.20 times over planar implementation.

We have used wormhole routing for switchingbecause it has low buffering requirements, and more importantly, it makes latency independent of the message distance. Since wormhole routing relies on a blocking mechanism for flow control, deadlock can occur because of cyclic dependencies over network resources during message routing. Since the hardware cost increases as the number of virtual channels increases, the unconstrained use of virtual channels is cost-prohibitive in parallel computers. Therefore, a deadlock free routing algorithm with a minimum number of virtual channels is needed. In this dissertation, we have presented a deadlock free routing algorithm using dimension order routing with a minimum number of virtual channels for the HTN. It has been proven that two virtual channels per physical channel are sufficient for the deadlock free routing algorithm of the HTN; two is also the minimum number of virtual channels for dimension order routing.

By using the deterministic, dimension-order routing algorithm and the uniform traffic pattern, we have evaluated the dynamic communication performance of the HTN as well as that of several other commonly used networks and hierarchical interconnection networks with two virtual channels. The average transfer time of HTN is lower than that of the H3D-mesh, TESH, mesh, and torus networks. Maximum throughput of the HTN is also higher than that of those networks. A comparison of dynamic communication performance reveals that the HTN outperforms the H3D-mesh, TESH, mesh, and torus networks because it yields low latency and high throughput, which are indispensable for high performance massively parallel computers.

A deadlock-free routing algorithm for an arbitrary interconnection network with a minimum number of virtual channels is preferred. However, there is a trade-off between dynamic communication performance and the number of virtual channels. The performance improves significantly as virtual channels are initially added. The benefits then diminish as more channels are added. Therefore, it is better to use as many virtual channels as will yield optimum performance. In this dissertation, we have analyzed the effect of the number of virtual channels and selected the number of virtual channels to achieve optimum performance. It is shown that 3 virtual channels per physical channel is the best choice to achieve optimum performance. We have investigated the impact of non-uniform traffic patterns on the HTN using dimension order routing with 3 virtual channels. Under various nonuniform traffic patterns, the dynamic communication performance of the HTN is always better than that of the H3D-mesh, H3D-torus, TESH, and mesh networks. It is also shown that the impact of non-uniform traffic patterns on the HTN is less than on the other networks.

In this dissertation, we have described a suite of low-cost adaptive routers, link-selection (LS), channel-selection (CS), and combination of link-selection and channel-selection (LS+CS), with dimension order routing, analyzed their cost, and evaluated the dynamic communication performance of the HTN. The hardware cost for the LS, CS, and LS+CS algorithms is exactly equal to dimension order routing. Based on a 0.8 micron gate array technology, we characterized the speed of those adaptive routers including the popular dimension order router. The inter-node router delay of path setup and data through are 9.69 ns and 6.69 ns for dimension-order routing and 12.99 ns and 6.69 ns for the LS, CS, and LS+CS algorithms. The only overhead imposed is router delay for header selection.

The proposed adaptive routing algorithms – CS, LS, and LS+CS – are simple and efficient for using the physical links and virtual channels of an HTN to improve dynamic communication performance. The freedom from deadlock of the proposed CS, LS, and LS+CS algorithms using 3 virtual channels has been proved. Using these adaptive routing algorithms and several traffic patterns, we have evaluated the dynamic communication performance of the HTN. In all traffic patterns, the average transfer time of the HTN using the CS, LS, and LS+CS algorithms is lower than when the dimension order routing is used, but the differences are not impressive. On the other hand, maximum throughput using the CS, LS, and LS+CS algorithms is higher than when the dimension order routing algorithm is used. Efficient use of physical links and virtual channels significantly improves the dynamic communication performance. A comparison of dynamic communication performance reveals that the LS+CS algorithm outperforms all other algorithms; an HTN using the LS+CS algorithm yields low latency and high throughput. The hardware cost of the LS+CS algorithm is exactly equal to that of dimension order routing. Therefore, an HTN with the LS+CS algorithm would be a good choice for future massively parallel computers.

A fault tolerant network is essential for the reliability of massively parallel computer systems, because a single node failure in a massively parallel computer system may make the computer crash or produce an erroneous computational result. We have presented a reconfiguration scheme for replacing faulty nodes with redundant nodes for the HTN. A hierarchical redundancy approach is explored, in which redundancy is provided at each level of the network. We have evaluated the yield of the HTN. The results indicate that, with a 25% redundancy at each level, the system yield at the basic module and second level are satisfactory. In short, we conclude that we have accomplished the reconfiguration by redundancy and successfully estimated the yield of the HTN.

The interconnection network used in a multicomputer system plays a key role in determining how fast applications can be executed on the system. To show the suitability of the HTN, we discuss mapping of some commonly used advanced applications, such as bitonic merge, FFT, and finding the maximum. The processing time of an application mapping in an interconnection network depends on the number of communication steps.

The versatility of the HTN in various application mappings is investigated by evaluating the total number of communication steps. It is shown that the number of communication steps required for various advanced application mappings on the HTN is lower than for conventional and other hierarchical interconnection networks. A comparison of the total number of communication steps reveals that the HTN outperforms the H3D-mesh, H3D-torus, TESH, and k-ary n-cube networks.

Pruning is the process of removing links from a basis network in a periodic fashion for reduced hardware complexity and increased performance. The wiring complexity of the system is an important issue since the silicon area is limited and in general networks are wiring intensive. HTN consists of multiple basic modules, where each basic module is a 3D-torus network. The wrap-around links of the end-to-end nodes result in a large number of links. The application of pruning techniques on the HTN reduces the wiring complexity and hence the hardware cost. The architecture of the pruned HTN and the 3D-Integration issue are discussed in detail in this dissertation. Pruned HTN reduces both the I/O interface cost and hardware cost and complexity. The pruned HTN is well suited for 3D-wafer stacked-implementations. It is shown that the peak number of vertical links in a 3D stacked-implementation is low for pruned HTN as compared to its non-pruned counterpart. The layout area of pruned HTN in 3D-WSI is less than that of non-pruned HTN. Therefore, HTN permits efficient VLSI/ULSI/WSI realization.

We have shown through the HTN that a hierarchical interconnection network where both the basic module and Level-2 networks have toroidal connection and yields better dynamic communication performance than other hierarchical networks. To show the versatility of the torus-torus combination for hierarchical networks, we have modified two hierarchical networks (H3D-torus and TESH) using torus-torus networks.

We have modified the H3D-torus network and presented a new hierarchical interconnection network, called the MH3DT network for massively parallel computers. The architecture of the MH3DT network, routing of messages, and static network performance were discussed in detail. A deadlock-free routing algorithm with a minimum number of virtual channels has been proposed for the MH3DT network. We proved that 2 virtual channels per physical channel are sufficient for the deadlock-free routing algorithm of the MH3DT network – 2 is also the minimum required number of virtual channels. From the static network performance, it was shown that the MH3DT network possesses several attractive features including constant node degree, small diameter, small cost, high connectivity, small average distance, and better bisection width. By using the dimension-order routing algorithm and the uniform traffic pattern, we have evaluated the dynamic communication performance of the MH3DT network as well as that of several other interconnection networks. The average transfer time of the MH3DT network is lower than that of the H3D-torus, TESH, and mesh networks. Maximum throughput of the MH3DT network is also higher than that of those networks. A comparison of dynamic communication performance reveals that the MH3DT network achieves better results than the H3D-torus, TESH$(2, 3, 0)$, and mesh networks.

We have also modified the Tori connected mESH (TESH) network. Analogous to the TESH network, we refer to it as the Tori connected Torus Network (TTN). The architecture of the TTN, routing of messages, and static network performance were discussed in detail. A deadlock-free routing algorithm with 4 virtual channels has been proposed for the TTN. From the static network performance, it has been shown that the TTN, like

MH3DT, possesses several attractive features that include constant node degree, small diameter, small cost, high connectivity, small average distance, and better bisection width. By using the dimension-order routing algorithm and the uniform traffic pattern, we have evaluated the dynamic communication performance of the TTN as well as that of several other interconnection networks. The average transfer time of the TTN is lower than that of the mesh, torus, TESH, and MH3DT networks. Maximum throughput of the TTN is also higher than that of those networks. A comparison of dynamic communication performance reveals that the TTN achieves better results than the mesh, torus, TESH, and MH3DT networks. The TTN yields low latency and high throughput with reasonable cost.

8.3 Future Directions

There are many areas in which the research presented in this dissertation could be extended. Some of these areas for further exploration are discussed below

- Many studies [157–162] have been conducted on the design, implementation, and simulation-based and experimental-based performance evaluation of collective communication algorithms on various networks. In this dissertation, for routing message in the HTN, we have considered only one-to-one communication. A deadlock-free wormhole routed multicast routing system for the HTN would also be an interesting future research topic.

- There are two ways to evaluate the dynamic communication performance of an interconnection network, viz., computer simulation and analytical model [163]. We have evaluated the dynamic communication performance by computer simulation. However, an analytical model is a cost-effective and versatile tool for evaluating system performance under different design alternatives. Using analytical models, one can see the effect of each parameter on the system performance including those parameters related to the network configuration, implementation choices and traffic load. In future, we would like to evaluate the dynamic communication performance of the HTN using a new analytical model and compare its performance with the simulation results.

- In real systems increasing size increases failure rates, and the incorporation of fault tolerant techniques will be of great importance. Parallel computers tend to be quite large and they could benefit greatly from fault-tolerant capabilities. Redundancy and yield of HTN is presented in this dissertation but is not sufficient for fault tolerance. Therefore, we would like to develop fault-tolerant routing algorithms for an HTN with faulty nodes.

- The ability of a network topology to efficiently simulate another topology is often of considerable interest when 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.

- We have pointed out pruning technique for HTN and studied the 3D-WSI issue for the pruned HTN. Due to a shortage of time, we could not explore various issues regarding pruned HTN. Thus, the static network performance and dynamic communication performance of the pruned HTN have been kept as future works.

- The prototyping approach, implemented on Field Programmable Gate Array (FPGA), provides another middle ground between paper design and full scale implementation efforts where a designer can quickly test the ideas without committing undue resources. Prototype implementation on an FPGA significantly reduce the effort, cost, and risk of hardware implementation. After analyzing both analytic and simulation performance, we will implement an HTN on FPGA.

- A deadlock free routing algorithm using dimension order routing with 4 virtual channels has been proposed for the TTN. However, the minimum number of virtual channels required for deadlock-free routing is 2. A deadlock free routing algorithm using dimension order routing with a minimum number of virtual channels and the evaluation of dynamic communication performance under various traffic pattern will be the subject of future works.

Of course, we would like to see the **H**ierarchical **T**orus **N**etwork **(HTN)** made fully operational and in use by the research and industrial community.

Bibliography

[1] T. Boku, K. Itakura, H. Nakamura, and K. Nakazawa, "CP-PACS: a massively parallel processor for large scale scientific calculations," *Proceedings of ACM Supercomputing Conference*, pp.108–115, 1997.

[2] W.C. Athas and C.L. Seitz, "Multicomputers: message passing concurrent computers," *IEEE Computer*, vol. 21, no. 8, pp.9–24, 1988.

[3] P. Mohapatra, "Wormhole routing techniques in multicomputer systems," *ACM Computing Surveys*, vol. 30, no. 3, pp.375–411, 1998.

[4] L.M. Ni and P.K. McKinley, "A survey of wormhole routing techniques in direct networks," *IEEE Computer*, vol. 26, pp. 62–76, 1993.

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

[6] W.J. Dally and B. Towles, "Principles and Practices of Interconnection Networks ", *Morgan Kaufmann Publishers*, San Francisco, California, USA, 2004.

[7] H. Fujii, Y. Yasuda, H. Akashi, Y. Inagami, M. Koga, O. Ishihara, M. Kashiyama, H. Wada, and T. Sumimoto, "Architecture and performance of the Hitachi SR 2201 massively parallel processor system," *Proceedings* 11th *International Parallel Processing Symposium*, pp. 233–241, 1997.

[8] Y. Yasuda, H. Fujii, H. Akashi, Y. Inagami, T. Tanaka, J. Nakagoshi, H. Wada, and T. Sumimoto,"Deadlock-free fault tolerant routing in the multidimensional crossbar network and its implementation for the Hitachi SR2201," *Proceedings of* 11th *International Parallel Processing Symposium*, pp. 346–352, 1997.

[9] J. Konicek, T. Tilton, A. Veidenbaum, C.Q. Zhu, E.S. Davidson, R. Downing, M. Haney, M. Sharma, P.C. Yew, P.M. Farmwald, D. Kuck, D. Lavery, R. Lindsey, D. Pointer, J. Andrews, T. Beck, T. Murphy, S. Turner, and N. Warter,"The Organization of the Cedar System," *Proceedings of International Conference Parallel Processing*, pp. 49–56, 1991.

[10] G.F. Pfister, W.C. Brantley, D.A. George, S.L. Harvey, W.J. Kleinfelder, K.P. McAuliffe, F.A. Melton, V.A. Norton, and J. WEiss,"The IBM research parallel processor prototype (RP3): introduction and architecture," *Proceedings of International Conference Parallel Processing*, pp. 764–771, 1985.

[11] M. Banikazemi, V. Moorthy, L. Herger, D. K. Panda, and B. Abali,“Efficient Virtual Interface Architecture Support for the IBM SP Switch-Connected NT Clusters,” *Proceedings International Parallel and Distributed Processing Symposium (IPDPS 2000)*, pp. 33–42, 2000.

[12] C. Leiserson, Z. Abuhamdeh, D. Douglas, C. Feynman, M. Ganmukhi, J. Hill, W. Hillis, B. Kuszmaul, M. St. Pierre, D. Wells, M. Wong-Chan, Y. Saw-Wen, and R. Zak,“The network architecture of the Connection Machine CM-5,” *Journal of Parallel and Distributed Computing*, vol. 33, no. 2, pp. 145-158, 1996.

[13] A. Ferreira, A.G. vel Lejbman, and S.W. Song, “Bus-based parallel computers: a viable way for massive parallelism,” *Proceedings of Parallel Architectures and Languages*, pp. 553–564, 1994.

[14] Intel Corp., iPSC/1 reference manual, 1986.

[15] R. Arlanskas, “iPSC/2 system: a second generation hypercube,” *Proceedings of 3rd ACM Conference on Hypercube Concurrent Computers and Applications*, pp. 38–42, 1988.

[16] S.F. Nugent, “The iPSC/2 direct-connect communication technology,” *Proceedings Conference Hypercube Concurrent Computers and Applications*, pp. 51–60, 1988.

[17] Intel Corporation, A Touchstone DELTA system description, 1991.

[18] Intel Corp., Paragon XP/S product overview, Supercomputer Systems Division, Beaverton, Oregon, 1991.

[19] C.L. Seitz, “The Cosmic cube,” *Communication of the ACM*, vol. 28, no. 1, pp. 22–33, 1985.

[20] nCUBE Systems, N-cube handbook, 1986.

[21] nCUBE Systems, nCUBE 2: nCUBE 6400 processor manual, 1990.

[22] nCUBE Systems, nCUBE-3, at http//www.ncube.com.

[23] A. Agarwal, R. Bianchini, D. Chaiken, K.L. Johnson, D. Kranz, J. Kubiatowicz, B.H. Lim, K. Mackenzie, and D. Yeung, “The MIT Alewife machine: architecture and performance,” *Proceedings of* 22^{nd} *Annual International Symposium Computer Architecture*, pp. 2–13, 1995.

[24] M. Noakes, D.A. Wallach, and W.J. Dally, “The J-machine multicomputer: an architectural evaluation,” *Proceedings of the* 20^{th} *International Symposium Computer Architecture*, pp. 224–235, 1993.

[25] M. Noakes and W.J. Dally, “System design of the J-machine,” *Proceedings of Advanced Research in VLSI*, pp. 179–192, 1990.

[26] M. Fillo, S.W. Keckler, W.J. Dally, N.P. Carter, A. Chang, Y. Gurevich, and W.S. Lee,"The M-Machine Multicomputer," *International Journal of Parallel Programming - Special Issue on Instruction-Level Parallel Processing Part II*, vol. 25, no. 3, pp. 183–212, 1997.

[27] C. Peterson, J. sutton, and P. Wiley, "iWARP: a 100-MPOS VLIW microprocessor for multicomputers," *IEEE Micro*, vol. 11, no. 13, 1991.

[28] D. Lenoski, J. Laudon, K. Gharachorloo, W. Weber, A. Gupta, J. Hennessy, M. Horowitz, and M.S. Lam., "The Stanford DASH multicomputer," *IEEE Computer*, vol. 25, no. 3, pp. 63–79, 1992. .

[29] J. Kuskin, D. Ofelt, M. Heinrich, J. Heinlein, R. Simoni, K. Gharachorloo, J. Chapin, D. Nakahira, J. Baxter, M. Horowitz, A. Gupta, M. Rosenblum, and J. Hennessy, "The Stanford FLASH multiprocessor," *Proceedings of the* 21^{st} *International Symposium Computer Architectures*, pp. 302–313, 1994.

[30] R.E. Kessler, J.L Swarszmeier, "Cray T3D: a new dimension for Cray research," *Proceedings CompCon*, pp. 176–182, 1993.

[31] Cray Research Inc., The Cray T3E scalable parallel processing system, on Cray s Web Page at http://www.cray.com/PUBLIC/product-info/T3E/ CRAY_T3E.html.

[32] SGI. Origin2000 Rackmount Owner's Guide, 007-3456-003, 1997. Web Psge at http://teckpubs.sgi.com/

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

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

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

[36] 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.

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

[38] 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.

[39] W.J.Dally, "Virtual-Channel Flow Control," *IEEE Transactions on Parallel and Destributed Systems*, vol.3, no.2, pp. 194–205, 1992

[40] S. Ramany, "Routing in wormhole networks," *Ph.D. Dissertation*, Computer Science Department, University of Saskatchewan, 1995.

[41] G.L. Frazier, "Buffering and flow control in communication switches for scalable multicomputers," *Ph.D. Thesis*, University of California, Los Angles, 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] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

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

[52] Peter Kok, Keong Loh, and Wen Jing Hsu,"Study of Hierarchical Interconnection Networks," *Technical report*, 2002.

[53] A. El-Amawy and S. Latifi, "Properties and Performance of Folded Hypercube," *IEEE Trans. on Parallel and Distributed Systems*, vol. 2, no. 1, pp. 31–42, 1991.

[54] A. Esfahanian, L.M. Ni, and B.E. Sagan, "The Twisted n-Cube with Application to Multiprocessing," *IEEE Trans. on Computers*, vol. 40, no. 1, pp. 88–93, 1991.

[55] J.M. Kumar and L.M. Patnaik, "Extended Hypercube: A Hierarchical Interconnection Network of Hypercube," *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, no. 1, pp. 45–57, 1992.

[56] N.F. Tzeng and S. Wei, "Enhanced Hypercube," *IEEE Trans. on Computers*, vol. 40, no. 3, pp. 284–294, 1991.

[57] S.G. Ziavras, "A Versatile Family of Reduced Hypercube Interconnection Network," *IEEE Trans. on Parallel and Distributed Systems*, vol. 5, no. 11, pp. 1210–1220, 1994.

[58] L.N. Bhuyan and D.P. Aggarwal, "Generalized hypercube and hyperbus structurtes for a computer network,"*IEEE Trans. on Computers*, vol.C-33, no. 4, pp. 323–333, 1984.

[59] 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.

[60] M.M. Hafizur Rahman, Y. 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.

[61] 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.

[62] S. Horiguchi and T. Ooki, "Hierarchical 3D-Torus Interconnection Network," *Proc. of ISPAN'00* , Texas, USA, pp.50–56, 2000.

[63] 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.

[64] 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.

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

[66] J.M. Wills and V.K. Jain, "Data Manipulator Interconnection Network for WSI design," *Proc. Int. Conf. on Wafer Scale Integration*, pp. 138–144, 1990.

[67] 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.

[68] L. Schwiebert and D. N. Jayasimha, "A Necessary and Sufficient Condition for Deadlock-Free Wormhole Routing," *Journal of Parallel and Distributed Computing*, vol. 32, no. 1, pp. 103–117, 1996.

[69] Y. Miura and S. Horiguchi, "An Adaptive Routing for Hierarchical Interconnection Network TESH," *Proc. of the 3rd Int'l Conf. on PDCAT0*, Kanazawa, Japan, pp. 335–342, 2002.

[70] Andrew A. Chien and Jae H. Kim, "Planer-Adaptive Routing: Low-cost Adaptive Networks for Multiprocessors," *Journal of the ACM*, vol.42, no.1, pp.91–123, 1995.

[71] Jae H. Kim, "Planer-Adaptive Routing: Low-cost Adaptive Networks for Multiprocessors," *M.Sc. Thesis*, University of Illinois at Urbana-Champaign, 1993.

[72] 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.

[73] D.A. Reed, "Cost-performance bounds for multicomputer networks," *IEEE Transactions Computers*, vol. C-32, no. 1, pp. 1183–1196, 1984.

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

[75] L.R. Dennison, "The Reliable Router: An Architecture for Fault Tolerant Interconnect," *Ph.D. Dissertation*, MIT, 1996.

[76] S. Konstantinidou and L. Snyder, "The Chaos Router," *IEEE Transactions on Computers*, vol. 43, no. 12, pp. 1386–1397, 1994.

[77] Vipin Kumar, Ananth Grama, Anshul Gupta, and George Karypis, "Introduction to Parallel Computing Design and Analysis of Algorithms", *The Benjamin/Cummings Publishing Company, Inc.*, Redwood city, California, USA, 1994.

[78] S.B. Akers, D. Harel, and B. Krishnamurthy, "The Star Graph: An Attractive Alternative to the n-Cube", *Proc. of the Int'l. Conf. on Parallel Processing*, pp.393–400, 1987.

[79] 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, 1989.

[80] F.T. Leighton, "Complexity Issues in VLSI", *MIT Press*, Cambridge, Massachusetts, 1983.

[81] F.T. Leighton, "Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercube", *Morgan Kaufmann Publishers*, San Mateo, California, USA, 1992.

[82] C. Leiserson, "Fat=trees: Universal Networks for Hardware-Efficient Supercomputing," *IEEE Trans. on Computers*, vol. 34, no. 10, pp. 892–901, 1985.

[83] J. Schwartz"Ultracomputers," *IEEE Trans. on Programming Languages and Systems*, vol. 2, no. 4, pp. 484–521, 1980.

[84] Ted Nesson, “Randomized, Oblivious, Minimal Routing Algorithms for Multicomputers,” *Ph.D. Dissertation*, Harvard University, Cambridge, 1995.

[85] 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.

[86] D.A. Reed, R.M. Fujimoto, “Multicomputer networks: message-based parallel processing,” MIT Press, 1987.

[87] D. Nassimi, S. Sahni, “Finding connected components and connected ones on a mesh-connected parallel computer,” *SIAM journal on Computing*, vol. 9, pp. 744–757, 1980.

[88] C.H. Yeh and B. Parhami, “Unified formulation of a wide class of scalable interconnection networks based on recursive graphs,” *Proc. Int'l. Conf. System Engineering*, 1996.

[89] C.H. Yeh and B. Parhami, “Swapped networks: unifying the architectures and algorithms of a wide class of hierarchical parallel processors,” *Proc. Internat. Conf. Parallel and Distributed Systems*, pp. 230–237, 1996.

[90] R. Muller and Q.F. Stout, “Parallel Algorithms for Regular Architectures: Meshes and Pyramids”, *The MIT Press*, Cambridge, Massachusetts, 1996.

[91] R. Miller and Q.F. Stout, “Data Movement Techniques for the Pyramid Computer,” *SIAM Journal on Computing*, vol. 16, no. 1, pp. 38–60, 1987.

[92] S. Campbell, M. Kumar, and S. Olariu, “The Hierarchical Cliques Interconnection Network,” *Journal of Parallel Distributed Computing*, vol. 64, no. 1, pp. 16–28, 2004.

[93] W.J. Hsu, “Fibonacci Cubes – A New Interconnection Topology,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, no. 1, pp. 3–12, 1993.

[94] K. Ghosh and K.R. Desai, “Hierarchical Cubic Networks,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 6, no. 4, pp. 427–435, 1995.

[95] 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.

[96] 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.

[97] M.M. Hafizur Rahman, Yasushi Inoguchi, and Susumu Horiguchi, “Modified Hierarchical 3D-Torus Network,” *IEICE Transactions on Information and Systems*, vol. E88-D, no. 2, pp.177–186, 2005.

[98] Y. Yamada, H. Amano, M. Koibuchi, A. Jouraku, K. Anjo, and K. Nishimura, "Folded Fat H-Tree: an interconnection topology for Dynamically Reconfigurable Processor Array," *Proc. of EUC04*, pp. 301–311, 2004. (LNCS 3207)

[99] M. M. de Azevedo, N. Bagherzadeh, Martin Dowd, and S. Latifi, "Some Topological Properties of Star Connected Cycles", *Information Processing Letters* vol. 58, pp. 81–85, 1996.

[100] Q.M.Malluhi and M.A.Bayoumi, "The Hierarchical Hypercube: A New Interconnection Topology for Massively Parallel Systems," IEEE Trans. on Parallel and Distributed Systems, vol. 5, no. 1, pp.17-30, 1994.

[101] Yamin Li, Shietung Peng, and Wanming Chu, "Metacube - A New Interconnection Network for Large Scale Parallel Systems," *Proc. 7th Asia-Pacific Computer Systems Architecture Conference (ACSAC'2002)*, Melbourne, Australia.

[102] Jie Wu, "Extended Fibonacci Cubes," *IEEE Trans. on Parallel and Distributed systems*, vol. 8, no. 12, pp. 1203–1210, 1997.

[103] R.A. Ayoubi, Q.M. Malluhi, and M.A. Bayoumi, "The extended cube connected cycles: an efficient interconnection for massively parallel systems," *IEEE Transactions on Computers*, vol. 45, no. 5, pp. 609 – 614, 1996.

[104] C. Chen, D.P. Agrawal, and J.R. Burke, "dBCube: A new class of hierarchical multiprocessor networks and it's area efficient layout," *IEEE Trans. on Parallel and Distributed System*, vol.4, no. 12, pp. 1332–1343, 1993.

[105] E.Ganesan and D.K.Pradhan, "The Hyper-deBruijn Networks: Scalable Versatile Architecture", *IEEE Trans. on Parallel and Distributed Systems*', vol. 4, no. 9, pp.962-978, 1993.

[106] ,"Network Performance of Hierarchical Interconnection Network de-Bruijn Connected Torus (BCT)," *Information Processing Society of Japan*, vol. , no. , pp. –, 2005.

[107] B.W.Arden and H.Lee, "Analysis of Chordal Ring Network", *IEEE Trans. on Computers*, vol. C-30, no. 4, pp.291-295, 1981.

[108] J.J.Park, K.Y.Chwa, "Recursive Circulant: A New Topology for Multicomputer Networks," *Proc. of ISPAN94*, pp.73–80, 1994.

[109] W.J.Dally, "Express Cubes: Improving the performance of k-ary n-cube interconnectin networks," *IEEE Trans. on Computers*, vol. 40, no. 9, pp.1016-1023, 1991.

[110] H.Li and M.Maresca, "Poly¿morphic-Torus Network", *IEEE Trans. on computers*, vol. C-38, no. 9, pp.1345-1351, 1989.

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

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

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

[114] J. Duato, "On the design of deadlock-free adaptive routing algorithms for multicomputers: design methodologies," *Proceedings Parallel Architectures and languages Europe*, pp. 390–405, 1991.

[115] J. Duato, "Deadlock-free adaptive routing algorithms for multicomputers: evaluation of a new algorithm," *Proceedings 3rd IEEE International Symposium on Parallel and Distributed Processing*, pp. 840–847, 1991.

[116] P. Lopez and J. Duato, "Deadlock-free adaptive routing algorithms for the 3D-torus: limitations and solutions," *Proceedings Parallel Architectures and Languages*, pp. 684–687, 1993.

[117] S. Warnakulasuriya and T. M. Pinkston, "A Formal Model of Message Blocking and Deadlock Resolution in Interconnection Networks," *IEEE Transactions Parallel and Distributed Systems*, vol. 11, no. 3, pp. 212–229, 2000.

[118] T. M. Pinkston and S. Warnakulasuriya, " On deadlocks in interconnection networks," *Proceedings 24th International Symposium on Computer Architecture (ISCA 97)*, pp. 38–49, 1997.

[119] J. Duato, "Improving the efficiency of virtual channels with time-dependent selection functions," *Proceedings Parallel Architectures and Languages*, pp. 635–650, 1992.

[120] K. Aoyama and Andrew A. Chien, "The cost of Adaptivity and Virtual Lanes in a Wormhole Router," *Journal of VLSI Design*, vol. 2, no. 4, pp. 315–333, 1995.

[121] D. Dai and D.K. Panda, "Effective use of virtual channels in wormhole routed DSM systems," *Technical Report, OSU-CISRC-10/97-TR46*, Department of Computer and Information Science, The Ohio-State University, 1997.

[122] W. Feng and K.G Shin, "The effect of virtual channels on the performance of wormhole algorithms in multicomputer networks," *University of Michigan directed Study Report*, May 1994.

[123] 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 UK Performance Engineering Workshop*, pp. 95-102, 2000.

[124] 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.

[125] Yasuyuki Miura, “Wormhole Routing for Hierarchical Interconnection Networks,” *Ph.D. Dissertation*, School of Information Science, Japan Advanced Institute of Science and Technology, 2002. (in japanese)

[126] M Schroeder and et al., “Autonet A high speed self configuring local area network using point to point links,” *IEEE Journal of Selected Areas in Communications*, vol. 9, no. 10, pp. 1318–1335, 1991.

[127] Xiaoding Zhang, “System Effets of Interprocessor Communication Latency in Multicomputers,” *IEEE Micro*, vol. 11, no. 2, pp.12–55, April 1991.

[128] W. Hsu, “Performance issues in wire-limited hierarchical networks,” *PhD Thesis*, University of Illinois-Urbana Champaign, 1992.

[129] L. Schwiebert, “A Performance Evaluation of Fully Adaptive Wormhole Routing including Selection Function Choice,” *IEEE International Performance, Computing, and Communications Conference*, pp. 117–123, 2000.

[130] G.F. Pfister and V.A. Norton, “Hot Spot Contention and Combining in Multistage Interconnection Networks,” *IEEE Trans. on Computers*, vol. 34, no. 10, pp. 943–948, 1985.

[131] F. Petrini and M. Vanneschi, “k-ary n-trees: High Performance Networks for Massively Parallel Architectures,” *Technical Report TR-95-18*, Universita di Pisa, Dec. 1995.

[132] K. Bolding, M. Fulgham, and L. Synder, “The Case of Chaotic Adaptive Routing,” *IEEE Trans. on Computers*, vol. 46, no. 12, pp. 1281–1292, 1997.

[133] H.H. Najaf-abadi and H. Sarbazi Azad, “The Effects of Adaptivity on the Performance of the OTIS-Hypercube Under Different Traffic Patterns,” *Proc. of IFIP Int’l. Conf. NPC2004*, LNCS, pp.390–398, 2004.

[134] P.R. Miller, “Efficient Communications for Fine-Grain Distributed Computers,” *Ph.D. Dissertation*, Southampton University, U.K., 1991.

[135] M. Grammatikakis, D. F. Hsu, M. Kratzel and J. F. Sibeyn, “Packet routing in fixedconnection networks: a survey,” *Journal of Parallel and Distributed Computing*, vol. 54, no. 2, pp. 77–132, 1998.

[136] W.J. Dally and H. Aoki, “Deadlock-free adaptive routing in multicomputer networks using virtual channels,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 4, pp. 66–74, 1993.

[137] K. Hwang, “Advanced computer architecture: parallelism, scalability and programmability”, McGraw-Hill (Ed.), 1993.

[138] J.H. Kim and A.A. Chien, “An Evaluation of Planar-Adaptive Routing (PAR),” *Proc. 4th IEEE Symp. on Parallel and Distributed Processing*, New-York, pp.470–478, 1992.

[139] A.A. Chien, “A Cost and Speed Model for k-ary n-cube Wormhole Routers,” *IEEE Trans. on Parallel and Distributed Systems*, vol.9, no.2, pp.150–162, 1998.

[140] K. Aoyama, “Design Issues in Implementing an Adaptive Router,” *M*aster’s thesis, Univ. of Illinois, Dept. of Computer Science, 1993.

[141] M. Slimine-kadi, A. Boubekeur, and G. Saucier, “Interconnection Networks with Fault Tolerance Properties”, *Proc. of Int. Conf. on Wafer Scale Integration*, pp. 213–222, 1993.

[142] B.M. Maziarz and V.K. Jain, “Yield Estimates for the TESH Multicomputer Network”, *Proc. of the 17th Int. Symp. Defect and Fault Tolerance*, pp. 1–9, 2002.

[143] 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.

[144] Selim G. Akl, “Parallel Sorting Algorithms,” *Academic Press*, vol. 11, no. 2, pp.17–37 & 81–108, 1985.

[145] Ding-Ming Kwai and Behrooz Parhami, “Pruned three-dimensional toroidal networks,” *Information Processing Letters*, vol. 68, pp. 179–183, 1998.

[146] Behrooz Parhami and Ding-Ming Kwai, “A Unified Formulation of Honeycomb and Diamond Networks,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 12, no. 1, pp. 74–80, 2001.

[147] Behrooz Parhami and Ding-Ming Kwai, “Incomplete k-ary n-cube and its derivatives,” *Journal of Parallel and Distributed Computing*, vol. 64, no. 2, pp. 183–190, 2004.

[148] Behrooz Parhami and Ding-Ming Kwai, “Comparing Four Classes of Torus-Based Parallel Architectures: Network :Parameters and Communication Performance,” *Mathematical and Computer Modeling*, vol. 40, no. 7-8, pp. 701–720 , 2004.

[149] Ding-Ming Kwai and Behrooz Parhami, “A Class of Fixed-Degree Cayley-Graph Interconnection Networks Derived by Pruning k-ary n-cubes,” *Proc. of the International Conf. on Parallel Processing (ICPP)*, pp. 92–95, 1997.

[150] Joseph Gil and Alan Wagner, “A New Technique for 3-D Domain Decomposition on Multicomputers which reduces Message Passing,” *Proc. of International Parallel Processing Symposium*, Honlulu, pp. 831–835 , 1996.

[151] Behrooz Parhami and Ding-Ming Kwai, “Periodically regular chordal rings,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 10, no. 6, pp. 658–672, 1999.

[152] W.J. Hsu, M.J. Chung, and Z. Hu, “Gaussian networks for scalable distributed systems,” *Computer Journal*, vol. 39, no. 5, pp. 417–426, 1996.

[153] A. Youssef, “Design and analysis of product networks,” *Proceedings of the Symposium Frontiers of Massively Parallel Computation*, pp. 521–528, 1995.

[154] R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Porter eld, and B. Smith, "The tera computer system," *Proceedings of the ACM International Conference on Supercomputing*, pp. 1–6, Amsterdam, 1990.

[155] J. Nguyen, J. Pezaris, G. Pratt, and S. Ward, "Three-dimensional network topologies," *Proceedings of the International Workshop Parallel Computer Routing and Communication*, pp. 101–115, 1994.

[156] L.M. Ni and D.K. Panda, "Sea of Interconnection Networks: What's Your Choice?," *A panel report of International Conference on Parallel Processing (ICPP)*, 1994.

[157] R.V. Boppana, S. Chalasani, and C.S. Raghavendra, "Resource deadlocks and performance of wormhole multicast routing algorithms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 6, pp. 535–549, 1998.

[158] X. Lin and L.M. Ni, "Deadlock-free multicast wormhole routing in multicomputer networks," *Proceedings 18th International Symposium Computer Architecture*, pp. 116–125, 1991.

[159] X. Lin, P. McKinley, and L.M. Ni, "Deadlock-free multicast wormhole routing in 2Dmesh multicomputers," *IEEE Transactions Parallel and Distributed Systems*, vol. 5, no. 8, pp. 793–804, 1994.

[160] M. Malumbres and J. duato, "An efficient implementation of tree-based multicast routing for distributed shared-memory multiprocessors," *Journal of Systems Architecture*, vol. 46, no. 11, pp. 1019–1032, 2000.

[161] P.K. Mckinley, H. Xu, A.H. Esfahanian, and L.M. Ni, "Unicast-based multicast communication in wormhole-routed networks,," *IEEE Transactions Parallel and Distributed Systems*, vol. 5, no. 12, pp. 1254–1265, 1994.

[162] D. Panda, S. Singal, and R. Kesavan, "Multidestination message passing in wormhole kary n-cube networks with base routing conformed paths," *IEEE Transactions Parallel and Distributed Systems*, vol. 10, no. 1, pp. 76–96, 2000.

[163] Hamid Sarbazi-Azad, "Performance Analysis of Wormhole Routing in Multicomputer Interconnection Networks," *Ph.D. Dissertation*, The Faculty of Science, University of Glasgow, 2001.

[164] H.J. Siegel, "Interconnection Networks Large Scale Parallel Processing," *McGraw-Hill*, 1990.

[165] H.J. Siegel and C.B. Stunkel, "Trends in parallel machine interconnection networks," *IEEE Computing in Science and Engineering*, pp. 69–71, 1996.

[166] I. Koren, Z. Koren, and C.H. Stapper, ",A unified negative-binomial distribution for yield analysis of defect-tolerant circuits" *IEEE Trans. on Computers*, vol. 42, no.6, pp. 724–733, 1993.

Publications

- **Journals:**

1. M.M. Hafizur Rahman, Masaru Fukushi, and Susumu Horiguchi, "Reconfiguration and Yield for the HTN: A New Hierarchical Interconnection Network," *International Journal of Embedded Systems*, 2005. (submitted)
2. M.M. Hafizur Rahman and Susumu Horiguchi, "A High Performance Hierarchical Torus Network," *International Journal of High Performance Computing and Networking*, vol.4, no.2, pp. – , 2006.
3. M.M. Hafizur Rahman and Susumu Horiguchi, "Routing Performance Enhancement in Hierarchical Torus Network by Link-Selection Algorithm," *Journal of Parallel and Distributed Computing*, vol.65, no.11, pp.1453-1461, 2005.
4. M.M. Hafizur Rahman, Yasushi Inoguchi, and Susumu Horiguchi, "Modified Hierarchical 3D-Torus Network," *IEICE Transactions on Information and Systems*, vol.E88-D, no.2, pp.177-186, 2005.
5. M.M. Hafizur Rahman and Susumu Horiguchi, "Dynamic Communication Performance of a Hierarchical Torus Network under Non-uniform Traffic Patterns," *IEICE Transactions on Information and Systems*, vol.E87-D, no.7, pp.1887-1896, 2004.
6. M.M. Hafizur Rahman and Susumu Horiguchi, "HTN: A new Hierarchical Interconnection Networks for Massively Parallel Computers," *IEICE Transactions on Information and Systems*, vol.E86-D, no.9, pp.1479-1486, 2003.

- **International Conferences Proceedings:**

1. M.M. Hafizur Rahman and Susumu Horiguchi, "High Performance Hierarchical Torus Network under Matrix Transpose Traffic Patterns" *Proc. of the 7th International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN)* , pp. 111-116, Hong-Kong, PRC, May 10-12, 2004.
2. M.M. Hafizur Rahman and Susumu Horiguchi, "Efficient Applications on High-Performance Network of Hierarchical Torus," *Proc. of the First Int'l Symposium on Towards Peta-Bit Ultra-Networks*, pp.68-77, Ishikawa, Japan, Sept. 08-09, 2003.
3. M.M. Hafizur Rahman and Susumu 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.

4. M.M. Hafizur Rahman and Susumu 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.
5. M.M. Hafizur Rahman, Yasuyuki Miura and Susumu Horiguchi, "Dynamic Communication Performance of Hierarchical Interconnection Network: H3D-Mesh," *Proc. of the 2nd Int'l. Conf. on Electrical and Computer Engineering (ICECE)*, pp.352-355, Dhaka, Bangladesh, Dec. 26-28, 2002.

- **National Conferences Proceedings:**

1. Susumu Horiguchi and M.M. Hafizur Rahman, "階層型トーラスネットワークのデッドロックフリー. ルーティング (Deadlock Free Routing of Hierarchical Torus Network)," *IEICE Technical Report, FIIS-03-115*, Miyazaki, Japan, March 7, 2003.
2. M.M. Hafizur Rahman and Susumu Horiguchi, "Dimension Order Routing on Hierarchical Torus Networks" *Proc. of the JCHC of IEE, Japan*, pp.241, Fukui, Japan, Sept. 18-19, 2002.

Index