Interconnection Networks: Dimensions in Design

Seth Abraham

School of Electrical and Computer Engineering
Purdue University
West Lafayette, Indiana 47907-1285

Abstract

The interconnection network is the switching fabric responsible for providing communication between all processors in a parallel computer. Much research has been directed towards developing superior interconnection networks, but there is no general agreement that this problem is solved. The speakers for this panel session were asked to address the following question: for a given range of number of commodity high-performance processors (e.g. 256 to 1024) what interconnection network should be used to build a general purpose MIMD parallel machine.

1. Introduction

One vital component of a parallel computer's hardware is its interconnection network. The interconnection network is the switching fabric responsible for providing communication between all processors in a parallel computer. The quest for an ideal interconnection networks is an old one, with many networks proposed, and (not as) many networks implemented. Much research effort has been expended, but there is no general agreement that this problem is solved. Three speakers were asked to comment for a panel session on this issue. They are Lionel Ni, from the Michigan State University, [Ni96] Craig Stunkel, from IBM TJ Watson Research Center, [Stun96] and Pen-chung Yew, from the University of Minnesota [HsYe96]. (Pen Yew's co-author is William Tsun-yuk Hsu from San Francisco State University.) The speakers were asked to address the following question: for a given range of number of commodity high-performance processors (e.g. 256 to 1024) what interconnection network should be used to build a general purpose MIMD parallel machine.

When a uniprocessor architect first examines a MIMD parallel computer, the part of the machine that is most alien is the interconnection network. The processors that comprise the machine are not significantly different than uniprocessors with which they are already acquainted. If only a small number of processors are involved, they can be interconnected with a familiar crossbar or high speed shared bus. However, if there are a large number of processors, the choices for interconnect become less clear. For this reason, the panelists have been asked to address their remarks towards the 256 to 1024 processor range.

Historically, many interconnection schemes have been proposed and extensively studied. Their performance has been analyzed with a variety of techniques ranging from simple back-of-the-envelope guesses to highly sophisticated and detailed mathematical models. Simulation performance studies have been conducted under a plethora of conditions, some of them realistic, some of them farfetched. Network behavior in the presence of transient or hard failures has been studied. Switching schemes have been developed and routing techniques devised. A variety of operating conditions and requirements have been proposed, each leading to a custom, and sometimes unique solution for the interconnection problem. Technology issues, packaging issues, and scalability issues have also not escaped notice or attention. Comparative studies, examining the relative merits of two or more design options abound. The literature on the subject is a veritable sea, yet agreement over a broad spectrum of issues is scarce.

Critics of the field are quick to seize upon this last circumstance. They (rightly) point out that many of the proposals are simply not feasible. They also point out that very few of the proposed networks have been implemented. Further, of the few systems that have been implemented, the results have been somewhat disappointing. Generally, the more successful interconnection networks have been the simpler topologies and the smaller sized systems. This has encouraged critics to claim that dreams of large systems and/or sophisticated topologies are impractical fantasies only loosely connected with real world constraints.

One reason for this feeling is that the interconnection networks in implemented systems sometimes seem as if extra hardware has been grafted onto the machine. A large number of sophisticated high performance processors are interconnected with a relatively unsophisticated network. One should realize that providing high performance interconnection is not as mature an endeavor as providing uniprocessor performance. The uniprocessor comes to today's parallel computer as a sophisticated engine that is the product of decades of theoretical study and practical experience. On the other hand, interconnection network designer do not have such an extensive wealth of knowledge to draw upon. Is it any wonder that the interconnection network fares poorly in the comparison?

0190-3918/96 $5.00

Furthermore, the most exciting and visible feature of a parallel supercomputer, (especially to the non-specialist), is the number of processors and the speed of each. With high performance commodity chips so reasonably priced, one is tempted to gather a large number of chips and put them into a single box. Then, one can claim that the new parallel supercomputer has a performance many times greater than previously achieved. In the effort to secure such bragging rights, issues such as network bandwidth, communication latency, programming model, software overhead, granularity of exploitable parallelism, synchronization time, fault tolerance and reliability are relegated to the fine print. These issues may not be very exciting, yet failure to carefully consider them can lead to a machine that is largely unusable.

More fundamentally, harsh "time to market" pressures pose a large obstacle to parallel machine development. The machines that are delivered late are guaranteed to become historical footnotes as conventional uniprocessor performance continues to improve. The pressure that the uniprocessor performance curve brings to bear on parallel computers is not conducive to experimenting with different interconnection solutions. Parallel machines with large numbers of processors require long development times, and machines with small numbers of processors are quickly overtaken by the relentless performance advances of uniprocessors. Until the uniprocessor performance curve flattens, this is unlikely to change. Historically, the performance increases have been delivered by increased clock rates and advances in exploiting instruction level parallelism (ILP). With on chip clock speeds already pushing 400Mhz, one expects the curve to flatten soon. It seems unreasonable to expect more than an order of magnitude improvement in clock speed and an order of magnitude improvement in ILP. With uniprocessor performance doubling every eighteen months, a maximum of two orders of magnitude improvement would imply the performance curve will flatten in about ten years.

In the three workshop papers, the panelists outline many dimensions in the design space for interconnection networks. Many important issues impacting interconnection network design are identified and discussed. Since the single most descriptive characteristic of an interconnection network is its topology, this introduction will describe the major network topology options, with reference to the three papers [HsYe96, Ni96, Stun96]. This will be followed by a summary of such non-topological issues such as network performance, scalability and incremental scalability, fault tolerance and reliability. Section 4 contains a short discussion about issues that impact the interconnection network but are beyond the scope of this panel. Lastly, Section 5 concludes this introduction.

2. Major Network Topologies

One can divide interconnection networks for large numbers of processors into the two major divisions indirect networks and direct networks. (We will omit ATM networks from this classification; however both Stunkel and Ni discuss how the use of ATM networks relate to the more traditional interconnection strategies.) A network is indirect if it is composed of switches connected with links, and a network is considered to be direct if it is constructed with point-to-point links between the processors. This classification is due to Pease [Peas77] with particular reference to the direct binary n-cube (or hypercube), and the indirect binary n-cube (a MIN variant). (The exact relationship between these two networks is explored in [Padm90].) Ni describes a more detailed classification of networks, summarized in Figure 1 of [Ni96]. Stunkel also describes a similar network classification summarized in Figure 1 of [Stun96]. Both authors note that it is possible to have hybrid schemes, yet both refrain from exploring this design option. However, Hsu and Yew [HsYe96] focus exclusively on hierarchical systems that are hybrids.

2.1 Indirect Networks

The most well known indirect network is the multistage network, or MIN. The earliest MINs include the indirect binary n-cube [Peas77], omega [Lawr73] and delta [Pate81], among others. These networks are constructed of stages of smaller switches, with wires connecting the stages together. Figure 1 shows an 8×8 omega constructed from 2×2 switches. These networks all have simple routing, and can be shown to be topologically equivalent to each other. A good summary of MIN properties can be found in [Sieg90].

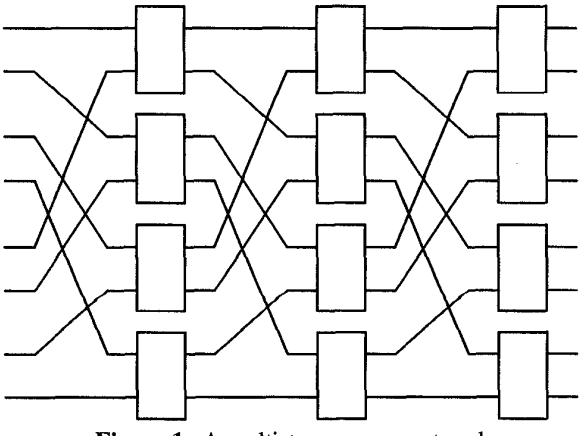

Figure 1. A multistage omega network

While MINs are often thought of as N×N networks ($N=2^n$) comprised of 2×2 switches, this is actually not a requirement. For example, omega networks of size N can be constructed using switch sizes corresponding to any

prime factorization of N [Lawr73]. It is even possible to construct networks for any even number N [Padm91]. While many MINs provide only one path between arbitrary source and destination, one can add redundancy to the network by adding redundant switching stages [AdSi82, PaLa83]

The networks are usually considered to be unidirectional; however, this is also not a requirement. Bidirectional MINs, often called "fat trees", are also an interconnection option. Stunkel describes the use of bidirectional MINs in the IBM SP2 [Stun96], He also elaborates on the practical aspects of implementing and manufacturing a parallel computer and explains why this particular topology was employed in the SP2.

The performance of MIN has been widely studied under both buffered and unbuffered conditions. Unbuffered network performance is well described by Patel [Pate81]; see Kruskal, Snir and Wyss for an excellent treatment of buffered network performance under a wide spectrum of traffic conditions [KrSW88]. There are also traffic conditions known as hot-spots that are particularly detrimental to MIN performance. This issue was first described by Pfister and Norton [PfNo85]. For comparisons of MIN performance to the performance of its direct connected counterpart, the hypercube, see [AbPa89].

2.2 Direct Networks

Direct networks are constructed with point-to-point links between processors. These are sometimes termed router networks although the router used at each processor is not fundamentally different from the switch used in indirect networks. The most widely known direct network topologies are the multi-dimensional meshes and tori, also called called k-ary n-cubes. The simplest of these networks is the one dimensional ring. If one allows every processor to belong to two distinct rings, a two dimensional torus is formed. Figure 2 shows this 2-dimensional structure for a 4×4 array of processors. If the end-around connections are removed, a 2-dimensional mesh results. The number of nodes in each dimension, and the total number of dimensions can be selected as desired. These structures are quite popular as they fit the computation structure of many scientific computations. Most of the implemented systems that use direct interconnection networks have been some sort of multi-dimensional mesh.

The links can be either unidirectional or bidirectional. Unidirectional links utilize a single communication channel between two nodes A and B. Messages can be sent from A to B, but not from B to A. In an N node unidirectional ring, messages may only be sent in one direction, say clockwise, so the average internode distance is N/2. Bidirectional links either require two communication channels so that messages can go from A to B and from B to A; or, they need to time multiplex a single communication channel to allow messages to be sent in both directions. With bidirectional links, an N node ring has average distance of N/4.

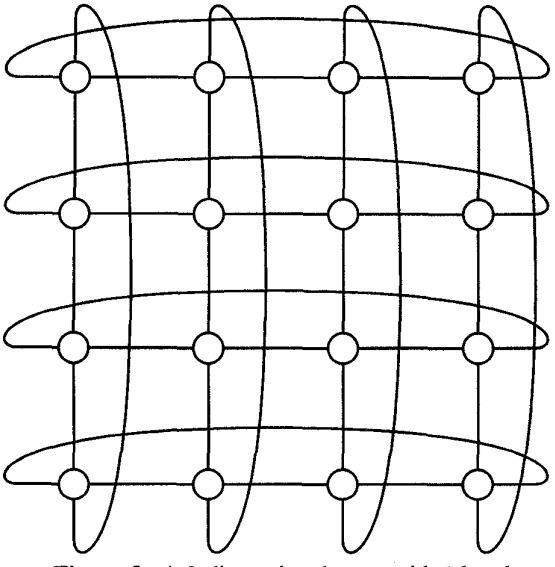

Figure 2. A 2-dimensional torus with 16 nodes

The well known n dimensional hypercube is a special case of a multi-dimensional torus. It can be viewed as either a unidirectional torus with two nodes in each of n dimensions, or it can be viewed as a bidirectional torus with four nodes in each of n/2 dimensions (for even numbers n). Thus, the ring and the hypercube represent the extremes of the multidimensional tori family.

Since the family of multi-dimensional networks, from rings to hypercubes is obviously closely related, it makes sense to study these networks comparatively. Wittie [Witt81] undertook such a study, examining the topology, routing, node degree, graph diameter, and average internode distance. The idea is that node degree is related to the network cost, and that graph diameter (or average internode distance) is related to performance. Some years later, Dally [Dall90] pointed out that the different members of the multi-dimensional torus family have very different implementation costs. Clearly, a ring is much easier to build than a higher dimensional structure such as a hypercube. Thus, to do a fair comparison between these networks, one should allow the lower dimensional structure to have wider communication links (i.e. communication channels with high bandwidth). This allows network performance to be compared between networks of constant cost. Dally chose the network bisection width as his constant cost constraint, and concluded that lower dimensional structures are best. A constant bisection width constraint is a measure of wiring complexity, and is particularly relevant when the network wiring is implemented on a single chip or board. Since that time, other authors have applied different cost constraints. Abraham and Padmanabhan [AbPa90] applied a constant pin-out

constraint which may be a more relevant cost constraint for todays pin limited chips, or in cases where connector costs for cabling between cabinets is a consideration. With such a constraint, higher dimensional networks look attractive again. Other authors have compared the performance of multidimensional tori using both of the above cost constraints, and also considered different wire delay models to account for the large delays required to implement higher dimensional structures. [Dall90, Agar91, ScGo94]. The topology that looks the best is highly dependent on the constraint chosen, and the wire delay model considered. It is difficult to do a meaningful comparison without considering such cost criteria.

Other topologies proposed for interconnection are often motivated by the search for dense graphs. Dense graphs have a large number of nodes in a graph for a given maximum node degree and graph diameter. The notion is that degree is related to network cost, and diameter is related to performance, so the "best" topology should be the one with the most nodes. Using static measures to predict cost and performance is approximate at best— several examples can be provided of networks that look appealing on the basis of these measures, but pale under closer inspection. However, many interesting topologies have been proposed, and Doty [Doty84] lists a number of dense graphs. There are also a host of other topologies that have been proposed for interconnection networks. A small sampling of these include star [AkKr89], pancake [AkKr89], rotator [Corb92], plus a number of hypercube variants [Kats88, HiKS87]. Stunkel [Stun96] comments on a few of these networks and their potential for use as an interconnection structure.

2.3 Hybrid and Hierarchical Networks

The hierarchical networks considered by Hsu and Yew [HsYe96] are essentially hybrid in nature. A good example of a hybrid topology is the cube-connected cycles network [PrVu81]. This topology is formed by replacing each of the hypercube's nodes with a ring of nodes. (Figure 3 shows an example.) For the remainder of this introduction, we will adopt Hsu and Yew's practice of calling these networks hierarchical.

In some sense, hierarchical networks are motivated by forces similar to those that drove the development of memory hierarchies. In the case of memory hierarchies, the speed of different memory technologies motivated the use of multi-level memory hierarchies. For interconnection networks, packaging constraints and technology issues invite the use of different topologies at each level of the hierarchy.

From a hardware design point of view, as the number of processors increases, the media implementing the switching fabric span the spectrum from VLSI routing and switches, to board level routing and possibly multi-chip switching, to backplane based routing, to multi-cabinet (cable) routing. Each movement along this spectrum may represent a change in one or more aspects of implementation technology. For instance, at one end of the spectrum signals are transmitted by VLSI wires, while at the other end signals are transmitted by twisted pair or coaxial cables. Each technology has its own set of fundamental propagation delay limits, basic switching speed limits, and connector costs. Designing a hierarchal system allows the designer the freedom to match a topology to the technology constraints at each level in the hierarchy. This can have a profound impact on system costs, ease of scalability, and even the programming techniques used on the parallel machine. Hsu and Yew develop a framework for characterizing and classifying hierarchical systems, with particular focus on two level systems. This is explained and more fully discussed in [HsYe96].

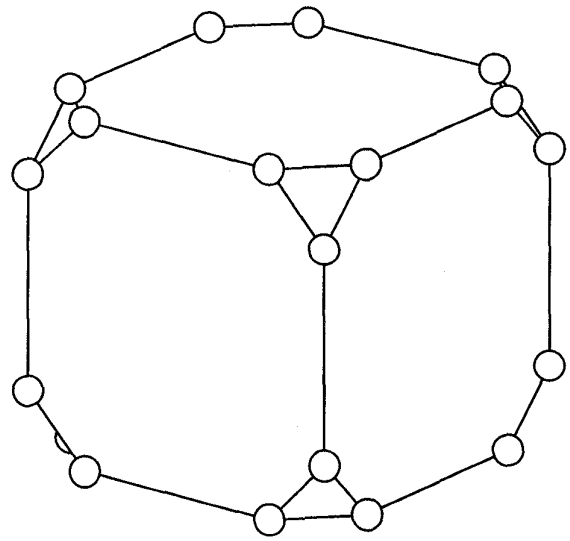

Figure 3. The Cube Connected Cycles Network

3. Important Interconnection Issues

Beyond topology, the panelists identify several other factors that must be considered when building an interconnection network. One important issue is performance. The relationship between performance and topology has already been discussed; however, there are several other components to interconnection network performance. The speed of the interconnect itself is an important factor. This speed depends on the technology and packaging used for implementation, and the width (number of wires) of the communication channel.

There are also several switching techniques that can be employed. The network can be circuit switched or packet switched. If the latter, then schemes can be classified as unbuffered or buffered. The simplest form of a buffered packet switched network is one that uses a store and forward strategy. This means that a packet is

completely received from one communication channel before it is routed to the next communication channel. However, if the routing information arrives before the end of the packet, then one can optimize the switching by making routing decisions before the packet is completely received. Thus, the head of the packet can be forwarded to the next communication channel while the end of the packet is still in transit on the previous channel. Of course, the next communication channel must be available. Kermani and Klienrock [KeKl79] describe this scheme, naming it virtual cut through. One can think of it as operating like circuit switching when the network load is low (and hence, the probability of a desired communication channel being busy is low), and operating like packet switching when the network load is high. When only limited buffer space is available and the packet is forced to remain on the communication channel, the scheme is called worm-hole [DaSe87]. (We note that the phrase "worm-hole routing" is somewhat of a misnomer because it mostly describes switching behavior, not routing behavior.)

For any switching scheme to be effective, it must be coupled with high speed routing that is simple enough to be implemented in a minimum of gate delays. Each topology has its own routing algorithm. If the network has more than one path between source and destination, then routing can either be adaptive or non-adaptive. Stunkel [Stun96] classifies non-adaptive routing algorithms as destination routing, table lookup and source based routing. Both Stunkel [Stun96] and Ni [Ni96] discuss adaptive routing as well.

The panelists also call attention to the issue of scalability, incremental scalability and partitionability. Ni [Ni96] observes that in practical terms, scalability need be considered over a particular range of processors, rather than from 1 to infinity, an observation that significantly simplifies the problem. All three of these issues have broad marketing implications, effecting the ability of customers to upgrade machines, and the ability of manufacturers to offer a wide variety of configurations at different price points.

Fault tolerance and reliability are also extremely important. Parallel computers contain, by their very nature, a large number of components. Since each component has a certain probability of failure, the chance of all components being operational becomes small. When the mean time to failure reduces to a few hours, the system becomes unusable in a production environment. Clearly large parallel processing systems and their interconnect must be capable of continued operation (with reduced performance) in the presence of faulty components. Interconnection networks must make routing and signaling provisions for faults within the switching fabric, as well as faults in the processing elements.

Lastly, interconnection networks may need to support special operations for efficient parallel computing. Examples of such operations may include broadcasts, multi-casts, and synchronization operations. These operations can be made significantly more efficient if they are integrated into the switching fabric. Unfortunately, there is little common agreement on what special operations are important, and what operations are not.

4. Issues Beyond the Scope

The panelists were asked to direct their remarks towards the interconnection network. However, there are several parallel computing issues which impact the interconnection network although they are not, strictly speaking, network issues. Layered on top of the network issues are two additional sets of issues. The first is the physical mechanism that the processor uses to make communication requests, and the second is the method that the application programmer uses to specify a communication request. The former encompasses issues related to the network interface unit, while the latter relates to the parallel programming model for interprocessor communication. In this section, these will be briefly discussed.

Since the fundamental function of the interconnection network is to provide interprocessor communication, it makes sense to briefly examine what sort of communication might be required. There are two dominant paradigms for requesting communication services in parallel applications. One way to request communication is obvious: ask that a message be sent from one process to another. This leads to the message passing programming model, usually implemented through the input/output subsystem of the processors. Typically, it also entails significant software overhead. The second method of communication is an extension of the stored program machine model where the result of one machine language instruction is written to memory and then subsequently read from memory by another machine language instruction. Extending the concept from communication between instructions to communication between processors leads to the shared memory programming model. The shared memory programming model mandates the need for explicit synchronization to insure the correct ordering of memory reads and writes. This appears to be a big advantage for the message passing model; however, it should be noted that the message passing model has synchronization implied by the receipt of messages. In message passing, the coordination of a message receipt with the application program read is typically accomplished by exploiting the (slower) synchronization provided by the operating system and I/O device drivers.

While some interconnection network topologies have been associated with one programming model or another, there is really no fundamental reason for this association. One should not confuse the interconnection network with the programming model. Usually, the most natural and

efficient parallel programming model (message passing or shared memory) for a machine will be determined by the network interface. When the network interface looks like an I/O device to the processor, then message passing is likely to be the favored communication mode. When the network interface looks like memory to the processor, then shared memory is likely to be favored. Of course, the system software can support any desired programming model, regardless of what the network interface supports.

However, the two programming models do pose a different set of requirements and demands upon the interconnection network. In general, a shared memory programming model entails greater cost for the network interface and must provide for hardware synchronization. This will complicate the design of the network interface, or may rule out the use of commodity processors altogether. The shared memory programming model allows the exploitation of fine-grain parallelism, and may require the interconnection network to provide low latency communication, with small packet sizes. If the network efficiency decreases with small packet sizes, then a shared memory programming model will be unsuitable if single word reads and writes are envisioned, but allowable if cache blocks are the unit of transfer. If a cache strategy is employed on a shared memory machine to hide network latency and increase packet size, then cache coherence may need to be assured. However cache coherence has severe network performance implications [GuAb95], and this further complicates the design of the network interface.

Message passing generally means a simpler network interface, but implies significant software overhead for message handling. Often, the interconnection network has a lower performance (i.e. higher communication latency). This situation discourages the exploitation of fine-grain parallelism. Communication traffic is likely to have a fewer number of (multi-word) packets. Packets may be of variable (and potentially large) size, further complicating interconnection network design.

Currently, there are no universal parallel programming models, and there is little agreement about what granularity of parallelism is desirable or feasible. In such a climate, researchers regularly experiment by modifying their programming style. Further, application programmers constantly tune their codes to match the properties of the interconnection network in their machine. This may lead to higher performance *on a particular machine* at the expense of the parallel programming paradigm. Machine architects must be on guard when they examine the style of application programmers to determine how to build a parallel computer. Is this the way the programmer *wanted* to program, or is this the way the performance minded programmer was *forced* to program? Failure to analyze this situation accurately can be quite harmful to the study of parallel processing.

5. Summary

In summary, the interconnection network is a major component of an MIMD parallel computer. The choice of interconnect has serious consequences for how a parallel processor will be used, and how it will perform. Many designs have been proposed to solve the interconnection problem, and many factors must be considered. Clearly, there is still a need for further study.

References

[AbPa89] S. Abraham and K. Padmanabhan, "Performance of the Direct Binary n-Cube Network for Multiprocessors," *IEEE Transactions on Computers,* (July 1989), Vol. 38, No. 7, pp. 1000-1011.

[AbPa90] S. Abraham and K. Padmanabhan, "Performance of Multicomputer Networks under Pin-out Constraints," *Journal of Parallel and Distributed Computing,* (July 1991), Vol. 12, No. 3, pp. 237-248.

[AdSi82] G. B. Adams III and H. J. Siegel, "The Extra Stage Cube: A Fault-tolerant Interconnection Network for Supersystems," *IEEE Transactions on Computers,* (May 1982), Vol. C-31, No. 5, pp. 443-454.

[Agar91] A. Agarwal, "Limits on Interconnection Network Performance," *IEEE Transactions on Parallel and Distributed Systems* (Oct. 1991), Vol. 2, pp. 398-412.

[AkKr89] S. B. Akers and B. Krishnamurthy. "A Group-Theoretic Model for Symmetric Interconnection Networks," *IEEE Transactions on Computers* (Apr. 1989), Vol. 38, No. 4, pp. 555-566.

[Corb92] P. F. Corbett. "Rotator Graphs: An Efficient Topology for Point-to-Point Multiprocessor Network," *IEEE Transactions on Parallel and Distributed Systems* (Sep. 1992), Vol. 3, pp. 622-626.

[DaSe87] W. J. Dally and C. L. Seitz. "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Computers* (May 1987), Vol. C-36, No. 5, pp. 547-553.

[Dall90] W. J. Dally, "Performance Analysis of k-ary n-cube Interconnection Networks", *IEEE Transactions on Computers* (Jun. 1990), Vol. 39, pp. 775-785.

[Doty84] K. W. Doty. "New Designs for Dense Processor Interconnection Networks," *IEEE Transactions on Computers* (May 1984), Vol. C-33, No. 5, pp. 447-450.

[GuAb95] S. Gupta and S. Abraham, "A Distributed Directory Cache Coherence Scheme and its Effects on Network Performance," *Journal of High Performance Computing,* (Nov/Dec

1995) Vol. 2, No. 1, pp. 3-16.

[HiKS87] P. A. J. Hilbers, M. R. J. Koopman and J. L. A. van de Snepscheut. "The Twisted Cube," *Parallel Architectures and Languages Europe, Lecture Notes in Computer Science* (Jun. 1987), pp. 152-159.

[HsYe96] W. T. Hsu and P. C. Yew, "An Introduction to Hierarchical Network Topologies," *1996 ICPP Workshop on Challenges for Parallel Processing* Aug. 1996.

[Kats88] H. P. Katseff, "Incomplete Hypercubes," *IEEE Transactions on Computers* (May 1988), Vol. 37, No. 5, pp. 604-608.

[KeKl79] P. Kermani and L. Kleinrock. "Virtual Cut-Through: A New Computer Communication Switching Technique," *Computer Networks*, (1979), Vol. 3, pp. 267-286.

[KrSW88] C. P. Kruskal, M. Snir and A. Weiss. "The Distribution of Waiting Times in Clocked Multistage Interconnection Networks," *IEEE Transactions on Computers* (Nov. 1988), Vol. C-37, No. 11, pp. 1337-1352.

[Lawr73] D. H. Lawrie. "Memory-processor Connection Networks," Ph.D. Thesis, University of Illinois at Urbana-Champaign, Feb. 1973.

[Ni96] L. M. Ni, "Issues in Designing Truely Scalable Interconnection Networks," *1996 ICPP Workshop on Challenges for Parallel Processing* Aug. 1996.

[PaLa83] K. Padmanabhan and D. H. Lawrie. "Fault Tolerance Schemes in Shuffle-Exchange Type Interconnection Networks," *1983 International Conference on Parallel Processing* (Aug. 1983), pp. 71-75.

[Padm90] K. Padmanabhan. "Cube Structures for Multiprocessors," *Communications of the ACM* (January 1990), Vol. 33, No. 1, pp. 43-52.

[Padm91] K. Padmanabhan. "Design and Analysis of Even-Sized Binary Shuffle-Exchange Networks for Multiprocessors," *IEEE Transactions on Computers* (Oct. 1991), Vol. 2, No. 4, pp. 385-397.

[Pate81] J. H. Patel. "Performance of Processor-Memory Interconnections for Multiprocessors," *IEEE Transactions on Computers* (Oct. 1981), Vol. C-30, No. 10, pp. 771-780.

[Peas77] M. C. Pease III. "The Indirect Binary n-Cube Microprocessor Array," *IEEE Transactions on Computers* (May 1977), Vol. C-26, No. 5, pp. 458-473.

[PfNo85] G. F. Pfister and V. A. Norton. "Hot Spot Contention and Combining in Multistage Interconnection Networks," *IEEE Transactions on Computers* (Oct. 1985), Vol. C-34, No. 10, pp. 943-948.

[PrVu81] F. Preparata and J. Vuillemin, "The Cube-connected Cycles: A Versatile Communication Network for parallel Computation," *Communications of the ACM* (1981), Vol. 24, No. 5, pp. 300-309.

[ScGo94] S. L. Scott and J. R. Goodman, "The Impact of Pipelined Channels on k-ary n-cube Networks," *IEEE Transactions on Parallel and Distributed Systems* (Jan. 1994), pp. 2-16.

[Sieg90] H. J. Siegel, *Interconnection Networks for Large-Scale Parallel Processing: Theory and Case Studies,* 2nd Edition, McGraw-Hill, New York, NY, 1990.

[Stun96] C. B. Stunkel "Commercially Viable MPP Networks," *1996 ICPP Workshop on Challenges for Parallel Processing* Aug. 1996.

[Witt81] L. D. Wittie. "Architectures for Large Networks of Microcomputers," *IEEE Transactions on Computers* (Apr. 1981), Vol. C-30, pp. 264-273.