SURVEY ON INTERCONNECTION NETWORKS

WORK DONE AS PART OF

PARALLEL ARCHITECTURE SYSTEMS

UNDER THE GUIDANCE OF

DR. EDWIN SHA

BY

GOMATHY GOWRI NARAYANAN
KARTHIK ALAGU

TABLE OF CONTENTS

Abstract

As processors continue to become faster and more machines are built using relatively large number if processors, buses are becoming less able to provide the required bandwidth for such machines, the solution is to use *Interconnection Networks* instead of buses. Basically, Interconnection Networks originated from the design of the high performance parallel computers. Interconnection networks are becoming increasingly pervasive in many different applications, with the operational costs and characteristics of these networks considerably depending on the application. They can be categorized according to a number of criteria such as topology, routing strategy and switching technique. In this paper we mainly concentrate on the Dynamic Interconnection Networks and more importantly on the Multistage Interconnection Networks. Towards the end of the paper we look into the tradeoffs and comparisons between the different types of interconnection networks.

Keywords: *Interconnection Network, Static Interconnection Networks, Dynamic Interconnection Networks, Multistage Interconnection Networks*

1. Introduction

Interconnection networks connect processors and memory banks together. The need for more network bandwidth has put them to use in network switches and they may even be used to connect peripherals to computers. The interconnection network plays a central role in determining the overall performance of a multi-computer system. Interconnection networks are becoming increasingly pervasive in many different applications, with the operational costs and characteristics of these networks considerably depending on the application. If the network cannot provide adequate performance, for a particular application, nodes will frequently be forced to wait for data to arrive. Interconnection Networks are used to route the packets appropriately depending on the application and needs of the particular network. Interconnection networks make a major factor to differentiate modern multiprocessor architectures. They can be categorized according to a number of criteria such as topology, routing strategy and switching technique. In this paper we mainly concentrate on the Dynamic Interconnection Networks. We are going to see a detailed survey on the Dynamic Interconnection Networks and the tradeoffs and advantages of each of them over the other. This comparative study would make the understanding of the Interconnection Networks better and help us determine which Interconnection Network suits the given application.

2. Interconnection networks

Processors and memory banks are connected by *interconnection networks*. The interconnection network plays a central role in determining the overall performance of a multi-computer system. A parallel system consists of a number of processing elements (PE) that are arranged in a configuration that is characterized by an interconnection network. Communication between PE's is accomplished by means of the interconnection network, which should not only provide a short path but also accommodate the communication needs of the application. A number of interconnection network topologies have been proposed and used; each network has features that make it suitable for a set or applications and algorithms.

One way to view the physical structure of an interconnection network is as a set of N processing elements interconnected by a network and fed instructions by a control unit. Each *processing element* (PE) consist of a processor with its own memory is called a *processing element to processing element* organization. The network is unidirectional and connects each PE to all or some subset of the other PEs. A transfer instruction moves data from each PE to one or more of the PEs to which that PE is connected by the network. To move data between two PEs that are not directly connected, the data must be passed through intermediary PEs by executing a programmed sequence of data transfers.

An alternative is to position the network between the processors and the memories. This type of configuration is called *processor-to memory* organization. In this case the interconnection network is bidirectional – it both connects each processor to all or some sunset of memories, as well as each memory to all or some subset of processors. A transfer instruction results in data being moved from each processor to one or more of the memories to which that processor is connected by the network, or from each memory to one or more of the processors to which that memory is connected by the network. One processor can transfer data to another processor through any memory to which both are connected. To pass data between two processors, a programmed sequence of data transfers must be executed. This sequence of transfers moves the data from one processor to the other by passing the data through intermediary memories.

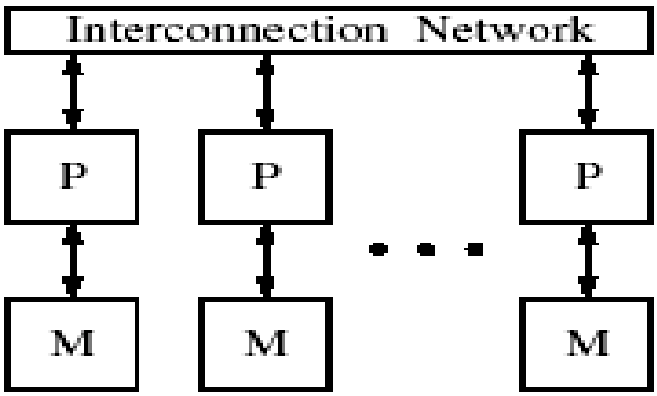


Fig 1: A Multicomputer

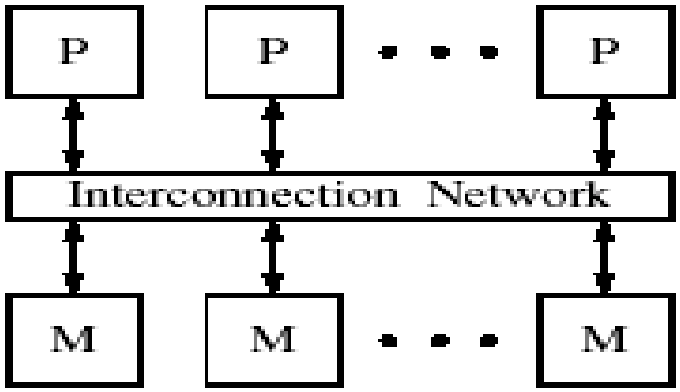


Fig 2: A UMA shared-memory multiprocessor

An interconnection network node has a router that provides means of handling messages on the network. The router receives, forwards and delivers messages as well as controlling message flow through the network. The router system transfers messages from its input port to the proper output ports based on the routing algorithm.

Interconnection networks are built up of switching elements; topology is the pattern in which the individual switches are connected to other elements, like processors, memories and other switches.

2.1 Switches

Traditional Interconnection Networks were limited by pin counts on chips. However, in on-chip networks wires can be packed so densely that this is no longer an issue. Instead we would like the area of the switch to be small, to give more area for actual computation. Buffer space is a critical part of the area taken by a switch, and as such of the critical interest for an on-chip network. The function of the switch is to take packets arriving at its input ports and route them to its output ports. As long as only one packet at a time arrives for a given output port, there will be no conflict, and the packets are routed with minimum latency. Unfortunately, as the throughput goes

up, so does the probability of conflict. When two packets destined for the same output port arrive at different ports of the switch at approximately the same time, they cannot both be forwarded immediately. Only one packet can be transmitted from an output port at a time, and hence one of the two packets is stored at the node for later transmission. The maximum throughput at which a switch can operate depends [i] directly on how efficient the switch can operate [ii] depends directly on how efficient the switch at storing the conflicting packets and forwarding them when the appropriate output port is no longer busy. An ideal switch has infinite buffer space, but will only buffer (delay) a packet as long as the output port to which the packet is destined is busy.

3. Requirements on interconnection networks

An Interconnection network should transfer a **maximum** number of messages in the **shortest time** with **minimum cost** and **maximal reliability**. Clearly, any design of an interconnection network is a tradeoff of various **contradictory requirements**. The most important requirements are the following:

3.1 Small diameter and small average distance

Small average distance allows small communication **latency**, especially for **distance sensitive** routing, such as **store-and-forward**. But it is also crucial for **distance insensitive** routing, such as **wormhole routing**, since short distances imply less used links and buffers, and therefore less communication **contention**.

3.2 Small and fixed vertex degree

Small and constant vertex degree allows simple and low-cost universal (i.e., independent of the network size) routers which amortizes the design costs. On the other hand, it implies less links and lower connectivity and larger distances. Ideally, we would like to get **low constant vertex degree** and at most **logarithmic diameter** simultaneously. Given N-vertex graph such that *triangle (G)* $\leq k$ for some constant k, the number of vertices reachable in G within i steps from any vertex is $O(k^i)$. Hence, *N=O (k^diam (G))*, which is equivalent to *diam (G) =Omega (log N)*. Fixed-degree networks cannot have better than logarithmic diameter.

3.3 Large bisection width

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

3.4 High connectivity and fault tolerance

The network should provide **alternative paths** for delivering messages in case of link or node **faults** or **communication congestions**. **Large packets** can be delivered faster if they can be **split** into smaller chunks sent along disjoint paths.

3.5 Small fault average distance and diameter

To have these alternatives or parallel disjoint paths as short as possible, small fault average distances and small fault diameter are naturally desirable.

3.6 Hamiltonianity

The existence of at least a Hamiltonian path is not the most important requirement, but it is useful whenever we need to label processors with numbers *1,..., p* so that adjacent processors get successive numbers (for example in sorting algorithms).

3.7 Hierarchical recursivity

Interconnection networks are usually defined using some independent parameters, called **dimensions**. The set of all graphs of different dimensions based on a given definition forms a **topology**. A topology is **hierarchically recursive** if a higher-dimensional graph contains lower-dimensional graphs with the same topology as sub graphs. Many Interconnection networks used in parallel systems are hierarchically recursive; topologies based on Cartesian product are one example. Recursiveness makes the design and technology of manufacturing interconnection networks easier. Large scale problems that can be solved in parallel by recursive decomposition into smaller sub problems can be easily and efficiently mapped on the hierarchically recursive topologies using induction.

3.8 Incremental extendibility and incremental scalability

If the definition of the topology allows graphs of any size, the topology is said to be **incrementally extendable**. There are incrementally extendable topologies, and some are at least partially extendable, since they allow size granularity only greater than 1. Some hierarchically recursive topologies allow graphs of specific discrete sizes, such as powers of two. If a topology is incrementally extendable, a very important question is how the structure of an instance of size n differs from the structure of an instance of size $n+k$ for some integer constant $k \geq 1$. An $(n+k)$-vertex instance can be obtained from a n-vertex one by removing $r(k)$ edges (to get a sub graph of the larger instance) and by adding additional vertices and corresponding edges to this sub graph. If $r(k) = O(k)$, the topology is said **incrementally scalable**. Very few topologies are incrementally scalable. For example, 2-D meshes are incrementally extendable, but not scalable.

3.9 Symmetry

This is a very important requirement. Many interconnection network topologies are vertex or edge symmetric. Intuitively, a symmetric network is easy to understand and the design of parallel and communication algorithms is very much easier, since it is irrelevant where the computation and/or communication starts or in which directions it will evolve. Also, the symmetry is helpful for solving issues related to VLSI design.

3.10 Support for routing and collective communication

The network topology should enable a simple shortest path routing, so that the basic routing algorithm could be implemented in hardware. Equally important parameters of an interconnection

network are complexities of **permutation routing** and **one-to-all** and **all-to-all** communication operations. The design of efficient communication algorithms is very simplified if the topology is symmetric and it also depends on the communication technology and router architecture.

3.11 Embeddability of and into other topologies

The efficiency of a parallel algorithm running on a parallel machine depends on the similarity between the process graph and the underlying physical interconnection network topology. We need a suitable mapping or **embedding** of the process graph into the topology. Desirable are those topologies which are able to simulate efficiently other topologies.

3.12 Simple VLSI or 3-D layout

Any VLSI implementation of an interconnection network implies VLSI-related requirements, such as easy mapping to rectangular grid, decomposition of a large interconnection network into building blocks so that the lengths and numbers of inter-chip wires is minimal (recall the bisection width above).

4. Network Design Considerations

Interconnection networks play a major role in the performance of modern parallel computers. There are many factors that may affect the choice of an appropriate interconnection network for the underlying parallel computer. These factors include the following:

4.1 Performance requirements

Processes executing in different processors synchronize and communicate through the interconnection network. These operations are usually performed by explicit message passing or by accessing shared variables. Message *latency*

locations. Also, the network may *saturate*—it may be unable to deliver the flow of messages injected by the nodes, limiting the effective computing power of a parallel computer. The maximum amount of information delivered by the network per time unit defines the *throughput*

4.2 Scalability

lies that as more processors are added, their memory bandwidth, I/O bandwidth, and network bandwidth should increase proportionally. Otherwise the components whose bandwidth does not scale may become a bottleneck for the rest of the system, decreasing the overall efficiency accordingly.

4.3 Incremental expandability

Customers are unlikely to purchase a parallel computer with a full set of processors and memories. As the budget permits, more processors and memories may be added until a system's maximum configuration is reached. In some interconnection networks, the number of processors must be a power of 2, which makes them difficult to expand. In other cases, expandability is

provided at the cost of wasting resources. For example, a network designed for a maximum size of 1,024 nodes may contain many unused communication links when the network is implemented with a smaller size. Interconnection networks should provide incremental expandability, allowing the addition of a small number of nodes while minimizing resource wasting.

Parallel computers are usually shared by several users at a time. In this case, it is desirable that the network traffic produced by each user does not affect the performance of other applications. This can be ensured if the network can be partitioned into smaller functional subsystems. Partitionability may also be required for security reasons.

4.5 Simplicity

Simple designs often lead to higher clock frequencies and may achieve higher performance. Additionally, customers appreciate networks that are easy to understand because it is easier to exploit their performance.

4.6 Distance span

This factor may lead to very different implementations. In multicomputers and *distributed shared-memory multiprocessors (*DSMs), the network is assembled inside a few cabinets. The maximum distance between nodes is small. As a consequence, signals are usually transmitted using copper wires. These wires can be arranged regularly, reducing the computer size and wire length. In *Networks of Workstations* (NOWs), links have very different lengths and some links may be very long, producing problems such as coupling, electromagnetic noise, and heavy link cables. The use of optical links solves these problems, equalizing the bandwidth of short and long links up to a much greater distance than when copper wire is used. Also, geographical constraints may impose the use of irregular connection patterns between nodes, making distributed control more difficult to implement.

4.7 Physical constraints

4.8 Reliability and repairability

4.9 Expected workloads

Users of a general-purpose machine may have very different requirements. If the kind of applications that will be executed in the parallel computer is known in advance, it may be possible to extract some information on usual communication patterns, message sizes, network load, and so on. That information can be used for the optimization of some design parameters. When it is not possible to get information on expected workloads, network design should be robust; that is, design parameters should be selected in such away that performance is good over a wide range of traffic conditions.

4.10 Operation mode

Two types of communication can be identified: synchronous and asynchronous. Synchronous communication is needed for processing in which communication paths are established synchronously for either a data manipulating function or a data/instruction broadcast. Asynchronous communication is needed for multiprocessing in which connection requests are issued dynamically. A system may also be designed to facilitate both synchronous and asynchronous processing. Therefore, typical operation modes of interconnection networks can be classified into three categories: synchronous, asynchronous and combined

4.11 Control strategy

A typical interconnection network consists of a number of switching elements and interconnecting links. Interconnection functions are realized by properly setting control of the switching elements. The control-setting function can be managed by a centralized controller or by the individual switching element. The latter strategy is called *distributed control*; the first strategy is called *centralized control*.

4.12 Switching methodology

The two major switching methodologies are *circuit switching* and *packet switching*. In circuit switching, a physical path is actually established between a source and a destination. In packet switching, data is put in a packet and routed through the interconnection network without establishing a physical connection path. In general, circuit switching is much more suitable for bulk data transmission, and packet switching is more efficient for short data messages. Another option, integrated switching, includes capabilities of both circuit switching and packet switching. Therefore, three switching methodologies can be identified: circuit switching, packet switching, and integrated switching.

Cost constraints

Design decisions often are trade-offs between cost and other design factors. Fortunately, cost is not always directly proportional to performance. Using commodity components whenever possible may considerably reduce the overall cost.

In the rest of the section, we give a survey of types and properties of several well-known topologies for interconnection networks. Some topologies are understood better than some others so that the description of properties will vary for various topologies.

5. Metrics for Interconnection Networks

Metrics provide a framework to compare and evaluate interconnection networks. A topology is evaluated in terms of a number of parameters. The metrics we have studied are:

- Network connectivity
- Network diameter
- Narrowness
- Network expansion increments

5.1 Network Connectivity

Network nodes and communication links sometimes fail and must be removed from service for repair. When components do fail the network should continue to function with reduced capacity.

Network connectivity measures the resiliency of a network and its ability to continue operation despite disabled components i.e. connectivity is the minimum number of nodes or links that must fail to partition the network into two or more **disjoint networks**. The larger the connectivity for a network the better the network is able to cope with failures.

5.2 Network Diameter

The diameter of a network is the maximum inter-node distance i.e. it is the maximum number of links that must be traversed to send a message to any node along a shortest path. 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.

5.3 Narrowness

This is a measure of congestion in a network and is calculated as follows: Partition the network into two groups of processors A and B where the number of processors in each group is N_a and N_b and assume $N_b \leq N_a$. Now count the number of interconnections between A and B call this I. Find the maximum value of N_b / I for all partitioning of the network. This is the narrowness of the network.

The idea is that if the narrowness is high (N_b > I) then if the group B processors want to send messages to group A congestion in the network will be high (since there are fewer links than processors)

5.4 Expandability

A network should be expandable i.e. it should be possible to create larger and more powerful multi computer systems by simply adding more nodes to the network. For reasons of cost it is better to have the option of small increments since this allows you to upgrade your network to the size you require (i.e. flexibility) within a particular budget. For example, an 8 node linear array can be expanded in increments of 1 node but a 3 dimensional hypercube can be expanded only by adding another 3D hypercube. (i.e. 8 nodes)

6. Types of Interconnect Networks

Interconnection Networks are characterized based on a number of different characteristics. A few of them are as follows.

- Topology
- Switching strategy
- Routing algorithm
- Flow control mechanism
- Queuing techniques

6.1 Topology

This says **what** the type of the Interconnection network is. There are two types of networks based on the topology. They are Direct and Indirect. *Direct* topologies connect each switch directly to a node, while in *indirect* topologies at least some of the switches connect to other switches.

6.2 Switching strategy

The switching strategy shows **how** the data in a message traverses a route. Using switching technique as a criterion one can mention some classes:

- *circuit switching*, in which the entire path through the network is reserved before a message is transferred,
- *packet switching* with *virtual cut through*, in which a packet is forwarded immediately after it determines an appropriate switch output,

6.3 Routing Algorithm

The routing algorithm shows **which** path the data follows to reach the destination. These routing algorithms restrict the set of paths that a message may follow to reach its destination. Routing could be either *deterministic* or *adaptive*. In *deterministic routing* the communication path is completely determined (a priori) by the source and destination addresses. In *adaptive routing* the path depends on network conditions. There are several types of algorithms with different properties.

- *Store and Forward Routing*, where each intermediate processor on the path forwards the message to the next processor after it has received and has stored in entire message. In this type of routing the communication time is high and there is a very poor utilization of communication resources.
- *Cut-Thorough Routing,* where message is advanced to the outgoing link as it is received from the incoming link. In this type of routing, the messages travel in small units or flow control digits called *flits.* An intermediate processor does not wait for the entire message to arrive before forwarding it. As and when the flits arrive they are forwarded to the next processor. All flits however follow the same path. There is thus no need for any buffer space to store the message. A *Wormhole routing* is essentially a cut-through routing which relaxes requirements of completely buffering of blocked packets in a single switch, typical for *packet switching*. Wormhole routing is currently the most popular technique in commercial parallel machines.

6.4 Flow Control Mechanism

The flow control mechanism determines **when** a message or portions of it traverse a route. It also determines what happens when traffic is encountered. Any solution must adjust to output rate.
Some of the flow control techniques are as follows:

- Ethernet: collision detection and retry after delay
- FDDI, token ring: arbitration token
- TCP/WAN: buffer, drop, adjust rate

6.5 Queuing Techniques

Queuing technique is another important factor in switching network architectures, since it strongly influences the aggregated bandwidth of the network. In the simple *input queuing* technique the packets queue at the switch input awaiting the availability of the desired switch output; higher performance is offered by *output queuing* which in turn is difficult to implement. A solution is to form multiple queues at each switch input or to create a central buffer shared among all switch inputs and outputs.

7. Classification of Interconnection Networks

Among other criteria, interconnection networks have been traditionally classified according to the operating mode (synchronous or asynchronous) and network control (centralized, decentralized, or distributed). Nowadays, multicomputers, multiprocessors, and NOWs dominate the parallel computing market. All of these architectures implement asynchronous networks with distributed control. Therefore, we will focus on other criteria that are currently more significant. A

classification scheme is shown in Fig 3, which categorizes the known interconnection networks into four major classes based primarily on network topology: shared-medium networks, direct networks, indirect networks, and hybrid networks. For each class, the figure shows a hierarchy of subclasses, also indicating some real implementations for most of them. This classification mainly focuses on networks that have been implemented. It is by no means complete, as other new and innovative interconnection networks may emerge as technology further advances, such as mobile communication and optical interconnections.

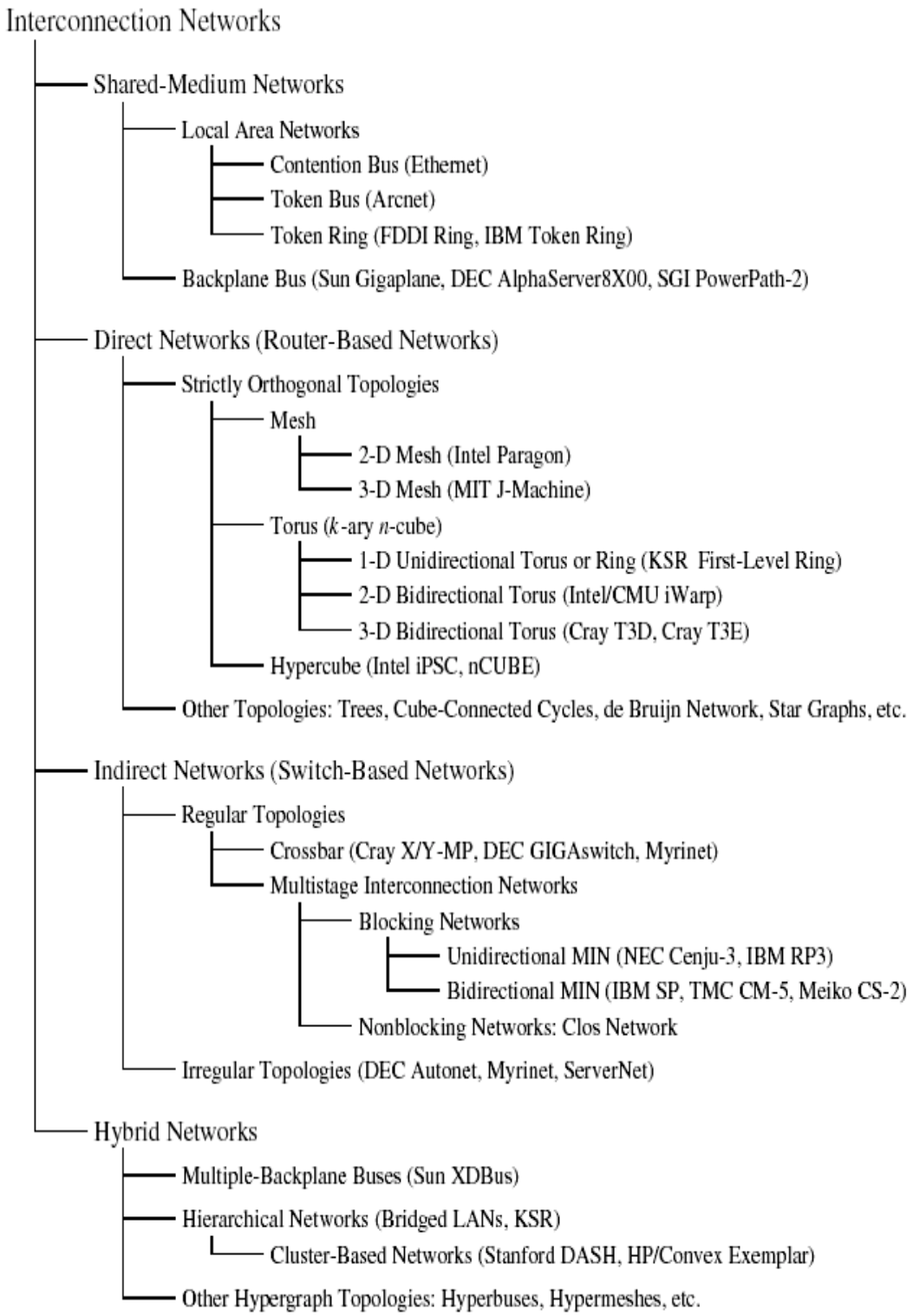


Fig 3: A broad classification of Interconnection Networks

In our project we study mainly on the two categories: *Direct* ***and*** *indirect* interconnection networks and do a detailed survey on indirect interconnection networks. Indirect or dynamic connection means the path between two entities (PE to memory or PE to PE) may change from one communication to next; Direct or static connection means the network topology stays the same all the time. Static interconnection networks are mainly used in message-passing architectures. We shall look in detail about these two types of networks.

7.1 Static Interconnection Networks

Processors are connected directly to other processors via point to point communication links. Static connection means the network topology stays the same all the time and does not change. In a static network the nodes are comprised of computers. Static interconnection networks are mainly used in message-passing architectures. The following are the different types of static interconnection networks.

- *Completely-connected network*. The network is a completely connected network. Each processor is connected to all the other processors in the networks. This type of network has good connectivity but expensive to build and to manage.

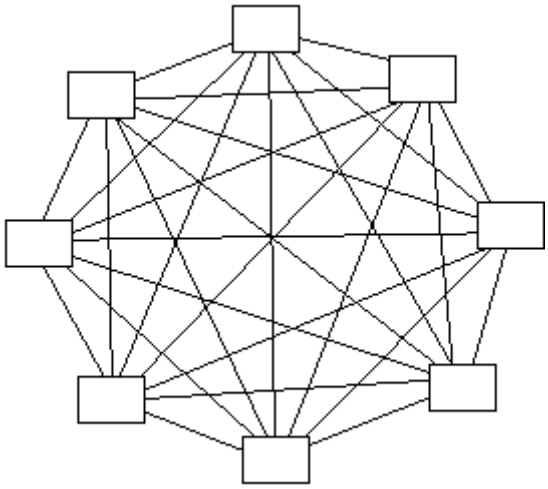


Fig 4: A Completely Connected network

- *Star-connected network*. The network connection follows a star pattern. This type of network may have a control bottle-neck. This network has relatively good connectivity.

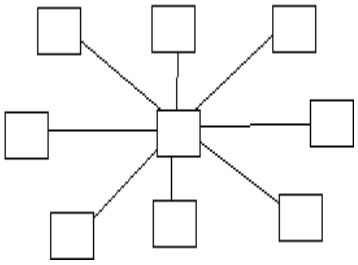


Fig 5: A Star Connected network

- *Linear array* or *ring* of processors. In this type of network any PE can directly communicate with its two neighbors, messages to any other PEs have to go through multiple PEs in-between. A ring is just a variation of linear array where the two end PEs are directly connected.

Fig 6: Linear array

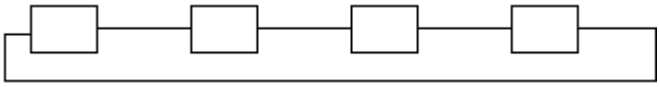


Fig 7: A Ring network

- *Mesh network* (in 2- or 3D). Each processor has a direct link to four/six (in 2D/3D) neighbor processors. Extension of this kind of networks is a *wraparound mesh* or *torus*. Commercial examples are Intel Paragon XP/S and Cray T3D/E. These examples cover also another class, namely *the direct network* topology.

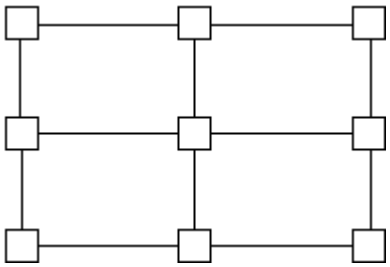


Fig 8: A Mesh network

- *Tree network* of processors. Communication bottleneck likely to occur in large configurations can be alleviated by increasing the number of communication links for processors closer to the root, which results in the *fat-tree* topology, efficiently used in the TMC CM5 computer. CM5 could be also an example of *indirect network* topology.

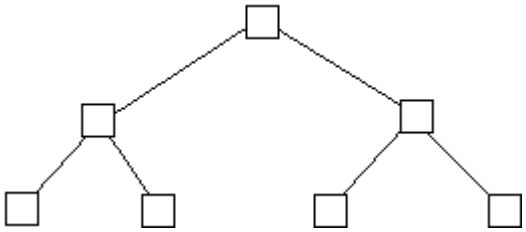


Fig 9: A tree network

- *Hypercube network*. Classically this is a multidimensional mesh of processors with exactly two processors in each dimension. An example of such a system is the Intel iPSC/860 computer. Some new projects incorporate the idea of several processors in each node which results in *fat hypercube*, i.e. *indirect network* topology. An example is the SGI/Cray Origin2000 computer.

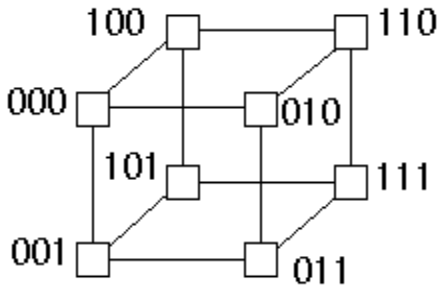


Fig 10: A 3D hypercube network

7. 11 Examples

listed below.

- Cray T3E: Bidirectional 3-D torus, 14-bit data in each direction, 375 MHz link speed, 600 Mbytes/s per link.
- Cray T3D: Bidirectional 3-D torus, up to 1,024 nodes (8 ×16 × 8), 24-bit links (16-bit data, 8-bit control), 150 MHz, 300 Mbytes/s per link.
- Intel Cavallino: Bidirectional 3-D topology, 16-bit-wide channels operating at 200 MHz, 400 Mbytes/s in each direction.
- SGI SPIDER: Router with 20-bit bidirectional channels operating on edges, 200 MHz clock, aggregate raw data rate of 1 Gbyte/s. Support for regular and irregular topologies.
- MIT M-Machine: 3-D mesh, 800 Mbytes/s for each network channel.
- MIT Reliable Router: 2-D mesh, 23-bit links (16-bit data), 200 MHz, 400 Mbytes/s per link per direction, bidirectional signaling, reliable transmission.
- Chaos Router: 2-D torus topology, bidirectional 8-bit links, 180 MHz, 360 Mbytes/s in each direction.
- Intel iPSC-2 Hypercube: Binary hypercube topology, bit-serial channels at 2.8 Mbytes/s.

7. 2 Dynamic Interconnection Networks

Processors are connected dynamically via switches or buses. It is usually associated to shared address space architectures. Dynamic network allow the interconnection network to be reconfigured during operation to allow various connections (routing) through the use of switches. In a dynamic network the nodes are comprised of switching elements. The cost of the network is thus measured by the number of switch boxes required. Dynamic connection means the path between two entities (PE to memory or PE to PE) may change from one communication to next. Dynamics interconnection networks implement one of four main alternatives.

7.21 Bus-based networks

Processors are connected to memory through a common data path. The simplest and efficient solution when the cost and moderate number of processors are involved. Its main drawback is a bottleneck to the memory when number of processors becomes large and also a single point of failure. To overcome the problems, sometimes several parallel buses are incorporated. The classical example of such machine is the SGI Power Challenge computer with packet data bus.

7.22 Crossbar switching networks

These are networks which employ a grid of switching elements. It is a digital analogous of a switching board. The network is non-blocking, since the connection of a processor to a memory bank does not block the connection of any other processor to any other memory bank. In spite of high speed, their use is limited, due to nonlinear complexity ($o(p^2)$, p - number of processors) and the cost. They are applied mostly in multiprocessor vector computers (like Cray YMP) and in multiprocessors with multilevel interconnections (e.g. HP/Convex Exemplar SPP). One outstanding example is the Fujitsu VPP500 which incorporates 224x224 crossbar switches.

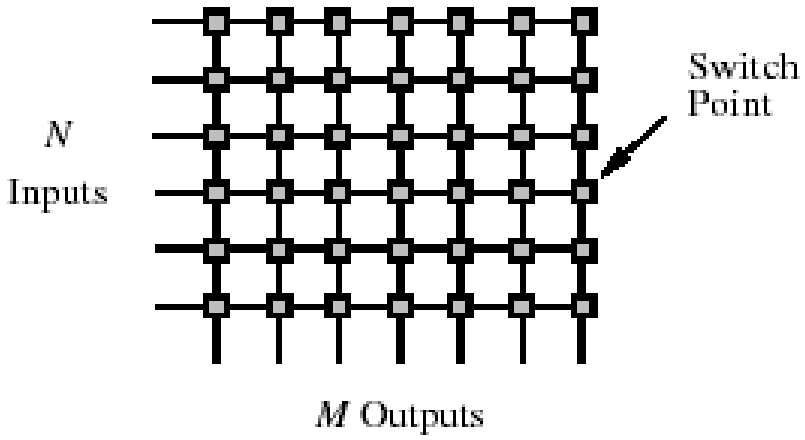


Fig 11: An M × N Crossbar Switch

7.23 Multistage interconnection networks

Multistage Interconnection Networks (MIN) are frequently proposed as connections in multiprocessor systems or in high-bandwidth network switches. A multistage network consists of more that one stage of switching elements and is usually capable of connecting an arbitrary input terminal to an arbitrary output terminal. These networks formulate the most advanced pure solution, which lies between the two extremes. The multistage networks are good compromise between cost and performance. Multistage Interconnection Networks are an efficient implementation of packet switching networks. Because MINs require less switching elements compared to a fully meshed crossbar switch, it is possible to create very large networks at low cost. MINs are composed of several small-sized switching elements that are arranged in stages. Multistage networks can be one-sided or two-sided. The *one-sided* network, sometimes called full switches, have input-output ports on the same side. The *two-sided* multistage network usually has an input side and an output side. Areas of application for MINs lie in multiprocessor systems or

high-bandwidth communication networks, internal buffering greatly increases the performance if a MIN. MINs are divided in the following types:

- *Concentrator:* A concentrator interconnects a specific ideal input to an arbitrary idle output. Often the concentrators are used for connecting several terminals to a computer.
- *Connector:* Establish a path from a specific input to specific output.
    - *Non-blocking:* Always possible to connect idle pairs of nodes without disturbing any connection currently in progress.
    - *Rearrangeable:* It may not be possible to connect idle pairs without disturbing the current connections or by rearranging the existing connections to suit the present routing. A well-defined network, the Benes network belong to this class. The Benes rearrangeable network topology has been extensively studied for use in synchronous data permutation and asynchronous inter-processor communication.
    - *Blocking:* It may not be possible to connect idle pairs in any way.

A typical example is the *omega network*, which consists of *log p* stages, where *p* is number of inputs and outputs (usually number of processor and of memory banks). Its complexity is *o (p log p),* less than for the crossbar switch. However, in the omega network some memory accesses can be blocked. Although machines of this kind of interconnections offer virtual global memory model of programming and ease of use they are still not much popular. Butterfly networks, Benes networks and Perfect shuffle are some of the examples of multistage networks. Examples from the past cover BBN Butterfly and IBM RP-3 computers. At present IBM RS6K SP incorporate multistage interconnections with Vulcan switch.

7.24 Multilevel interconnection network

The multilevel network seems to be a relatively recent development. The idea comes directly from clusters of computers and consists of two or more levels of connections with different aggregated bandwidths. Typical examples are SGI/Cray Origin2000, IBM RS6K SP with PowerPC604 SMP nodes and HP/Convex Exemplar. This kind of architecture is getting the most interest at present.

8. Study of different types of networks

A single bus and non-blocking crossbar occupy the two ends of the performance spectrum of dynamic interconnection networks. A single bus is a cost-effective means of interconnecting a small number of processors, and for connecting processors to the memory and I/O subsystems. Owing to their simplicity and low cost, buses are the most common interconnection means in computer systems today. Crossbar networks, on the other hand, are natural candidates when low latency and non-blocking access are critical. On the other hand, Multistage Interconnection Networks extend an extensive applicability and usage in the real parallel machines. Let us have a detailed study of each of these architectures in the following section and at the end of the section let us have a look at the topological details of some of the real parallel machines.

8.1 Bus Architecture

Buses are the most common type of interconnection networks in use in today's computer systems; they are used in systems over a wide range of cost and performance, ranging from hand-held computers to supercomputers. Although a single bus allows only one pair of attached devices to communicate at any time, the performance is still adequate for many applications.

Despite their wide-spread use, buses suffer from a number of limitations. The most severe of them, perhaps, is the *capacity bottleneck*. The designer of a multiprocessor system based on a single bus often finds the bus as the most severe obstacle to scaling up the performance of the system, either by increasing the number of nodes or by increasing the performance of the existing nodes. A simple solution to the capacity bottleneck is to use multiple shared buses. When multiple buses are used to connect processors to memory modules in a shared-memory system, a throughput equivalent to that of a non-blocking crossbar can be obtained with the number of buses only slightly larger than half the number of processors, assuming that the traffic to memory is uniformly distributed among the modules. Although physical constraints such as the size of the connectors and backplanes impose severe constraints on the number of buses in today's systems, future technologies such as optical backplanes may allow the packaging of a large number of buses in a small area.

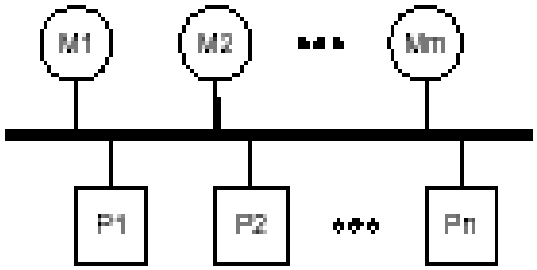


Fig 12: Sample bus architecture

A second problem with a single shared bus is that it introduces a single point of failure in the system, unacceptable for several applications that demand high reliability. Fortunately, the multiple-bus approach also solves this problem; a bus-failure in a multiple-bus system results in degraded performance, but no loss of connectivity.

A third problem that imposes a limit on the number of nodes that can be attached to bus is the fan out capability of the bus. Once again, the use of multiple buses suggests a solution to the problem by allowing each bus to be connected to only a subset of the nodes instead of the entire set. Sequent symmetry and Alliant(double) use bus architecture.

8.2 Crossbar Networks

Crossbar switches allow non-blocking access from any input port to any output port and therefore forms the natural building block of any low-latency interconnection system. Crossbar networks are the basic building blocks used to form other dynamic networks such as multistage networks. Crossbar finds application even within the individual nodes of static networks such as the hypercube, where they facilitate fast routing of data through intermediate nodes.

A conventional two-sided crossbar consists of a set of input lines and a set of output lines, conceptually placed perpendicular to each other, with switches placed at each point where the lines cross. Large crossbar switches are implemented by partitioning the matrix into smaller rectangular blocks and assigning each block to a chip. C - YMP and Fujitsu VPP500 use crossbar switches.

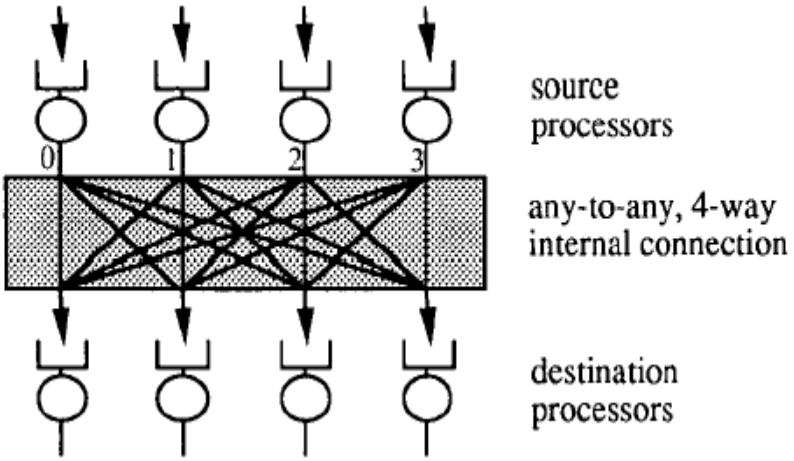


Fig 13: 4- way Crossbar Switch

Crossbar networks have not found wide use in large multiprocessor systems because of two primary reasons:

- A crossbar requires O (N^2) gates to interconnect N ports, as compared to ($N \log_2 N$) needed for a multistage network. Thus, the circuit-complexity of large crossbar networks was considered to be beyond the capability of available technologies.
- The available number of pins on an integrated package did not permit large crossbar networks to be integrated on a single chip.

8.3 Multistage Interconnection Networks

Multistage networks have several sets of switches in parallel, so data only needs to pass through several *switches*, not several processors. In this section let us study the MINs in detail. We have done an extensive study on the MIN for the project and the following are the essence of the survey we have done on the MINs.

8.31 Shuffle Exchange Networks

Interconnection networks with the capability of passing all the *N!* permutations on *N* elements in one pass through the network are known as *re- arrangeable networks*. Shuffle/exchange networks are rearrangeable networks and are initially proposed by Stone.

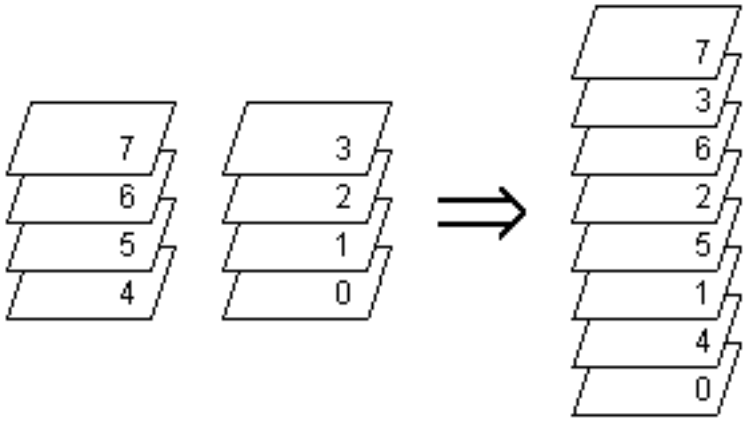
In a shuffle-exchange network, one stage of the network consists of a perfect shuffle permutation followed by an exchange stage consisting of *N/2* switches, each of size 2 x 2. The r-dimensional shuffle-exchange graph has 2^r nodes and $3*2^{r-1}$ edges. Label the nodes from 0 to $2^r - 1$ in binary. Then two nodes *U* and *V* are connected directly by an edge if either:

- *U* and *V* differ in only in the last bit (*exchange edge*)
- *U* is a left or right cyclic shift of *V* (*shuffle edge*)

For example, the omega network is constructed by cascading $log_2 N$ shuffle/exchange stages; this network, besides being capable of passing some important classes of permutations useful in parallel processing, can be controlled by a simple algorithm. However, when *N* is large, the network can perform only a small fraction of the *N!* possible permutations. The universality of shuffle/exchange networks was studied by researchers in an attempt to determine how many stages are required to attain the capability of performing arbitrary permutations, *i.e., rearrangeability.*

Perfect-shuffle interconnection: Before we study the Shuffle exchange networks let us understand the concept of perfect-shuffle interconnect. This interconnection network is defined by the routing function

$$S\left((a_{n-1}\ a_1 a_0)^2\right) \equiv (a_{n-2}\ a_1 a_0\ a_{n-1})^2$$



It describes what happens when we divide a card deck of, e.g., 8 cards into two halves and shuffle them perfectly.

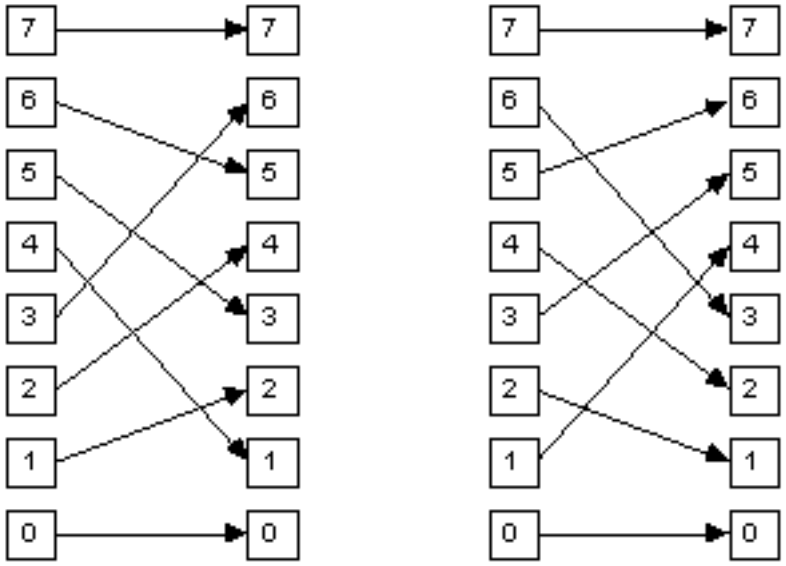
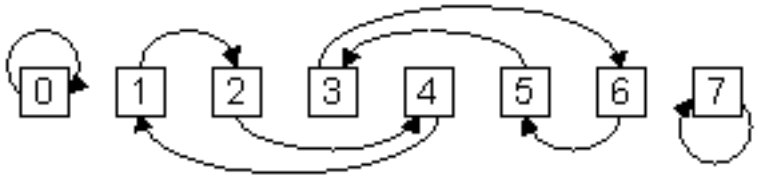


Fig 14: A general Shuffle Exchange Network

We can draw the processor interconnections required to obtain this transformation (left). If the links are bidirectional, the *inverse perfect shuffle* is obtained (right).

8.3.1.1 Omega Networks

The omega network is a multistage implementation of the single-stage shuffle-exchange network. By itself, a shuffle network is not a complete interconnection network. This can be seen by looking at what happens as data is *recirculated* through the network.



An *exchange* permutation can be added to a shuffle network to make it into a complete interconnection structure:

$$E\,(a_{n-1}\ a_1 a_0)^2 \equiv a_{n-1}\ a_1 a_0$$

A shuffle-exchange network is isomorphic to a cube network, with a suitable renumbering of boxes. An omega network is a $log_2\ N$-stage shuffle-exchange interconnection, with individual cells that can perform four different operations:

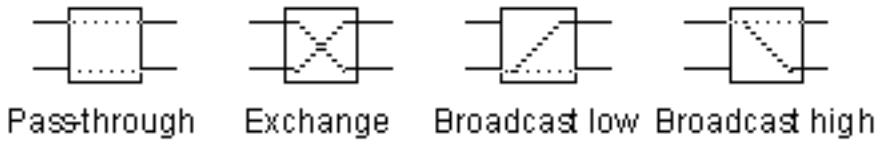


Fig 15: Four operations performed by Shuffle Exchange networks

Sum (or other operations involving all the elements) can be performed in log *N* steps. In addition, with a shuffle-exchange network; arbitrary cyclic shifts of an *N*-element array can be performed in log *N* steps.

Omega Networks are blocking networks; routes to different memory banks share a link a message might be blocked by another. A size N omega network consists of *m* identical stages, where each stage is a shuffle connection followed by a column of *N/2* interchange boxes. The inter-stage connection pattern is a perfect shuffle. IBM in its SP series of parallel computer uses a compromising strategy: it uses an Omega network but the individual switches are 8×8 cross bar.

Blocking Effects

In an Omega switch, paths are shared so it is possible for a connection to be blocked due to the connection between another pair. A network that has this feature is called a *blocking network*. Thus blocking exists in an Omega network when the requested permutation would require that a single switch be set in two positions simultaneously. Obviously this is impossible, and requires that one of the permutation requests be blocked and tried in a later pass. In general, with 2×2 switches, an Omega network can implement $n^{n/2}$ permutations in a single pass. For $n = 8$, this is about 10% of all possible permutations. In general, a maximum of $log_2 n$ passes are needed for an *n*-input Omega network, where *n* is the number of processors and also the number of banks. The base of the log is two (2) because the switch in Omega network is 2×2. Each stage connects *n* inputs and *n* outputs

In an Omega network of size *p*, a link exists between input *i* and output *j* if the following is true:

$$j = \begin{cases} 2i & 0 \le i \le p/2 - 1 \\ 2i + 1 - p & p/2 \le i \le p - 1 \end{cases}$$

This is a left-rotation operation on the binary representation of *i* to obtain *j*. This is called a *perfect shuffle*.

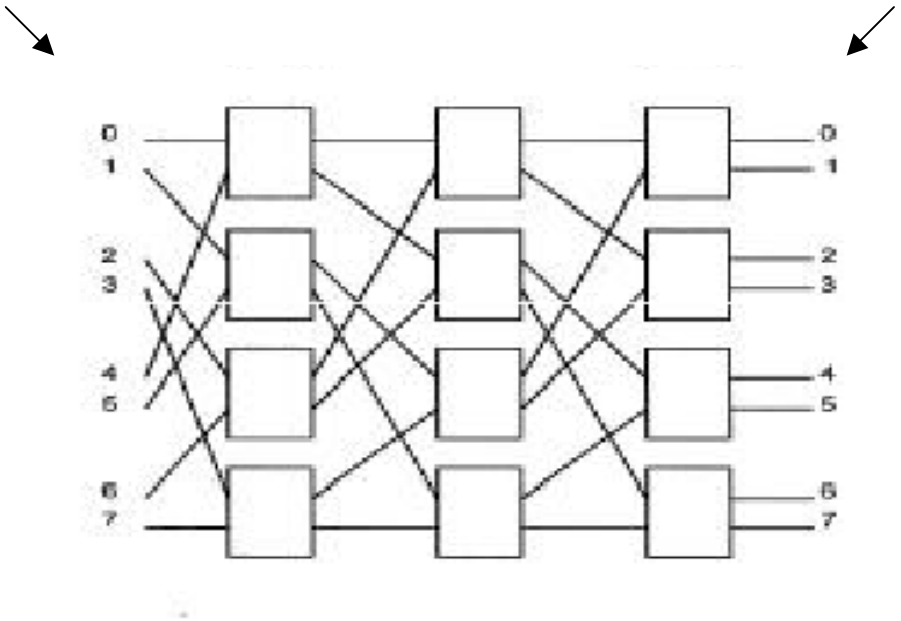


Fig 16: An 8 × 8 Omega Network

The links are numbered from 0 to *N-1* based on their vertical ordering (position). There are *log p* stages each with *p/2* switching elements each which leads to *p/2 * log p* totally. The Omega networks have a simple routing algorithm where at each stage, looking at the corresponding bit (starting with the mot significant bit) of the source and destination address, the routing is done accordingly. If the bits are the same, messages passes through, otherwise is crossed-over.

The input links to each stage follow the shuffle connection pattern. For example the link labeled 0 connects to box input labeled shuffle (0) = 0, the link (position) labeled 1 connects to box input (position) labeled shuffle (1) = 2, and so on. In general, the link (position) labeled *i* connects to the box input (position) labeled shuffle (*i*), $0 \leq i < N$. Thus, each stage of the omega network consists of the shuffle connection pattern entering *N/2* interchange boxes.

The omega network is redrawn with the same link-labeling convention as used for the generalized-cube network. In particular, a link labeled *j* at the input retains this label number, $0 \leq j < N$. Using this labeling it can be seen that stage i of the omega network implements $cube_i$, $0 \leq i < m$; that is, a stage *i* box set to swap performs the $cube_i$ function. The generalized-cube network is organized in exactly the same way, with the input stage (stage *m–1*) implementing $cube_{m-1}$, the next stage (stage *m – 2*) implementing $cube_{m-2}$, and so on.

An Omega network can also be used to broadcast data to multiple destinations. The switch to which the input is connected is set to the broadcast position (input connected to both outputs). Each additional switch (in later stages) to which an output is directed is also set to the broadcast position.

8.3.1.2 Generalized Cube Networks

The generalized-cube network is a multistage cube-type network topology. Assume the network has N inputs and N outputs. The generalized-cube topology has $m = log_2 N$ stages, where each stage consists of a set of N lines (links) connected to $N/2$ interchange boxes. Each interchange box is a two-input, two-output device. The labels of the links entering the upper and lower inputs of an interchange box are used as the labels for the upper and lower outputs, respectively. The labels are the integers from 0 to $N-1$.

The connections in this network are based on the cube interconnection functions. Let $P = p_{m-1}....p_1p_0$ be the binary representation of an arbitrary link label. Then the m cube interconnection functions can be defined as:

$$\text{cube}_i(p_{m-1}...p_1p_0) = p_{m-1}..p_{i+1}\ \overline{pi}_i\ p_{i-1}....p_1p_0$$

where $0 \leq i < m$, $0 \leq P < N$, and $\overline{pi}$ denotes the complement of p_i. That is, the cube_i interconnection function connects link P to link $\text{cube}_i(P)$, where $\text{cube}_i(P)$ is the link whose label differs from P in just the ith bit position. Stage i of the generalized-cube network topology contains the cube_i interconnection function; that is, it pairs links that differ in the ith bit position.

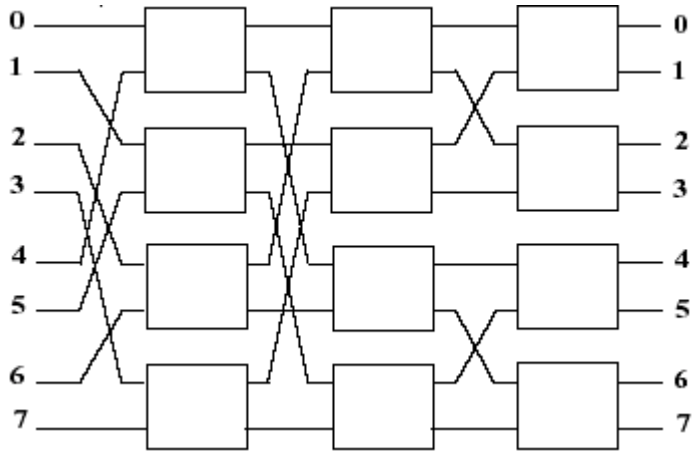


Fig 17: An 8 node Generalized Cube Network

Each interchange box can be set to one of four legitimate states. Let the upper input and output links be labeled J and lower input and output links be labeled K. The four legitimate states are (1) *straight*-input J to output J, input K to output K; (2) *swap*-input J to output K, input K to output J; (3) *lower broadcast*-input K to outputs J and K; and (4) *upper broadcast*-input J to outputs J and K. The straight and swap settings produce a one-to-one connection at the box, while the lower and upper broadcasts produce a one-to-two connection at the box. A *two-function* interchange box can be in either the straight or swap state and a *four-function* interchange box can

be in either the straight or swap state and a four-function interchange box can be in any of the four states.

8.3.1.3 Clos Network

A Clos network is a rearrangable non-blocking network. Clos networks can be used to efficiently implement low latency, high-bandwidth, connection-oriented ATM switching. Clos-type switching fabrics have also been found to be economically and technically attractive. Due to these properties they are extensively used in economical telephone switching. Combined with their inherent fault-tolerant and multi-path routing properties, they pose as a very appealing choice for reliable broadband switching.

A 3-stage Clos network consists of three successive stages of switching elements, which are interconnected by point-to-point links. In a symmetric three-stage network, all switching elements r switches of size (*n* *m*) in the first stage, *m* switches of size (*r* x *r*) in the second, and *r* switches of size (*m* x *n*) in the third stage. This network thus interconnects *n** input ports with * output ports in one direction. For an x Clos network, the network is rearrangeably non-blocking when the output ports x . When this condition is true, all the incoming calls can be routed. Fig 18 shows an example of a (m, n, r) rearrangable network with m = 3, n = 3 and r = 4.

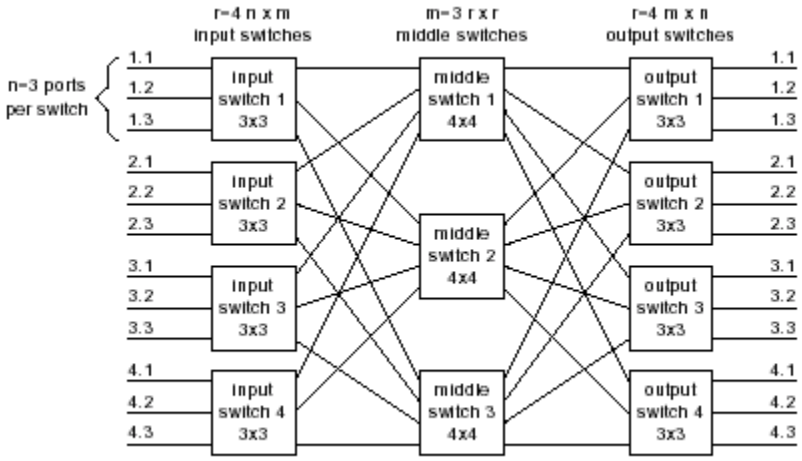


Fig 18: (3,3,4) Clos network used in our example.

Clos Networks with More Than Three Stages

To build a 4 x 4 switch, first level of switches has sixteen 4 x 4 switches. Each of them has 4 outputs, which should be connected to 4 different middle switches. Thus, the middle level must have four 16 x 16 switches. The third level of switches is similar to the first level. Now the problem is reduced to a previously solved problem.

Similarly to build 16 x 16 switches we already know how to create a 16 x 16, 3-stage Clos Network using 4 x 4 switches. Also, we were able to build a 64 x 64 network by plugging in a 16 x 16 network for each middle switch. If we want to build a 256 x 256 network we can do a similar thing. The result is a network with an odd number of stages: 16 x 16 has 3 stages, 64 x 64

has 5 stages, 256 x 256 has 7 stages, etc. We use scheduling to set up a call from a particular input to a particular output. First, we should determine which big middle stage could be used. If the first and last switches have unused channels of the same color (i.e. free channels to the same middle stage), then the appropriate middle switch should be used. However, if there are no such channels, then rearrangements should be done to free channels to the same middle stage. After the outer level is defined, the next step is to determine how to route a call inside the big middle stage. In general, the number of small middle switches will be $(2n-1)^i$ in the strictly non-blocking network versus n^i in the rearrangable non-blocking network.

In order to reach the ideal of a network that handles all traffic patterns equally well, one can take advantage of a property of the Clos network. The Clos topology provides multiple routes between hosts, and all shortest routes are deadlock-free. When a host interface can send successive packets to another host along multiple routes, the traffic is dispersed in a way that statistically avoids "hot spots," high utilization of specific network links that can result from single-route mappings of the communication patterns of application programs to the topology. This same dispersive routing technique can provide fault tolerance on a much smaller time scale than by periodic mapping, and can exploit multiple ports on host interfaces.

8.3.1.4 Benes Networks

A special case of Clos networks is one made of 2 x 2 switches, called a Benes network. It is a 2-flies back-to-back network with two middle stages fused into one stage (in fact, all Clos networks are back-to-back buttery networks). A Benes Network, recursively constructed from the Clos Network, exhibits a symmetric topological structure. In other words, the Benes network can be considered as a cascaded combination of an omega network and a reverse omega network. The complete structure of an 8x8 Benes network is given in fig 19. Since there is only one path between any input and output in both the omega and the reverse omega network, each path in the benes network actually consists of two sub paths – one in the Omega network and the other in the reverse omega network. These two sub paths can be joined at any one of the central modules to form a complete path. Therefore, there are *N/2* alternative paths for each connection in the Benes network, where *N* is the network size.

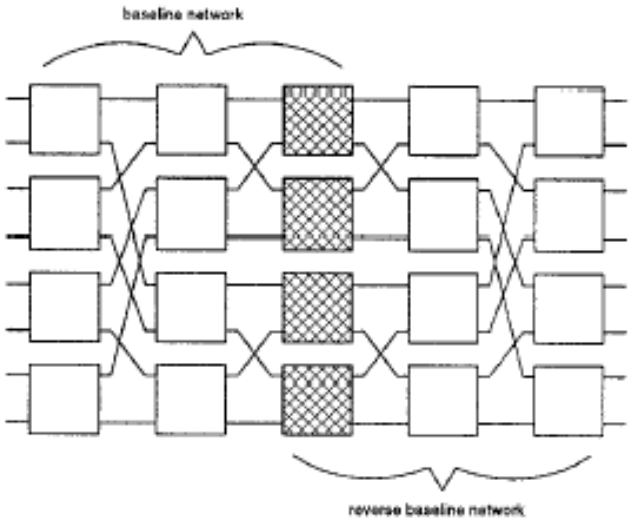


Fig 19: An 8x8 Benes network

The Benes networks have the minimal number of cross points and this property was considered very important some time ago when the cost of the system was determined primarily by the number of switches used. For *NxN* switches the cost was N^2. If the size of switches was increased, then the cost increased as square. If instead the number of stages was increased, then the cost increased logarithmically in some sense. This is not true anymore because transistors are cheap while pins or ports are expensive. So the result is now exactly the opposite - the fewer stages - the fewer pins that are necessary for each input - the lower the cost. This is because you will need 2 pins for input and output at each stage, and thus, the number of pins grows proportionally with the number of stages. That's why today designers build Clos networks with maximum size switches. They want a larger number of cross points which reduces the number of stages. The final thing about Clos and Benes networks is they have an axis of symmetry. That suggests a particular way of packaging such networks if input and output points are the same. They can be folded along the axis of symmetry and the first and the fifth stages can share the same package because the links are exactly symmetrical. The second and the fourth stage can share the package. The third stage is packaged by itself. A drawback to such packaging (unless there is simultaneous bidirectional signaling) is that now the network should be built from smaller switches because the switches have to share the number of pins on a package.

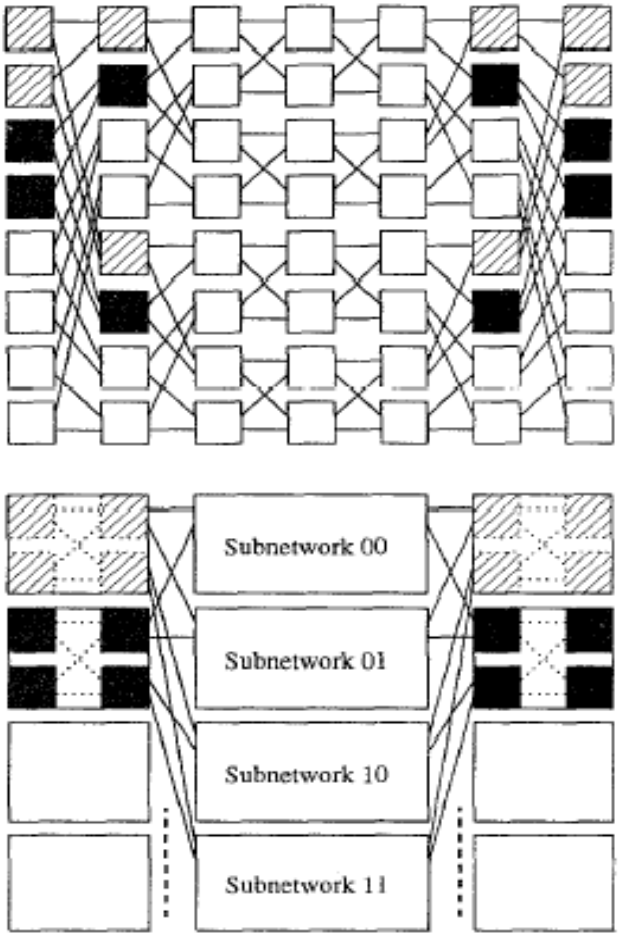


Fig 20: A 16x16 Benes Network and a fictitious structure of 16x16 Clos Networks

8.3.1.5 Banyan Networks

A network with a unique path from each input to each output is called a *banyan network*. Banyan networks belong to the class of Multistage Interconnection Networks (MINs). They are characterized by the property that there is exactly one path from any input to any output. The *indegree* of a node is the number of edges leading into it, and the *outdegree* of a node is the number of edges leading out of it. A banyan network is *layered* if the nodes can be arranged in successive layers, with inputs at the first layer, outputs at the last layer, and edges connecting nodes from one layer to nodes at the next layer. A layered network with n stages of nodes is called an *n-stage network*. A *rectangular banyan network of degree k* is a layered banyan network where all nodes (with the exception of inputs) have indegree exactly k, and all nodes (with the exception of outputs) have outdegree exactly k. A rectangular banyan network has $n + 1$ layers, each consisting of kn nodes.

A simple example of an 8×8 banyan interconnection network (i.e. which has 8 inputs and 8 outputs) is shown in fig 21. This network consists of nodes and links. Every node is a 2×2 switch, which can receive packets at each of its two input ports and send them through each of its two output ports. Generally, switches may have k input ports and k output ports (k × k switches). Switches are grouped in stages. This means that every switch in stage j is connected to switches in stages $j - 1$ and $j + 1$. Of particular interest are networks in which all switches are identical, except perhaps the switches in the first and last stages. The other building element of a MIN is links. Links are used to connect switches of successive stages. So far, we have made some assumptions about the switches. Now we will concentrate on the network. We assume that the problem is to interconnect N inputs to N outputs, thus we want to construct an $N \times N$ *MIN* which has a banyan topology. Using $k \times k$ switches, we need a network with $\log_k N$ stages and $N/k \log_k N$ switches. We, also, assume that we want to transport packets from inputs to outputs without pre establishing a path; our transport network will be connectionless.

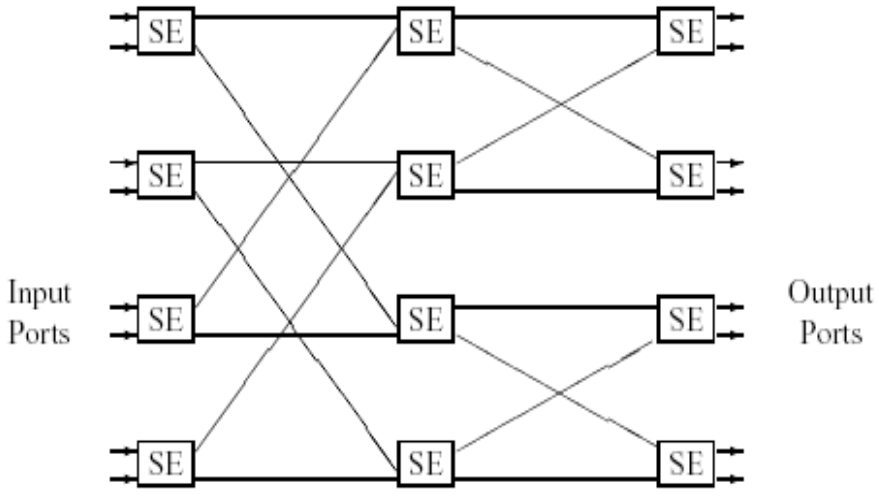


Fig 21: An 8×8 Banyan network

Advantages of Banyan Networks

Banyan networks have many advantages. First of all they require only $N/k \log_k N$ switches and $N \log_k N$ connections, as opposed to the crossbar network which requires $O(N^2)$ switches and links. The path from a source node to a destination node can be described by the sequence of digits that label the successive outputs of the switches at each stage of the network. This sequence of n digits is called a *path descriptor*.We can also define a *reverse path descriptor* describing the route backwards through the network from a destination to an input node. Delta networks are a class of banyan network with the property that the path descriptor for a destination node is a permutation of the n radix-k digits of the node address. Also, they have the property of *self-routing*. This means that every switch that accepts a packet in one of its inputs can decide in which of its outputs to forward this packet depending only on the destination address of the packet. For example, in the Delta-2 networks, which are a subclass of Banyan networks consisting of 2×2 switches, every switch of stage j can decide in which output port to forward a packet based on the j bit of the destination address. If this bit is 0 then the packet is forwarded to the "upper" output port, and if it is 1, it is forwarded to the "lower" output port. The property of self-routing is very important because it means that the implementation of a complex routing algorithm, either centralized or distributed, is not needed. Another advantage of Banyan networks is their *regularity* and *interconnection patterns* which are very *attractive* for VLSI implementation.

But, these advantages come with a cost. It is well known that Banyan networks are blocking. This means that it is possible for two or more packets to content for the same link somewhere in the network. Only one of them will win this contention and will be transmitted. The others must be buffered and try again in a later time (we assume that the network must not discard packets if there are enough buffers). So, a choice that a designer may have is where to place the buffers. If the buffers are placed in the input ports, which means that a packet stays there until it is finally transported to an output port, then due to head-of-line blocking (the packet in the head of the queue will prevent the other packets from reaching an output port even if they can find a path in the network) the throughput of the network will be much lower. Another option is to place buffers inside the switching fabric of the switches. It is interesting to note that these results are independent of the link pattern which is used to construct the network. Generally, the performance of Banyan networks is independent of the specific topology if the traffic is equally distributed among the input and output ports.

8.3.1.6 Butterfly networks

Butterfly networks are built using crossbar switches instead of those found in Omega networks. These are closely related to shuffle-exchange networks.There are no broadcast connections in a butterfly network, making them a restricted subclass of the Omega networks. Many commercial and experimental parallel computers, including the NYU Ultra computer, the IBM RP3, the BBN Butterfly, and NEC's Cenju, use butterfly networks to route packets between processors. Several proposed designs for the switching fabric of scalable high-speed ATM networks use butterfly and other closely related multistage interconnection networks.

The butterfly permutation is defined as

$$B(a_{n-1}\ a_{n-2}\ a_1 a_0) \equiv B(a_0\ a_{n-2} a_1\ a_{n-1})$$

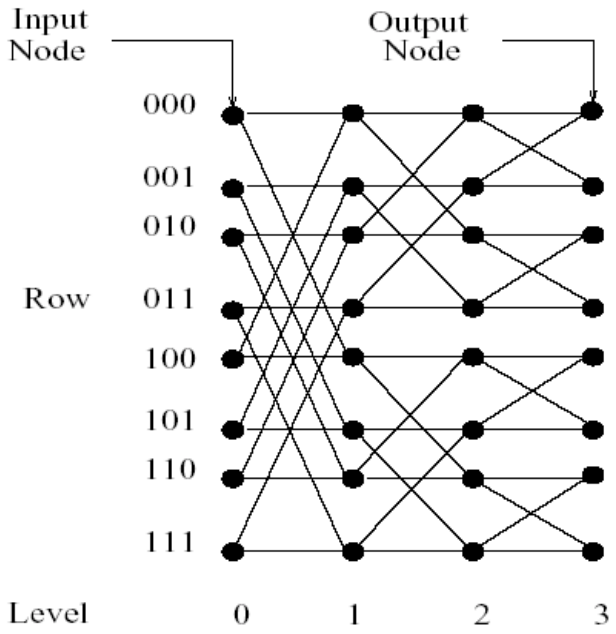
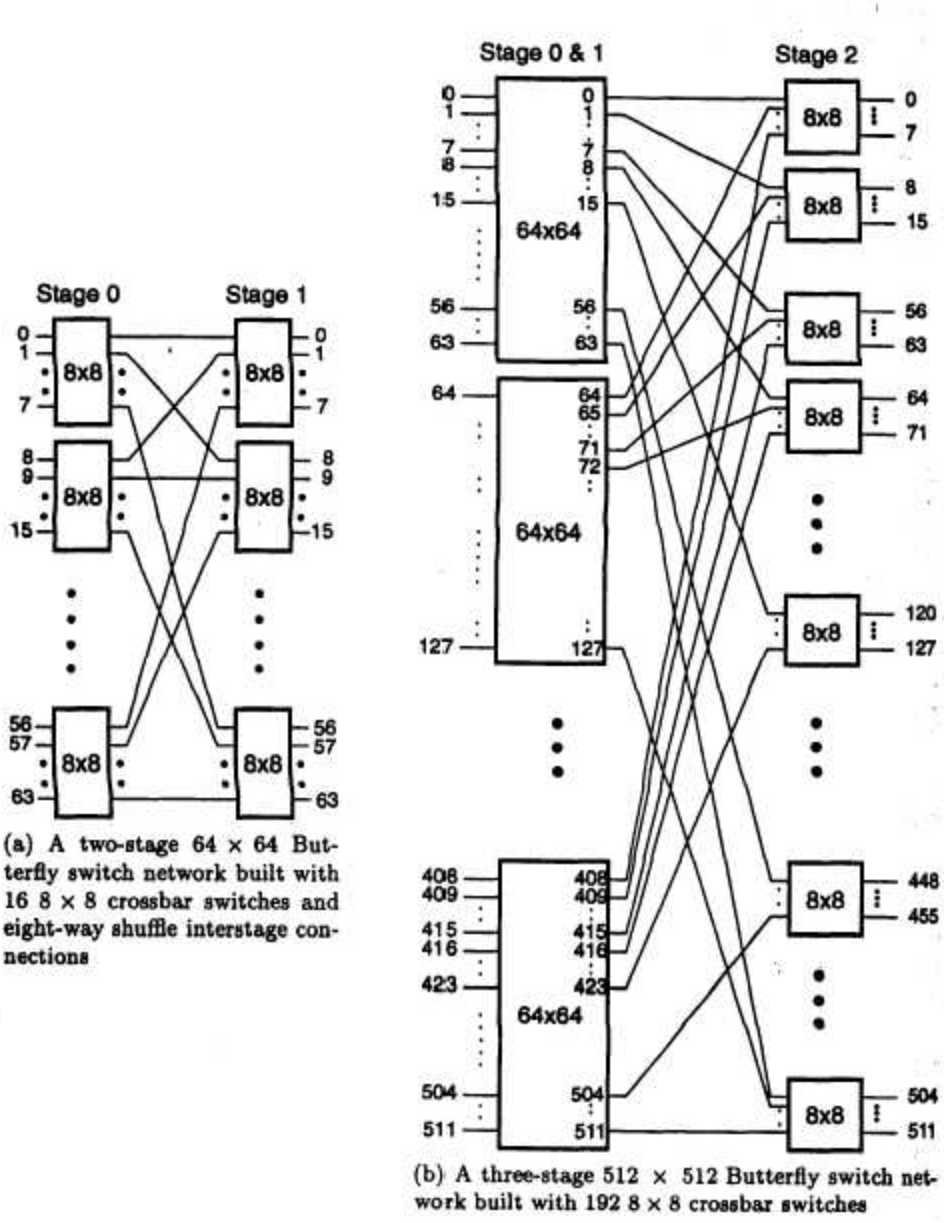


Fig 22: An 8-input Butterfly Networks

An example of an N-input butterfly ($N = 8$) with depth log$N$ (log$N = 3$) is shown in Fig 22. The edges of the butterfly are directed from the node in the smaller numbered level to the node in the larger numbered level. The nodes in this directed graph represent switches, and the edges represent communication links. We use the terms node and switch interchangeably in the rest of the paper. Each node in a butterfly has a label $\langle l, c_0 \ldots c_{\log N - 1} \rangle$, where the *level*, l, ranges from 0 to log$N$, and the *row*, $c_0 \ldots c_{\log N - 1}$, is a log N-bit binary string. The switches on level 0 are called *inputs*, and those on level log N are called *outputs*. For $l < \log N$, node $\langle l, c_0 \ldots c_{\log N - 1} \rangle$ is connected to node $\langle l + 1, c_0 \ldots c_l \ldots c_{\log N - 1} \rangle$ by a *straight* edge, and to node $\langle l + 1, c_0 \ldots \bar{c}_l \ldots c_{\log N - 1} \rangle$ by a *cross* edge. (The notation $\bar{c}_l$ denotes the complement of bit c_l.) At each time step, each switch is permitted to transmit one packet along each of its outgoing edges. In a butterfly network, packets are typically sent from the inputs on level 0 to the outputs on level log N. One of the nice properties of the butterfly is that there is a unique path of length log N between any input and any output, and there is a simple rule for finding that path. This path selection algorithm is called *source oblivious* because, at each node, the next edge taken by a packet depends only on its current location and its destination, and not on its source, or on the locations or paths taken by any of the other packets.



(a) A two-stage 64 × 64 Butterfly switch network built with 16 8 × 8 crossbar switches and eight-way shuffle interstage connections

(b) A three-stage 512 × 512 Butterfly switch network built with 192 8 × 8 crossbar switches

Fig 23: Modular construction of Butterfly switch networks with 8×8 crossbar switches Courtesy of BBN Advanced Computers, Inc.,

8.3.1.7 Gemini Network

The Gemini interconnect is a dual technology interconnection network designed for use in tightly coupled multicomputer systems. It consists of a circuit-switched optical data path in parallel with a packet-switched electrical control/data path. The optical path is used for transmission of long data messages and the electrical path is used for switch control and transmission of short messages. The Gemini interconnect is a novel processor-to-processor interconnection network for tightly-coupled multicomputers. It includes an end-to-end optical data path for high bandwidth, large data volume message delivery. The optical switching is accomplished using $LiNbO_3$ electro-optical 2x2 switches. In addition, Gemini includes an electrical path that both controls the optical path and delivers low-latency, small data volume messages.

Gemini Architecture

The Gemini interconnect consists of a dual network consisting of an optical interconnect for passing large data blocks and a parallel electronic interconnect for both controlling the optical switching elements (and, thus, message routing) and also for passing small blocks of data. Given the high speeds of the optical interconnect and the unavailability of low cost optical storage and logic components, circuit switching is used for the optical path. An electrical control message sets up and tears down this path. The electrical network is a self-routing, packet switched network used for short, low-latency data, and control messages.

A simple Banyan topology is used for both the optical and electrical networks. While this is a blocking network, it uses only O *(N log N)* switching elements, rather than the O (N^2) required for a crossbar topology. Additionally, the number of switches through which the signal must pass is a constant *log N* as opposed to a maximum of *2N-1* in the crossbar case. Minimizing the total number of switches is important since, in today's optical technology, optical switches are costly. Reducing the number of switches through which a signal must pass is also important since, for a given optical source power level and a given receiver detector sensitivity, there are a maximum number of switches through which the signal may pass before detection becomes unreliable.

Gemini Interconnect

Electro optic 2x2 switching elements are the key devices used in the fabrication of the Gemini *N x N* optical data path. These $LiNbO_3$ switching elements rely on the electro optic effect (i.e., the application of an electric field changes the refractive index of a material within the field) to provide for pass through and crossover connections between the input and output ports. Thus, the state of the 2x2 optical switching elements is determined by an electrical control signal. Larger 4x4 switching elements are available and we can expect the levels of integration to improve rapidly over the next several years.

Fig 24 shows an 8x8 Gemini network and also illustrates the use of dual electronic and optical networks. To construct the optical data path, the electro optic 2x2 switching elements are connected together on a single printed wiring board using polymer channel waveguides. Connections between boards use optical fiber. The Gemini interconnect consists of a number of waveguide bends, crossovers, and optical switches. After the last stage of switching, the waveguide is coupled into an outbound fiber, which is connected to a photodiode associated with the destination processor.

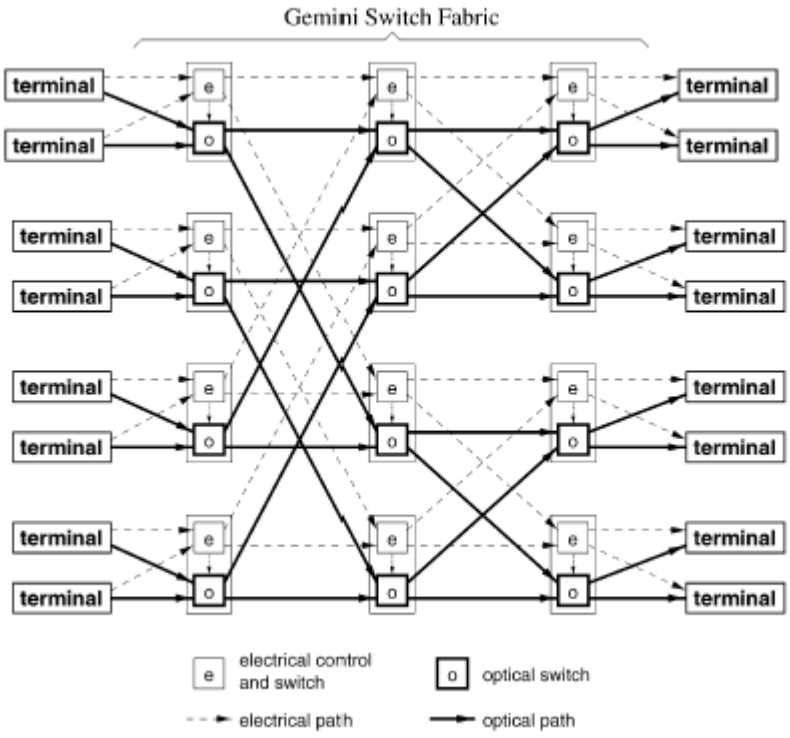


Fig 24: An 8x8 Gemini Network

8.3.1.8 Honeycomb Networks

The Honeycomb mesh, based on hexagonal plane tessellation, is considered as a multiprocessor interconnection network. A honeycomb network with n nodes has degree 3 and diameter ≈ 1.63 $\sqrt{n}$ -1. Vertex ad edge symmetric honeycomb torus network is obtained by adding wraparound edges to the honeycomb mesh. The network cost, defined as the product of degree and diameter, is better for honeycomb networks that for the two other families based on the square and triangular tessellations.

The Honeycomb networks begin with hexagonal tessellation but use cells (instead of nodes) as processors. The honeycomb torus network is obtained by adding wraparound edges to the corresponding mesh networks.

Honeycomb and hexagonal meshes and mesh-connected computers clearly belong to the same family of networks. Their comparison in all criteria should give the same asymptotic performance. For example, they all have fixed degree and O ($\sqrt{n}$) diameter. Thus, honeycomb meshes compare to other networks (e.g., hypercube) asymptotically as well as mesh-connected computers (for such criteria). Hypercubic networks are better than mesh-like ones in terms of diameter, average distance, and product of diameter and degree. Still, mesh is the only one of two topologies manufacturers have used. One of the main reasons behind this is, certainly, is because meshes are planar graphs, therefore, they have easy physical layout.

Topological properties of Honeycomb

Honeycomb networks can be built from hexagons in various ways. The degree of any such network is three. The simplest way to define them is to consider the portion of the hexagonal tessellation which is inside a given convex poly. Honeycomb hexagon mesh (HHM) is inside a regular hexagon, honeycomb rhombic mesh (HRoM) is inside a rhombus, and honeycomb rectangular mesh (HReM) is inside a rectangle.

To maximize symmetry, honeycomb (hexagonal) meshes can be built as follows: One hexagon is a honeycomb mesh of size one, denoted HM_1. The honeycomb mesh HM_2 of size two is obtained by adding six hexagons to the boundary edges of HM_1. Inductively, honeycomb mesh HM_t of size t is obtained from HM_{t-1} by adding a layer of hexagons around the boundary of HM_{t-1}. The honey-comb mesh is a bipartite graph. All nodes can be subdivided into two groups, which will be called black and white nodes, such that any edge joins a black and a white node.

Honeycomb torus network can be obtained by joining pairs of nodes of degree two (i.e., their unused ports) of the honeycomb mesh. In order to achieve edge and vertex symmetry, the best choice for wrapping around seems to be the pairs of nodes that are mirror symmetric with respect to three lines, passing through the center of hexagonal mesh, and normal to each of three edge orientations.

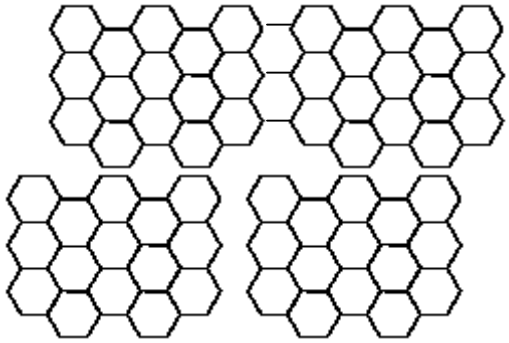


Fig 25: Honeycomb mesh

8.3.1.9 Delta Networks

A banyan network, for which the label sequence of every path to the same output is the same, is known as a delta network. Delta networks are shown to have a recursive structure. Delta networks are useful for packet routing applications since each switching decision made for a packet at a particular node depends on only a single bit of the destination address of the packet. In a general network, the paths leading from different input nodes to the same output node may have different path descriptors. Thus, a routing table, containing a path descriptor for each output node, is needed at each input node. It is convenient to have all these tables identical. Then, we can take the path descriptor associated with paths leading to the output node *s* to be the *address* of *s*, and the unique information needed to route a packet to an output node is the address of that node.
We define a *digit controlled* or *delta network* to be a network with the following properties:

(1) There is a path from each input node to each output node.
(2) The path descriptors associated with paths leading to the same output node are identical.

The second condition implies that there is at most one path from an input to an output, since two different paths from the same input have different descriptors. This along with the first condition implies that delta networks are banyan networks, i.e. there is exactly one path from each input to each output. The second condition also implies that all the paths leading from the inputs to any particular node in the network have the same length. In particular, the nodes of a delta network can be arranged by stages so that the input nodes are all at stage 1 and edges connect nodes at stage i only to nodes at stage i+1. Note, however, that the output nodes of a delta network need not all be at the same stage. Delta networks in which all outputs are at the same stage are layered banyan networks.

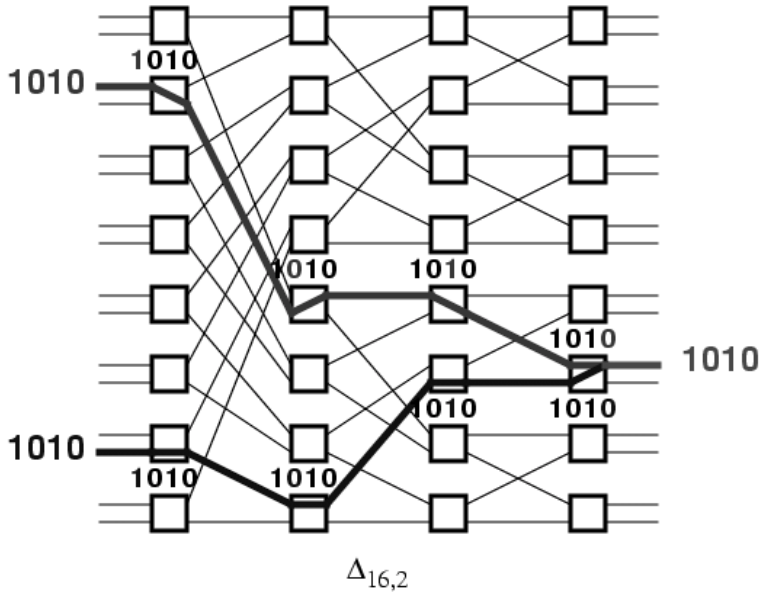


Fig 26: Delta or Baseline routing network

8.3.1.10 Bidelta Networks

Bidelta networks are shown to have a unique topology. If the endpoints of the edges of a banyan network G can be labeled so that both the network and its reverse are delta networks, then G is said to be a bidelta network. For instance, a butterfly network is also an example of a bidelta network. The definition of bidelta network is used to derive in a uniform manner the labeling schemes that define the omega networks, indirect binary cube networks, flip networks, baseline networks, modified data manipulators, and two new networks; these schemes are generalized to arbitrary radices.

In many applications, traffic through the network is bidirectional. It is convenient then that the traffic in the reverse direction also traverses a delta network — especially if the ‘‘output’’ nodes can initiate requests. The *reversal* G_R of the network G is the network obtained from G by

reversing the direction of each edge, and replacing each input by an output and each output by an input. We assume that the reverse network G_R is labeled or, equivalently, that the inputs of G carry distinct addresses, and that each edge of G carries two numbers, one associated with each of the two incident nodes. A network G is a *bidelta network* if both G and G_R are delta networks. In a bidelta network all paths connecting inputs to outputs have the same length, all paths leading from inputs to a given node have the same length and the same path descriptor, and all reverse paths leading from output to the same node have the same length and the same path descriptor. We use *multistage shuffle-exchange network* for any n-stage bidelta network of degree k (although in the literature the term often refers only to networks of degree 2).

In packet-switching networks, when messages are sent from inputs to outputs, replies are often returned to the sender. We remark, in passing, that in a packet-switching network that has labels on both the input edges and the output edges, a message need not (initially) carry the sender address. Rather, this address can be created on the fly when the message is routed: whenever one digit from the ‘‘forward’’ path descriptor is discarded, it is replaced by one digit that identifies the edge through which the message has arrived. When the message arrives at its destination, it carries a path descriptor for the reverse path to the sender; in a bidelta network this is the address of the sender. Two bidelta networks are isomorphic if there is a label preserving graph isomorphism between them (labels for both directions are preserved). Now we achieve at last our hope of having a functional description that defines a unique network.2

8.3.1.11 Beta Networks

A β-network results from simply interconnecting a set of β-elements; no particular topology is specified. A β-element is an interchange box that can perform only straight and exchange permutations.

The β-network fault model assumes 1) only switches fail, by becoming stuck in one of their two allowed states, 2) faulty switches are usable, and 3) faults occur independently. A β-network is said to tolerate a fault if the fault does not destroy dynamic full-access property if any given network input can be connected to any single network output in a finite number of passes through the network. Between passes it is assumed that each output can connect to its corresponding input via a path outside the network. This is a considerably less restrictive fault-tolerance criterion than those of the other networks surveyed.

9. Tradeoffs between different kinds of DIN

- The most common shared memory dynamic interconnection networks is bus-based network, which processors are connected to a global memory unit using a common data path called a bus. With increasing number of processors, the performance of the bus-based network deteriorate drastically because of the fixed amount of data a data bus can handle

- The crossbar interconnection network will give good data transfer performance even the number of processors and memories unit increase. They are extensive however; it is quite expensive to build.

- A multistage network scheme is usually employed to compensate performance and cost. A commonly used multi-stage connection network is the omega network.

Properties of various types of multiprocessor interconnections

Property	Bus	Crossbar	Multistage
Speed	low	high	high
Cost	low	high	moderate
Reliability	low	high	high
Configurability	high	low	moderate
Complexity	low	high	moderate

9.1 Comparisons between different Multistage Interconnection Networks

The following table shows a comparison between some of the MIN networks we studied above and gives a broad outlook on the hardware details of each of the networks.

network type	number of stages	number of switches at a stage	topology of links between stages	switch size	operation mode
omega	$\log_2 N$	N/2	2-way shuffle	2x2	blocking
butterfly	$\log_8 N$	N/8	8-way shuffle	8x8	blocking
genera-lised-cube	$S=\log_2 N$	N/2	[0,1]: shuffle [1,S]: exchange	2x2	blocking
Benes	$S=2*\log_2 N-1$	N/2	[2,S-1]: exchange	2x2	rearrangeable nonblocking

10. Examples

To wrap up with the dynamic interconnection networks, we present some of the examples of parallel computers with indirect networks and commercial switches to build indirect networks. These examples are listed below:

- Myricom Myrinet: Supports regular and irregular topologies, 8×8 crossbar switch, 9-bit channels, full-duplex, 160 Mbytes/s per link.
- Thinking Machines CM-5: Fat tree topology, 4-bit bidirectional channels at 40 MHz, aggregate bandwidth in each direction of 20 Mbytes/s.
- Inmos C104: Supports regular and irregular topologies, 32×32 crossbar switch, serial links, 100 Mbits/s per link.
- IBM SP2: Crossbar switches supporting Omega network topologies with bidirectional, 16-bit channels at 150MHz, 300 Mbytes/s in each direction.
- SGI SPIDER: Router with 20-bit bidirectional channels operating on edges, 200 MHz clock, aggregate raw data rate of 1 Gbyte/s. Offers support for configurations as non-blocking multistage network topologies as well as irregular topologies.
- Tandem ServerNet: Irregular topologies, 8-bit bidirectional channels at 50 MHz, 50 Mbytes/s per link.

11. Topologies in some Real Machines

As a closing and concluding part we thought it would be good to give the summary of the topologies used in some of the real machines in the parallel world. Here is a table that shows the topology, cycle time, channel width and routing delay in some of the supercomputers.

Machine	Topology	Cycle Time (ns)	Channel Width (bits)	Routing Delay (cycles)	flit (data bits)
nCUBE/2	Hypercube	25	1	40	32
TMC CM-5	Fat-Tree	25	4	10	4
IBM SP-2	Banyan	25	8	5	16
Intel Paragon	2D Mesh	11.5	16	2	16
Meiko CS-2	Fat-Tree	20	8	7	8
CRAY T3D	3D Torus	6.67	16	2	16
DASH	Torus	30	16	2	16
J-Machine	3D Mesh	31	8	2	8
Monsoon	Butterfly	20	16	2	16
SGI Origin	Hypercube	2.5	20	16	160
Myricom	Arbitrary	6.25	16	50	16

12. Conclusion

In this project we have done an extensive search on the different kinds of Interconnection Network. Based on the study we have done on the interconnection networks we have understood the need for the interconnection networks and how the interconnection networks affect the parallel system. So a careful selection of the interconnection network is a very vital thing to be done when we design a parallel system. In general, an interconnection network must have a low cost and must be scalable and be able to carry large data. Depending on the specific applications a compromise between the cost and performance is made. Thus the interconnection network plays a important role in the design of the parallel system and this project ahs given us an opportunity to figure out the pros and cons of each of the types of the interconnection networks.

References

[1] Howard Jay Siegel, *Interconnection networks for large-scale parallel processing: theory and case studies*.

[2] Isaac D. Scherson, Abdou S. Youss, *Interconnection networks for high-performance parallel computers* [edited by].

[3] *Interconnection networks for multiprocessors and multicomputers: theory and practice*: manuscript for IEEE tutorials.

[4] Chuan-lin Wu and Tse-yun Feng, *Tutorial, interconnection networks for parallel and distributed processing*

[5] Roger D. Chamberlain, Mark A. Franklin and Ch'ng Shi Baw, "*Gemini: An Optical Interconnection Network for Parallel Processing*", IEEE Trans. Vol.13, No.10, Oct 2002.

[6] C. Salisbury and R. Melhem, "*Distributed, Dynamic Control of Circuit-Switched Banyan Networks*".

[7] Jose G. Delgado-Frais, "*A Programmable Dynamic Interconnection Network Router with Hidden Refresh*", IEEE Vol. 45, No. 11, Nov 1998

[8] Marcus Brenner, Dietmar Tutsch and Gunter Hommel, "*Measuring Transient Performance of a Multistage Interconnection Network Using Ethernet Networking Equipment*"

[9] Morris Marden, "*A Genetic Algorithm for Designing Parallel Processor Interconnection Networks*"

[10] Amit Kumar Gupta, Francois Labonte, Paul Wang Lee, Alex solomatnikov, "*Dynamic Buffer Organization Methods for Interconnection Networks Switches"*

[11] Ivan Stojmenovic, "*Honeycomb Networks: Topological Properties and Communication Algorithms*", IEEE Trans. Vol. 8, No. 10, Oct 1997.

[12] Jean Carle, Jean Frederic Myoupo and Ivan Stojmenovic, "*Higher Dimensional Honeycomb networks*"

[13] Christos Bouras and Christos Gkantsidis, "*Cost of Implementing Banyan Networks for use in ATM switching fabrics*"