Studies on Design, Routing and Fault-Tolerance of Interconnection Networks

Doctoral Dissertation

By

Krishnendu Mukhopadhyaya

Under the supervision of

Professor Bhabani P. Sinha

Indian Statistical Institute
203, B.T. Road, Calcutta 700 035
INDIA

DEDICATION

To my parents

Acknowledgement

It was my priviledge to have worked under the supervision of Professor Bhabani P. Sinha. He made me believe in myself and it was through his encouragement, that I decided to go for the Ph. D. degree. He has been a friend, philosopher and guide to me, in the true sense of the phrase.

I would like to thank Dr. Bhargab B. Bhattacharya for his constant help and support in spite of his extremely busy schedule. I thank Dr. Jayasree Dattagupta and Mrs. Nabanita Das for their co-operation and collaboration. I also thank Dr. Mihir K. Chakrabarti and Mr. Ashok R. Dasgupta for their encouragement and support. I would also like to thank my friends Susmita Sur-Kolay, Subhas C. Nandi, Saswati Mukherjee, Sandip Das, Rajib K. Das, Debashis Das, Srabani Sengupta and Mabhin Ghosh for their delightful company, stimulating discussions and ever-dependable helping hands. My thanks are due to Mr. Anil K. Chakrabarti for his assistance in every official matter.

I express my sincerest gratitude to Mrs. Sumita Sinha for constantly pushing me out of my lethargy. I also thank all other persons who have been helpful in some way or other.

Last but not the least, I thank my parents, my brother and my sister for being themselves and making me what I am today.

[illegible]
(Krishnendu Mukhopadhyaya)

September 19, 1993
Electronics Unit
Indian Statistical Institute
Calcutta 700 035.

Contents

Chapter 1
Introduction

1.1 Introduction

A recent trend in computing is to distribute the computations among a set of processing elements. There are two basic approaches to this – one is to build a *loosely-coupled system* and the other is to form a *tightly-coupled system* [PS85].

In a loosely-coupled system, the processors do not share common memory or a common clock; but sharing of important resources like data files, softwares, special hardware components etc., is possible without duplicating the resources themselves. The processing nodes may even be geographically separated from each other and are connected through databuses, telephone/radio links, satellite, etc. Such loosely-coupled systems are also commonly referred to as *distributed systems.*

In a tightly-coupled system, the processors share a common clock and/or common memory resulting in a *parallel processing* environment. Such a system enables one to meet the requirement for enormous amount of fast real-time computations in many practical applications, e.g., weather forecasting, image processing, etc. Examples of parallel processing systems include pipelined computers, array processors and multiprocessors. A distributed system or a parallel processing system involving array processors requires an *interconnection network* for communication among the processing elements.

Interconnection Networks may be classified into two broad categories [Fe81]. They are :

i) *static interconnection network*

and ii) *dynamic interconnection network.*

A static interconnection network is a collection of processors and links among them. Such an interconnection network is usually represented by a graph. The nodes of the graph represent the processing elements and the links of the graph stand for the communication links. This graph is usually called the *network graph* or the *network topology*. A dynamic interconnection network has a set of sources and a set of destinations (may be the same as the set of sources) which are connected through switches and links. Different input-output connection patterns may be achieved by changing switch-settings.

The desirable properties of a good network topology are :

i) *Low degree of nodes* : The upper bound on the degree is, in practice, restricted by the number of I/O ports that can be provided by a node.
ii) *Low number of links* : This helps in reducing the cost of the interconnection.
iii) *Low diameter* : This is to reduce the inter-node communication time.
iv) *High degree of fault-tolerance.*
v) *Regular structure* : This is for easier implementation in VLSI.
vi) *Incremental extensibility*

and vii) *Simple routing algorithms* in both fault-free and faulty conditions.

Some of these properties are mutually conflicting. For example, the third property is in conflict with the first two. The fourth one in the above list is impeded by the previous three. Some of the problems of recent research interests in the area of static networks are as follows :

1. Design of a network topology : An integrated design approach, simultaneously optimizing all of these aspects, is very difficult. The usual practice is to consider one or some of them, but not all at a time, to arrive at an optimal or near-optimal design. Many such topologies, e.g., tree, mesh [Le93], de Bruijn [Br46], Möbius [LS82], hypercube [Já92], cube-connected-cycle [PV81], double-loop [DHL+90] etc., are available in the literature.

2. Modification of the existing topologies : Some properties of some existing topologies may be improved upon, by suitable modification, without affecting much of the other desirable properties. For example, the diameter of a hypercube can be reduced by adding some extra links or exchanging some suitable pairs of links [ENS91].

3. Analysis of network topologies : Properties like diameter, average path length, fault-tolerance etc. may be derived for the existing or new network topologies.

4. Routing : Simple routing is one of the most important needs of a network topology. Routing problem may be classifed into four categories :

i) *node-to-node routing* (sending a message from one node to another node),
ii) *broadcasting* (sending the same message from one node to all other nodes),
iii) *scattering* (sending different messages to different nodes from one single node),
iv) *multi-node scattering* (scattering from a set nodes).

Algorithms for all the four categories exist in the literature [BOS+91].

Some important considerations for a dynamic interconnection network are :

1. Permutation capability : Some networks cannot realize all possible permuations in a single pass. These are called *blocking* networks. Examples of blocking networks are *Baseline* [WF80], *Omega* [La75] etc. For blocking networks it is important to find the minimum number of passes that may be required to realize a given permutation.

Again, some networks can realize any permutation if the whole permutation is given together. But if the requests for input-output connection are given one by one, some requests may be blocked by some of the existing connections. Such networks are called *rearrangeable* networks. Very popular among the rearrangeable networks is the Benes network [Be65].

Networks which can accommodate any request at any stage are called *non-blocking* networks. Examples of non-blocking networks are Crossbar and Clos network [Cl53].

Studies on the permutation capabilities of different dynamic networks constitute a major area of research interest.

2. *Routing algorithm* : Finding simple routing algorithms is an important consideration for a dynamic interconnection network. For blocking networks, it is desirable that a given permutation is realized in minimum number of passes. For some networks, permutations can be realized by distributing the routing algorithm to the switches with some small extra hardware. Such networks are called *self-routing* networks. For the others, a given self-routing strategy may realize a subset of permutations.

3. Fault-tolerance : Detection and diagnosis of different types of faults is an important problem. Different fault models may also be considered [FW81], [Ag82]. When the faults are diagnosed, the routing algorithms may suitably be modified to by-pass the faulty areas, with graceful degradation of performance.

1.2 Scope of the Thesis

In this thesis we will address some problems relating to the design, analysis, routing and fault-tolerance issues of both static and dynamic interconnection networks. In the follwing subsections we discuss some of the problems we have addressed.

1.2.1 Static Networks

Results on the static networks include i) the design problems involving fault-tolerant hamiltonian topologies, ii) modifications on the hypercube topology for reduction of diameter, iii) an in-depth study on distributed loop networks and iv) reliability analysis of networks.

A. Fault-Tolerant Hamiltonian Topologies

Fault-tolerance of a network graph can be expressed in several ways. In a typical application, a network graph may be called fault-tolerant if a specific structure, for example, a hypercube or a hamiltonian cycle, is embedded in the network even after the occurrence of the fault(s). In this thesis we propose a family of network topologies, each of which has a hamiltonian cycle in the fault-free situation as well as when there is a single node or link failure. The proposed topologies require the minimum number of links among all possible interconnection structures having the above properties.

B. Bridged and Twisted Hypercubes

Hypercube is a very popular network topology. In this thesis, we propose two techniques of reducing the diameter by

i) adding some extra links
and ii) exchanging some pairs of links (*twisting*) without adding any extra link.

In the first technique, the number of additional links, referred to as bridges, constitutes a very small fraction of the total number of links. We show that by adding 8 bridges to an n-cube ($n \geq 4$), its diameter can be reduced by 2; also by adding 16 bridges to an n-cube ($n \geq 6$), its diameter can be reduced by 3. In general, we can show that by adding $\binom{4m}{m+1} + 1$ bridges to an n-cube ($n \geq 4m$ and $m \geq 2$), its diameter can be reduced by 2m; by adding $2\binom{4m-3}{m} + 1$ bridges to an n-cube ($n \geq 4m-2$ and $m > 2$), its diameter can be reduced by $2m-1$.

In the second technique, we consider the reduction of the diameter of an n-cube by exchanging some independent links. Two links are called independent if they are not incident on a common node. We show that by exchanging 4 pairs of independent links in an n-cube ($n \geq 5$), we can reduce its diameter by 2. Exchange of 16 pairs of independent links, reduces the diameter of an n-cube ($n \geq 7$), by 3. For $n \geq 9$, the diameter can be reduced by 4, by exchanging 57 pairs of independent links. In general, to reduce the diameter by n/2, where n is an even number ≥ 10, we need to exchange $\binom{n-1}{r} + 1$ pairs of independent links, where $r = \lfloor n/4 \rfloor + 1$.

C. Distributed Loop Networks

The ring network is a popular network topology used in local area networks. But it has the disadvantage of high diameter and hence large communication delay. So loop networks were introduced with some extra fixed-jump links added to the ring. Some work has already been done on the optimal choice of the jump-size. The minimum diameter of a loop network with N nodes for different choices of jump size is bounded below by $lb(N) = \lceil (\sqrt{(2N-1)} - 1)/2 \rceil$. In this thesis, we characterize a subset of the values of N for which the lower bound on the diameter of the network can be achieved. An algorithm for finding a shortest path between any two nodes of a general loop network is also reported. Lastly, we propose a scheme for finding a near optimal path (path length not more than one over the optimal) in the case of a single node or link failure.

D. Reliability Analysis of Networks

Fault-tolerance of a general-purpose network is usually measured in terms of the connectivity of the underlying graph. For a better quantitative measure of the degree of fault-tolerance, an alternative concept of *reliability* is also considered. In this approach, failures of the different nodes and/or links are described using a probabilistic model. We propose a method for evaluating this reliability when only the nodes may fail but the links are fault-free. Analytical expressions for the reliabilities of some common networks are derived. Non-recursive formulae for the reliability of a few families of network topologies are also found.

1. 2. 2 Dynamic Networks

In the case of dynamic networks, some new results have been obtained involving i) self-routing in Benes network, and ii) non-blocking Multistage Interconnection Networks (MINs).

A. Self-Routing in Benes Network

Among the dynamic interconnection networks, an $N \times N$ Benes network can realize all N! permutations. Though the communication time in a Benes network is $O(\log_2 N)$, the routing algorithm takes $O(N. \log N)$ time. For some permutations, it is possible to set the required switch-setting by setting each switch independently by considering only the inputs to that particular switch. Such permutations are called self-routable permutations. In this thesis, we classify the self-routable permutations according to some possible self-routing strategies. For such a class S_i of self-routable permutations, we establish that $|S_i| > 2^n (2^{N/2} + n! - 1)$, where $n = \log_2 N$. In fact, it is a lower bound on the size of the intersection of all S_i's considered here. Some interesting properties of these self-routing strategies are analyzed, which essentially help characterize the set of permutations realizable by these techniques. We also propose an $O(n)$ time algorithm that may apply to any of these classes S_i; for a given permutation P, it checks whether $P \in S_i$ and if so, it generates the necessary controls for routing.

B. Analysis of Non-blocking MINs

Another important consideration for the dymaic interconnection networks is their blocking nature. In this thesis, we present some new properties of non-blocking interconnection networks. We also propose some classes of non-blocking multi-layered interconnection networks and their corresponding routing strategies.

Chapter 2
A Brief Review

2.1 Introduction

As already mentioned in chapter 1, there are several points to be considered in the design of a network topology. It is impossible to optimize all the aspects to arrive at an universally optimal design. Some of the optimization criteria are even mutually conflicting in nature. For example, the requirement of low valency of nodes gives rise to higher diameter i.e., the maximum shortest distance between any pair of nodes and also reduces the fault-tolerance of the network. There are many topologies available in the literature, which have their positive as well as negative sides with respect to different parameters, e.g., valency of nodes, diameter, fault-tolerance, incremental extensibility etc.. Depending upon the specific application area, one may need to optimize one or several parameters of the network, and based on that, one chooses either one of the existing topologies or designs a new one to fit the requirements.

2.2 Loop Networks

One very common network topology is the ring. The ring has many attractive properties like simplicity of structure, incremental extensibility, low valency, ease of implementation etc. But it has some drawbacks as well. It is highly vulnerable to faults in the network. Also the diameter of a ring of N processors (nodes), is $\lfloor N/2 \rfloor$ ($\lfloor x \rfloor$ denotes the maximum number $\leq$ x) which leads to large transmission delay. There have been several approaches to bring down the diameter of a ring by adding some more links to it. One such idea, *chordal ring*, was proposed by Arden and Lee [AL81], where there is one chord from every node of the ring. This is a 3-regular graph with diameter $O(N^{0.5})$. Another approach is to use two chords from every node. We define the length of a chord as the distance (along the ring) between the nodes that are joined by the chord. Using this metric, chords are made to be of fixed length. These graphs are 4-regular, provided that the chords are not of length N/2. These structures are called *Double-Loop Networks* or simply *Loop Networks*. .

Loop networks are special cases of an important class of graphs, called *Circulants.* Circulants have been known in the graph theory for a long time. According to Davis [Da79], they were first introduced by Catalan in 1846. A Circulant $C_N(s_1, s_2)$ is a graph with N nodes numbered from 0 to N−1 and node i is connected to nodes $(i \pm s_1) \bmod N$ and $(i \pm s_2) \bmod N$. There have been several works on their properties [Da79], [BT84], [BW85].

We consider a set of N nodes labelled $V_0, V_1, \dots, V_{N-1}$. Each node V_i is adjacent to 4 other nodes, $V_{i+1}, V_{i-1}, V_{i+s}$ and V_{i-s}, where s is the length of a chord. Using standard notation, [DHL+90] let us call this graph G(N; 1, s). The question that arises is : *what value of s should be chosen so that the diameter of* G(N; 1, s) *is minimum among all possible double-loop networks with N nodes?*

Let d(N; 1, s) denote the diameter of G(N; 1, s) and
$d(N) = \text{minimum}_s \{d(N; 1, s)\}$

Wong and Coppersmith [WC74] gave a lower bound $(\sqrt{(2N)}-3)/2$ for d(N). Boesch and Wang [BW85] made the bound tighter to $lb(N) = \lceil(\sqrt{(2N-1)}-1)/2\rceil$, where $\lceil x \rceil$ denotes the minimum integer $\geq x$. However, the lower bound lb(N) may not be achievable for all values of N. For example, Du et al. [DHL+90] showed that for $N = 24$, $d(24) = 4$ with $s = 7$, but $lb(24) = 3$. The graphs whose diameters are equal to d(N) are *optimal* for the given value of N, and those whose diameters are equal to lb(N) are called *tight optimal.* Thus, a graph G(N; 1, s) may be optimal for some s, but may not be tight optimal if $d(N) > lb(N)$. Du et al. gave some classes of values of N, for which the lower bound lb(N) can be achieved. They also gave some other classes, for which the lower bound cannot be achieved but an optimal choice was given for such graphs.

2.3 Hypercubes

The hypercube interconnection scheme is a very popular network topology. An n-dimensional hypercube Q_n consists of $N = 2^n$ nodes interconnected as follows :

i) each node is labeled by an n-bit binary number $(a_1 a_2 \dots a_n)$,

ii) two nodes are connected by a link if and only if their binary labels differ in exactly one bit position.

The n-cube has become an interesting topic of research in recent years due to its versatile applications in parallel and distributed processing. An n-cube with $N = 2^n$

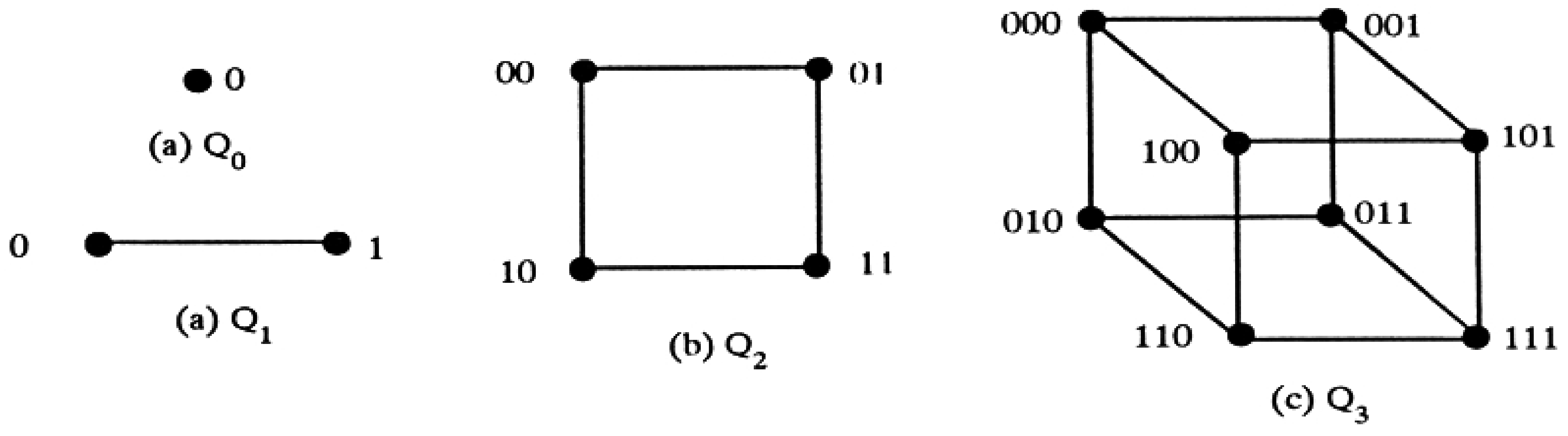


Figure 2.1 : Examples of hypercube networks of lower dimensions

nodes has the following properties :

i) It has diameter $n = \log_2 N$.
ii) Degree of each node is n.
iii) It has a very symmetric structure. Any two nodes are isomorphic with each other.
iv) An n-cube consists of two disjoint (n−1)-cubes. An alternative definition of n-cube is $Q_n = Q_{n-1} \times K_2$, for $n > 0$ where Q_0 is an isolated node, and '×' denotes the Cartesian product of two graphs [Ha69].

Figure 2.1 shows examples of hypercubes of dimension 0, 1, 2 and 3. Many interesting properties of the n-cube have been reported in the literature [AG81], [AP89], [Le93].

In [AL90], it has been shown that by adding $\binom{4m}{m}$ extra links to a 4m-dimensional cube ($m \geq 2$), its diameter can be reduced by 2m−1. In [TW91], the effect of adding some extra links on the performance measures, such as diameter, mean internode distance, traffic density etc. have been discussed.

Two links of a hypercube are called *independent* if they are not incident on a common node. For example in figure 2.2 links (1000, 1001) and (1100, 1101) are two independent links. In [ENS91] it has been shown that by exchanging a pair of independent links in a 4-cycle of an n-cube ($n \geq 3$), known as twisting, its diameter can be reduced by 1. Esfahanian et. al. [ENS91] have shown that by replacing two links by two new links, the diameter of an n-cube ($n \geq 2$) can be reduced by one. Figure 2.2 shows a twisted hypercube of dimension 4. In figure 2.2 the links (0000, 0100) and (0010, 0110) have been twisted to get links (0000, 0110) and (0010, 0100). In [HKS87], it is shown that by exchanging $(n-1)2^{n-4}$ link pairs in a n-cube (n

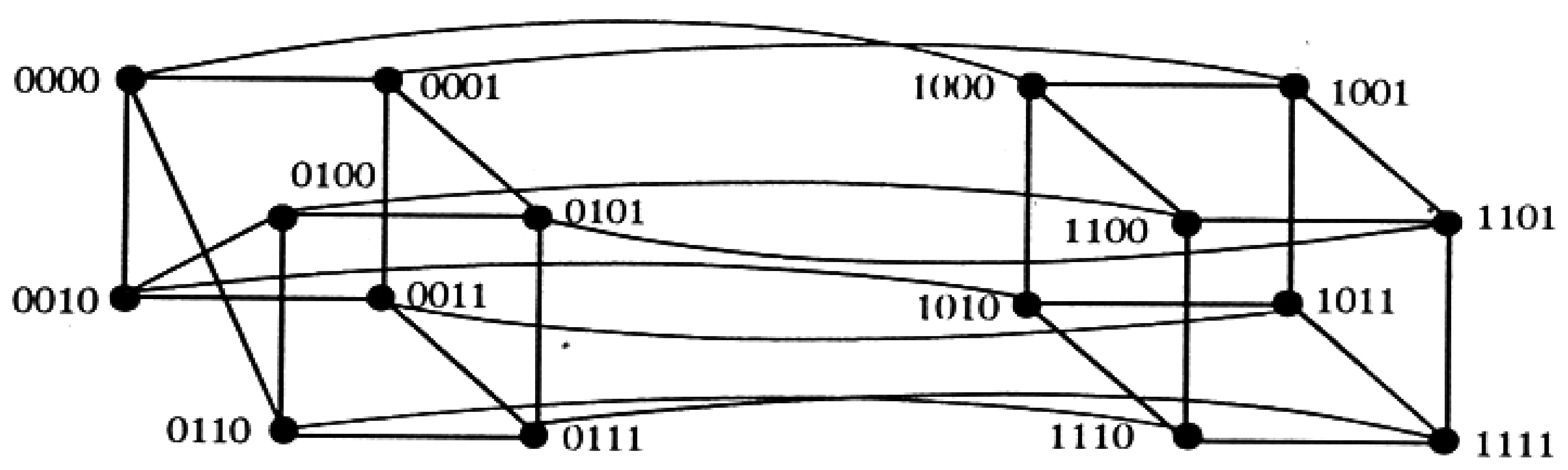


Figure 2.2 : A twisted hypercube of dimension 4.

= 2m+1), its diameter can be reduced to (n+1)/2. In [AP91], performance measures in terms of routing, fault-tolerance etc., of such twisted cubes have been studied.

An n-dimensional hypercube has exactly 2^n nodes, thus strongly restricting the possible system sizes. So peaple have considered what is called an *incomplete hypercube,* where the number of nodes can be more flexible [Ka88], [Tz90]. Katseff has presented simple and deadlock-free algotithms for routing and for broadcasting messages in the incomplete hypercube with an arbitrary number of nodes [Ka88]. Tzeng analyzed the structural properties in incomplete hypercubes with size $2^n + 2^k$, $0 \leq k < n$, i.e., incomplete hypercube composed of two complete hypercubes [Tz90].

2.4 Fault-Tolerance and Reliability

In designing or selecting a network topology, one fundamental consideration is system-level fault-tolerance. At this level, the types of faults to be tolerated are processor (node) or link failures. The system is said to be *fault-tolerant* if it can remain *operational* in the presence of failures. It is, however, the functional requirements on the topology set by the application environment that determine when a network is considered operational.

Two basic operationality criteria have widely been accepted in the literature. According to one of these criteria, a network is considered operational as long as a certain topology is logically embedded in the system. The desired topology may be a hypercube, ring, binary tree etc. Much work has been done in developing parallel algorithms and the best topologies for their executions [Qu87], [Le93], [Já92].

The second functionality criterion considers the network operational as long as there is a non-faulty communication path between each pair of non-faulty nodes. One simple measure for fault-tolerance is the node-connectivity or the link-connectivity of the underlying network graph. Though this measure is relatively easier to compute, it does not reflect the complete picture. For example, a proper subgraph of a graph may have the same node- or link-connectivity.

Another approach is to use a probabilistic model. Here each node and each link is assigned a probability of failure. The measure of fault-tolerance is the network *reliability*, which is defined as the the probability that the system is connected. A popular simplification in this model is the assumption that the failures are independent. Also, for the sake of simplicity, most of the works available in the literature either considers faulty links with fault-free nodes or faulty nodes with fault-free links.

The reliability problem where the links fail independently and nodes are perfectly reliable has been studied extensively in the literature for directed as well as undirected graphs. Let V(G) be the set of network nodes. Consider the general case where every operational pair in a set of nodes K, $K \subseteq V(G)$, needs to communicate with each other. The reliability of the network is defined as the probability that there is subgraph G′ of G whose nodes are operational and all operational nodes in K lie in the same component of G′. In the special case when $K = V(G)$ the reliability is known as the *all-terminal reliability* or the *residual node connectedness problem* [El92]. For $|K| = 2$, the reliability is called the *two-terminal reliability*. Valiant [Va79] has shown that computing the two-terminal reliability is a #P-complete problem. Ball [Ba80], and Provan and Ball [PB83] have proven similar complexity results assuming different forms of reliability evaluations and approximations. Subsequently, Provan [Pr86] has shown that the problem remains NP-hard even if G is a planar graph or an acyclic graph and $|K| = 2$. In view of the apparent intractability of the problem, many researchers have focussed on developing efficient algorithms for computation of reliability for restricted classes of graphs and efficiently computable lower and upper bounds for reliability. Some results in this effort have been reported by Colbourn [Co87a]. Ball and Provan [BP82] reported one such class of reliability bounding techniques for a restricted class of graphs. Satyanarayana et al. [SSS92] introduce an interesting transformation called the *swing surgery*. If H is the graph obtained by deleting m independent links from the complete graph, then for any other graph G with the same number of nodes and the same number of links as in H, $R(H, p) > R(G, p)$ for all $0 < p < 1$, where $R(G, p)$ denotes the all-terminal reliability of G with all nodes having identical failure rate p.

Another important case which has received much attention is the case when nodes fail independently but the links are fault-free. Similar complexity results for this case can be obtained from the unreliable link case, by replacing each unreliable link by two reliable links incident on a new unreliable node. AboElFotoh and Colbourn [AC90] have shown that a variant of the problem where nodes in K are perfectly reliable remains NP-hard for chordal graphs and comparability graphs. Sutner et al. [SSS91] proved that the residual node connectedness reliability problem is NP-hard for split graphs and bipartite planar graphs.

Another measure of network reliability is the *network resilience*. The resilience of a network is defined as the expected number of pairs of distinct nodes that can communicate. This is also a #P-complete problem, even when the network is planar and the nodes are fail-safe. The apparent complexity of computing the resilience has led to the development of efficient algorithms on the class of partial 2-tree and k-tree networks [Co87b], [Ma91], [Sl87]. Mata-Montero [Ma91] describes a linear time algorithm to compute the resilience of partial k-tree networks given with a suitable embedding in a k-tree (for a fixed k).

Najjar and Gaudiot proposed an alternative definition for network resilience [NG90]. According to them, for some given value p, $0 \leq p \leq 1$, *Network Resilience NR(p)* is ‘‘the maximum number of node failures that can be sustained while the network remains connected with a probability $(1 - p)$’’. *Relative Network Resilience RNR(p)* is defined as NR(p) / N, where N is the number of nodes in the network. They showed that the single-node disconnection probability is the dominant factor irrespective of the topology under consideration. They derived an analytical approximation for the disconnection probability and verified it with a Monte Carlo simulation.

In case the components are associated, the assumption of independence of failures leads to underestimation of reliability if the system is in series, whereas the converse holds for parallel systems. Egeland and Huseby [EH91] consider monotone systems and study the error resulting from the independence assumption when the component states are, in fact, distributed to certain dependence model. Two models considered by them are the *shock model* and *standby model*. Shock model has also been considered by others [BS84], [Hu86].

2.5 Dynamic Interconnection Networks

Extensive studies on dynamic interconnection networks indicate the following issues as being the most fundamental in the design of interconnection networks :

i) network topologies,
ii) permutation capabilities
iii) routing
and iv) fault-tolerance.

In general, a dynamic interconnection network can be depicted by a graph, in which a node represents a switching point and a link represents a communication link. The overall graph representation is called the *network topology*. Many topologies have been proposed so far for use in parallel computer systems.

Characteristics used to categorize the topologies of dynamic interconnection networks are namely, the *stages of switching elements* and *functional capability*.

According to *stages of switching elements*, dynamic interconnection networks can be characterized into two categories : *single stage* and *multistage interconnection networks*

The recirculating shuffle-exchange network [St71] is an example of a single-stage network. Multistage interconnection networks (MIN's) include baseline [WF80], omega [La75], delta [Pa81], indirect-binary-n-cube [Pe77], flip [Ba76], banyan [GL73], data manipulator [Fe74], cube [SM81], Benes [Be62], Clos [Cl53] and Cantor [Ca71] networks.

Depending upon the capability of a dynamic interconnection network (IN) to achieve different permutations of input-output combinations, dynamic IN's may be classified into a) *blocking* and b) *non-blocking*. If we assume that input-output connection requirements are coming one by one, in blocking IN's it may so happen that a particular input-output connection request can not be satisfied till some of the existing connections finish their communications. In other words, some input-output connection request may be blocked by the existing connections. In such networks, multiple passes may be necessary to achieve all possible permutations from input to output lines. In *blocking networks*, simultaneous connections of more than one path may result in conflicts in the use of communication links [WF80], [La75], [Pe77], [Ba76], [GL73], [Fe74], [SM81] and [Pa81]. MINs for which, each source

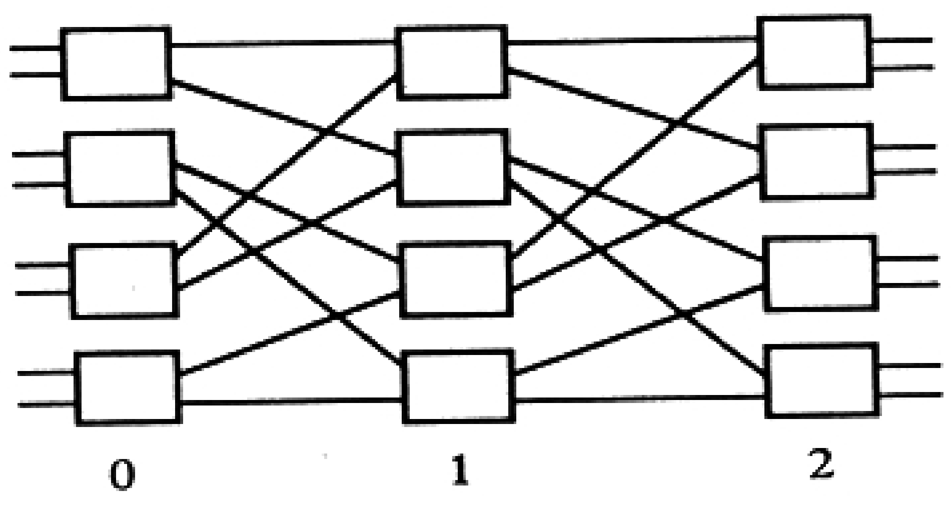


Figure 2.3 : An 8×8 omega network

may reach every possible destination (full-access) and exactly one path exists between any source-destination pair (unique-path), are referred to as *unique-path full-access MIN's* [RV86]. Baseline, omega, cube, reverse baseline and inverse omega are some renowned examples of this class. Figure 2.3 shows an example of an 8×8 omega network. Some IN's are blocking but *rearrangeable.* In rearrangeable networks, a request for a particular input-output connection may not be possible, because of the existing connections; but by rearranging the existing connections it is always possible to accommodate a new request. Benes network is the most popular rearrangeable network. However, the setting of switches to realize the paths requires time $O(N \log_2 N)$ in contrast to its propagation delay $O(\log_2 N)$. Figure 2.4 shows an 8×8 Benes network. In non-blocking IN's, it is always possible to accommodate a new request between a free input line and a free output line, without disturbing the existing connections. Extensions have been done to consider non-blocking networks for one-to-many connections as well [Th78].

The fact that blocking MIN's can not realize every possible connections, due to conflicts in links, leads to some studies on functional relations among such networks. Siegel and Smith examined and demonstrated some functional relations among a number of MIN's [SS78]. In [Si79], explicit maps are constructed to provide a simulation environment within a class of MIN's. Wu and Feng further examined

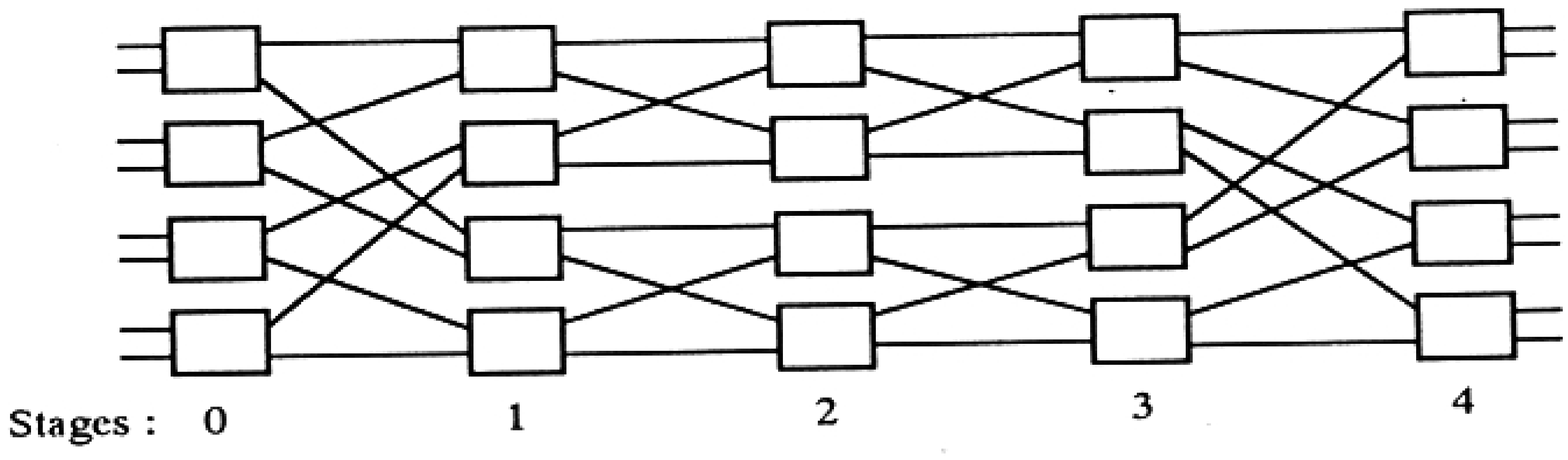


Figure 2.4 : An 8×8 Benes network

the relations among existing MIN's and introduced the notion of *topological* and *functional equivalence* [WF80], [WF80a]. According to them, two MIN's are *topologically equivalent,* if one can be obtained from the other by properly permuting switching elements and associated links within the same stage. Some studies on topologically non-equivalent MIN's are also reported in [AKS88]. The functional equivalence is a more practical notion as it amounts to just renaming the terminal nodes of a MIN without disturbing its internal switching structures. The research efforts reported in [PK80], [Ag83] and [OO85], made significant contributions to the understanding of functional equivalence relations among interconnection networks. Many of these studies, however, have been limited to MIN's with 2×2 switches only.

A different and more general problem is addressed in [OO85], [OOB85] and [Sr89]. They tackled the problem of determining equivalence between any two arbitrary interconnection networks with N inputs and N outputs and realizing k permutations each and construct the maps which conjugate one network onto the other [Sr89].

MIN's are usually used to interconnect a set of processors among themselves or a set of processors with a set of resources. In an interleaved memory organization [Hwang Briggs], different permutations of memory-processors may be neded to run a given algorithm. In a particular environment, only a typical subset of the set of all permutations may actually be needed. But, larger the set of permutations achievable by the MIN, the more powerful and flexible is the system.

The permutation capability of a MIN generally refers to the fraction of all possible permutation requests that can be realized without blocking [SH87]. It is needless to mention that permutation capability is an important parameter in the evaluation of performance of an IN.

Though non-blocking networks are better from the point of view of performances, they are more expensive too. Two examples of non-blocking IN's are crossbar [Fe81] and Clos networks [Cl53]. In a crossbar network, the hardware cost is $O(N^2)$ for N inputs and N outputs. In a 3-stage Clos network C(n, m, r), the number of crosspoints for $n = \sqrt{N}$ is, $6N^{3/2} - 3N$, i.e., $O(N^{3/2})$. Compared to this, an N×N Benes network, which is rearrangeable, has cost $O(N \log_2 N)$. Another non-blocking network is the Cantor network [Ca71]. Cantor network consists of several layers of N×N Benes networks. Each input line is connected to all the corresponding positions in the different layers. Similarly each output line is connected to all the multiple layers.

Now, the nonblocking and the rearrangeable IN's can perform all possible permutations in a single pass, whereas, an N×N blocking MIN, like baseline, omega, cube or any other equivalent networks, can realize only $N\,2^{N/2}$ permutations out of total N! permutations in single pass [AA80], [AS82]. Therefore, depending on the set of permutations necessary in a system, one can select an appropriate MIN. It has been shown that perfect shuffle is an important interconnection pattern for a parallel processor, solving FFT (Fast Fourier Transform), polynomial evaluation, sorting and matrix transposition [St71]. Batcher have studied the flip network in STARAN; the allowable sets of permutations are barrel shifts, barrel shifts on substrings and FFT-butterflies, which are useful for sorting, FFT, image wraping and solving partial differential equations on multi-mesh regions [Ba76]. Seznec [Se87] proposed a new interconnection network, namely the Sigma network, which is essentially an omega network followed by an inverse omega network, very much like Benes network. He has shown that in SIMD machines, this MIN can implement a general family of permutations, covering the standard needs of scientific vector processing, such as rearrangement, compression, expansion, perfect-shuffle, bit-reversal etc. using a simple and efficient control algorithm. However, in real applications, it is not enough to know that a particular permutation is realizable in a MIN, but it is also important to find how efficiently it can be routed in the MIN.

Ultimately, the universality of an IN, i.e., the capability of it to realize every possible permutation function becomes a concern. In blocking MIN's, one can achieve universality by increasing the number of passes, such that conflicting transmissions are realized in different passes [WF80], [WF81]. The shuffle-exchange connection provides an efficient interconnection scheme for parallel computation for many problems [St71], [WF80]. A single-stage multiple pass recirculation technique is an economic way to construct these connections to realize any arbitrary permutation. An asymptotic lower bound of $(2n - 1)$ stages of 2×2 switching elements for rearrangeability has been known for long [Wa68], $n = \log_2 N$. Now an interesting problem is that, whether $(2n - 1)$ stages of shuffle-exchange are necessary and sufficient for universality. Stone [St71] observed that using an algorithm due to Batcher [Ba68], sorting of arbitrary sequences of data can be performed in n^2 passes. Parker has shown that 3n passes are sufficient [Pa80] and Wu and Feng provide an upper bound of 3n–1 passes [WF81]. In [HT86], it has been proved that 2n–1 passes are necessary and 3n–3 passes are sufficient to realize all permutations. Raghavendra and Verma established an upper bound of 3n–4 passes for rearrangeability [RV87]. For some well-known classes of permutations, such as BPC, LC etc., it has been already shown that (2n–1) stages of shuffle-exchange are sufficient [Na89], [RB91]. The rearrangeability of (2n-1)-stage shuffle-exchange network was undecided for a long time. Recently Abdennadher and Feng [AF92] have proved some general results regarding the rearrangeability of MIN's. Using them, them have shown that omega-omega and (2n-1)-stage shuffle-exchange networks are rearrangeable.

Every dynamic interconnection network needs a control strategy to route data or messages from a source to various destinations. Usually, in MIN's with dynamic topology, a path is to be established before data is actually transmitted over the path. Therefore, the routing problem must be solved in the path connection phase by specifying switch settings in the path. The controls for setting the switches are usually derived from source and destination addresses.

Nonblocking networks do not need extensive care on their control algorithms, since in such networks, no input-output connection block any other path. But control strategies for blocking and rearrangeable networks are quite involved due to sharing of common links among different input-output paths.

For the full-access unique-path class of MIN's, destination tag routing is widely used to realize any input-output connection [La75], [WF80], [WF80a], [RV86]. But problem arises, when more than one simultaneous paths conflict with each other i.e., they demand some common links in the network and therefore can not be realized in the same pass. Hence, for cost-effective routing, it is important to determine, how a given set of input-output paths can be realized in minimum number of passes. This is known as the *conflict resolution problem* [WF80], [Ag83], [RV86], [BR91]. For an N×N MIN, maximum N transmission paths are to be set up simultaneously, which can be represented as an N×N permutation (assuming that communications from input to output are one-to-one and onto). Given any arbitrary permutation P : inputs $(0, 1, \ldots, N-1) \rightarrow$ outputs $(0, 1, \ldots, N-1)$, the conflict resolution problem is equivalent to the problem of partitioning P into a minimum number of subsets, such that transmissions in each subset are conflict-free and hence are routable in the same pass. Unfortunately, the problem, in general, is NP-hard [BR91].

A non-optimal conflict resolution scheme has been proposed in [WF80]. In [Ag83], an upper bound of $2^{\lfloor n/2 \rfloor}$ passes was derived for realizing any arbitrary permutation on the omega network. Optimal solutions are reported only for well-known classes of permutations.

In [RV86], given any N×N permutation P, the conflict information is represented by a graph, namely the *conflict graph* G(V, E), such that G consists of N nodes, each representing a transmission and two nodes are adjacent if and only if the two transmissions are in conflict. With this model, the problem of conflict resolution is mapped to the well-known graph-coloring problem. For BPC (bit-permute-complement) class of permutations in delta network, they observed some typical characteristics of conflict graphs and hence developed an algorithm for optimal routing of permutations of that class.

The (s,d) mask formalism is introduced in [BR91]. It represents a message pattern, that is a generalization of BPC permutations. The optimal routing for this class of permutations on bundled omega network has been solved in [BR91].

However, the set of permutations for which the problem has been solved optimally includes a very small fraction of the whole set of possible permutations and as N increases, this fraction asymptotically becomes vanishingly small.

An $O(N^3)$ heuristic algorithm for routing arbitrary permutations in baseline network is developed in [DF87]. Therefore, the search for optimal number of passes required to realize any arbitrary permutation still continues.

In rearrangeable networks, every possible permutation is realizable in a single pass but the routing is not very straight forward. For any input-output pair, there exists a number of paths and depending on the whole permutation, an appropriate path is to be established for each, such that all paths are conflict-free.

Benes network [Be62] is a well-known rearrangeable network and the best known routing algorithm for an N×N Benes network is of time complexity $O(Nn)$ on a uniprocessor system [Wa68], compared to its propagation delay $O(n)$ only, where $n = \log_2 N$. Even with parallel setup algorithms the time needed to realize any permutation is dominated by the setup time [NS81], [LPV81].

However, in practice, it is found that, the permutations used very often in a parallel computer system, are in general regular in structure and may be routable by some simpler algorithms. Extensive studies have been made to develop fast routing algorithms for many useful classes of permutations required in parallel computations. A parallel control algorithm for the *Frequently Used Bijections* (FUB) have been developed in [Le78]; it includes $O(2^{2n})$ permutations only out of total $2^n!$ all possible permutations. A simpler algorithm for self-routing a class of permutations F, that includes BPC (bit-permute-complement) class, IΩ (inverse omega) class and the 'FUB' families, is proposed in [NS81]. It has been shown there, that for any permutation in class F, one need not look over the whole permutation to find a conflict-free path but the switching elements independently can set up their configuration of their own. A new self-routing technique with the same idea has been presented in [RB91] that can route the LC (linear-complement) class of permutations as well as IΩ class of permutations. Das et al. [DBD90] have introduced the *group interchanges* and *group transformations* on the set of permutations. They have shown that the conflict graph remains isomorphic under group transformations. Thus the existing algorithms for the conflict resolution problem can be extended to much bigger classes. For instance, the algothms applicable to the BP (bit-permute) or BPC classes are extendable to a much bigger class called the BPCL (bit-permute-closure) class.

Many research works have been reported so far, presenting different self-routing techniques in rearrangeable networks. However, the scope of these techniques have not been explored fully; it has always been mentioned that the proposed algorithm may realize many more permutations than it has been specified in the literature.

However, the blocking and rearrangeable network structures considered so far, have evolved with the idea of optimizing the number of switching elements used in the system (an N×N IN consists of $O(N \log N)$ switches). It has led to an interesting paradox. The recent research results appear to conjecture that the setup algorithm to realize any arbitrary permutation on these networks requires at least $O(N \log N)$ sequential time in contrast to its proapgation delay $O(\log N)$ only. It indicates that optimizing the performance of an IN over the number of its switches alone is not necessarily the best strategy.

Chapter 3
Optimal Loop Networks

3.1 Introduction

Loop networks present a good improvement over the simplistic ring topology. It preserves the desirable properties of symmetry, expandability, uniform building blocks for switching mechanism and uniform token passing. Diameter of a loop network depends on the choice of the jump size. Given the number of nodes, say N, one may be interested to know whether it is possible to select a value for the jump such that the lower bound on the diameter is achieved. Another important problem is to find a good algorithm for finding a shortest path between two arbitrary nodes. It would be desirable that the routing algorithm takes care of any fault that may occur in the network.

The tight optimal classes mentioned in [DHL+90] are not exhaustive. The complete characterization of the chord lengths for the design of the tight optimal double-loop network is yet to be solved. Apart from the choice of the jump-size s, there are also some other problems related to the loop networks, which remain to be solved. Du et al. suggested some problems. Two of them are :

1) Classify those N's for which the tight optimal loop networks can be found.
2) Study the routing algorithms for $G(N; 1, s)$ when there is no fault and also when a node or a link fails.

About the classification, we suggest some classes of values for N, which achieve the lower bound on the diameter. These classes cover a large class of values of N. They also include many N's not classified by Du et al. Then we focus our attention on the problem of routing. We propose an algorithm to find a shortest path between any two nodes. We also propose how to find a near optimal path (not more than 1 over the optimal) in case of a single node or a link failure.

3.2 Optimal Design Criteria

As we have mentioned in chapter 2, circulants have been studied quite extensively. Let $C_N(s,t)$ represent the graph on N nodes labelled $V_0, V_1, V_2, \ldots, V_{N-1}$ such that node V_i is connected to V_{i+s}, V_{i-s}, V_{i+t} and V_{i-t}. Boesch and Wang [BW85, Theorem 5] have found out that for $N > 6$, $d(N; lb(N), lb(N)+1) = lb(N)$. We try to use the above result, when it is given that one of the jumps is 1 and we have to minimize the diameter over the other jump of length t.

We shall refer to a link from V_i to V_{i+x} as an x-jump, $x = s$ or t.

Lemma 3.1 : If $\gcd(N, s) = 1$, then in $C_N(s, t)$ there is a hamiltonian cycle using only s-jumps.

Proof: If we prove that for $0 < m < n < N$, $ms \bmod N \neq ns \bmod N$, then V_0, V_s, $V_{2s}, \ldots, V_{Ns} = V_0$ gives us the required hamiltonian cycle.

Let, $ms \bmod N = ns \bmod N$ for $0 < m < n < N$. Then $(m - n)s = kN$ for some $k > 0$. But as $\gcd(N, s) = 1$, N must divide $(m - n)$ where $0 < m - n < N$, which is a contradiction.

It can easily be shown that the other way implication is also true. ◆

Lemma 3.2 : If $\gcd(N, lb(N)) = 1$ then $C_N(lb(N), lb(N) + 1)$ is equivalent to G(N; 1, s) for some s.

Proof : Since $\gcd(N, lb(N)) = 1$, by lemma 3.1, $V_0, V_{lb(N)}, V_{2.lb(N)}, \ldots, V_{N.lb(N)}$ form a hamiltonian cycle in $C_N(lb(N), lb(N)+1)$. We relabel the nodes $V_{i.lb(N)}$ as V_i for $0 < i < N$. Obviously, there are links from V_i to both V_{i+1} and V_{i-1}. Now as (i.lb(N) mod N) are all distinct for $0 \leq i < N$, choose j such that $j.lb(N) \bmod N = lb(N)+1$. We have links from V_i to V_{i+j} for $0 \leq i < N$. This implies that we have link from V_{i-j} to V_i also. That is, $C_N(lb(N), lb(N)+1)$ is equivalent to G(N; 1, j). ◆

Lemma 3.3: If $\gcd(N, lb(N)+1) = 1$ then $C_N(lb(N), lb(N)+1)$ is equivalent to G(N; 1, s) for some s.

Proof : Similar to that of lemma 3.2. ◆

If we combine lemmas 3.2 and 3.3 with Theorem 5 of Boesch and Wang [BW85], we have the following result.

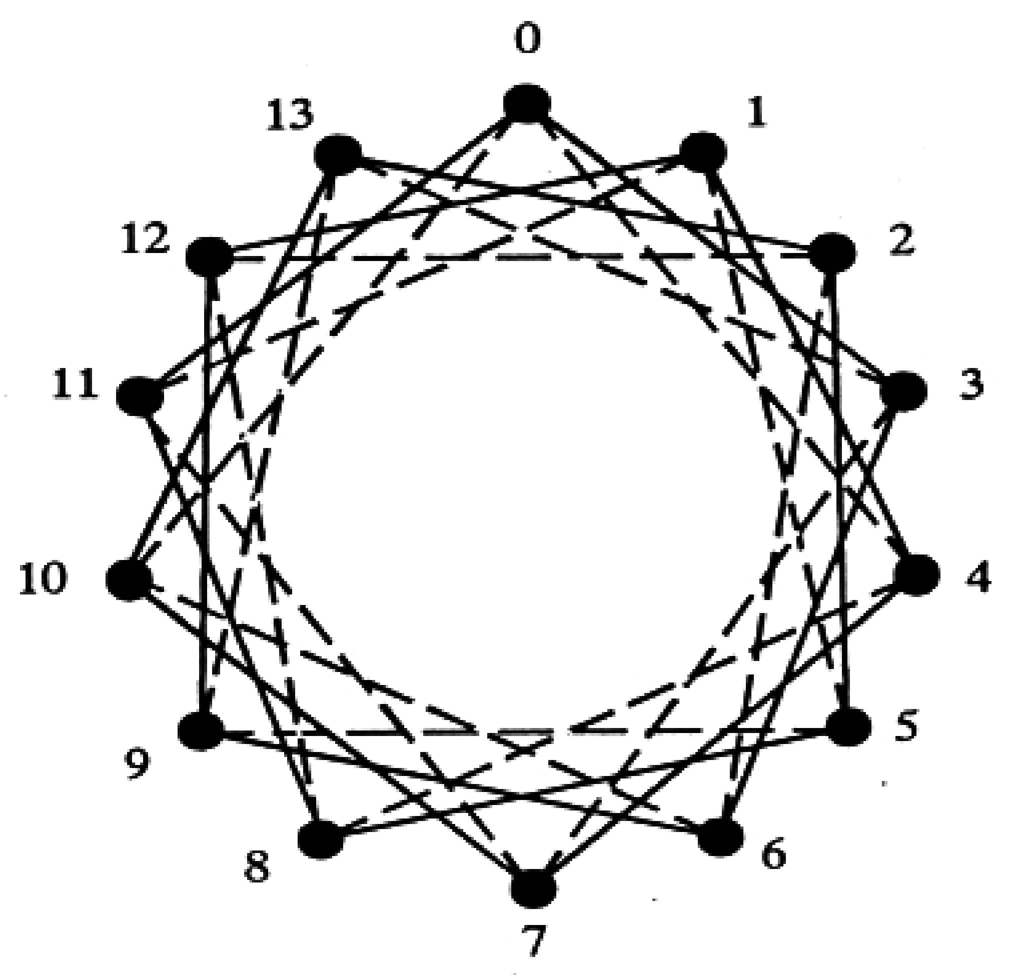


Figure 3.1(a) : G(14;3,4)

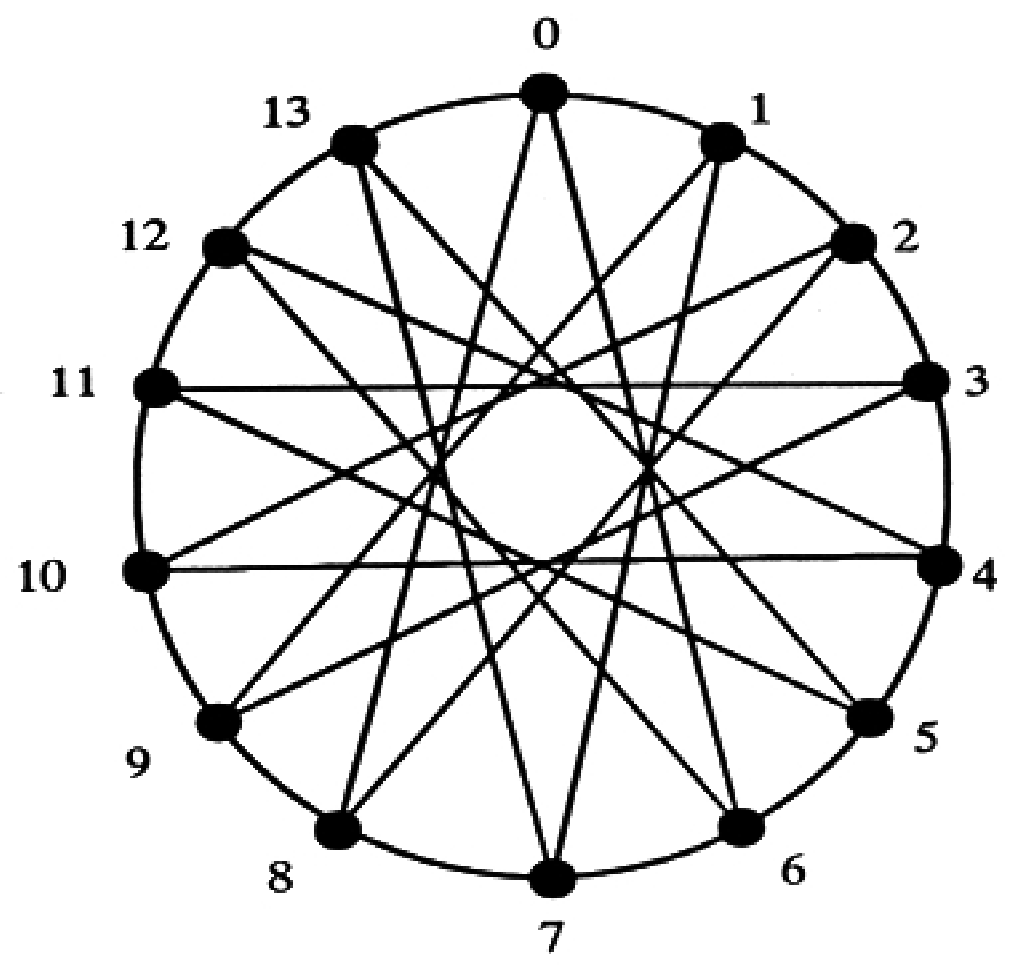


Figure 3.1(b) : G(14;1,6)

Theorem 3.1: For $N > 6$, if $\gcd(N, lb(N)) = 1$ or $\gcd(N, lb(N)+1) = 1$, then $d(N) = lb(N)$.

Example 3.1 : Consider the case of $N = 14$ nodes. Here we see that $lb(14) = 3$, i.e., $\gcd(N, lb(N)) = 1$. So, by theorem 3.1, G(14; 3, 4) has diameter 3 and it can be redrawn as G(14; 1, s) for $s = 6$.

In figure 3.1(a), we see G(14; 3, 4) with the 4-jumps shown in broken lines. Here nodes 0, 3, 6, 9, 12, 1, 4, 7, 10, 13, 2, 5, 8, 11, 0 form a hamiltonian cycle. In figure 3.1(b) these have been relabelled as 0, 1, 2, ... , 13, 0. The 4-jumps are converted into 6-jumps.

Using theorem 3.1, we get an excellent coverage over the possible values of N, the number of nodes in the network. We have exhaustively searched the optimal designs upto N=16,000 and found that the tight optimal designs can be obtained for more than 80% of the values of N by following the scheme of theorem 3.1. For $7 \leq N \leq 5305$, the classes given by Du [DHL⁺90] cover only about 13% of the values of N, whereas theorem 3.1 covers about 88.6%; nearly 10% of the values remain unclassified by either scheme. As N increases, the classes defined by Du give lesser coverage.

3.3 Optimal Routing

For two nodes with a link connecting them, communication is carried out through that link. In absence of a direct link, the message is transmitted through some intermediate nodes. The number of links traversed in such a path represents the transmission delay. So for any two nodes, it is important to find a path with minimum number of links. Such a path is called a *shortest path.* Note that shortest path between two nodes may not be unique. Here we consider the problem of finding a shortest path from V_i to any arbitrary node V_j. We note that because of the symmetry in the underlying topology it is enough to consider the problem of finding a shortest path from V_0 to an arbitrary node V_u.

For our convenience we shall differentiate between two s-links from V_m, depending on whether they are used to go to V_{m+s} or V_{m-s} by using a '+' or '–' sign respectively. Similarly we define +1 and –1 links. Consider a path involving w, x, y and z (all non-negative integers) number of [+s], [–s], [+1] and [–1] links respectively. Let the endpoints of the path be V_i and V_j. Then the relation '$j = (ws - xs + y - z) \bmod N$' holds irrespective of the order in which the links appear in the path. Since we are interested only in the lengths of the paths, we shall denote such a path by *(w)[+s] + (x)[–s] + (y)[+1] + (z)[–1].*

Lemma 3.4 : Let (w)[+s] + (x)[–s] + (y)[+1] + (z)[–1] be a shortest path from V_i to V_j. Then at most one of w and x and at most one of y and z is non-zero.

Proof : Let both w and x be non-zero. Without loss of generality let $w \geq x$. Consider the path (w–x)[+s] + (0)[–s] + (y)[+1] + (z)[–1]. As (w)[+s] + (x)[–s] + (y)[+1] + (z)[–1] was a path from V_i to V_j, $(i + w.s - xs + y - z) \bmod N = j$. Hence (w–x)[+s] + (0)[–s] + (y)[+1] + (z)[–1] is also a path from V_i to V_j and it is shorter than (w)[+s] + (x)[–s] + (y)[+1] + (z) [–1], which contradicts the hypothesis that (w)[+s] + (x)[–s] + (y)[+1] + (z)[–1] is a shortest path. Similarly, at most one of y and z may be non-zero. ◆

In view of lemma 3.4, at most two of w, x, y and z can be non-zero. From now on, we shall drop the terms with zero coefficient.

As a consequence of lemma 3.4, we note that a shortest path from V_0 to V_u would be using either (+s, +1) or (+s, –1) or (–s, +1) or (–s, –1) links. So if we find the shortest of the paths of each combination of links, that path will be the required shortest path. We shall discuss in details a method for finding a shortest path using +s and +1 links. The case of +s and −1 links would be very similar. The other two cases would be the same as finding a shortest path from V_0 to V_{N-u} using (+s, −1) and (+s, +1) links. From now on, by a *(+s, +1)-shortest path* we shall mean a shortest path among the paths using +s and +1 links only.

Lemma 3.5 : Let (w)[+s] + (x)[+1] be a (+s, +1)-shortest path from V_0 to V_u. Then $x < s$.

Proof : If $x \geq s$ then (w+1)[+s] + (x−s)[+1] is a shorter (+s,+1)-path from V_0 to V_u. ◆

Lemma 3.6 : A (+s,+1)-shortest path from V_0 to V_u has at least $\lfloor u/s \rfloor$ number of +s-links.

Proof : If we use less than $\lfloor u/s \rfloor$ number of +s-links, then we have to use more than s number of +1-links. But a group of s number of +1-links can always be replaced by one +s-link. ◆

Let $S_1 = s$ and W_1 be the cost of reaching the node at S_1 from V_0 using +s-links only. That is $W_1 = 1$. For $u > s$, we can use lemma 3.6 and reduce it to a problem of reaching V_u from V_0 with $u < S_1$.

Lemma 3.7 : For $u < s$, the number of +s-links in a (+s, +1)-shortest path is either zero or at least $W_2 = (\lfloor N/s \rfloor + 1)$.

Proof : Let (w)[+s] + (x)[+1] be a shortest path from V_0 to V_u for some w, $0 < w < W_2 = \lfloor N/s \rfloor + 1$. Then $s \leq ws \leq N$. Length of this path is $w + (u + N - ws) > u$. But (0)[+s] + (u)[+1] is a (+s, +1)-path between V_0 and V_u and its length is u. Hence the contradiction. ◆

Remark : W_2 is the cost of reaching the node at $S_2 = (W_2.s) \bmod N = s(\lfloor N/s \rfloor + 1) - N$, from V_0 by using +s-links only. Clearly, $S_2 = s - N \bmod s \leq s = S_1$.

Now, if $S_1 > u \geq S_2$, and $W_2 < S_2$, then we may use W_2 +s-links from V_0 to reach the node number S_2. We may use groups of W_2 +s-links repeatedly, till we would reduce the problem to one of routing to a node within S_2 distance. If, however, $W_2 > S_2$, then (+s, +1)-shortest path will not have any +s-link at all, because instead of using W_2 +s-links we can use S_2 +1-links to reach the node at S_2 by a shorter path. If the distance of V_u from V_0 is still $\geq S_2$ we can repeat the replcement of

W_2 +s-links by S_2 +1-links and repeating this we can reach a node within S_2 distance from V_u, by using +1-links only. Again by lemma 3.7, the number of +s-links is either zero or at least W_2. But a direct +1-path has length $u \leq S_2 < W_2$. So we may refine our lemma 3.7 as follows.

Lemma 3.8 : For $u < s$, the number of +s-links in a (+s,+1)-shortest path is

i) zero $\{\text{if } W_2 \geq S_2\}$

or, ii) at least $\lfloor u/S_2 \rfloor W_2$ $\{\text{if } W_2 \leq S_2\}$

Example 3.2 : Consider G(258; 1, 100). Suppose we have to find a shortest path from V_0 and V_{70}. So $N = 258$, $S_1 = s = 100$ and $u = 70$. By lemma 3.8, we may use $W_2 = 3$ +s-links to reach node number $S_2 = 3 \times 100 - 258 = 42$. If we take W_2 +s-links once more, we shall raech V_{84}, crossing our destination. Since we cannot use -1-links, reaching V_{70} from V_{84} using +1-links would not generate a shortest path. What we can do is, we can use $3 \times 3 = 9$ +s-links from V_{42} to V_{168} and use one -s-link (i.e., use $9 - 1 = 8$ +s-links from V_{42}) to reach V_{68}.

Now we proceed to generalize the results in lemmas 3.6 - 3.8. In order to do that we define some terms :

i) $S_0 = N, \quad S_1 = s,$

$S_k = S_{k-1} .(\lfloor S_{k-2} / S_{k-1} \rfloor + 1) - S_{k-2}$

ii) $W_0 = 0, \quad W_1 = 1,$

$W_k = W_{k-1} \cdot (\lfloor S_{k-2} / S_{k-1} \rfloor + 1) - W_{k-2}$

We now describe some properties of the sequences $\{S_i\}$ and $\{W_i\}$.

Lemma 3.9 : The sequence $\{S_i\}$ satisfies the following properties.

i) $S_i \geq 0$.

ii) $S_0 \geq S_1 \geq S_2 \geq \dots$

iii) If $S_i = S_{i+1}$, for some $i \geq 0$, then $S_{i+k} = S_i$ for all $k \geq 0$.

Proof :

i) As $S_{k-1} .(\lfloor S_{k-2} / S_{k-1} \rfloor + 1) \geq S_{k-2}$, the result follws from the definition of S_k.

ii) Again the result follows from the definition and the observation that

$$S_{k-1} (\lfloor S_{k-2} / S_{k-1} \rfloor + 1) \leq S_{k-2} - S_{k-1} .$$

iii) If for some k, $S_k = S_{k-1}$, then

$S_{k+1} = S_k .(\lfloor S_{k-1} / S_k \rfloor +1) - S_{k-1} = 2. S_k - S_{k-1} = S_k$, and so on. ◆

Lemma 3.10 : $W_0 \leq W_1 \leq W_2 \leq \ldots$

Proof : From definition, $W_0 \leq W_1$. Let $W_0 \leq W_1 \leq \ldots \leq W_{k-1}$. As $S_{k-2} \geq S_{k-1}$ (lemma 3.9), $W_k \geq 2.W_{k-1} - W_{k-2} \geq W_{k-1}$. ◆

Lemma 3.11 : $(W_i. s) \bmod N = S_i$ for $i = 0, 1, 2, \ldots$

Proof : The result is easily verified to hold for $i = 0$ and 1.
Let the result be true for $i = k$, for some $k > 0$.

Then, $(W_{k+1}.s) \bmod N$

$= [(W_k \cdot (\lfloor S_{k-1} / S_k \rfloor +1) - W_{k-1}). s] \bmod N$ {From definition of W_{k+1}}

$= (S_k \cdot (\lfloor S_{k-1} / S_k \rfloor +1) - S_{k-1}) \bmod N$ {By induction hypothesis}

$= S_{k+1}$

Theorem 3.2 : For $1 \leq p < W_i$, $ps \bmod N \geq S_{i-1}$, $i = 2, 3, \ldots$

Proof : For $i = 2$, $W_2 = \lfloor N/s \rfloor + 1$. Take p such that, $1 \leq p \leq \lfloor N/s \rfloor$. So, $s \leq ps \leq N$, i.e., $ps \bmod N \geq s = S_1$.

Let the result be true for $2 \leq i \leq k$. We shall show that the result holds for k+1.

Let $1 \leq p < W_{k+1} = \lfloor S_{k-1} / S_k \rfloor W_k + (W_k - W_{k-1})$. From induction hypothesis, the result is true for $1 \leq p < W_k$. For $p = W_k$, $ps \bmod N = S_k$ (lemma 3.11). So we have to consider only $W_k < p < W_{k+1}$. Suppose the result is not true. Then there exists $W_k < p < W_{k+1}$ such that $ps \bmod N = v$ for some $0 \leq v < S_k$.

Case I : $p = x.W_k + y$, where $x < (\lfloor S_{k-1} / S_k \rfloor)$ and $y < W_k$. So,

$$\begin{aligned}
& [ps + (W_k - y) s] \bmod N \\
&= [(x. W_k + y + (W_k - y)) s] \bmod N \\
&= (x+1) S_k
\end{aligned}$$

$$\begin{aligned}
\Rightarrow \quad & [(W_k - y) s] \bmod N \\
&= [(x+1) S_k - ps] \bmod N \\
&= (x+1) S_k - v \\
&\leq x S_k < S_{k-1} \quad \{\text{as } x < (\lfloor S_{k-1} / S_k \rfloor)\}
\end{aligned}$$

But this contradicts the induction hypothesis, as $W_k - y \leq W_k$.

Case II : $p = x\,W_k + y$, where $x = \lfloor S_{k-1}/S_k \rfloor$ and $y < W_k - W_{k-1}$.

$$\begin{aligned} &[(x\,W_k + y + W_{k-1})\,s] \bmod N \\ &= [v + S_{k-1}] \bmod N \\ &= [v + x.S_k + (S_k - S_{k+1})] \bmod N \qquad \{\text{From definition of } S_{k+1}\} \end{aligned}$$

$$\begin{aligned} \Rightarrow\quad &[(y + W_{k-1})]\,s \bmod N \\ &= v + (S_k - S_{k-1}) \end{aligned}$$

As $y + W_{k-1} < W_k$, by induction hypothesis, $v + (S_k - S_{k+1}) \geq S_k$, i.e., $v \geq S_{k+1}$.

Now, $[(p + W_k - y).s] \bmod N = (x+1)\,S_k$.

$$\begin{aligned} \Rightarrow [(W_k - y)s] \bmod N &= (x+1)\,S_k - v \\ &\leq (x+1)\,S_k - S_{k+1} \\ &= S_{k-1} \qquad \{\text{From definition of } S_{k+1}\} \end{aligned}$$

But this contradicts the induction hypothesis, as $W_k - y < W_k$. ◆

Theorem 3.3 : For $u < S_i$, the number of +s-links in a (+s,+1)-shortest path is

i) zero $\{\text{if } W_{i+1} \geq S_{i+1}\}$

or, ii) at least $\lfloor u/S_{i+1} \rfloor\, W_{i+1}$ $\{\text{if } W_{i+1} \leq S_{i+1}\}$

Proof : Let (p)[+s] + (q)[+1] be a (+s, +1)-shortest path from V_0 to V_i. Since (0)[+s] + (u)[+1] is a (+s, +1)-path from V_0 to V_i, $p + q \leq u$. In particular, $q \leq u < S_i$. So, if $p > 0$, $0 < ps \bmod N < S_i$. By theorem 3.2, we must have $p \geq W_{i+1}$.

If $W_{i+1} > S_{i+1}$, then (+s, +1)-shortest path will not have any +s-link at all, because instead of using W_{i+1} +s-links we can use S_{i+1} +1-links to reach the node at S_{i+1} by a shorter path. If the distance of V_u from V_0 is still $\geq S_{i+1}$ we can repeat the replacement of W_{i+1} +s-links by S_{i+1} +1-links and repeating this we can reach a node within S_{i+1} distance from V_u, by using +1-links only. As $S_{i+1} \leq S_i$, again by theorem 3.2, the number of +s-links is either zero or at least W_{i+1}; but a direct +1-path has length $u \leq S_{i+1} < W_{i+1}$.

Now consider the case when $S_{i+1} > W_{i+1}$. If we do not use any +s-link, the length of the path is u. By theorem 3.2, if we use any +s-link, we must use at least W_{i+1} +s-links. If $u > S_{i+1}$ then we may use W_{i+1} +s-links and reach S_{i+1}. Even if we take all the remaining links as +1-links, we have path of length $W_{i+1} + (u - S_{i+1}) < u$. Again, if the distance of V_i from S_{i+1} is more than S_{i+1}, we may further reduce the path length by taking groups of W_{i+1} +s-links repeatedly till we reach within a distance of S_{i+1} from V_i. So, in this process we take $\lfloor u/S_{i+1} \rfloor W_{i+1}$ +s-links. We note that in order to reach V_i we may take some more +s-links. But the number of +s-links is at least $\lfloor u/S_{i+1} \rfloor W_{i+1}$. ◆

By repeated application of theorem 3.3, we can get a (+s,+1)-shortest path as follows.

Algorithm (+s,+1)-shortest path :

```
STEP 1 : $p \leftarrow 0$;  $i \leftarrow 0$;
STEP 2 : $i \leftarrow i+1$; If $W_i > S_i$ then goto step 5;
STEP 3 : If $u < S_i$ then goto step 2;
STEP 4 : $p \leftarrow p + \lfloor u/S_i \rfloor W_i$;  $u \leftarrow u \bmod S_i$; goto step 2;
STEP 5 : $q \leftarrow u - (ps \bmod N)$; output $((p)[+s] + (q)[+1])$; stop.
```

3.4 Routing Under Fault

In this section we consider the problem of routing when one of the nodes (or links) is faulty. We note that the paths as we have defined, specifies only the number of links used for different types of links. It says nothing about the order in which they are traversed. The question is : *can we always bypass the faulty node (link) by some ordering of the links traversed?* In some cases we can bypass the faulty node (link). The result is stated in the following theorem.

Theorem 3.4 : The links of any shortest path $(p)[\pm s] + (q)[\pm 1]$ with $p, q > 0$, can always be ordered in a way such that it does not pass through a specified node.

Proof : Without loss of generality, let the path be $(p)[+s] + (q)[+1]$. We also assume that one of the endpoints of the path is V_0. First we consider the following realization R of the path, where we traverse all the p [+s]-links and then the q [+1]-links. If the faulty node V_f is not on R, then R gives us the path bypassing the faulty node. Suppose V_f is a node on R.

Csae I : $f = zs \bmod N, \quad 0 < z \leq p.$

We consider another realization R' of the path $(p)[+s] + (q)[+1]$, where we first traverse a [+1]-link, then p [+s]-links and lastly the remaining $(q - 1)$ [+1]-links. We claim that V_f is not a node on R'. Suppose V_f is a node on R'. The segments of R and R' from V_0 to V_f must have equal length. Otherwise we can replace the longer segment by the shorter one to get a shorter path. Note that a typical node on R' is V_i, where $i = (ts + 1) \bmod N$, $0 \leq t \leq p$, or $i = (ps + j) \bmod N$, $2 \leq j \leq q$.

Subcase I a : $f = (ts + 1) \bmod N, \qquad 0 \leq t \leq p.$
From the equality of path lengths from V_0 to V_f, we have, $z = t + 1$.
So, $\quad f = (t + 1)s \bmod N.$
$\Rightarrow \quad (ts + 1) \bmod N = (t + 1)s \bmod N = (ts + s) \bmod N$
$\Rightarrow \quad (s - 1) \bmod N = 0,$ Contradiction !

Subcase I b : $f = (ps + j) \bmod N, \qquad 2 \leq j \leq q.$
Again form the equality of path lengths from V_0 to V_f, we have $z = p + j$. But we know that $z \leq p$ and $j \geq 2$. Contradiction !

Case II : $f = (ps + j) \bmod N, \qquad 1 \leq j \leq q - 1.$
Here we take R' to be the realization of the path where first we travarse $(j + 1)$ [+1]-links, then the p [+s]-links and then the rest of the $(q - j + 1)$ [+1]-links. A typical node on R' is V_i, where $i = t$, $0 \leq t \leq j + 1$, or $i = (ts + j + 1) \bmod N$, $1 \leq t < p$ or $i = (ps + t) \bmod N$, $j + 2 \leq t \leq q$.

Subcase II a : $f = t, \qquad 0 \leq t \leq j + 1.$
Again from the equality of path lengths from V_0 to V_f, we have, $p + j = t$.
So, $\quad p + j = (ps + j) \bmod N$
$\Rightarrow \quad (s - 1) \bmod N = 0,$ Contradiction !

Subcase II b : $f = (ts + j + 1) \bmod N, \ 1 \leq t < p.$
From the equality of path lengths from V_0 to V_f, we have, $p + j = t + j + 1$.
So, $\quad ((p - 1)s + j + 1) \bmod N = (ps + j) \bmod N.$
$\Rightarrow \quad (s - 1) \bmod N = 0.$ Contradiction!

Subcase II c : $f = (ps + t) \bmod N, \ j + 2 \leq t \leq q.$
From the equality of path lengths from V_0 to V_f, we have, $p + j = p + t$ or $j = t$.
Contradiction ! ◆

For nodes which do not have a mixed (using both types of links) shortest path, length of a shortest path in faulty situation will be at least one more than that in the fault-free case. We may, however, add one each of '+' and '–' links of the type of link not used in the fault-free shortest path and this path may be at most one link longer than a shortest path in the faulty case.

3.5 Conclusion

In this chapter we have classified many values of N for which tight loop networks exist. Though this gives a much wider coverage than the classes defined by Du et al, some values of N remain yet to be classified. Some further works on the classification have also been reported in the literature [Tz91], [BT91]. We also give an algorithm to find a shortest path between any pair of nodes and a near optimal routing in the presence of single node or link failure.

For improvement over the ring we have considered the addition two chords from every node. One may consider a further generalization where there are 2k chords from every node. Let $G(N; 1, s_1, s_2, \dots, s_k)$ denote the supergraph of ring where from each node V_i there are links to the nodes $V_i \pm 1, V_i \pm s_1, V_i \pm s_2, \dots, V_i \pm s_k$. Below, we list some of the problems which remain to be solved.

1) Deriving an analytical formula for the diameter of G(N; 1, s),
2) Design of optimal loop networks for all values of N,
3) Optimal routing under single as well as multiple faults,
4) Analysis of generalized loop networks $G(N; 1, s_1, s_2, \dots, s_k)$ etc.

Chapter 4
Bridged and Twisted Hypercubes

4.1 Introduction

The hypercube interconnection scheme is a very popular network topology. An n-dimensional hypercube Q_n consists of $N = 2^n$ nodes interconnected as follows : i) each node is labeled by an n-bit binary number $(a_1 a_2 \dots a_n)$, ii) two nodes are connected by an link if and only if their binary labels differ in exactly one bit position. The n-cube has become an interesting topic of research in recent years due to its versatile applications in parallel and distributed processing. Many interesting properties of the n-cube have been reported in the literature [AG81], [AP89], [Le93].

Esfahanian et al. [ENS91] have shown that by adding two new links, the diameter of an n-cube $(n \geq 2)$ can be reduced by one. In [AL90] it has been shown that by adding $\binom{4m}{m}$ extra links to a 4m-dimensional cube $(m \geq 2)$, its diameter can be reduced by $2m-1$. In [TW91], the effect of adding some extra links on the performance measures, such as diameter, mean internode distance, traffic density etc. have been discussed.

Two links of a hypercube are called independent if they are not incident on a common node. In [ENS91] it has been shown that by exchanging a pair of independent links from a 4-cycle of an n-cube $(n \geq 3)$, known as twisting, its diameter can be reduced by 1. In [HKS87], it is shown that by exchanging $(d-1)2^{d-4}$ link pairs in a d-cube $(d = 2m+1)$, its diameter can be reduced to $(d+1)/2$. In [AP91], performance measures of such twisted cubes have been studied.

In this chapter, we have proposed a new family of network topologies, by modifying the original hypercube structure, which will have diameters lesser than that of a hypercube, but still retaining the other desirable features of the hypercube, e.g., ease of routing under fault free and faulty situations, high connectivity, i.e., high degree of fault-tolerance, etc. Two possible approaches have been considered for this – one involving addition of a few extra links called *bridges,* and the other involving exchanges of pairs of independent links (i.e., *twists*) without the need for extra links.

We shall first show that by adding$\binom{d}{\lceil d/4 \rceil+1}+1$ extra links, termed as bridges, to a d-cube (d > 4), its diameter can be reduced by $\lfloor d/2 \rfloor$. Then we genaralize this scheme to add $\binom{4m}{m+1}+1$ $(m > 2)$ bridges to an n-cube $(n > 4m$ and $n > 8)$ to reduce its diameter by 2m. To reduce the diameter by 2m−1 we add $2\binom{4m-3}{m}+1$ $(m > 2)$ *bridges to an* n-cube $(n > 4m-2$ and $n > 10)$. We have also given routing algorithm for the bridged hypercubes Q_d with diameter $\lceil d/2 \rceil$. The proposed routing algorithm ensures path length less than or equal to the diameter without much overhead. Routing in other cases can similarly be dealt with.

Next we shall show that by exchanging 4 pairs of independent links in a d-cube $(d \geq 5)$, we can reduce its diameter by 2. By exchanging 16 pairs of independent links, the diameter of a d-cube $(d \geq 7)$ can be reduced by 3. By exchanging 57 pairs of independent links, the diameter can be reduced by 4 for $d \geq 9$. To reduce the diameter by $\lfloor d/2 \rfloor$, where $d \geq 10$, we need to exchange $\binom{d-1}{r}+1$ pairs of independent links, where $r = \lfloor d/4 \rfloor + 1$. In [HKS87], one type of twisted hypercube with lower diameter has been devoloped. But there the number of link pairs exchanged is much more. Starting with a d-cube, where $d = 2m+1$, $(d-1)2^{d-4}$ link pairs are exchanged to get a graph of diameter $(d+1)/2$. Accordingly, in an 11-cube one needs to exchange $10.2^7 = 1280$ link pairs to reduce its diameter to 6 by the method given in [HKS87]. In our scheme we need only 121 link pairs to be exchanged in an 11-cube to get a graph of diameter 6.

We introduce a few definitions and notations in section 4.2. In section 4.3 we show how the extra links, referred to as bridges, can be connected to a d-dimensional hypercube for reducing the diameter to $\lceil d/2 \rceil$. Section 4.4 deals with the generalization of the idea of adding bridges for reducing the diameter of a hypercube by any given value. We discuss twisted hypercubes in section 4.5. Section 4.6 deals with the routing algorithms in bridged and twisted hypercubes.

4.2 Notations and Terminologies

A node in a hypercube is represented by a string of binary digits. A subcube is represented by a string over the alphabet {0,1,∗}, where '∗' stands for '0' or '1'. For example, 01∗1∗ represents the subcube formed by the nodes 01010, 01011, 01110, 01111. In order to make the representation more compact, we would replace p consecutive occurences of a given symbol by that symbol raised to the power of p. For example, 00011 will be written as $0^3 1^2$.

Let f(x) be the number of 1's in the binary representation of a node x. We now define the following sets:

$$W_r = \{x \mid f(x) = r\}$$
$$A_r = \{x \mid x \in W_r \text{ and } x \in *^{n-1}0\}$$
$$B_r = \{x \mid x \in W_r \text{ and } x \in *^{n-1}1\}$$

Example 4.1 : Take a 5-dimensional cube. For $r = 3$,
$W_3 = \{00111, 01011, 01101, 01110, 10011, 10101, 10110, 11001, 11010, 11100\}$
$A_3 = \{01110, 10110, 11010, 11100\}$
$B_3 = \{00111, 01011, 01101, 10011, 10101, 11001\}$

Also, let HD(x, y) denote the Hamming distance between two nodes x and y.

If a node x is represented by $x = a_1 a_2 \ldots a_n$, $a_i \in \{0,1\}$, a node diametrically opposite to x is denoted by $\bar{x} = \bar{a}_1 \bar{a}_2 \bar{a}_3 \ldots \bar{a}_n$.

For a source destination pair (s, t) where $s = a_1 a_2 \ldots a_n$ and $t = b_1 b_2 \ldots b_n$ we define four sets of bit positions S_{00}, S_{01}, S_{10} and S_{11} as follows :

$$S_{00} = \{h \mid a_h = 0,\ b_h = 0\}$$
$$S_{01} = \{h \mid a_h = 0,\ b_h = 1\}$$
$$S_{10} = \{h \mid a_h = 1,\ b_h = 0\}$$
$$S_{11} = \{h \mid a_h = 1,\ b_h = 1\}$$

Example 4.2 : Let $s = 01001001$
and $t = 10011011$
Then $S_{00} = \{3, 6\}$, $S_{01} = \{1, 4, 7\}$, $S_{10} = \{2\}$, $S_{11} = \{5, 8\}$.

We denote the cardinalities of the sets S_{00}, S_{01}, S_{10} and S_{11} by s_0, s_1, s_2 and s_3 respectively. Hence the number of 0's in $s = (s_0 + s_1)$ and the number of 1's in $s = (s_2 + s_3)$. Also $HD(s, t) = s_1 + s_2$.

4.3 Bridged d-Cube with Diameter $\lceil d/2 \rceil$

The extra links that we propose to add to a cube are always in the form of $(x, \bar{x})$. We refer to these extra edges as *bridges*. In a d-dimensional cube $(d > 4)$, we take a fixed value of r, $0 < r < d/2$, and then connect bridges from all the nodes in W_r

$\cup W_0$. The cardinality of the set W_r is $\binom{d}{r}$. Hence the cardinality of the set $W_r \cup W_0$ is $\binom{d}{r} + 1$, which will be equal to the number of such bridges that will be added to the hypercube. To find an expression for the diameter of the bridged hypercube, we first try to unfold the possible paths between two nodes s and t through a bridge.

Let x be a node at which a bridge is connected. To reach t from s, we can reach x by changing some bits in s. Let $S_{ij}' \subseteq S_{ij}$, $i, j \in \{0, 1\}$, be the sets of bit positions where these changes are done. Let s_0', s_1', s_2' and s_3' be the cardinalities of the sets S_{00}', S_{01}', S_{10}', and S_{11}' respectively. Then we have the following lemma.

Lemma 4.1 : The shortest path between two points s and t, via the bridge $(x, \bar{x})$ is of length $p = 1 + 2(s_1' + s_2') + s_0 + s_3$.

Proof: From s, we reach x in $(s_0' + s_1' + s_2' + s_3')$ steps. Another single step via the bridge leads to the node $\bar{x}$. $\bar{x}$ differs from t in all bits of S_{01}', S_{10}', $S_{00} - S_{00}'$ and $S_{11} - S_{11}'$ and nothing else. Hence,

$$\begin{aligned} p &= (s_0' + s_1' + s_2' + s_3') + 1 + (s_1' + s_2' + s_0 - s_0' + s_3 - s_3') \\ &= 1 + 2(s_1' + s_2') + s_0 + s_3. \end{aligned}$$ ◆

Example 4.3 : Consider an 8-cube with $s = 01001001$ and $t = 10011011$. Then $S_{00} = \{3, 6\}$, $S_{01} = \{1, 4, 7\}$, $S_{10} = \{2\}$, $S_{11} = \{5, 8\}$. Take $r = 3$ and $x = 01100010$. Then $S_{00}' = \{3\}$, $S_{01}' = \{7\}$, $S_{10}' = \varnothing$, $S_{11}' = \{5, 8\}$. From the formula in lemma 4.1, the length of the shortest path between s and t using the bridge $(x, \bar{x})$ is $1 + 2(1 + 0) + 2 + 2 = 7$. Also we see that the distance between s and x is 4 and the distance between t and $\bar{x}$ is 2, giving the path length as $4 + 1 + 2 = 7$.

Remark : To get a shorter path length between s and t, we would change as lesser number of bits in S_{01} and S_{10} as possible.

Lemma 4.2 : There exists a path of length $p_0 = d + 1 + s_2 - s_1$ between s and t, via the bridge connected to the node in W_0.

Proof: Starting from s, we reach the node in W_0 by changing all bits in S_{10} and S_{11}. By lemma 4.1, $p_0 = 1 + s_0 + s_3 + 2s_2$. Putting $s_0 + s_3 = d - (s_1 + s_2)$ we get $p_0 = d + 1 + s_2 - s_1$. ◆

Example 4.4 : In the example 4.1, the distance between s and 00000000 is 3 and then the distance between 11111111 and t is 3. Thus the minimum path length via the bridge from W_0 is $3 + 1 + 3 = 7$. The path length using the formula in lemma 4.2 is again $8 + 1 + 1 - 3 = 7$.

Lemma 4.3 : For $s_1 > d - r$ there exists a path of length $p_1 = -d + 2r + 1 + s_1 - s_2$ between s and t, via a bridge connected to some node in B_r.

Proof: $s_1 > d - r$ implies $r > d - s_1$. If we want to reach a node in B_r from s, where $r > d - s_1 = s_0 + s_2 + s_3$ we must change all bits in S_{00} to 1 and also some more bits in S_{01}. The number of bits in S_{01} to be changed is equal to $s_1' = r - (d - s_1) = s_1 - (d - r)$. Note that if the d-th bit is in S_{01}, these s_1' bits must include the d-th bit to reach a node in B_r. The path length $p_1 = 1 + 2s_1' + s_0 + s_3$ (by lemma 4.1). Putting $s_0 + s_3 = d - (s_1 + s_2)$ and $s_1' = s_1 - (d - r)$, we get $p_1 = 1 + 2r - d + s_1 - s_2$. ◆

Example 4.5 : Take $s = 00001000$, $t = 11110110$ and $r = 3$. Then $s_1 = 6 > 5 = d - r$. After changing the bit in S_{00} we reach 00001001. We must change another 0 to 1 in order to reach a node in W_3. So we change the seventh bit to reach 00001011. From there we take the bridge to 11110100 and then to t in another step. Thus the path length is 4. Using the formula in lemma 4.3, $p_1 = 1 + 6 - 8 + 6 - 1 = 4$.

Lemma 4.4 : For $s_1 \leq d - r$ and $s_2 \leq r$, there exists a path of length $p = 1 + s_0 + s_3$ between s and t, via a bridge connected to a node in W_r.

Proof: Case I : $s_2 + s_3 \leq r$, i.e., the number of 1's in s is less than or equal to r. The number of 0's in a node in W_r is greater than or equal to s_1, since $s_1 \leq d - r$, by the given condition. Hence, starting from s, we can reach a node in W_r by changing all (for $s_1 = d - r$) or some (for $s_1 < d - r$) bits in S_{00}. By lemma 4.1, $p = 1 + s_0 + s_3$.

Case II : $s_2 + s_3 > r$.
By the given condition $s_2 \leq r$, i.e., the number of 1's in a node in W_r is greater than or equal to s_2. Hence starting from s, we can reach a node in W_r by changing some bits in S_{11} only. By lemma 4.1, path length $p = 1 + s_0 + s_3$. ◆

Example 4.6 : Take $s = 01001001$, $t = 10011011$. If we take $r = 4$, then $s_1 = 3 \leq 4 = d - r$ and $s_2 = 1 \leq 4 = r$. Since $s_2 + s_3 = 3 < r$, by changing one bit of S_{00} (say, the third bit) in s we reach 01101001, a node in W_r. Then we take the bridge to reach 10010110 and then to t in three steps. Thus the path length is $1 + 1 + 3 = 5$. From lemma 4.4 $p = 1 + 2 + 2 = 5$.

Lemma 4.5 : For $s_2 \leq d - r$ and $s_1 \leq r$, there exists a path of length $p = 1 + s_0 + s_3$ between s and t, via a bridge connected to a node in W_r.

Proof: Similar to lemma 4.4, except that we start from t. ◆

Lemma 4.6 : For $r < s_1 < d - r$ and $r < s_2 < d - r$, there exists a path of length $p = d - 2r + 1 + s_2 - s_1$ via a bridge connected to a node in W_r.

Proof: Since $s_2 > r$, the number of 1's in a node in W_r is less than that in s. Thus, to reach a node from s we have to change some '1' bits to '0'. We can change all bits in S_{11} and some $s_2' = s_2 - r$ bits in S_{10}, to reach a node in W_r. By lemma 4.1, $p = 1 + s_0 + s_3 + 2s_2'$. Putting $s_0 + s_3 = d - (s_1 + s_2)$ and $s_2' = s_2 - r$, we get $p = 1 + d - 2r + s_2 - s_1$. ◆

Theorem 4.1 : A bridged hypercube of dimension d ($d \geq 8$), with bridges connected to all the nodes in $W_0 \cup W_r$ ($r = \lfloor d/4 \rfloor + 1$), has diameter equal to $\lceil d/2 \rceil$.

Proof: We can assume w.l.o.g. (without loss of generality) that $s_2 \leq s_1$. Also we need to consider only those s, t for which $HD(s, t) > \lceil d/2 \rceil$, i.e., $s_0 + s_3 \leq \lceil d/2 \rceil - 1$. There can be three possible cases,

Case 1 : $s_1 > d - r$

Case 2 : $r < s_1 \leq d - r$

Case 3 : $s_1 \leq r$

Case 2 can, however, be divided into two subcases

2a) $s_1 \leq d - r$, $s_2 \leq r$

2b) $r < s_1 < d - r$, $s_2 > r$

Note that the case $s_1 = d - r$ and $s_2 > r$ cannot arise, since in that case $s_1 + s_2 > d - r + r = d$, but the total number of bits is only d.

Case 1. By lemma 4.2, $p_0 = d + 1 + s_2 - s_1$ and by lemma 4.3, $p_1 = 1 + 2r - d + s_1 - s_2$. Hence, $\min(p_0, p_1) \leq (p_0 + p_1)/2 = r + 1 \leq \lceil d/2 \rceil$.

Case 2a). By lemma 4.4, the path length $p = 1 + s_0 + s_3 \leq \lceil d/2 \rceil$. When $s_0 + s_3 = \lceil d/2 \rceil - 1$, the path length $p = \lceil d/2 \rceil = HD(s, t)$.

Case 2b). By lemma 4.6, $p = d - 2r + 1 + s_2 - s_1 \leq d - 2r + 1$. For $r = \lfloor d/4 \rfloor + 1$, $p \leq d/2$, as d is even.

Case 3. By lemma 4.5, the path length $p = 1 + s_0 + s_3 \leq d/2$.

Hence the proof. ◆

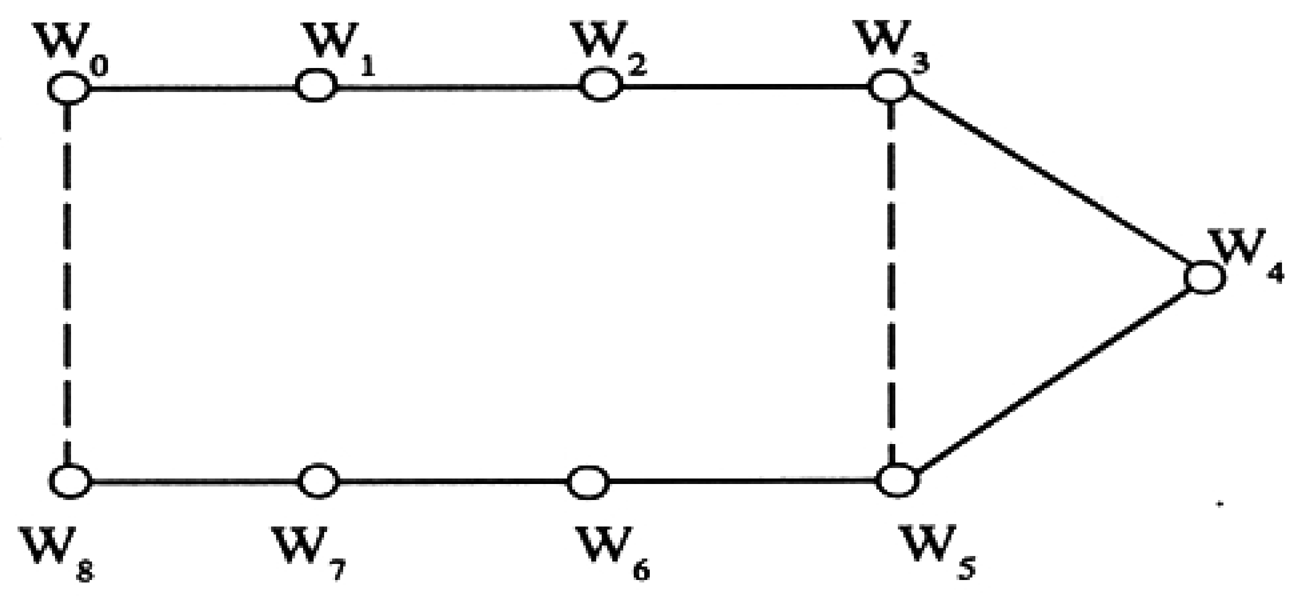


Figure 4.1 : Illustration of a Bridged 8-Cube.

Example 4.7 : An illustrative example is given in figure 4.1 with $n = 8$, where the solid lines represent the already existing lines in the 8-cube and the dotted lines represent the bridges added to the cube.

The number of extra links that we add is small compared to the total number of links in the d-cube. As we increase d, the ratio $\binom{d}{r} + 1 : d.2^{d-1}$ becomes smaller, where $r = \lfloor d/4 \rfloor + 1$. Table 4.1 shows the values of this ratio for different d.

Table 4.1

d	$\binom{d}{r}+1$	$\binom{d}{r}+1 : d.2^{d-1}$
6	16	.083
8	57	.05
10	121	.02

4.4 Reduction of Diameter by an Arbitrary Value

In the previous section we have considered the bridged hypercube whose diameter is reduced to half of its dimension. Now we will show how to reduce the diameter of a cube by any arbitrary value k, $1 \le k < d/2$.

4.4.1 Reduction by an Even Value

First we consider the case when k is even. Assuming that there are bridges connected to all the nodes in W_r for a given value of r, $0 < r < d/2$, we investigate some more properties regarding the paths between s and t.

Lemma 4.7 : For $s_2 \geq r$ and the d-th bit in S_{01} there is a path of length $p_0 = d - 2r + 1 + s_2 - s_1$, between s and t via a bridge connected to A_r.

Proof: Starting from s, we can change all bits in S_{11}, and $s_2' = s_2 - r$ bits in S_{10} to reach a node in A_r. Hence $p_0 = 1 + s_0 + s_3 + 2s_2'$ (by lemma 4.1). Putting $s_0 + s_3 = d - (s_1 + s_2)$ and $s_2' = s_2 - r$ we get $p_0 = d - 2r + 1 + s_2 - s_1$. ◆

Lemma 4.8 : For $s_1 \geq r$ and the d-th bit in S_{10} there is a path of length $p_0 = d - 2r + 1 + s_1 - s_2$, between s and t via a bridge connected to A_r.

Proof: Proof is similar to that of lemma 4.7, except that we start from t. ◆

Lemma 4.9 : For $s_2 \geq r$ and the d-th bit in S_{11} or S_{00} there is a path of length $p_0 = d - 2r + 1 + s_2 - s_1$, between s and t via a bridge connected to A_r.

Proof: Similar to that of lemma 4.7. ◆

Lemma 4.10 : For $s_1 \geq r$ and the d-th bit in S_{01} there is a path of length $p_1 = d - 2r + 1 + s_1 - s_2$, between s and t via a bridge connected to B_r.

Proof : Starting from t, we change all bits in S_{11}, and $s_1' = s_1 - r$ bits in S_{01} (excluding the last bit) to reach a node in B_r. Hence $p_0 = 1 + s_0 + s_3 + 2s_1'$ (by lemma 4.1). Putting $s_0 + s_3 = d - (s_1 + s_2)$ and $s_1' = s_1 - r$, we get $p_0 = d - 2r + 1 + s_1 - s_2$. ◆

Lemma 4.11 : For $s_2 \geq r$ and the d-th bit in S_{10} there is a path of length $p_1 = d - 2r + 1 + s_2 - s_1$, between s and t via a bridge connected to B_r.

Proof: Similar to that of lemma 4.10 except that we start from s. ◆

Lemma 4.12 : For $s_1 \geq r - 1$ and the d-th bit in S_{00} or S_{11}, there is a path of length $p_1 = d - 2r + 3 + s_1 - s_2$ via a bridge connected to B_r.

Proof: If the d-th bit is in S_{11}, we cannot change the d-th bit to reach a node in B_r. So, starting from t, we can change only $s_3 - 1$ bits in S_{11} and the remaining $s_1' = s_1 - (r - 1)$ bits in S_{01}. Hence $p_1 = 1 + s_0 + s_3 + 2s_1' = d - 2r + 3 + s_1 - s_2$. If the d-th bit is in S_{00}, we must change it to 1. So, starting from t we can change all s_3 bits in S_{11} and $s_1' = s_1 - (r - 1)$ bits in S_{01} to reach a node in B_r. Hence, $p_1 = d - 2r + 3 + s_1 - s_2$. ◆

Theorem 4.2 : By adding 8 extra links to an n-cube ($n \geq 4$), its diameter can be reduced by 2.

Proof : Case I : $n = 4$.
Let us connect all diametrically opposite pairs by bridges. For this 8 extra links will be added to the cube. Let us take any two points s and t. Let p be the length of the shortest path between s and t. To show that the diameter of this bridged 4-cube is 2 we consider only those (s, t) for which $HD(s, t) > 2$. If $HD(s, t) = 3$, $HD(\bar{s}, t) = 1$ and hence $p = 2$. If $HD(s, t) = 4$, $t = \bar{s}$ and hence $p = 1$.

Case II : $n > 4$. We take a 4-dimensional subcube $C_0 = 0^{n-4} *^4$. We connect bridges within this subcube as in case I, i.e., the bridges are from $0^{n-4} x$ to $0^{n-4} \bar{x}$, where x is a binary string of length 4. Take two nodes $s = ax$ and $t = by$, where x and y are binary strings of length 4, a and b are binary strings of length $n-4$. Let f be the number of bit positions at which s and t are same. To show that the diameter of this bridged n-cube is $n-2$, we consider only those (s, t) for which $HD(s, t) > n - 2$, i.e., $f \leq 1$.

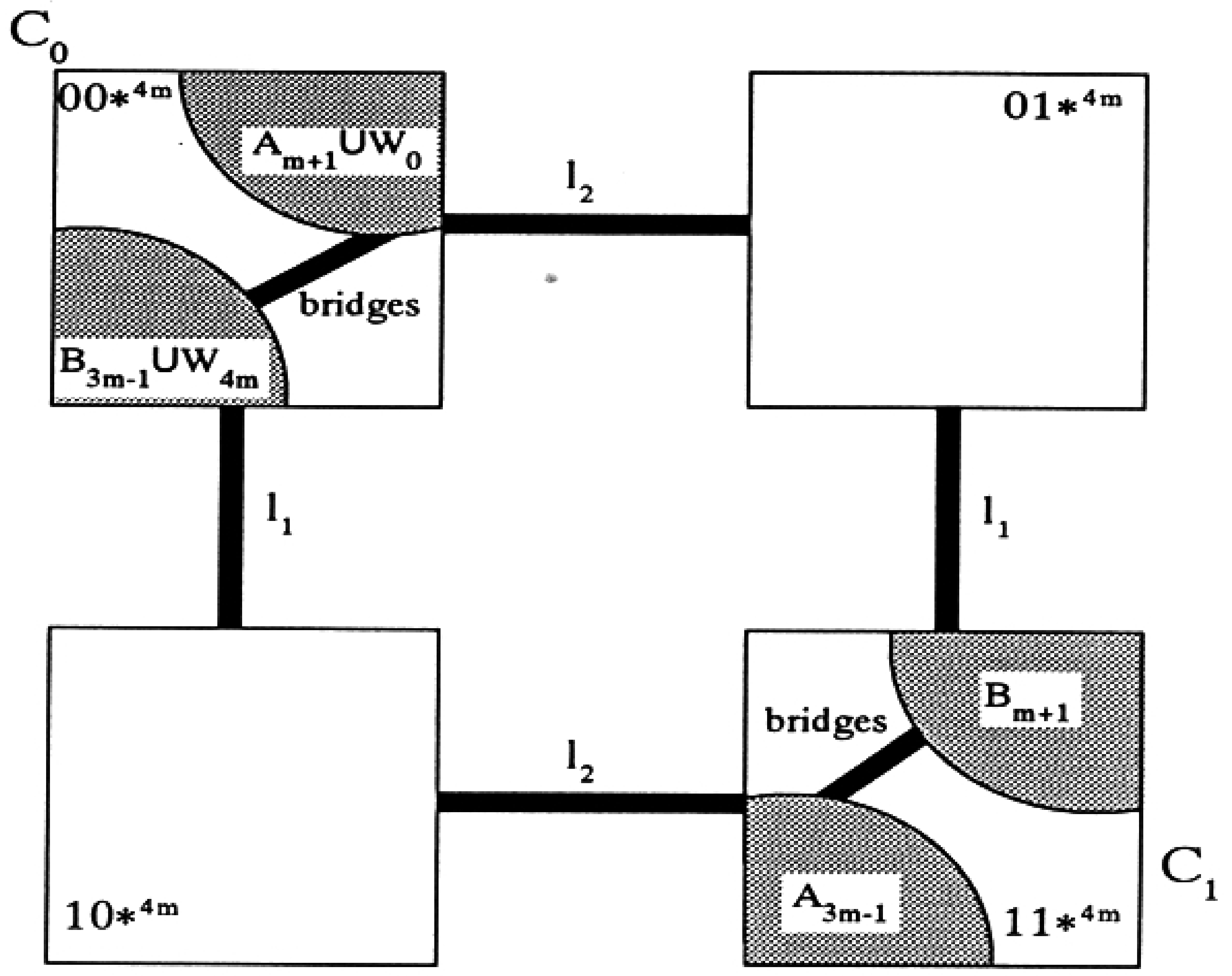
If $f = 0$, then $x = \bar{y}$. The shortest path between x and y is through C_0 and is of length $n - 4 + 1 = n - 3$. If $f = 1$, there can be two possible cases :

a) If the bit position at which s and t are having the same value, is within the last 4 bits, then shortest path between $0^n x$ and $0^n y$ is of length 2 (by case I). So the shortest path between x and y is through C_0 and is of length $n - 4 + 2 = n - 2$.

b) If the bit position at which s and t are having the same value, is within the first $n-4$ bits, $x = \bar{y}$. The shortest path between s and t is through C_0 and is of length $n-4$ or $n-2$ respectively, depending on whether the bit value is 0 or 1. ◆

We generalize this idea to reduce the diameter of a hypercube by an even number $2m$ ($m \geq 2$) as follows. We take an n'-cube ($n' = n + 4m$). We take two subcubes $C_0 = 0^n *^{4m}$ and $C_1 = 1^n *^{4m}$. In C_0, we add bridges of the form $(0^n x, 0^n \bar{x})$, where x is a bit string of length 4m. Similarly, in C_1 the bridges of the form $(1^n x, 1^n \bar{x})$ are also inserted. For the subcubes C_0 and C_1, we define the sets W_r, A_r, B_r as for a 4m-dimensional cube by ignoring the leftmost n bits of the node representation. In C_0, we connect bridges from all the nodes in A_{m+1} and W_0. In C_1, we connect bridges from all the nodes in B_{m+1}. An illustration for connecting such bridges is shown in figure 4.2 for the case when $n = 2$.

Theorem 4.3 : In an n'-cube ($n' = n + 4m$, $m \geq 2$), by adding $\binom{4m}{m+1} + 1$ extra links in the above mentioned way, we can reduce its diameter by 2m.



l_i : links connecting nodes which differ in the i-th bit, i = 1, 2.

Figure 4.2 : An Illustration for Connecting Bridges in a (4m+2)-Cube

Proof: We can represent any two nodes s and t in this n' cube as $s = a_1 a_2 \dots a_n a_{n+1} \dots a_{n+4m}$ and $t = b_1 b_2 \dots b_n b_{n+1} \dots b_{n+4m}$. We define two strings x and y as $x = a_{n+1} a_{n+2} \dots a_{n+4m}$ and $y = b_{n+1} b_{n+2} \dots b_{n+4m}$. Let p_0 be the length of a path between $0^n x$ and $0^n y$ in C_0 and p_1 be that between $1^n x$ and $1^n y$ in C_1.

Among the first n bits from left, let p be the number of bits those are 0 in both s and t and q be the no of bits those are 1 in both s and t. We define s_0, s_1, s_2 and s_3 for the two strings x and y as before. We consider only those (s, t) for which $HD(s, t) > n + 2m$, i.e., $p + q + s_0 + s_3 \leq 2m-1$. We now consider two paths between s and t, one using a bridge in C_0 and the other using a bridge in C_1. Let the path through C_i be of length P_i, $i \in \{0,1\}$.

We can easily verify that the total number bits to be changed to get $0^n x$ from s, and t from $0^n y$ is equal to $(n-p-q)+2q = n-p+q$. Hence, $P_0 = n-p+q+p_0$. Similarly, $P_1 = n + p - q + p_1$.

Now, $\min(P_0, P_1) \leq (P_0 + P_1)/2 = n + (p_0 + p_1)/2$.

In finding p_0, p_1 we apply the lemmas where $d = 4m$ and $r = m+1$. There can be 4 possible cases as in theorem 4.1.

Case 1	:	$s_1 > d - r = 3m - 1$
Case 2a	:	$s_1 \leq d - r = 3m - 1,\ s_2 \leq r = m+1$
Case 2b	:	$m + 1 = r < s_1 < d - r = 3m - 1,\ s_2 > r = m + 1$
Case 3	:	$s_1 \leq r = m + 1$

Case 1 : By lemma 4.2, $p_0 = 4m + 1 + s_2 - s_1$ and by lemma 4.3, $p_1 = -2m + 3 + s_1 - s_2$. Hence, $\min(P_0, P_1) \leq n + m + 2 \leq n + 2m$, since $m \geq 2$.

Case 2a : By lemma 4.4 at least one of p_0, p_1 is equal to $1 + s_0 + s_3$. Hence, at least one of P_0 and P_1 is less than or equal to $n + p + q + s_0 + s_3 + 1 \leq n + 2m$, [as $p + q + s_0 + s_3 \leq 2m-1$]. When $p = q = 0$ and $s_0 + s_3 = 2m-1$, either P_0 or $P_1 = n + s_0 + s_3 + 1 = n + 2m < HD(x, y)$.

Case 2b : If the $(n + 4m)$-th bit is in S_{00} or S_{11}, $p_0 = 2m - 1 + s_2 - s_1$ (by lemma 4.9) and $p_1 = 2m + 1 + s_1 - s_2$ (by lemma 4.12). Hence, $\min(P_0, P_1) \leq n + 2m$.

If the $(n + 4m)$-th bit is in S_{01}, $p_0 = 2m - 1 + s_2 - s_1$ (by lemma 4.7) and $p_1 = 2m - 1 + s_1 - s_2$ (by lemma 4.10). Hence, $\min(P_0, P_1) \leq n + 2m - 1$.

If the $(n + 4m)$-th bit is in S_{10}, $p_0 = 2m - 1 + s_1 - s_2$ (by lemma 4.8) and $p_1 = 2m - 1 + s_2 - s_1$ (by lemma 4.11). Hence, $\min(P_0, P_1) \leq n + 2m - 1$.

Case 3 : If $s_2 \leq 3m - 1$, by lemma 4.5, one of p_0, p_1 is equal to $1 + s_0 + s_3$. Hence, at least one of P_0, P_1 is less than or equal to $n + p + q + s_0 + s_3 + 1 \leq n + 2m$, [as $p + q + s_0 + s_3 \leq 2m-1$].

If $s_2 > 3m - 1$, the case is similar to case 1 if we interchange s and t.

Hence the proof. ◆

4.4.2 Reduction by an Odd Value

To reduce the diameter by an odd number say $2m - 1$ ($m \geq 2$) we take an $n' = n + 4m - 2$ dimensional cube. We add bridges to the subcubes $C_0 = 0^n *^{4m-2}$ and $C_1 = 1^n *^{4m-2}$ as before. Now we will see that this construction does not give a graph of diameter $n + 2m - 1$ for $m > 2$. But, for $m = 2$ this construction gives a graph of diameter n+2m−1. To do that, in theorem 3 we substitute d by 4m−2 and r by m. We note that between two points x and y the path length is at most (n+2m−1) in all cases except in case 2b when the (n+4m−2)-th bit is in S_{00} or S_{11}.

In case 2b. when the last bit is in S_{00} or S_{11}

$$P_0 = n - p + q + 2m - 1 + s_2 - s_1$$
$$P_1 = n + p - q + 2m + 1 + s_1 - s_2$$

If $p - q < s_2 - s_1 - 1$, then $P_1 < n + 2m$ and if $p - q > s_2 - s_1 - 1$, then $P_0 < n + 2m$. $P_1 = P_0 = n + 2m$ only if $p - q = s_2 - s_1 - 1$ ...(a).

For $HD(x, y) > n + 2m - 1 = n + 3$, $p + q + s_0 + s_3 \leq 2$. Also $s_0 + s_3 \geq 1$, since the $(n + 4m - 2)$-th bit is in S_{00} or S_{11}.

Hence, $p + q \leq 1$. Then $p + q$ can be 1 or 0. If $p + q = 1$, $p - q$ is an odd number. But $s_0 + s_3$ is equal to 1. Hence, $s_2 - s_1$ is an odd number. So condition (a) does not hold. If $p + q = 0$, i.e., $p = q = 0$. We can take either of the path through C_0, which are of lengths $n + 2m - 1 + s_2 - s_1$ and $n + 2m - 1 + s_1 - s_2$.

To reduce the diameter of a hypercube by some odd number $2m - 1$ $(m > 2)$, we make a slight modification to our previous scheme. Before going into that, we make some more observation regarding paths between two nodes s and t in a d cube, where bridges are connected to all the nodes in W_r.

Lemma 4.13 : For $s_1 \leq d - r - 1$, $s_2 \leq r$ and the d-th bit of s is 0, there exists a path of length $p_0 = 1 + s_0 + s_3$ between s and t, via a bridge connected to a node in A_r.

Proof: Case I : $s_2 + s_3 \leq r$. The number of 0's in a node in A_r is greater than or equal to $s_1 + 1$, since $d - r \geq s_1 + 1$, by the given condition. Hence we can reach a node in A_r by changing at most $s_0 - 1$ bits in S_{00} to 1. Hence we never change the last bit even if it's in S_{00}. By lemma 4.1, $p_0 = 1 + s_0 + s_3$.

Case II : $s_2 + s_3 > r$. Starting from s, we change some bits in S_{11} to reach a node in A_r. $p_0 = 1 + s_0 + s_3$ (by lemma 4.1). ◆

Lemma 4.14 : For $s_1 \leq d - r$ and $s_2 < r$ and the d-th bit of s equal to 1, there exists a path of length $p_1 = 1 + s_0 + s_3$, between s and t, via a bridge connected to a node in B_r.

Proof: Case I : $s_2 + s_3 \le r$. Starting from s, we change some bits in S_{00} to reach a node in B_r. $p_1 = 1 + s_0 + s_3$ (by lemma 4.1).

Case II : $s_2 + s_3 > r$. Starting from s, we change at most $s_3 - 1$ bits in S_{11} to reach a node in B_r as $s_2 < r$. Hence we never change the last bit even if it is in S_{11}. $p_1 = 1 + s_0 + s_3$ (by lemma 4.1). ◆

Now, we describe our scheme as follows. We take an n'-cube ($n' = n + 4m - 2$). We take two subcubes $C_0 = 0^n *^{4m-2}$ and $C_1 = 1^n *^{4m-2}$. In C_0, we add bridges of the form $(0^n x, 0^n \bar{x})$, where x is a bit string of length $4m - 2$, and in C_1 the bridges are of the form $(1^n x, 1^n \bar{x})$. For the subcubes C_0 and C_1 we define the sets W_r, A_r, B_r as for a $(4m - 2)$-dimensional cube by ignoring the leftmost n bits of the node representation. In C_0, we connect bridges from all the nodes in A_m and W_0. In C_1, we connect bridges from all the nodes in B_{m+1}.

Theorem 4.4 : In an n'-cube, where $n' = n + 4m - 2$ $(m > 2)$, by adding $2\binom{4m-3}{m} + 1$ bridges in the above mentioned way we can reduce its diameter by $2m - 1$.

Proof: We represent two nodes s and t as $s = a_1 a_2 \ldots a_n a_{n+1} \ldots a_{n+4m-2}$ and $t = b_1 b_2 \ldots b_n b_{n+1} \ldots b_{n+4m-2}$. We define two strings x and y as $x = a_{n+1} a_{n+2} \ldots a_{n+4m-2}$ and $y = b_{n+1} b_{n+2} \ldots b_{n+4m-2}$. We define p, q, p_0, p_1, s_0, s_1, s_2, s_3, P_0 and P_1 as in theorem 4.3. We consider only the case when $HD(s, t) > n + 2m - 1$, i.e., $p + q + s_0 + s_3 \le 2m - 2$. There can be three possible cases.

Case 1	:	$s_1 > 3m - 3$
Case 2a	:	$m < s_1 \le 3m - 3,\ s_2 \le m$
Case 2b	:	$m < s_1 \le 3m - 3,\ s_2 > m$.
Case 3	:	$s_1 \le m$

Case 1 : In lemma 4.2 we put $d = 4m - 2$ and $r = m + 1$, to get $p_0 = 4m - 1 + s_2 - s_1$. Similarly, from lemma 4.3 we get $p_1 = -2m + 5 + s_2 - s_1$. Hence, $\min(P_0, P_1) \le n + m + 2 \le n + 2m - 1$, as $m > 2$.

Case 2a : If the rightmost bit of s is 0, putting $d = 4m - 2$ and $r = m$ in lemma 4.13, we have $p_0 = 1 + s_0 + s_3$. Hence, $P_0 \le n + p + q + s_0 + s_3 + 1 \le n + 2m - 1$. Alternatively, if the rightmost bit of s is 1, putting $d = 4m - 2$ and $r = m + 1$ in lemma 4.14, we have $p_1 = 1 + s_0 + s_3$. Hence $P_1 \le n + 2m - 1$. When $p = q = 0$ and $s_0 + s_3 = 2m - 2$, $P_1 = n + 2m - 1 < HD(s, t)$.

Case 2b : If the rightmost bit of s is in S_{01}, $p_0 = 2m - 1 + s_2 - s_1$ (by lemma 4.7) and $p_1 = 2m - 3 + s_1 - s_2$ (by lemma 4.10). Thus $\min(P_0, P_1) \leq n + 2m - 2$. The proof is similar for the case when the rightmost bit of s is in S_{10}. If the rightmost bit of s is in S_{00} or S_{11}, $p_0 = 2m - 1 + s_2 - s_1$ (by lemma 4.9) and $p_1 = 2m - 1 + s_1 - s_2$ (by lemma 4.12). Hence, $\min(P_0, P_1) \leq n + 2m - 1$.

Case 3 : If $s_2 > 3m - 3$, we interchange s and t so that case 1 becomes applicable. If $s_2 \leq 3m - 3$, we interchange s and t so that case 2a becomes applicable.

Hence the proof. ◆

4.5 Twisted Hypercubes

We now consider the reduction of the diameter of a d-cube, by exchanging a few pairs of independent links. Consider two independent links (u, v) and (x, y) of the cube. If the links (u, x) and (v, y) are not present in the cube, then to exchange two independent links (u, v) and (x, y), we delete these links and add two new links (u, x) and (v, y). In this way, the degree of each node remains the same. We call such an exchange of a pair of links, a *twist*. In [ENS91] only 4-cycle twists are considered, where the nodes u, v, x and y form a 4-cycle in the hypercube. There it is shown that a single 4-cycle twist can reduce the diameter of an n-cube (n > 2) by 1. We consider in general, reduction in diameter by any given value k, 1 < k < m where m depends on the dimension of the cube. First, we show that by making 4 twists in a d-cube ($d \geq 5$), its diameter can be reduced by 2.

Definition : A twist of type r applied on a d-cube consists of the following operations. Let y be a bit string of length (d – 1). For all $y0 \in A_r$, we delete the link pair (y0, y1) and ($\bar{y}0$, $\bar{y}1$) and connect (y0, $\bar{y}1$) and (y1, $\bar{y}0$). We denote the twist of type r by T_r. The total number of link pairs exchanged by such a twist is equal to $\binom{d-1}{r}$.

Lemma 4.15 : To effect the twists, if we delete links of the type (y0, y1), where $y0 \in A_r$, for some given r, then the path length between any two points s and t is HD(s, t), provided HD(s, t) > 1.

Proof: Suppose there is a path $s \rightarrow a_1 \rightarrow a_2 \rightarrow \dots \rightarrow a_i \rightarrow a_j \rightarrow \dots \rightarrow a_k \rightarrow t$, of length HD(s, t), in the original cube from s to t, where a_1, a_2 etc. are intermediate nodes.

Suppose the link (a_i, a_j) is deleted due to the twist. Hence a_i and a_j must differ in

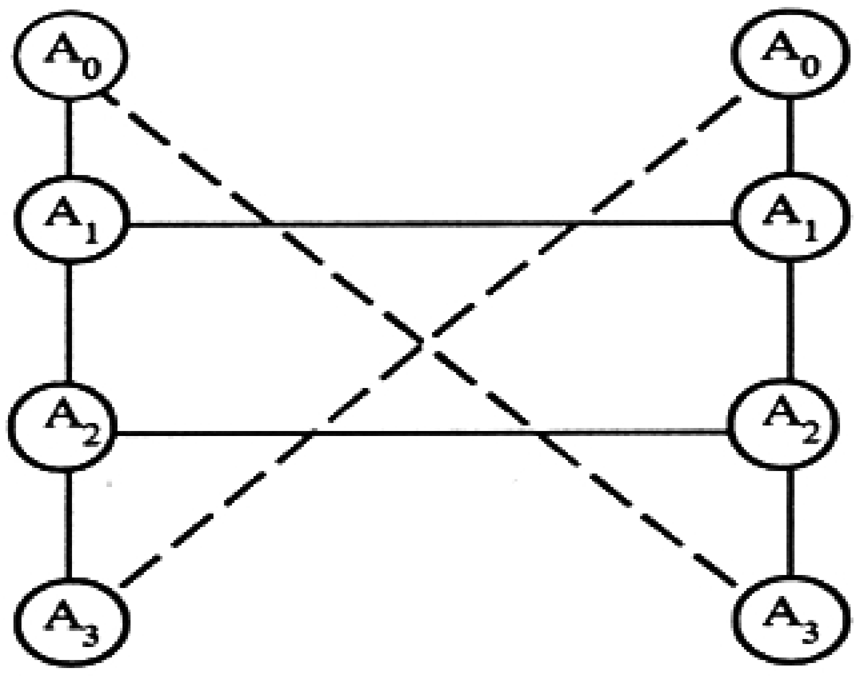


Figure 4.3 : An Illustration for a 4-Cube with a Twist of Type 0 (T_0)

the rightmost bit, and so also s and t. For HD(s, t) > 1, both $s = a_i$ and $t = a_j$ can not be true. We can assume w.l.o.g. that $s \neq a_i$. Hence by first changing the rightmost bit in s, followed by the changes in the other bits, we can reach t along a path of length HD(s, t). ◆

Remark : If we delete the links of the type (y0, y1), $y0 \in A_r$ for $r = r_1, r_2, ..., r_k$ where

$$r_2 - r_1 > 1$$
$$r_3 - r_2 > 1$$
$$. \quad . \quad . \quad .$$
$$r_k - r_{k-1} > 1,$$

then for any two points s and t where HD(s, t) > 1, we have a path of length HD(s, t), between s and t. When HD(s, t) = 1 and the link (s, t) is deleted, let s = y0 and t = y1. There exists a node $y_1 0$ such that HD(s, $y_1 0$) = 1 and the link ($y_1 0$, $y_1 1$) is intact. Now we can get a path $s \rightarrow y_1 0 \rightarrow y_1 1 \rightarrow t$, which is of length 3.

Lemma 4.16 : If we apply a twist of type 0 on a 4-cube, its diameter becomes equal to 3.

Proof: Figure 4.3 shows a 4-cube with a twist of type 0, where new links are shown by dotted lines. Let s and t be any two nodes in the above cube. If HD(s, t) > 1, there always exists a path of length HD(s, t) after the twist. To show that the diameter of this bridged cube is equal to 3, we need to consider only the cases where HD(s, t) = 1 or 4. We can assume w.l.o.g. that s is in A_k, $0 \leq k \leq 3$.

Case 1 : HD(s, t) = 1.
The link (s, t) is deleted only for $s \in A_0$ or $s \in A_3$. If $s \in A_0 = \{0000\}$ then t is equal

to 0001 for HD(s, t) = 1 and the link (s, t) deleted, We take the path (s $\rightarrow$ 0010 $\rightarrow$ 0011 $\rightarrow$ 0001 = t) which is of length 3. The proof is similar for the case when $s \in A_3$.

Case 2 : HD(s, t) = 4.
We assume s to be in A_k, $0 \leq k \leq 3$.
If $s \in A_0$ or $s \in A_3$, s and t are directly connected by a bridge added due to the twist.
If $s \in A_1$, then $t \in B_3$. The path (s $\rightarrow 0^4 \rightarrow 1^4 \rightarrow$ t) is of length 3. The proof is similar for the case when $s \in A_2$. ◆

Lemma 4.17 : If we apply a twist of type 1 on a 4-cube, its diameter becomes equal to 3.

Proof: Referring to the figure 4.4, let s and t be any two points in the 4-cube. Path length between s and t can be greater than HD(s, t) only if s and t differ in the last bit. We can assume w.l.o.g. that s is in A_k, $0 \leq k \leq 3$.

a) $s \in A_1$
If $t \in B_2$, there can be two cases. If HD(s, t) = 1, we take the path s $\rightarrow 0^4 \rightarrow 0^3 1$ $\rightarrow$ t which is of length 3. If HD(s, t) = 3, we take the path s $\rightarrow \bar{s} \rightarrow$ t which is of length 2.
If $t \in B_3$, there can be two cases. If HD(s, t) = 4, path length = 1, due to a direct link provided by the bridge. If HD(s, t) = 2, we take the path s $\rightarrow \bar{s} \rightarrow 1^4 \rightarrow$ t, which is of length 3.
The proof is similar for the case when $s \in A_2$.

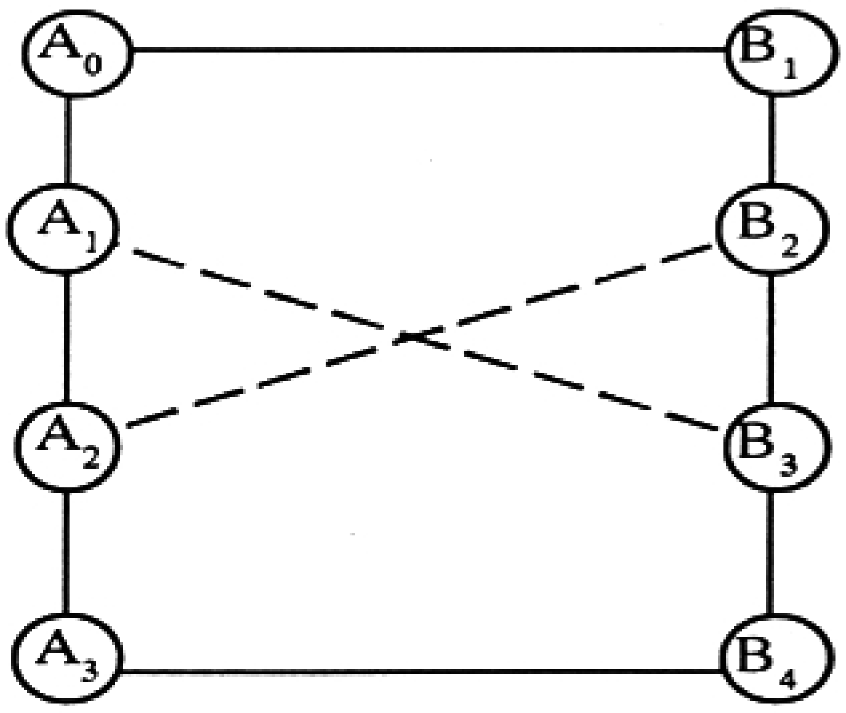


Figure 4.4 : An Illustration of a 4-Cube with Twist of Type 1 (T_1)

b) $s \in A_0$
If $t \neq \bar{s}$, there is always a path between s and t of length HD(s, t). If $t = \bar{s} = 1111$, we take the path $0^4 \rightarrow 10^3 \rightarrow 01^3 \rightarrow t$ which is of length 3. The proof is similar for the case when $s \in A_3$. ◆

Theorem 4.5 : By exchanging 4 link pairs in Q_{n+4}, $(n > 0)$ we can reduce its diameter by 2.

Proof: We apply T_0 within the 4-dimensional subcube $C_0 = 0^n *^4$. This involves an exchange of 1 link pair. We apply T_1 within another one dimensional subcube $C_1 = 1^n *^4$. This involves exchanges of 3 link pairs.

Take any two nodes s and t in Q_{n+4}. Let $s = ax$ and $t = by$ where a and b are strings of length n. Let the path length between s and t be represented by p(s, t);

Case I : If $a = b = 0^n$, $p(s, t) \leq 3$ (by lemma 4.16).

Case II : If $a = b = 1^n$, $p(s, t) \leq 3$ (by lemma 4.17).

Case III : If $HD(s, t) \leq n + 2$, then $p(s, t) \leq n + 2$.

Case IV : $HD(s, t) = n + 4$, then $x = \bar{y}$ and there exists a path from s to t via one of C_0 and C_1 of length $n + 1$.

Case V : **HD(s, t) = n + 3. In this case s and t must agree in one bit position.**
i) Suppose the bit common to s and t is within the lefttmost n bit position i.e., $x = \bar{y}$. **Either $0^n x$ is connected to $0^n \bar{x}$ or $1^n x$ is connected to $1^n \bar{x}$.** Hence, the path length is at most $n + 2$.

ii) The bit common to s and t is within the rightmost 4 bit position, i.e., $a = \bar{b}$

Subcase a) : If $\bar{x}$ and y do not differ in the rightmost bit, then let $z \in \{0,1\}$ and the path $s = ax \rightarrow z^n x \rightarrow z^n \bar{x} \rightarrow z^n y \rightarrow by = t$, is of length $n + 2$.

Subcase b) : If $\bar{x}$ and y differ in the rightmost bit, then there may be the following cases :
i) If $b \neq 0^n$ or 1^n, then the path $s \rightarrow z^n x \rightarrow z^n \bar{x} \rightarrow b\,\bar{x} \rightarrow by$ is of length n + 2.
ii) If $b = 1^n$, then for $x \in A_0 \cup A_3 \cup B_1 \cup B_4$, $y \in A_0 \cup A_3 \cup B_1 \cup B_4$ and the

path $s = 0^n x \rightarrow 0^n \bar{x} \rightarrow 10^{n-1} \bar{x} \rightarrow 10^{n-1} y \rightarrow 1^n y$ is of length $n + 2$.
On the other hand for $x \in A_1 \cup A_2 \cup B_2 \cup B_3$, $y \in A_1 \cup A_2 \cup B_2 \cup B_3$ and the path $s = 0^n x \rightarrow 0^n \bar{y} \rightarrow 1^n \bar{y} \rightarrow 1^n y$ is of length $n + 2$.

iii) If $a = 1^n$, it can similarly be proved that $p(s, t) = n+2$. ◆

Before investigating the effect of a twist of type r on a d-cube we prove another lemma regarding the path between two points s and t when bridges are connected to all the nodes in A_r.

Lemma 4.18 : For $s_1 > d - r - 1$ and the d-th bit in s is 0, there exists a path of length $p_1 \leq 2r + 2$, between s and t, via a bridge connected to some node in A_r.

Proof: $r \geq d - s_1$, for $s_1 > d - r - 1$. If the d-th bit is in S_{01}, we can reach a node in A_r from s, by changing all bits in S_{00} to 1 and also some $s_1' = s_1 - (d - r)$ bits in S_{01}. The path length $p_1 = 1 + 2s_1' + s_0 + s_3$ (by lemma 4.1). Putting $s_0 + s_3 = d - (s_1 + s_2)$ and $s_1' = s_1 - (d - r)$, we get $p_1 = -d + 2r + 1 + s_1 - s_2$. Since $s_1 - s_2 \leq d$, $p_1 \leq 2r + 1$. If the d-th bit is in S_{00}, we change all but the d-th bit in S_{00} and $(s_1' + 1)$ bits in S_{01} to reach a node in A_r. Thus $p_1 = -d + 2r + 3 + s_1 - s_2$. In this case $s_1 - s_2 \leq d - 1$. Hence $p_1 \leq 2r + 2$. ◆

Lemma 4.19 : In a d-cube (d being an even number), if we apply a twist of type r, where r is an integer between 0 and $d/2 - 2$, path length between any two points s and t is less than or equal to $\max(2r + 2, d - 2r + 1, d/2, 4)$

Proof: First consider that only bridges involved in the twist of type r are added, but no links are deleted. We can assume without loss of genarality that $s_1 \leq s_2$. Because of the symmetry of the twisted cube, we assume s to be in some A_k, $0 \leq k \leq d - 1$ (proof for the other case is similar).

We define s_0, s_1, s_2 and s_3 as before. As in theorem 4.1, there are 4 possible cases

Case 1 : $s_1 > d - r - 1$
Case 2a : $s_1 \leq d - r - 1$, $s_2 \leq r$
Case 2b : $r < s_1 < d - r - 1$, $s_2 > r$
Case 3 : $s_1 \leq r$

Case 1 : By lemma 4.18, the path length $p \leq 2r + 2$.

Case 2a : By lemma 4.13, the path length $p = 1 + s_0 + s_3 \leq d/2$ if $HD(s, t) > d/2$.

Case 2b : If the rightmost bit of s is in S_{01}, by using lemmas 4.7 and 4.10, we can show that the path length $p \le d - 2r$. If the rightmost bit of s is in S_{00}, by using lemmas 4.9 and 4.12, we can show that the path length $p \le d - 2r + 1$.

Case 3 : If $s_2 > d - r - 1$, we interchange s and t so that case 1 becomes applicable. If $s_2 \le d - r - 1$, we interchange s and t, so that case 2a becomes applicable.

Now it follows that the path length is less than or equal to $\max(2r + 2, d - 2r + 1, d/2)$.

Regarding the deletion of links we make the following observations :

Observation 1 :For $HD(s, t) > 1$, there is always a path of length $HD(s, t)$ between s and t (by lemma 15).

Observation 2 : For $HD(s, t) = 1$, and the d-th bit of s and t are same, the link (s, t) is not deleted.

Observation 3 : For $HD(s, t) = 1$, and the link (s, t) is deleted, there exists a path of length 3, between s and t. In this case s is connected to $\bar{s}$ and t is connected to $\bar{t}$.

Now, consider a path between s and t via a bridge $(x, \bar{x})$ which is of length $HD(s, x) + 1 + HD(\bar{x}, t)$ before deletion. If $HD(s, x) > 1$ and $HD(\bar{x}, t) > 1$, the path length does not change after deletion (by observation 1). If $HD(s, x) = 1$ and the link (s, x) is deleted, then s is connected to $\bar{s}$. If $HD(\bar{s}, t) > 1$, then there is a path of length $1 + HD(\bar{s}, t)$ between s and t (by obseravtion 1), which is less than or equal to d/2 for $HD(s, t) > d/2$. If $HD(\bar{s}, t) = 1$, the path between s and t is of length at most 4 (by observation 3). ◆

Example 4.8 : Take an 8-cube and apply twists of type 1. Take s = 00000000 and t = 11111110. Since $s_1 = 7 > 6 = d - r - 1$, this falls under case 1. In order to reach a node W_1, we first use one link to reach 00000010, say. Then we take the twisted link to reach 11111101. Now note that we can not go to 11111100 because that link has been deleted due to the twist. So we reach our destination via 11111111 in another 2 steps. Thus the path length is 4, which is equal to $2r + 2$.

Lemma 4.20 : In a d-cube (d being an even number), if we apply twists of type 0 and type r, where r is an integer between 1 and $d/2 - 1$, path length between any two points s and t is less than or equal to $\max(r + 2, d - 2r + 1, d/2, 5)$.

Proof: First consider that only bridges are added, but no links are deleted. We can assume w.l.o.g. that $s_1 \leq s_2$. Because of the symmetry of the twisted cube, we assume s to be in some A_k, $0 \leq k \leq d-1$ (proof for the other case is similar).

We define s_0, s_1, s_2 and s_3 as before. There can be 4 possible cases as in theorem 4.1.

Case 1 : $s_1 > d - r - 1$
Case 2a : $s_1 \leq d - r - 1,\ s_2 \leq r$
Case 2b : $r < s_1 < d - r - 1,\ s_2 > r$
Case 3 : $s_1 \leq r$

Case 1 : Using lemmas 4.2 and 4.3, we can show that the path length $p \leq r + 2$.

Other cases are treated as in lemma 4.19.

Now it follows that the path length is less than or equal to $\max(r + 2, d - 2r + 1, d/2)$.

Regarding the deletion of links we have the following observations :

Observation 1 : For $HD(s, t) > 2$, there is always a path of length $HD(s, t)$ between s and t.

Observation 2 : For $HD(s, t) \leq 2$, and one of s and t is not connected to a bridge, there exists a path of length $HD(s, t)$ between s and t.

Observation 3 : For $HD(s, t) = 1$, and both of s and t have bridges connected to them, there exists a path of length at most 3, between s and t.

Observation 4 : For $HD(s, t) = 2$, and both of s and t have bridges connected to them, there exists a path of length at most 4, between s and t.

Now, consider a path between two points s and t via a bridge $(x, \bar{x})$ which is of length $HD(s, x) + 1 + HD(\bar{x}, t)$ before deletion. If $HD(s, x) > 2$ and $HD(\bar{x}, t) > 2$, the path length does not change after deletion (by observation 1). Consider $HD(s, x) \leq 2$, and s is connected to $\bar{s}$. If $HD(\bar{s}, t) > 2$, then there is a path of length $1 + HD(\bar{s}, t)$ between s and t (by observation 1), which is less than or equal to d/2 for $HD(s, t) > d/2$. If $HD(\bar{s}, t) \leq 2$, the path between s and t is of length at most 5 (by observation 3). ◆

Lemma 4.21 : If we apply a twist of type 1, on a d-cube (d being an even number ≥ 6), its diameter is reduced by 2.

Proof: By lemma 4.19 the path length is less than or equal to $(d - 2r + 1, d/2, 2r + 2, 4)$. Putting $r = 1$ we get a maximum path length of $d - 1$ corresponding to $d - 2r + 1$. But the path length is less than $d - 2r + 1$ if the d-th bit of s and t are different. If the d-th bit is in S_{00} or S_{11} the path length $p_0 = d - 2r + 1 + s_2 - s_1$ (by lemma 4.9) and $p_1 = d - 2r + 1 + s_1 - s_2$ (by lemma 4.12). In this case, the path length becomes equal to $d - 2r + 1$ only if $s_1 = s_2$, i.e., when $HD(s, t) \leq d - 2$. ◆

Lemma 4.22 : In a 6-cube, if we apply twists of type 0 and type 2, maximum path length between any two points is less than or equal to 4.

Proof: According to lemma 4.20 the maximum path length is 5 for $d = 6$ and $r = 2$. But the path length can be 5, only if $HD(\bar{s}, t) = 2$. In that case $HD(s, t) = 4$. Hence path length is at most 4. ◆

We take a d-cube ($d = n + 6$, $n \geq 1$). In this d-cube we take two subcubes, $C_0 = 0^n*^6$ and $C_1 = 1^n*^6$. We apply twists of type 0 and 2 in C_0 and twist of type 1 in C_1. The total number link pairs exchanged in the process is 16. Now we have the following theorem.

Theorem 4.6 : By applying twists on a d-cube as above ($d \geq 7$), we can reduce its diameter by 3.

Proof: Take two points (s, t) in the above cube. When both s and t are in C_0 or C_1, there is a path of length at most 4 between them. We can assume w.l.o.g. that t is outside C_0, C_1. From the discussion in section 4.3.2, it follows that if no links were deleted in C_0, C_1 the d-cube would have diameter $d - 3$. Now also this result holds, as whatever links are deleted are of the type which change the d-th bit. We can change the d-th bit, if required, outside C_0, C_1. ◆

Lemma 4.23 : In an 8-cube, if we apply twists of type 0 and 3 its diameter becomes equal to 5.

Proof: Follows from lemma 4.20. ◆

Lemma 4.24 : If we apply a twist of type 2 in an 8-cube, its diameter becomes equal to 6.

Proof: Follows from lemma 4.19. ◆

Lemma 4.25 : If we apply twists of type 0 and 3 on a 9-cube, its diameter becomes equal to 5.

Proof: Follows from lemma 4.20 where we replace d/2 by $\lceil d/2 \rceil$. ◆

In a d-cube ($d = n + 8$, $n \geq 2$) we exchange 57 link pairs in the following way. We apply twists of type 0 and 3 within one 8 dimensional subcube $0^n *^8$. We apply a twist of type 2 within another subcube $1^n *^8$. Now we have the following theorem.

Theorem 4.7 : The d-cube twisted as above has diameter equal to $d - 4$.

Proof: The proof is similar to that of theorem 4.6. ◆

Theorem 4.8 : In a d-cube (d being an even number ≥ 10) if we apply twists of type 0 and r where $r = \lfloor d/4 \rfloor + 1$, its diameter becomes equal to d/2.

Proof: Follows from lemma 4.20.

4.5.1 Reduction in Diameter by d/2, d an Even Number and $d \geq 10$

If we want to reduce the diameter by d/2 in an (n+d)-cube ($n \geq 1$ and $d \geq 10$) we apply some twists in a d-dimensional subcube $C_0 = 0^n *^d$ so that any two nodes within this subcube have a path between them of length at most d/2. In another d-dimensional subcube $C_1 = 1^n *^d$ we apply some twists so that any two nodes within this subcube have a path between them of length at most d/2+1. Now we state the following Theorem.

Theorem 4.9 : The (n+d)-cube twisted as above will have diameter equal to $n + d/2$.

Proof: We define x, y, p_0, p_1, p, q as in theorem 4.3. Hence, $p_0 \leq d/2$ and $p_1 \leq d/2 + 1$. If $p \geq q$, the path between s and t, via C_0 is of length $n - p + q + p_0 \leq n + d/2$. If $p < q$, the path between s and t, via C_1 is of length $n + p - q + p_1 < n + d/2 + 1$. ◆

4.5.2 Reduction in Diameter by 2m ($m \geq 4$) in Q_{n+4m} ($n > 0$)

We take two 4m-dimensional subcubes $C_0 = 0^n *^{4m}$ and $C_1 = 1^n *^{4m}$. We apply T_0 and T_{m+1} within C_0 so that any two nodes in C_0 are apart by a distance of at most 2m. We apply some twists within C_1 so that any two nodes in C_1 are apart by a distance of at most 2m+2. We call the resultant graph Q'_{n+4m} and claim that the diameter of Q'_{n+4m} is $n + 2m$.

Theorem 4.10: The (n+4m)-cube twisted as above will have diameter equal to n+2m.

Proof: We represent two nodes s and t as $s = a_1 a_2 \ldots a_n a_{n+1} \ldots a_{n+4m-2}$ and $t = b_1 b_2 \ldots b_n b_{n+1} \ldots b_{n+4m-2}$. We define two strings x, y as $x = a_{n+1} a_{n+2} \ldots a_{n+4m-2}$ and $y = a_{n+1} a_{n+2} \ldots a_{n+4m-2}$. We define p, q, p_0, p_1, s_0, s_1, s_2, s_3, P_0 and P_1 as in theorem 4.3. Hence $p_0 \le 2m$ and $p_1 \le 2m + 2$. We consider only those s, t for which HD(s, t) > n + 2m, i.e., $p + q + s_0 + s_3 \le 2m - 1$.

When $p_0 \neq 1 + s_0 + s_3$, $p_0 \le \max(m+2, 2m-1) \le 2m-1$.
Hence if $q - p \le 1$, $P_0 = n + (q - p) + p_0 \le n + 2m$
if $q - p > 1$, $P_1 = n - (q - p) + p_1 < n + 2m + 1$, i.e., $P_1 \le n + 2m$.
When $p_0 = 1 + s_0 + s_3$, $P_0 = n - p + q + 1 + s_0 + s_3 \le n + p + q + 1 + s_0 + s_3 \le n + 2m$.
◆

Let X(k) denotes the number of link pairs to be exchanged to reduce the diameter by k in Q_{n+2k} (n > 0) then we can write

$$X(k) = \sum_{i=3}^{m} \left\{ \binom{4i-1}{i+1} + 1 \right\} + 57 \ \{x(4) = 57\} \qquad \text{for } k = 2m, \text{ and } m > 2,$$
$$X(2m) + \binom{4m+1}{m+1} + 1, \qquad \text{for } k = 2m+1, \text{ and } m > 2,$$

4.6 Routing Algorithms

4.6.1 Routing in a Bridged Q_d of Diameter $\lceil d/2 \rceil$

In this section we describe the routing strategy in a bridged hypercube of dimension d (d > 4). The routing algorithm merely consists of

i) deciding whether a bridge is to be used or not.
ii) if a bridge $(x, \bar{x})$ is to be used, then finding that node x from which a bridge will be used.

A packet to be routed from the source node s to the destination node t, is transmitted along with a tag like this :

t	b	C

where t is the destination node, b is a flag bit and C is a d-bit vector used for the following purposes.

b = 1 indicates that a bridge is to be used and b = 0 indicates that we can route the packet without using any bridge.

C is called the routing vector. Each bit of C is associated with a specific dimension of the hypercube so that if a bit of C is '1', then we transmit the packet along a link of the hypercube in that dimension. Thus, given a routing vector C, we scan the bits of C from one end, say the left end and whenever we encounter a '1', the packet is transmitted from the intermediate node along the corresponding dimension.

When $s_1 + s_2 \leq \lceil d/2 \rceil$ we set b = 0 and c is set to $s \oplus t$. When $s_1 + s_2 > \lceil d/2 \rceil$ we set b = 1. In this case, we have to find the node x from which a bridge will be used. For this, we first enumerate s_0, s_1, s_2 and s_3. For different values of s_1 and s_2 we enumerate p_0, p_1 and p as done in the proof of theorem 4.1. Then we find the minimum path length. Now to find the node x to achieve this minimum path length, we have to suitably change certain bits in s in a way as described in the lemma which corresponds to this minimum pathlength. The routing vector C is then set to $s \oplus x$. Routing from s to x is done by scanning the bits of C from left. The bridge $(x, \bar{x})$ is then used. Then we set b = 0 and set $C = \bar{x} \oplus t$ and route according to this C.

4.6.2 Routing in a Twisted Hypercube

We now describe the routing strategy for a twisted hypercube Q_d^T on which we apply T_0 and T_r (for $r = \lfloor d/4 \rfloor + 1$). We discuss here the routing technique for hypercubes of even dimension $d\,(d \geq 10)$. The cases for twisted hypercubes of smaller dimension can similarly be dealt with. The scheme for routing a message from the source node s to the destination node t is described as below.

1) If $HD(s, t) \leq d/2$, define the routing vector $C = s \oplus t$. Let $C = c_1 c_2 \ldots c_n$. For two nodes $a = a_1 a_2 \ldots a_n$ and $b = b_1 b_2 \ldots b_n$, if a and b differ only in the i-th bit then transmitting from a to b is equivalent to transmitting along the dimension i.

1a) If HD(s, t) = 1 and the link (s, t) is present, then transmit to t.

1b) If HD(s, t) = 1 and the link (s, t) is deleted (because of twisting), then the routing will be via a node s′ adjacent to s. From s′ send the packet along the d-th dimension to t′ (t′ will be adjacent to t) and then route from t′ to t.

1c) If HD(s, t) > 1, then let $c_i, c_j, \ldots, c_k$, $i < j < \ldots < k$ be the bits in C which are 1. There can be following cases :

i) s and t do not differ in the rightmost bit. In this case the deleted links do not affect the path between s and t. The routing is the same as in the original (untwisted) cube Q_d.

ii) s and t differ in some bits including the rightmost bit. If s is not connected to $\bar{s}$, then transmit the packet from s along the dimensions k, ... , j, i and in that order. If s is connected to $\bar{s}$, then first transmit along the dimension i and then along the dimension d. Afterwards, transmit to t.

2) If HD(s, t) > d/2, then we use the link $(x, \bar{x})$ and route from s to x, then from x to $\bar{x}$ and finally from $\bar{x}$ to t. Our aim is to find x from s. If t is such a node that $(t, \bar{t})$ is connected then $x \leftarrow \bar{t}$; otherwise x is found as follows depending on the values of s_1 and s_2.

2a) $s \in A_k$, $0 \leq k \leq d - 1$.

Case 1 : $s_1 > d - r - 1$
Case 2 : $s_1 \leq d - r - 1$, $s_2 \leq r$
Case 3 : $r < s_1 < d - r - 1$, $s_2 > r$
Case 4 : $s_1 \leq r$

In each case we find the minimum path length p as done in the proof of lemma 4.19. To find the node x to achieve this minimum path length, we have to suitably change certain bits in s in a way as described in the lemma which corresponds to this minimum path length.

2b) $s \in B_k$, $1 \leq k \leq d$.

Because of the symmetry of the structure, routing can be done as follows :

For a source destination pair (s, t), define a new pair (s′, t′) where s′ differs from s in the d-th bit and t′ differs from t in the d-th bit. Then find a bridge $(y, \bar{y})$ to be used for this new pair (s′, t′) as in case 2a. Now for s the corresponding bridge to be used is $(x, \bar{x})$ where x differs from y in the d-th bit.

Routing from s to x and from x to t, in both the cases 2a and 2b above will be done following the method described above for $HD(s, t) \leq d/2$.

4.7 Conclusion

Hypercubes have various applications in parallel processing. This chapter describes different methods for reducing the diameter of a hypercube by adding some extra links, called bridges and also by exchanging some pairs of independent links (without any extra link). The addition of bridges will not only reduce the diameter but also the average internode distance. In this chapterer we have aimed at reducing the diameter, by adding as few links as possible. Also, we have given an algorithm for routing in such bridged hypercubes. We add $\binom{d}{r} + 1$ bridges (where $r = \lfloor d/4 \rfloor + 1$) to a d-cube ($d > 4$) to reduce its diameter by $\lfloor d/2 \rfloor$. The extra links constitute a small fraction of the total number of links and this fraction is $\binom{d}{r} + 1 : d.2^{d-1}$, which decreases with increase in d. One important result is that the number of bridges to be added to reduce the diameter by k remains constant for all hypercubes of dimension greater than 2k.

Twisting of hypercubes has an extra advantage of reducing the diameter without changing the degree of nodes. Though the routing in this case becomes a little more complicated we have devoloped a suitable algorithm for routing in a twisted hypercube. Also, the number of link pairs exchanged to reduce the diameter by a given value is much small compared to that given in [HKS87]. For example, we apply twists of type 0 and type 3 in an 11-cube to get a graph of diameter 6. The number of link pairs exchanged in the process is 121. In comparison, the number of link pairs to be exchanged in the method given in [HKS87] is equal to $10.2^7 = 1280$. It will be interesting to study the improvement of some other performance measures such as average routing distance, traffic density etc. of the architecture proposed in this chapter over the original hypercube.

Chapter 5

Fault-Tolerant Hamiltonian Topologies

5.1 Introduction

Design of the topology of a computer network is guided by many mutually conflicting requirements. It is not always possible to get a network design, which is optimum from all angles. Depending on the requirements and their priorities, one has to design a suitable network structure. Certain applications need some particular properties to be present in the design. We consider, in this chapter, the application areas requiring the presence of a hamiltonian cycle in the structure. An example of such an application may be a distributed operating system where mutual exclusion of certain shared resources is implemented by a 'Token Passing' approach [PS87]. In such cases, the token passes along a logical ring. As long as the structure is connected, it is possible to construct a logical ring by revisiting some of the nodes. But it is desirable that a hamiltonian cycle be present in the network so that all the nodes in the network get equal chance to grab the token.

Let $G = (V, E)$ be a graph, where V is the set of nodes and E is the set of links. Let $|V| = N$. Two nodes are said to be *adjacent* if there is a link joining them. A *cycle* is a sequence of three or more nodes such that i) two consecutive nodes are adjacent and ii) the first and the last nodes are the same. A cycle is called a *hamiltonian cycle,* if its nodes are distinct and they span V. A graph having a hamiltonian cycle is called a *hamiltonian graph.*

Definition : A graph is said to be *1-node-deleted hamiltonian* if the graph is hamiltonian after the deletion of any node and its adjacent links.

Definition : A graph is said to be *1-link-deleted hamiltonian* if it is hamiltonian after the deletion of any link.

Many network topologies have hamiltonian cycles embedded in them. In this chapter, we try to construct a network graph which is hamiltonian as well as 1-link-deleted hamiltonian and 1-node-deleted hamiltonian. After a node or a link fails, the degree of some of the nodes may decrease by one. But the presence of a hamiltoian cycle needs that the degree of every node should be at least 2. Hence, for a graph to be 1-link-deleted hamiltonian or 1-node-deleted hamiltonian, all nodes must have degree at least 3. Here we propose a design such that i) when N, the total number of nodes is even, all nodes are of degree 3 and ii) when N is odd, all but one of the nodes are of degree 3 and the remaining node is of degree 4. The proposed graph has minimum number of links among all the graphs which are 1-link-deleted hamiltonian or 1-node-deleted hamiltonian.

5.2 Design of the Graph

First we consider the construction of the graph for $N = 6n+4$ nodes for $n \geq 0$.

1) Connect $6n+3$ nodes to form a ring.
2) Place the remaining node, labelled as $V(0)$, at the centre of the ring and connect it to three equidistant nodes on the ring. Label these nodes on the ring as $V(1)$, $V(2)$ and $V(3)$ in the clockwise direction.
3) Starting from each $V(i)$, $i = 1, 2, 3$, label the n nodes encountered in traversing the ring counterclockwise as $V(i, 1), V(i, 3), \dots, V(i, 2n-1)$. Similarly, the n nodes encountered in the clockwise traversal from $V(i)$, $i = 1, 2, 3$, are labelled as $V(i, 2), V(i, 4), \dots, V(i, 2n)$. As the distance along the ring between $V(i)$ and $V(j)$, $i, j \in \{1,2,3\}$, $i \neq j$, is $2n$, there will not be any conflict in such labelling of the nodes.
4) Join $V(i, 2j-1)$ to $V(i, 2j)$ by a *chord-link* for all i, j, $i = 1, 2, 3$ and $1 \leq j \leq n$.

Let us call the resultant graph as G. Figure 5.1 shows an example of such a graph with $N = 16$ nodes.

G can easily be generalized for any even number of nodes. Starting from $6n+4$ nodes we can put two more nodes in G. We introduce the two new nodes on the ring as follows : i) $V(1, 2n+1)$ is placed between $V(1, 2n-1)$ and $V(3, 2n)$, ii) $V(1, 2n+2)$ is put between $V(1, 2n)$ and $V(2, 2n-1)$. Then we join $V(1, 2n-1)$ and $V(1, 2n)$ by a link. This gives us the graph for $N = 6(n+1)$ nodes. Figure 5.2 shows an example for $N = 18$ nodes. Similarly, we can put two more nodes, $V(2, 2n+1)$ (in between $V(2, 2n-1)$ and $V(1, 2n+2)$ on the ring) and $V(2, 2n+2)$ (in between $V(2, 2n)$ and $V(3, 2n-1)$ on the ring). An example for $N = 20$ nodes is shown in figure 5.3. Finally, two more nodes, $V(3, 2n+1)$ (in between $V(3, 2n-1)$ and $V(2, 2n+2)$ on the ring) and $V(3, 2n+2)$ (in between $V(3, 2n)$ and $V(1, 2n+1)$ on the ring) can be added to

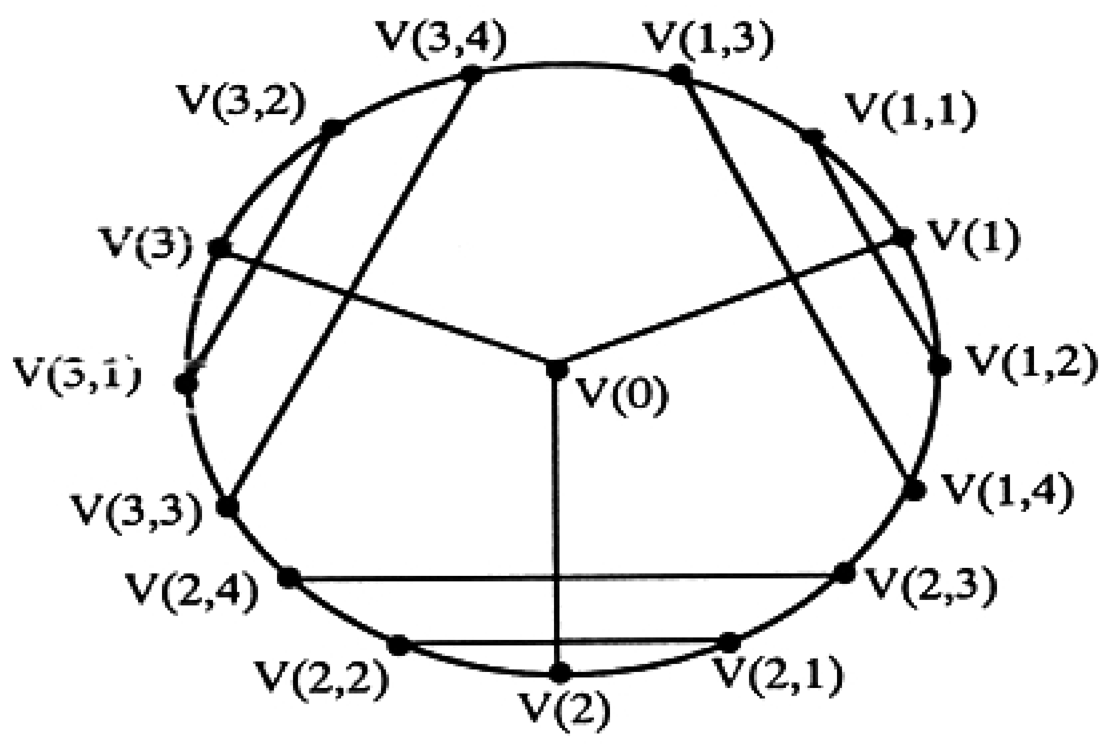


Figure 5.1 : An example of the proposed graph for N=16 nodes

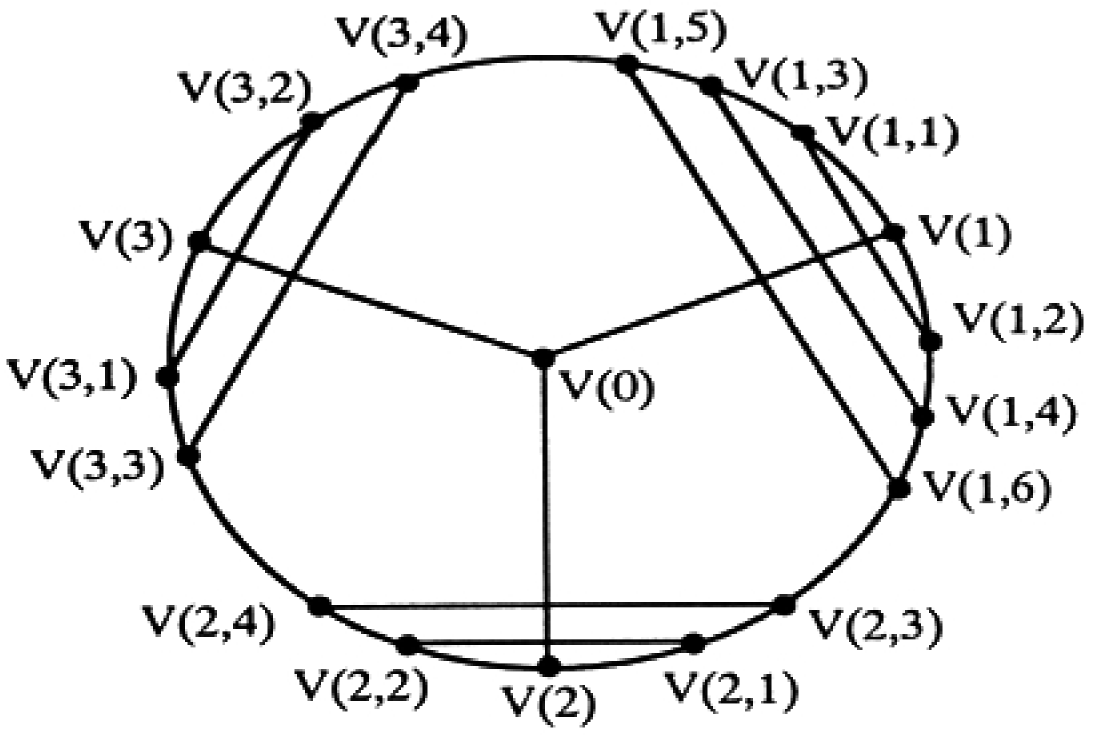


Figure 5.2 : The proposed graph with N=18 nodes

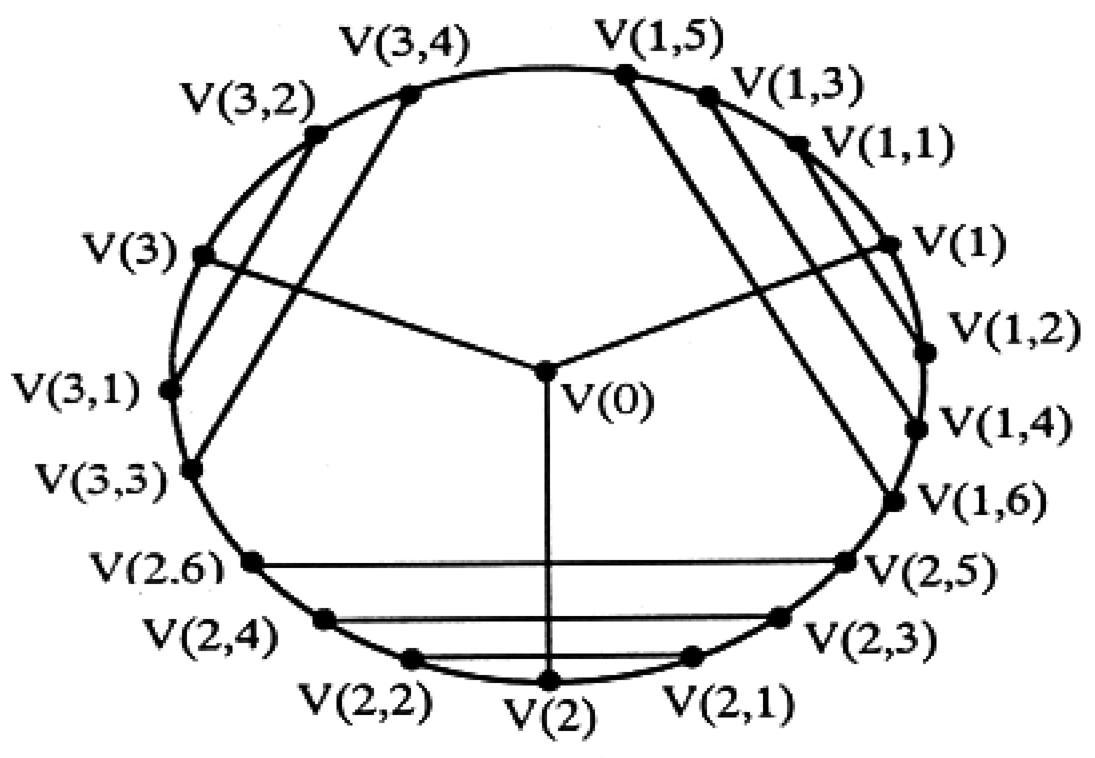


Figure 5.3 : The proposed graph for N=20 nodes

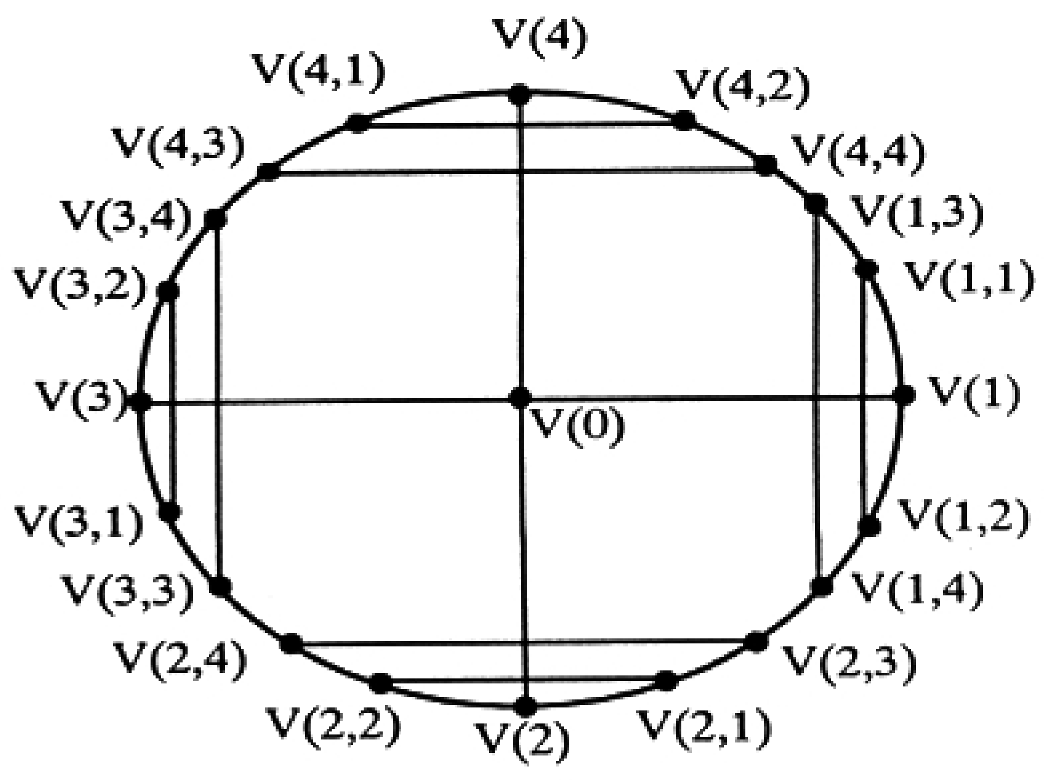


Figure 5.4 : The proposed graph for N=21 nodes

generate the graph for N = 6(n+1) + 4 nodes.

We now extend the design technique to the case where the total number of nodes, N, is odd. For odd values of N, it follows from the degree-sum criterion that we can not make the graph 3-regular. However, allowing one node of degree 4 and the rest of degree 3, we can easily generalize the previous design algorithm to get a graph G'. We take the case of 8n+5 nodes. First we place 8n+4 nodes on a ring. Then we place a central node and join it to 4 nodes on the ring such that the spokes divide the ring into 4 equal parts (as opposed to 3 in the previous case). Then we similarly name the nodes around each spoke and join them. Again, as in the case of even number of nodes, we can add to it two nodes at a time, thus designing the network for any odd N. Figure 5.4 shows an example of the graph with 21 nodes.

5.3 Properties

Theorem 5.1 : The graph G with even number of nodes has the following properties :

a) It is trivalent.
b) It is planar.
c) It is hamiltonian.
d) It is 1-link-deleted hamiltonian.
e) It is 1-node-deleted hamiltonian.
f) It has a diameter $\lfloor N/6 \rfloor + 2$.
g) It is incrementally extensible by two nodes.

Proof: We prove the results for $N = 6n + 4$. For other even values of N, the proof is similar.

a) Obvious.

b) Planarity follows if we draw the chord-links between V(i, 2j−1) and V(i, 2j), i = 1, 2, 3, $1 \leq j \leq n$, outside the ring. Figure 5.5 shows a planar emebedding of the graph for N = 16 nodes.

c) First we consider the case when n is odd.

The following sequence of nodes gives one hamiltonian cycle :

V(0), V(1), V(1, 1), V(1, 2), V(1, 4), V(1, 3), V(1, 5), V1, 6), ... , V(1, 2n−1), V(1, 2n), V(2, 2n−1), V(2, 2n−3), V(2, 2n−5), ... , V(2,1), V(2), V(2, 2), V(2, 4), V(2, 6), ... , V(2, 2n), V(3, 2n−1), V(3, 2n), V(3, 2n−2), V(3, 2n−3), V(3, 2n−5), ... , V(3, 1), V(3, 2), V(3), V(0). ... (5.1)

Similarly, for even values of n, a hamiltonian cycle is given by :

V(0), V(1), V(1, 1), V(1, 2), V(1, 4), V(1, 3), V(1, 5), V1, 6), ... , V(1, 2n), V(1, 2n−1), V(3, 2n), V(3, 2n−2), V(3, 2n−4), ... , V(3, 2), V(3), V(3, 1), V(3, 3), V(3, 5), ..., V(3, 2n−1), V(2, 2n), V(3, 2n−1), V(2, 2n−3), V(2, 2n−2), V(2, 2n−4), ... , V(2, 1), V(2, 2), V(2), V(0).

d) We first note that we can get hamiltonian cycles, other than that described in sequence (5.1), by interchanging the roles of V(1), V(2) and V(3).

Case I : Let the faulty link be one between V(k, 2j–1) and V(k, 2j), for some k and j, $k \in \{1, 2, 3\}$, $1 \leq j \leq n$. If k = 2, the link is anyway not included in the hamiltonian cycle described in part (c). If k = 1 or 3, we can interchange the roles of V(2) and V(k) in sequence (5.1) to get a hamiltonian cycle without using the faulty link.

Case II : Let the faulty link be between V(0) and V(k), $k \in \{1, 2, 3\}$. If k = 2, the link is anyway not included in the hamiltonian cycle described in part (c). If k = 1 or 3, we can interchange the roles of V(2) and V(k) in sequence (5.1) to get a hamiltonian cycle without using the faulty link.

Case III : Let the faulty link be between two nodes V(k, j) and V(k, j+2), for some k and j, $k \in \{1, 2, 3\}$, $1 \leq j \leq n-2$. For the hamiltonian cycle, we start from V(0) and go to V(k) first. The choice of the next node is guided by the value of j so as to avoid the faulty link. If j = 4p or 4p+1, then we go to V(k, 1) else we go to V(k, 2) from V(k). The rest of the sequence can be computed in the manner similar to that discussed in part (c).

e) First we take the case when n is odd.

Case I : *V(0) is faulty.*
The hamiltonian cycle exists along the ring.

Case II : *V(k), $k \in \{1, 2, 3\}$ is faulty.*
Without loss of generality, let V(2) be faulty. We can modify the hamiltonian cycle as described in part (c), by replacing the subsequence 'V(2, 1), V(2), V(2, 2)' by 'V(2, 1), V(2, 2)'.

Case III : *V(1, i), $1 \leq i \leq 2n$, is faulty.*
Let V(1, i) be connected to V(1, i') by a chord-link, where i' = i+1 or i–1. To get a hamiltonian cycle, we start with the sequence V(0), V(1). Next we move either to V(1, 1) or to V(1, 2) depending on the value of i so that V(1, i) can be bypassed by the subsequence given as follows :

For n = 1,

i) V(1), V(1, i'), V(2, 1) for i' = 2
ii) V(1), V(1, i'), V(3, 2) for i' = 1

For $n > 1$,

i) $V(1), V(1, i'), V(1, i'+2)$ for $i' \leq 2$.
ii) $V(1, i'-2), V(1, i'), V(1, i'+2)$ for $3 \leq i' \leq 2n-2$.
iii) $V(1, i'-2), V(1, i'), V(3, 2n)$ for $i' = 2n-1$.
iv) $V(1, i'-2), V(1, i'), V(2, 2n-1)$ for $i' = 2n$.

The rest of the sequence can be computed in a manner similar to that described in part (c).

f) From each node we first find the maximum number of steps needed.

i) From $V(0)$ we can reach any other node within $n+1$ steps.

ii) From $V(1)$, $V(2)$ or $V(3)$ we can reach any other node (via $V(0)$) within $n+2$ steps.

iii) From $V(1, i)$ (without loss of generality let $i = 2m$), $V(0)$ can be reached within $m+1$ steps.

So, $V(k, j)$, $k = 2, 3$, can be reached via $V(0)$, within $n+2$ steps for $1 \leq j \leq 2(n-m)$. Now, $V(2, 2n-1)$ can be reached via $V(1, 2n)$ in $(n - m + 1)$ steps. So, $V(2, 2j-1)$ for $n - m < j \leq n$ can be reached within n steps. Hence $V(2, 2j)$, for $n - m < j \leq n$, can be reached within $n+1$ steps. Similarly, we can go to $V(1, 2m-1)$ in one step. From there we can go to $V(3, j)$ for $2(n - m) \leq j \leq 2n$ in maximum of $n+1$ more, i.e., $n+2$ total steps.

We also note that the distance between some pairs of nodes, for example $V(2)$ and $V(3, 2n)$, is exactly $n+2$. Hence the diameter is exactly $n+2$.

g) Follows from the design algorithm.

Theorem 5.2 : The graph G' with odd number of nodes has the following properties :

a) All nodes except $V(0)$ has degree 3 and $V(0)$ has degree 4.
b) It is planar.
c) It is hamiltonian.
d) It is 1-link-deleted hamiltonian.
e) It is 1-node-deleted hamiltonian.
f) It has a diameter $\lfloor N/8 \rfloor + 3$.
g) It is incrementally extensible by two nodes.

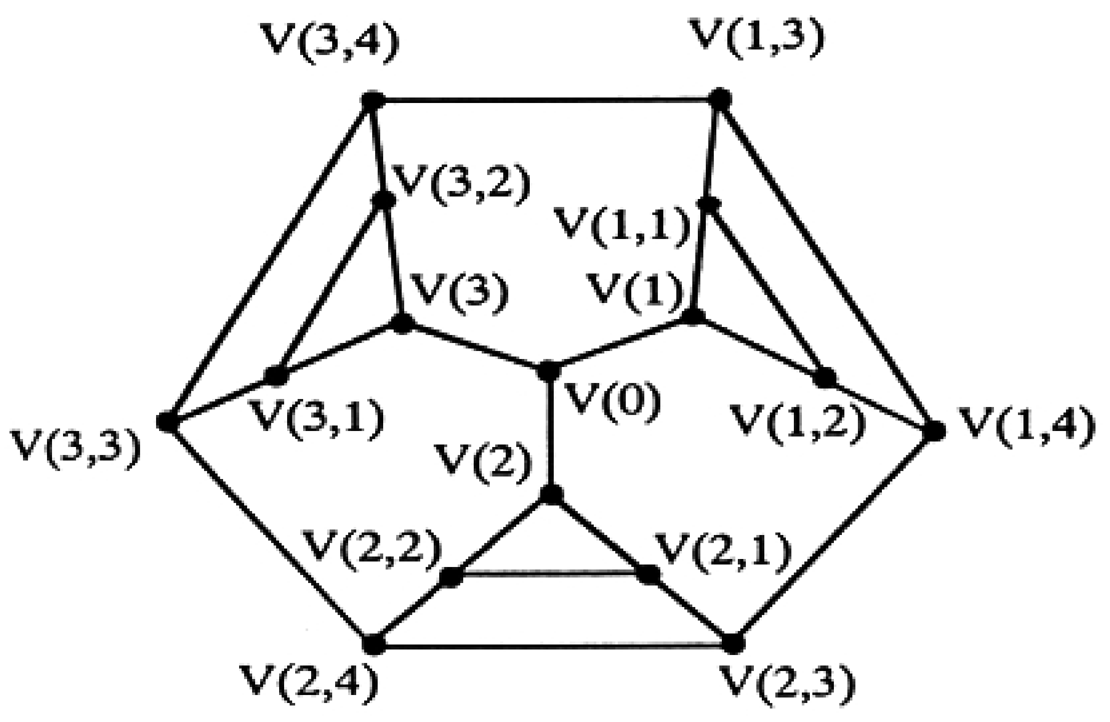


Figure 5.4 : A planar embedding of the graph in figure 5.1.

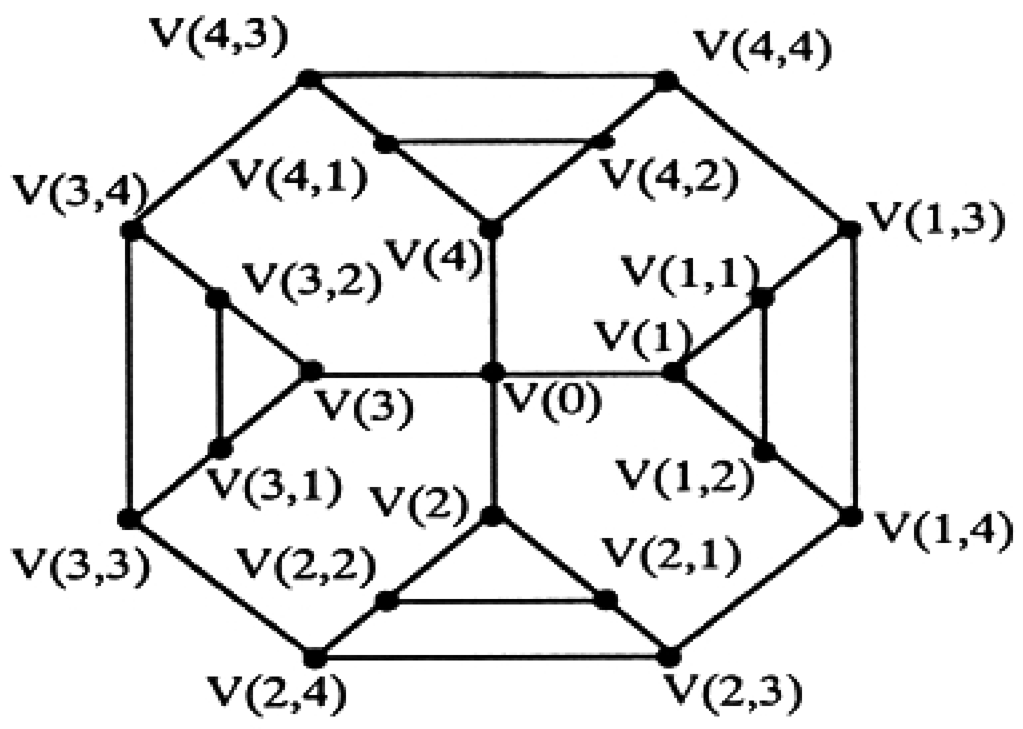


Figure 5.6 : A planar embedding of the graph in figure 5.5.

Proof : Similar to that of Theorem 5.1.

Figure 5.5 shows an example of the graph with N = 21 nodes and figure 5.6 shows a planar embedding of that graph.

5.4 Conclusion

Although the graphs we have proposed in this chapter are hamiltonian, 1-node-deleted hamiltonian and 1-link-deleted hamiltonian, they are not optimal with regard to diameter. There are many denser trivalent and tetravalent graphs reported in the literature [SSB+91], [LS82]. As an example, if we take the case of the trivalent Möbius graph, it has diameter $O(\log N)$ and it also has a hamiltonian cycle. However, the existence of a hamiltonian cycle with a single node or link failure in a Möbius graph remains yet to be established. Also the problem with the Möbius graph is that it is defined only for $N = 2^n$ nodes. In other words Möbius graph is not incrementally extensible. Further, for odd n, two of its links overlap, thus making it not exactly trivalent. Similar problems regarding incremental extensibility also appear in case of dense tetravalent de Bruijn graph [PR82]. The graph structure that we have proposed here is, on the other hand, incrementally extensible.

Chapter 6
Reliability Analysis of Static Networks

6.1 Introduction

A good network structure should have a number of desirable properties some of which are mutually conflicting in nature. Fault-tolerance of a network topology is of fundamental importance. The faults to be considered may be processor or link failures. A network topology is said to be fault-tolerant if it remains 'operational' in the presence of faults. It is, however, the topological requirements set by the application environment that essentially determine when a network is considered operational. For special purpose networks, the requirement may be that the induced subgraph on the live or non-faulty nodes satisfies some specific property, e.g., embedding some particular structure (say, a complete binary tree or a binary cube). For general purpose networks, usually it is considered 'operational' as long as the induced subgraph [Ha69] on the live nodes is connected.

Though there is no universally accepted measure for reliability, a simple measure for general topology may be the connectivity (i.e, the minimum number of nodes which are to be removed in order to make the network graph disconnected [Ha69]). A more intricate approach may be to use a stochastic model [BC89]. Here, with each node and link we assign a probability of failure. The failures are assumed *to occur independently. Details of the model are described later. Under this set-*up, our aim is to find the probability that the network is connected. Some work [CH88], [SF72], [AR81], [GG81], [KA90], [Po71], [LP71] has been reported on finding the reliability when only the links have positive failure probability.

In this chapter, we formally define the reliability of a network. Then we give a method for recursively computing the reliability when the links are fault-free. Lastly, we derive analytical expressions for finding the reliabilities of certain common networks like complete graph, path, cycle, star, wheel, complete m-ary tree, ladder etc.

6.2 Basic Concepts

In the past, the fault-tolerance of a network has usually been expressed in terms of the node connectivity of the corresponding network. Though such a measure is relatively easier to calculate, it does not provide enough information regarding the fault-tolerance of the network. It suffers from two shortcomings :

(1) Being a discrete measure, it cannot compare between two graphs with same node connectivity, though one may be obviously more robust than the other. For example, consider the network in figure 6.1(a). It has connectivity 2. Putting an additional link between nodes 1 and 3 in the structure, we get the graph as in figure 6.1(b), where connectivity is also 2. But definitely the graph in figure 6.1(b) is more fault-tolerant than the graph in figure 6.1(a). For example, if nodes 2 and 4 fail, the graph in figure 6.1(a) becomes disconnected, whereas that in figure 6.1(b) is still connected. This observation demands a new measure of network reliability, which would show that a super-graph on the same set of nodes is more reliable.

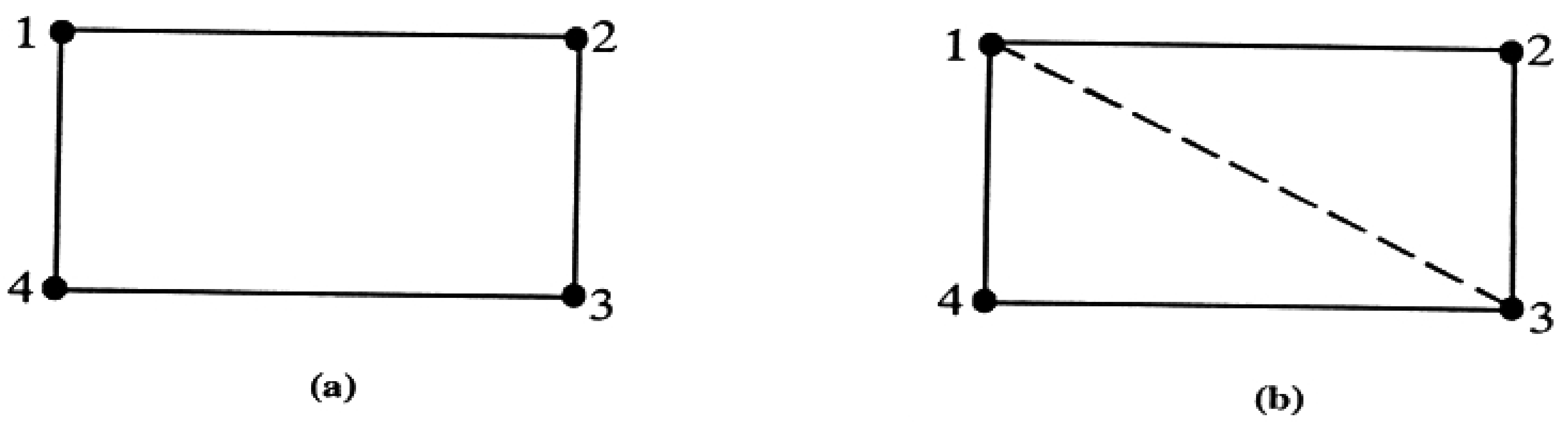


Figure 6.1 : Network Graphs

(2) Failure-probabilities of individual nodes may have a significant effect on the reliability of a network. As an example, in the wheel-graph, the failure-rate of the central node has the maximum effect on the reliability of the network. But the node connectivity of the network graph does not reflect such dependency.

Being so motivated, we formally define the network reliability under the following model :

Model : Consider a network graph G = (V, A), where V is the node set and A is the link set, with the following attributes :

(1) With every node we assign a probability of failure.

(2) Node faults are assumed to occur independently. When two nodes are alive (non-faulty) we assign a probability that the link between them is alive. (If there is no connection at all, we assume this probability to be zero.) When one of the terminal nodes of a link is dead, the link automatically dies.

Definition : For a network G = (V, A), we define the reliability of the network (denoted by R(G)) as R(G) = P (E), where P(.) denotes probability, and E is the event that the induced subgraph on the live nodes is connected.

In terms of sets, this beomes

$$E = \bigcap_{\substack{u,v \in V \\ u \neq v}} L_u^c \cup L_v^c \cup (L_u \cap L_v \cap C_{u,v})$$

where, L_u = The event that the node u is alive.
$C_{u,v}$ = The event that there is a path from the node u to the node v.

[For a set S we denote its complement by S^c]

This definition equates reliability to the probability that a live node can commmunicate with any other live node. As a special case, if G is null, we define R(G) = 1.

6.3 Recursive Evaluation of Reliability

Now that we have a quantitative definition of what the reliability of a network is, we must have some way to compute it. Let us consider the case when the links are fault-free, i.e., only the nodes may have positive rate of failure, and the failure rate of any link is zero. We also assume that the failure of a node does not affect, in any way, the failure of another node. That is, L_u and L_v are independent of each other, for any distinct pair u, v ∈ V.

Let us take any node v ∈ V.

$$R(G) = P(E) = P(E \cap L_v^c) + P(E \cap L_v)$$

But $E \cap L_v^c$ is the event that the node v is dead and for any x, y in $V - \{v\}$ if x and y are both alive, there still exists some route of communication between the two (which, of course, does not pass through v).

So, $P(E \cap L_v^c) = R(G - \{v\}).\, P(L_v^c)$

Also, $P(E \cap L_v) = P(E \mid L_v) \,.\, P(L_v)$

Let V1 be the set of all nodes in V, which have a direct link with v and $V2 = V - (V1 \cup \{v\})$.

In the case when v is alive, at least one communication path, between any two live nodes in V1 exists through the node v. Thus one may try to simulate its effect by removing v and making V1 a complete graph. Let G_v be the graph constructed from G by removing v and putting links between all pairs of nodes in V1.

Let E_v = the event that the induced subgraph is connected, given that v is alive,
$E1$ = the event that the induced subgraph on the live nodes of G_v is connected
and $E2$ = the event that all nodes in V1 are dead, and the set of live nodes in V2 is non-empty and they form a connected component.

Lemma 6.1 : Given v is alive, $E_v = E1 - E2$.

Proof : First we show that $E_v \subseteq E1 - E2$.

If E_v happens, then E1 must happen. Because for E1 to happen, there should be a path between two live nodes in $G - \{v\}$. Take two live nodes s and d in $G - \{v\}$. Let 's ... u_1 v u_2 ... d' be the path between s and d in $G - \{v\}$. Then s and d will also have a path in G_v given by, 's ... u_1 u_2 ... d', implying that E1 happens. Thus $E_v \subseteq E1$.

If E2 happens, then there will be at least one live node in V2 which will be disconnected from v in G, implying that E_v cannot happen. Thus $E_v \cap E2 = \emptyset$. Merging these two relations we get $E_v \subseteq E1 - E2$.

Now we show that $E_v \supseteq E1 - E2$.

For E_v to happen, any two live nodes s and d in G should have a connecting path.

Case I : If s and d both are in $G - \{v\}$, there must be a path connecting them in G_v (as E1 happens). If this path involves a link (u_1, u_2) non-existent in G_v, then we replace this link by two links (u_1, v) and (v, u_2) to get the corresponding path between s and d in G, i.e., E_v happens.

Case II : Let $s = v$. If d is in V1, then there is a direct link between s and d in G. If d is in V2 then there must be a live node u′ in V1, otherwise E2 would be true. There also exists a path from u′ to d in G (by Case I). Hence there exists a path from s to d in G. Configuring the two cases, we get $E_v \supseteq E1 - E2$.

Hence the proof. ◆

Lemma 6.2 : Let <V2> denote the induced subgraph on the node set V2. Then,

$$P(E \mid L_v) = R(G_v) - [R(\langle V2 \rangle) - \prod_{u \in V2} P(L_u^c)] \prod_{u \in V1} P(L_u^c)$$

Proof : The proof follows directly from lemma 6.1, if we note that E2 is a subset of E1 and put the values of P(E1) and P(E2). ◆

Hence we get the following theorem,

Theorem 6.1 : For any network, $G = (V,A)$, if the links are fault-free, then for any $v \in V$,

$$R(G) = R(G - \{v\}) \, . \, P(L_v^c) + P(L_v) \left[R(G_v) - (R(\langle V2 \rangle) - \prod_{u \in V2} P(L_u^c)) \prod_{u \in V1} P(L_u^c)\right] \quad ◆$$

Remark : Theorem 6.1 provides us a recursive way of computing the reliability of a network in terms of the reliabilities of networks with fewer nodes.

6.4 Reliabilites of Some Simple Networks

In this section, we find out analytical expressions for the reliabilities of certain families of very simple networks, by applying theorem 6.1.

6.4.1 Complete graph (K_n)

Theorem 6.2 : $R(K_n) = 1$, for $n \geq 1$.

Proof : Let us consider a node u with $P(L_u) = p$. Since all other nodes are connected to u, V2 is null. So $R(<V2>) = 1$. From theorem 6.1,

$$\begin{aligned} R(K_n) &= R(K_{n-1}) \cdot (1-p) + p\,[R(K_{n-1}) - (1-1) \prod_{u \in V1}(L_u^c)] \\ &= R(K_{n-1}) = \ldots = R(K_1) = 1. \end{aligned}$$

◆

6.4.2 Path (P_n)

Here we assume that all the n nodes in the path P_n have equal failure-rate, q (=1 – p, say).

Theorem 6.3 : $R(P_n) = q^n + \sum_{i=1}^{n} (n+1-i)\, p^i q^{n-i}$, for $n \geq 4$.

Proof : We shall prove by induction on n. For the base step of the induction, one can easily check the result for n = 4 and 5.

Let the result be true for all $n < m$, $m \geq 6$. We shall prove the result for m.

Applying theorem 6.1, by choosing the node v as that at one end of the path P^m, we have,

$$R(P_m) = R(P_{m-1}) - pq\, R(P_{m-2}) + pq^{m-1}$$

By induction hypothesis, applying the expressions for $R(P_{m-1})$ and $R(P_{m-2})$, we get,

$$\begin{aligned} R(P_m) &= q^{m-1} + \sum_{i=1}^{m-1} (m-i)\, p^i q^{m-i-1} - pq\,[q^{m-2} + \sum_{i=1}^{m-2} (m-i-1)\, p^i q^{m-2-i}] + pq^{m-1} \\ &= q^{m-1} + \sum_{i=1}^{m-1} (m-i)\, p^i q^{m-i-1} - \sum_{i=2}^{m-1} (m-i)\, p^i q^{m-i} \\ &= q^{m-1} + (m-1)\, pq^{m-2} + \sum_{i=2}^{m} (m-i)\, p^{i+1} q^{m-(i+1)} \\ &= q^m + \sum_{i=1}^{m} (m-i+1)\, p^i q^{m-i} \end{aligned}$$

◆

6.4.3 Cycle (C_n)

We consider the network with n nodes, connected in a cyclic structure. Here also we assume that all nodes have independent and identical failure rate q (= 1 – p).

Theorem 6.4 : $R(C_n) = q^n + \sum_{i=1}^{n-1} n\, p^i q^{n-i} + p^n$ for $n \geq 4$.

Proof : Similar to that of theorem 6.3. ◆

6.4.4 Star (S_n)

We consider a network with n nodes, where (n–1) of the nodes are attached only to one particular single node.

Let the reliability of the central node be p_1 (=1 – q_1) and that of every other node be p (= 1 – q). We also assume that node failures are mutually independent.

Theorem 6.5 : $R(S_n) = q_1 q^{n-1} + (n-1)\, q_1\, p\, q^{n-2} + p_1$.

Proof : Follows from a direct application of theorem 6.1. ◆

Corollary 6.1 : For $p_1 = p$, $R(S^n) = q^n + (n-1)\, pq^{n-1} + p$.

6.4.5 Wheel (W_n)

The wheel Wn with n nodes is defined as

$W_n = K_1 + C_{n-1}$, where '+' denotes the join operation [Ha69].

Theorem 6.6 : $R(W_n) = p_1 + q_1\, q^{n-1} + q_1 \sum_{i=1}^{n-2} (n-1)pq^{n-i} + p^{n-1}q_1$

Proof : Follows from a direct application of theorem 6.1. ◆

Corollary 6.2 : For $q_1 = q$, $R(W_n) = p + q^n + \sum_{i=1}^{n-2} (n-1)\, p^i q^{n-i+1} + p^{n-1} q$

From Theorems 6.3 and 6.4, $R(C_n) - R(P_n) = \sum_{i=1}^{n-1} (i-1)\, p^i q^{n-i}$

which is quite significant for small values of q and large n; although there is just one extra link in C_n.

Again, both P_n and S_n are one-connected and they have the same number of links. But for small q, $R(S_n) > R(P_n)$. However there will be heavy congestion of traffic in the central node.

6.5 Reliabilites of Some More Networks

In the previous chapter we found the analytical expressions of certain simple network topologies, by application of Theorem 6.1. In this chapter we consider some more complicated network graphs and try to get some expressions for their reliabilities.

6.5.1 Complete m-ary Tree ($T_{m,n}$)

Let $T_{m,n}$ denote a complete m-ary tree with height n, i.e, with $1 + m + m^2 + \ldots + m^n = [(m^{n+1} - 1) / (m-1)]$ nodes.

Here we assume that all the nodes have identical failure rate $q = 1 - p$. Let v be the root of the tree.

$R(T_{m,n}) = p\, P(T_{m,n} \text{ is connected} \mid L_v) + q\, P(T_{m,n} \text{ is connected} \mid L_v^c)$,

Given that the root is dead, the tree can be connected if and only if all but one of the subtrees of the root are totally dead and the remaining one is connected (it may also be totally dead).

So, $P(T_{m,n} \text{ is connected} \mid L_v^c) = m\, R(T_{m,n-1})\, (q^{m^n - 1}) - (m-1)\, (q^{(m^{n+1}-1)/(m-1)})$

When the root v is alive, the tree is connected if and only if all the non-null subtrees have their root nodes alive.

So, $P(T_{m,n} \text{ is connected} \mid L_{root})$

$$= \sum_{i=0}^{m} \binom{m}{i} q^{i(m^n-1)/(m-1)} . [\, p.P(T_{m,n-1} \text{ is connected} \mid L_{root})\,]^{m-i}$$

$= [\, p.P(T_{m,n-1} \text{ is connected} \mid L_{root}) + q^{(m^n-1)/(m-1)} \,]^m$

Thus we can compute $R(T_{m,n})$ and $R(T_{m,n} \mid L_{root})$ recursively using $R(T_{m,n-1})$ and $R(T_{m,n-1} \mid L_{root})$.

Now, let $f_n = P(T_{m,n} \text{ is connected} \mid L_{root}) = [p\, f_{n-1} + q^{(m^n-1)/(m-1)}]^m$

$$\Rightarrow \qquad f_n^{1/m} = p\, f_{n-1} + q^{(m^n-1)/(m-1)}$$

Let $x_k = f_k^{1/m}$. So, $x_n = p\,(x_{n-1})^m + q^{(m^n-1)/(m-1)}$

Dividing both sides by $q^{(m^n-1)/(m-1)}$ and substituting $y_n = x_n / q^{(m^n-1)/(m-1)}$

$$y_n = (p/q)\,[y_{n-1}]^m + 1 \qquad \ldots (6.1)$$

Starting from the initial value of y_1, y_n can thus be very easily computed from eqn. (6.1).

For all practical values of p ($0.5 < p < 1$), $(p/q)\,[y_{n-1}]^m >> 1$ after a few steps of recursion. As a result, we may ignore the '+1' term beyond a certain value of n. As an example, the percentage errors in the computed reliability of a binary tree, of height 6 ($m = 2$, $n = 6$) for different values of p, are listed in Table 6.1. The 'cut-off n' in Table 6.1 is the value of n from which we neglect the '+1' term to compute y_n.

Table 6.1 : Percentage error in the reliability for different p and cut-off n

cut-off n \ p	0.5	0.6	0.7
2	9.39	3.22	0.32
3	0.22	9×10^{-3}	6×10^{-5}
4	2×10^{-4}	1×10^{-7}	0.00

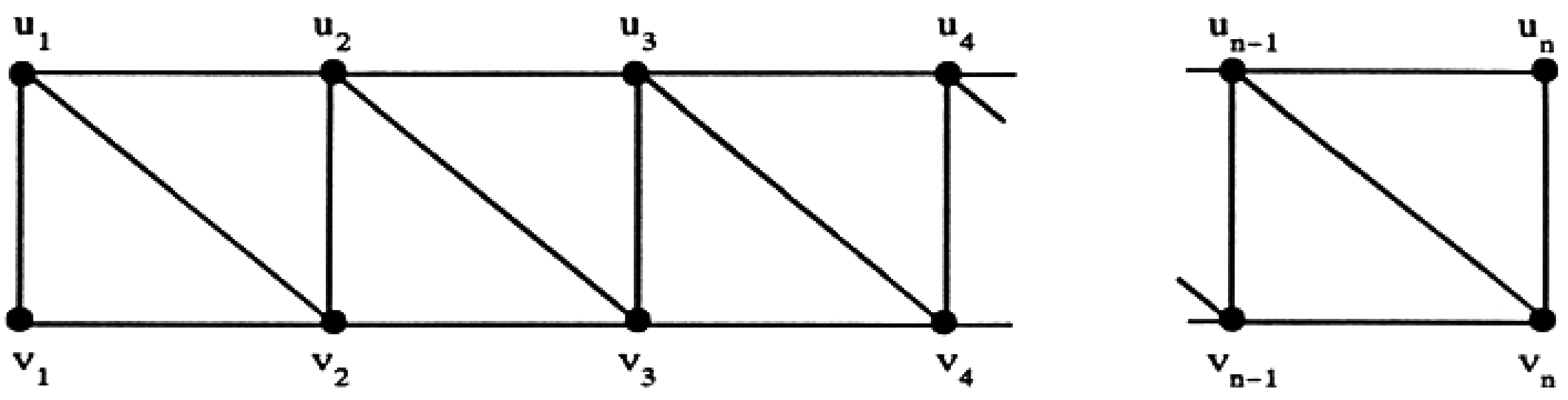


Figure 6.2 : Modified ladder network

6.5.2 Ladder (L_n)

We want to find the reliability of the modified ladder (Figure 6.2) graph of length n, i.e., 2n nodes.

Let R_n = Reliability of L_n.

Define l_m = Probability that L_m is connected and it has length m, where, by 'length m' we mean that if we breadthwise collapse the ladder, or whatever part of it is alive, it produces a continuous line of length m.

Then, $R_n = P(L_n \text{ is connected})$

$$= \sum_{i=1}^{n} P(L_n \text{ is connected and has length } i) + q^{2n}$$

$$= \sum_{i=1}^{n} (n+1-i)\, q^{2(n-i)}\, l_i + q^{2n} \qquad \text{... (6.2)}$$

So, to get R_n it is enough to compute l_i, $i \geq 1$.

Define s_m = Probability that L_m is connected, has length m and v_1 is alive.

So, $l_m = p\, l_{m-1} + pq\, s_{m-1}$, for $m \geq 2$ ... (6.3)

Also for $m \geq 2$, $s_m = p\,[p\, l_{m-1} + q\, s_{m-1}]$

$$= p^2 l_{m-1} + pq\,[\,p^2 l_{m-1} + pq\, s_{m-2}]$$

$$= p^2 \sum_{i=1}^{m-1} l_{m-i}\,(pq)^{i-1} + (pq)^{m-1} s_1$$

$$= p^2 \sum_{i=1}^{m-1} (pq)^{m-i-1}\, l_i + (pq)^{m-1}\, p \quad [\text{ As } s_1 = p\,] \qquad \ldots (6.4)$$

From equation (6.3) we get, $l_{m+1} = p\, l_m + pq\, s_m$

Putting the value of s_m from equation (6.4), we get,

$$l_{m+1} = p\, l_m + p^2 \sum_{i=1}^{m-1} (pq)^{m-i}\, l_i + p\,(pq)^m$$

We define, $l_0 = 1/p$

Then, $l_{m+1} = p^2 \sum_{i=0}^{m} (pq)^{m-i}\, l_i + pq\, l_m$

Dividing both sides by $(pq)^{m+1}$ and putting $l_k / (pq)^k = t_k$, we get,

$$t_{m+1} = w \sum_{i=0}^{m} t_i + t_m \qquad m \geq 2, \qquad \ldots (6.5)$$

where $w = p/q$.

So, $\quad t_m - t_{m-1} = (\, t_{m-1} - t_{m-2}\,) + w\, t_{m-1}$

$\Rightarrow \quad t_m = (2 + w)\, t_{m-1} - t_{m-2}\,, \quad$ for $m \geq 3$. $\qquad \ldots (6.6)$

Let $G(x)$ be the generating function for the t_i's. $G(x) = \sum_{i \geq 0} t_i\, x^i$

So, $\quad [1 - (2 + w)\, x + x^2]\, G(x)$

$= t_0 + [t_1 - (2{+}w)\, t_0]\, x + [t_2 - (2{+}w)\, t_1 + t_0]\, x^2 \qquad$ [By equation (6.6)]

So, $\quad G(x) = \{t_0 + [t_1 - (2 + w)\, t_0]\, x + [t_2 - (2 + w)\, t_1 + t_0]\, x^2\} \,/\, \{1 - (2 + w)\, x + x^2\}$

Now, puttiing the values of l_0, l_1 and l_2, we get,

$t_0 = 1/p$

$t_1 = 1 + 1/q$

$t_2 = (1 + 2q) / q^2$

So, $\quad G(x) = [C_0 + C_1\, x + C_2\, x^2] \,/\, [1 - (2{+}w)\, x + x^2]$

where $C_0 = t_0$

$C_1 = t_1 - (2{+}w)\, t_0$

and $C_2 = t_2 - (2{+}w)\, t_1 + t_0$

Now, $(2+w)^2 - 4 > 0$, as $w > 0$

So the roots of the denominator of $G(x)$ are real. Let the roots be r_1 and r_2.

Hence,

$$1 / [(x-r_1)(x-r_2)] = \sum_{i\geq 0} f_i x^i, \qquad \text{where } f_i = [r_1^{i+1} - r_2^{i+1}] / [(r_1 r_2)^{i+1} (r_1 - r_2)]$$

Therefore, $G(X) = [C_0 + C_1 x + C_2 x^2] [\sum_{i \geq 0} f_i x^i]$

But, $G(x) = \sum_{i\geq 0} t_i x^i$

So, for $i \geq 2$, $t_i = C_0 f_i + C_1 f_{i-1} + C_2 f_{i-2}$

6.5.3 End-Connected Strings

Here, we consider a network in the form of m strings of length k+2 each, where the extreme points of all the strings are common and no other node is common between two strings. Let us call this structure G^*.

Assume that all the nodes have identical failure-rate $q = 1 - p$.

Let the two end points be u and v.

There are three possible cases, depending on the nodes u and v.

Case I : Both u and v alive.
Given that both the nodes are alive, G^* can be connected if and only if at least one of the strings is completely alive and for all incomplete strings they have two connected components (which, may be null), connected with u and v.

So, the probability of G^* being connected in this case is,

$$\Pi_1 = \sum_{i=1}^{m} \binom{m}{i} p^{ik} f^{m-i}$$

$$= [p^k + f]^m - f^m$$

where f = Probability that a string of length k is incomplete and its components are starting from one of the extreme nodes.

$$= \sum_{j=1}^{k} (k+1-j)\, p^{k-j} q^{j}$$

Case II : Only one of u and v is alive.

Given that only one of u and v is alive, G^* may be connected, only if all the strings are connected from the specified live node.

So, the probability of G^* being connected in this case is

$$\Pi_2 = [\sum_{i=0}^{k} p^i q^{k-i}]^m$$

Case III : Both u and v are dead.

Given that both u and v are dead, the network may remain connected if either all but one of the strings are totally dead and the last one has exactly one connected component or all of the m strings are totally dead.

So, the probability of G^* being connected in this case is

$$\Pi_3 = m\, q^{(m-1)k} \sum_{j=1}^{k} (k+1-j)\, p^j q^{k-j} + q^{mk}$$

So the probability of G^* being connected is,

$$R(G^*) = p^2 . \Pi_1 + 2pq . \Pi_2 + q^2 . \Pi_3$$

This structure may be generalized by considering links which are not strings (say, by introducing a bypass in the string). We may also consider cases, when all the paths between u and v may not be of the same type. This way we may find reliabilities of some structures like tree-machine, Binomial graph [RRK83], etc.

6.6 Conclusion

In the definition of the term ‘operational’ we considered only the subgraphs which are connected. Another approach may be to assign certain weights to all subgraphs, depending on some measure of local connectedness, rather than considering global connectedness. These weights may be assigned depending on the number of components or the percentage of possible communications achieved, etc. But such a measure of reliability would be quite difficult to compute.

Chapter 7
Self-Routable Permutations in Benes Network

7.1 Introduction

Multistage Interconnection Networks (MIN's) are used in multiprocessor systems to interconnect the different modules (PE's and/or memory modules) dynamically. An N×N Benes Network is a well-known rearrangeable MIN that can connect its N inputs to its N outputs in all possible ways. The recursive structure of an N×N Benes network B(n) is shown in figure 7.1, ($n = \log_2 N$). The best known routing algorithm for an arbitrary permutation in a Benes Network is of time-complexity $O(Nn)$ on a uniprocessor system [Wa68], compared to its propagation delay $O(n)$ only. Even with parallel algorithms for switch set-up, the time needed to realize an arbitrary permutation in a Benes network is dominated by the set-up time [NS82].

However, many useful permutations, often required in parallel processing environments are *self-routable* [Le78], [NS81a], [Na89], [BR88] in a Benes network, in the sense that as the vector of data passes through the network, the two-state switches are controlled on the fly by the destination tags of the inputs to the respective switches. Lenfant proposed a simple routing algorithm for some *Frequently Used Bijections* [Le78], namely the *FUB* family (|FUB| is $O(2^{2n})$). Nassimi and

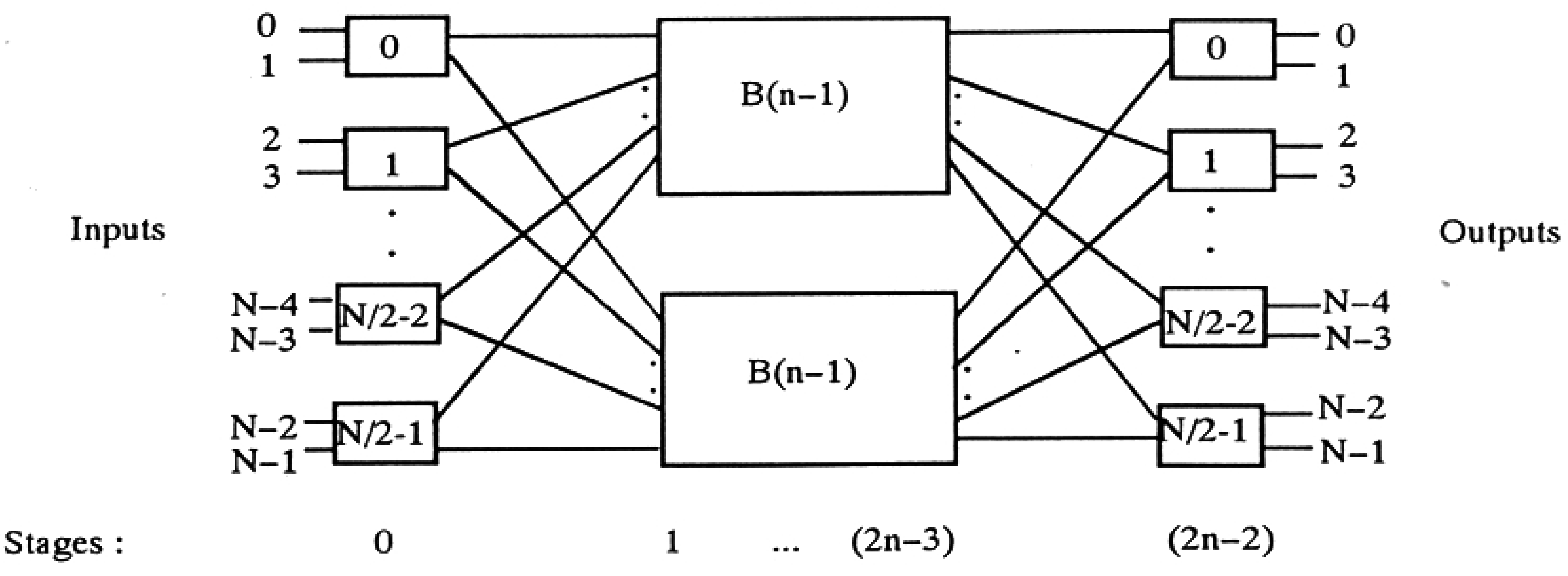


Figure 7.1 : An N×N Benes Network B(n), $n = \log_2 N$

Sahni [NS81a] proposed an O(n) algorithm that routes the BPC (*Bit-Permute Complement*) class of permutations ($|\text{BPC}| = n!\,2^n$) and IΩ (*Inverse Omega*) class of permutations ($|I\Omega| = 2^{n \cdot N/2}$). Nassimi [Na89] also developed a fault-tolerant routing scheme for BPC class of permutations. Boppana and Raghavendra [BR88] solved the problem for LC (*Linear-Complement*) class of permutations ($|\text{LC}|$ is $O\,(2^{n^2})$), which also routes the IΩ class of permutations as well.

In these earlier works, the authors started from some known classes of permutations and developed suitable self-routing strategies for each. In effect, we just know about some classes of permutations, self-routable by some known technique. But the applicability of the different self-routing techniques developed so far is yet to be fully explored. In fact, there exists many more permutations that are routable by the existing self routing algorithms; but these characterizations are not yet complete. Again, to route any arbitrary permutation P by any of these routing techniques, firstly we are to recognize whether or not P belongs to the permutation class routable by it (*identification problem*). As a result, the overall routing complexity may increase. Moreover, these algorithms require some extra hardware facilities in the switches of Benes network to decide which of the two inputs to a switch should be used for setting it.

In this chapter, we tackle the problem in a more realistic way. We classify the self-routable permutations into four categories, namely :

i) *Top-Control Routable set of permutations (TCR),*
ii) *Bottom-Control Routable set of permutations (BCR),*
iii) *Least-Control Routable set of permutations (LCR)*
and iv) *Highest-Control Routable set of permutations (HCR).*

We will show that each of these classes contain at least $2^n(2^{N/2} + n! - 1)$ permutations which is much more than the size of any one of the classes FUB, BPC, IΩ or LC. Thus our results will produce a better characterization of the permutations that are passable by the different self-routing algorithms. However, our characterization is also far from complete. Each of the above classes actually contains many more permutations. In fact, this lower bound of $2^n\,(2^{N/2} + n! - 1)$ is the size of the intersection of all classes (i.e., TCR, BCR, LCR and HCR) considered here.

We also develop an algorithm that will detect whether any N×N permutation P belongs to any of the four classes or not. If it is routable by any of the four self-routing algorithms, this algorithm also determines at the same time the controls necessary for the routing of P. That is, the identification problem and switch-setting, both are done by this algorithm. This algorithm can be implemented on a multiprocessor system with a time complexity $O(n)$. The routing technique does not require any additional hardware facility in the switching elements of the Benes network, as in the cases of the other self-routing techniques mentioned earlier. Even in the presence of some faults in the network, our algorithm can easily be modified to generate the necessary controls for routing P through the faulty network.

The idea of *Group Transformations* on permutations developed earlier [DBD90], finds some interesting applications on the self-routable sets of permutations. It helps us to find a one-to-one correspondence between the permutations in the set TCR (LCR) with those in BCR (HCR). Moreover, it leads us to develop a new self-routing algorithm for all the permutations of the BPCL class [DBD90], which is the set of permutations generated from BP (bit-permute) permutations, by the application of all possible group transformations. The BPCL class includes many important classes of permutations; the BPC class and the LC class are subsets of the BPCL class. An algorithm is already presented in [DBD90], to test whether any given permutation P belongs to BPCL or not. We have utilized the outputs of that algorithm to route the permutations in the BPCL class with a time complexity $O(n)$ on a multiprocessor system.

7.2 Classification of Self-Routable Permutations in a Benes Network

In n-stage $2^n \times 2^n$ blocking MIN's, like omega, baseline, inverse-omega, reverse baseline etc., a unique path exists between any input-output pair [WF80]. For a switch at any stage i, $0 \leq i \leq n-1$, let $x_{n-1} \cdots x_1 x_0$ be one of the destination tags. Then the switch is set by the bit x_j, where j depends only on i. Therefore it is evident that a permutation P is passable by the network, if and only if, for each switch at any stage i, the x_j bits of the destination tags attached to its two inputs are complement of each other. For reverse baseline network, $j = i$.

However, a Benes network consists of a baseline network followed by a reverse-baseline network with the last stage of baseline and the first stage of reverse baseline being merged together. The routing through the reverse-baseline section, i.e., the last n-stages of the Benes network, must follow the normal destination tag routing scheme. We are to determine the routing strategy for the first $(n-1)$ stages only, so that ultimately it becomes passable through the network.

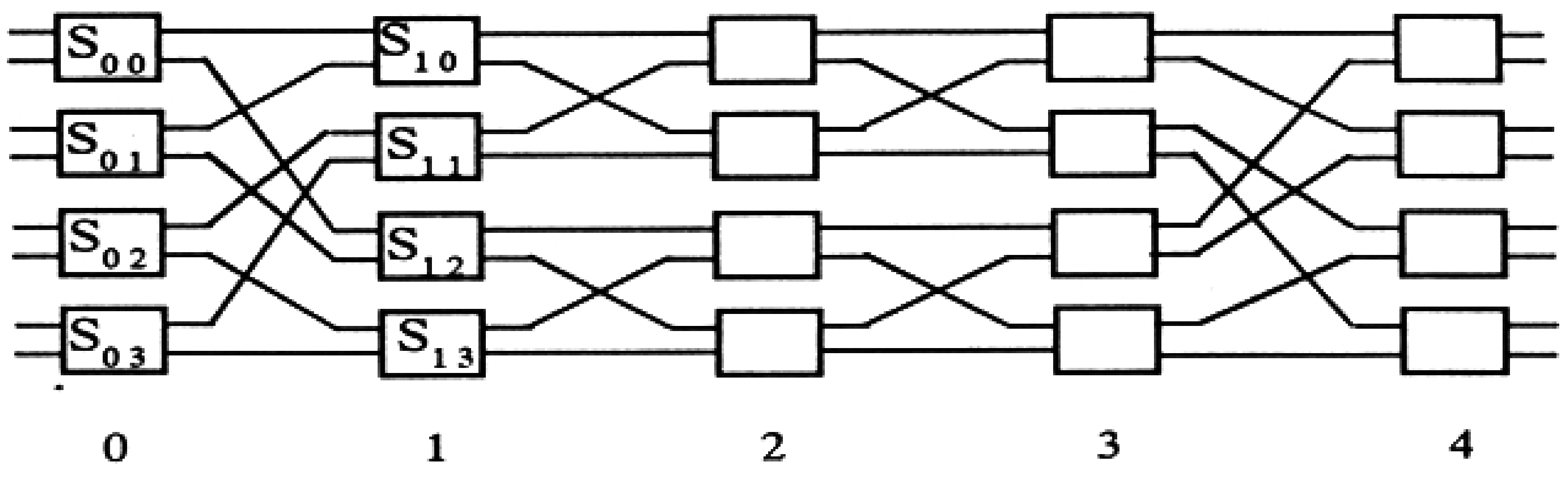


Figure 7.2 : An 8×8 Benes Network and the Numbering of the Switches for Stages i < n−1

We will assume that a switch at stage i is denoted by S_{ij}, where $0 \leq i \leq n-2$ and $0 \leq j < N/2$ and the destination tags attached to the two inputs of a particular switch S_{ij} are identified as T_{ij} (the top one) and B_{ij} (the bottom one) respectively. For an 8×8 Benes network, the numbering of the switches are shown in figure 7.2.

In self-routing algorithms, the setting of a switch is done locally using the destination tags of its two inputs. We shall concentrate on algorithms in which one of the two destination tags is selected using a global property (say, that of the top input or the lesser of the two destination values etc.). This input will be called the *R-input.* One particular bit of the R-input is chosen, depending on the stage in which the switch lies and that bit is used for setting the switch. This particalar bit will be referred to as the *R-bit.*

For the classes of self-routable permutations, discussed here, whatever be the R-input at any stage i, $0 < i < n-2$, *the R-bit is the bit* x_i *of the destination tag* $x_{n-1} \ldots x_i \ldots x_1 x_0$ *of the R-input.*

Definition : In an N×N Benes network the self-routing algorithms considered here are of the following types :

i) *TCR* (*Top Control Routing*) algorithm, if the R-input is T_{ij}
ii) *BCR* (*Bottom Control Routing*) algorithm, if the R-input is B_{ij}
iii) *LCR* (*Least Control Routing*) algorithm, if the R-input is $\min[T_{ij}, B_{ij}]$
and iv) *HCR* (*Highest Control Routing*) algorithm, if the R-input is $\max[T_{ij}, B_{ij}]$.

Figures 7.3 - 7.6 show examples of these routing schemes for different permutations. [We will represent an N×N permutation by the sequence of outputs corresponding to the sequence of inputs (0 1 2 ... N−1).]

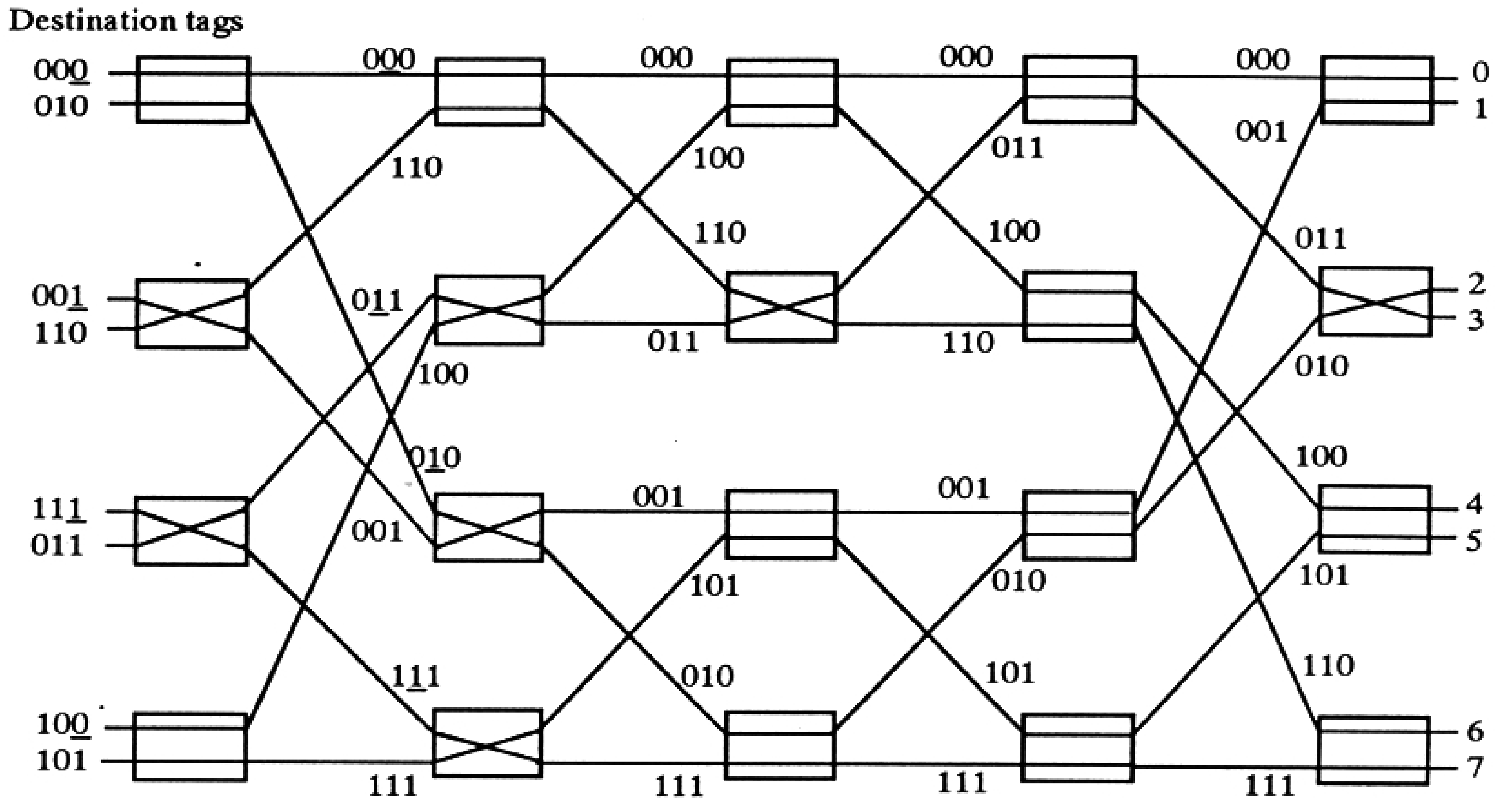


Figure 7.3 : Routing of Permutation P : (0 2 1 6 7 3 4 5) by Top-Control-Routing (At Any Stage-i, 0 ≤ i < 2, the Underlined Input Bit Determines the Switchsetting)

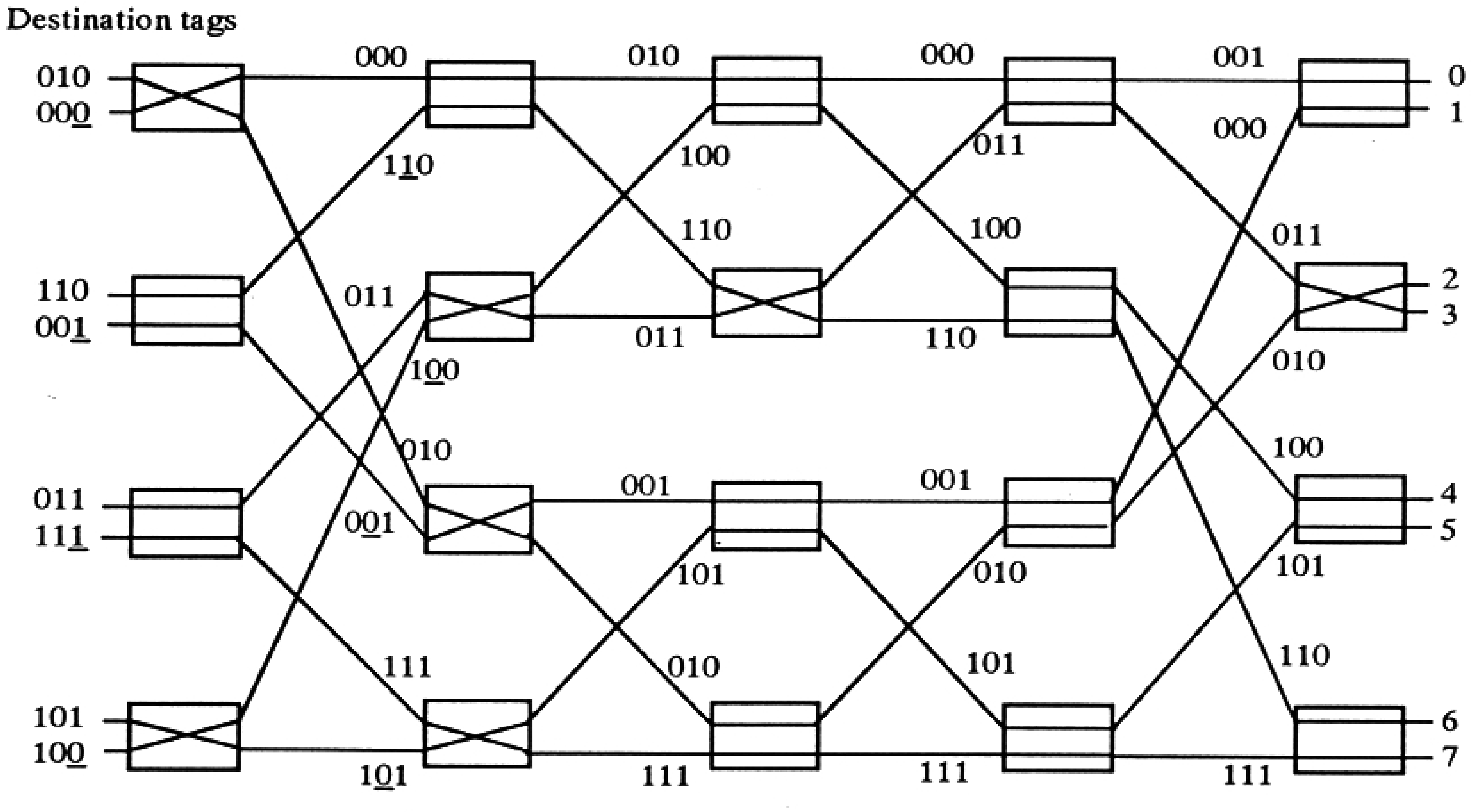


Figure 7.4 : Routing of Permutation P : (2 0 6 1 3 7 5 4) by Bottom-Control-Routing (At Any Stage-i, 0 ≤ i < 2, the Underlined Input Bit Determines the Switchsetting)

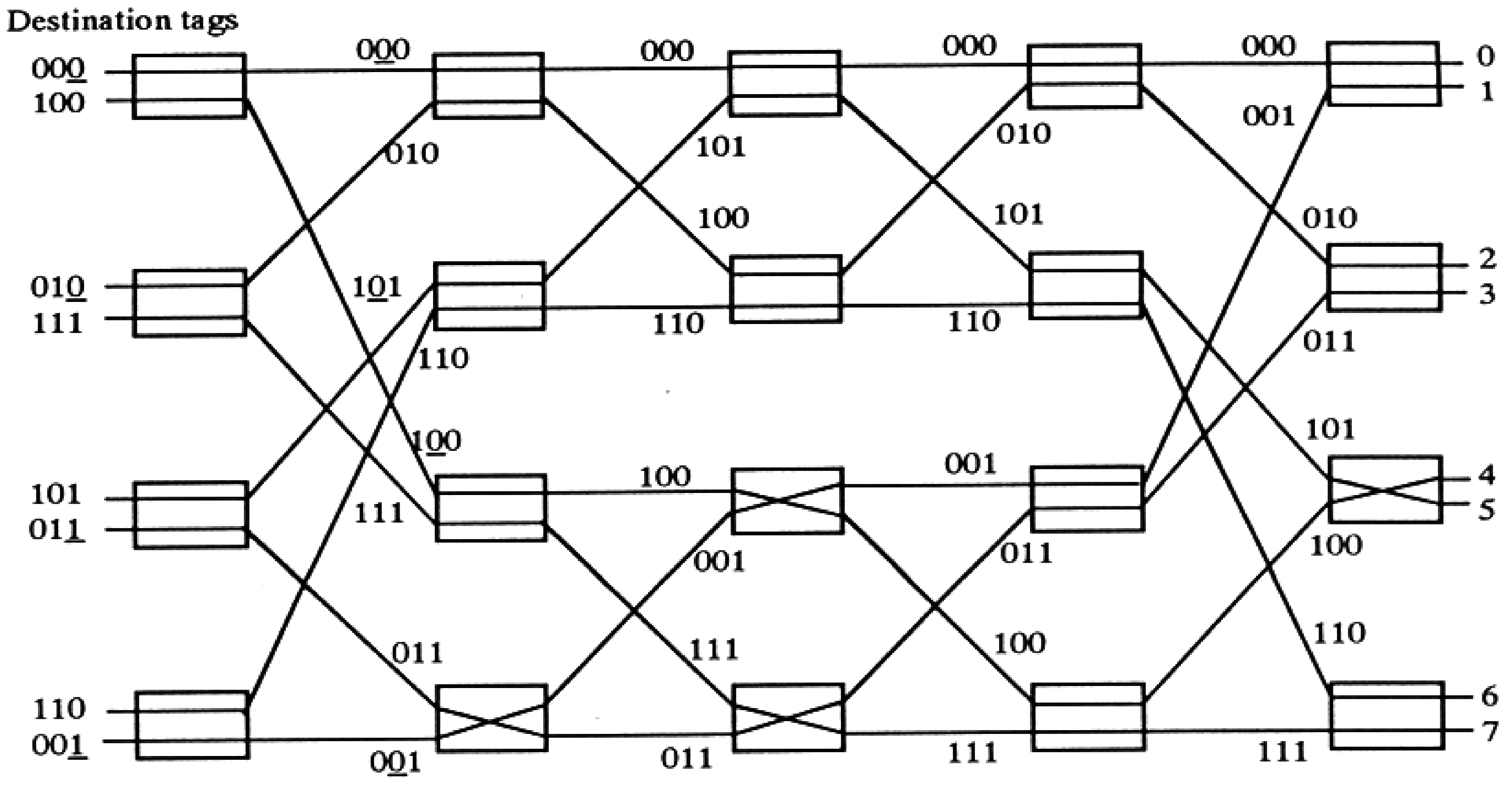


Figure 7.5 : Routing of Permutation P : (0 4 2 7 5 3 6 1) by Least-Control-Routing (At Any Stage-i, 0 ≤ i < 2, the Underlined Input Bit Determines the Switchsetting)

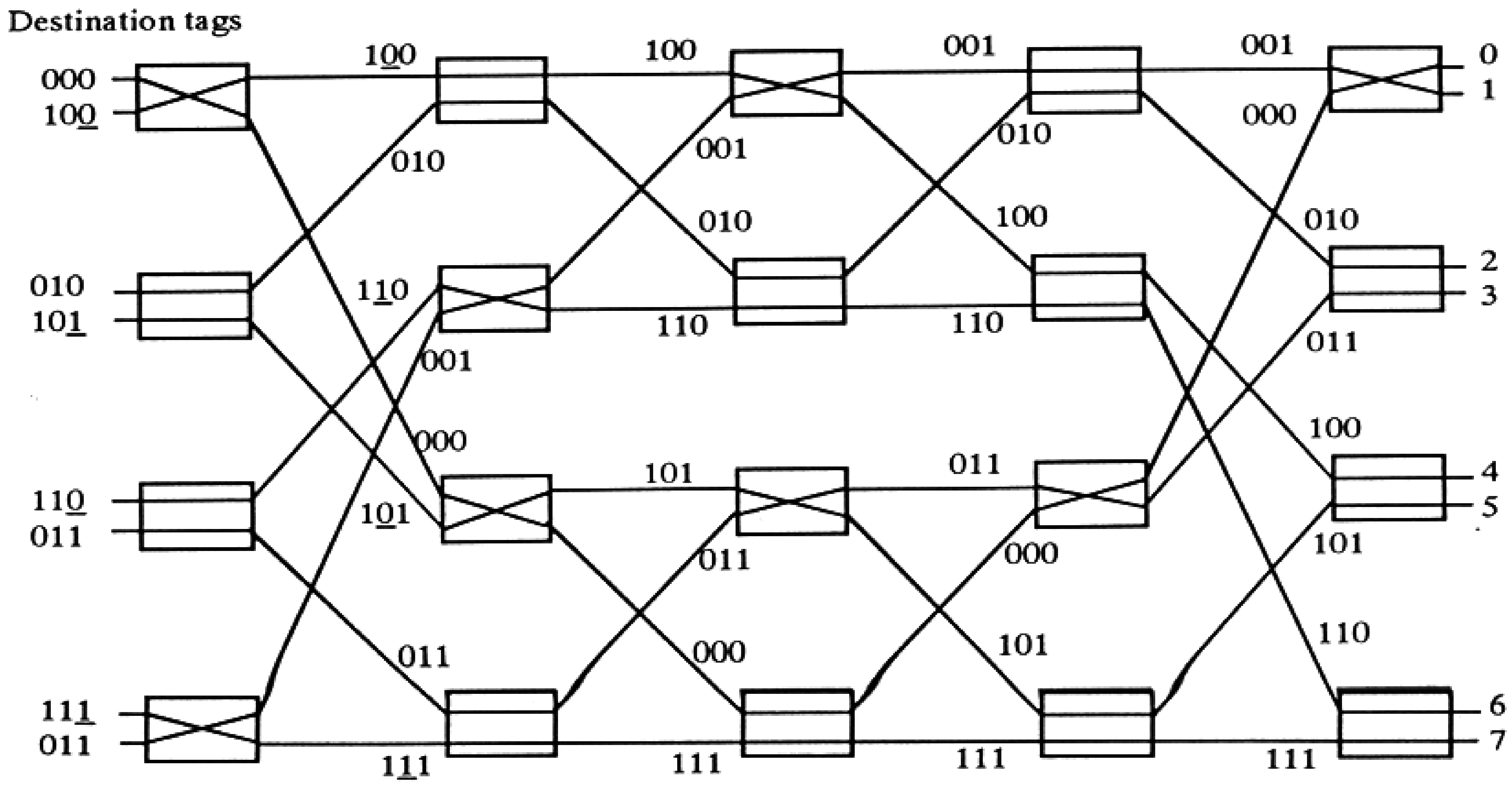


Figure 7.6 : Routing of Permutation P : (0 4 2 5 6 3 7 1) by Highest-Control-Routing (At Any Stage-i, 0 ≤ i < 2, the Underlined Input Bit Determines the Switchsetting)

Remark : For each of the permutations P shown in examples 7.3 - 7.6, it can be verified that no self-rouitng strategy, other than that chosen, from the above four types can successfully route P. In other words, none of the classes TCR, BCR, LCR or HCR is contained in any of the other ones.

Definition : A permutation will be called *Top / Bottom / Least / Highest Control Routable* if it can be routed on an N×N Benes network using the Top / Bottom / Least / Highest Control Routing scheme.

Among the four classes of permutations mentioned above, the classes TCR and LCR have already been introduced earlier [NS81a], [BR88]. Nassimi and Sahni [NS81a] have shown that a rich class of permutations, namely the F class, that includes the BPC class, IΩ class and also the five classes of permutations considered by Lenfant [Le78], is top control routable, i.e., TCR $\supseteq$ F. Boppana and Raghavendra studied the LC (Linear-Complement) and IΩ class of permutations, which are least control routable [BR88]. The four simplest self-routing strategies as defined above, extend the set of self-routable permutations beyond these known classes. Here follows some interesting properties of the classes which finally lead us to develop the general algorithm for routing a given permutation P, if P belongs to any of the self-routable classes, mentioned above.

Definition : In an N×N Benes network, the set of *Free-choice Self-Routable (FSR)* permutations is defined as the intersection of TCR, BCR, LCR and HCR.

Example 7.1 : The permutation P = (0 1 4 5 3 2 6 7) is routable in the 8×8 Benes network by any one of the proposed self-routing algorithms, namely the top-control, bottom-control, least-control and highest-control. Therefore, P $\in$ FSR.

Definition : In an N×N Benes network, the set of *R-invariant Self-Routable (RSR)* permutations is defined as the subset of FSR, which leads to the switch settings that are invariant with the nature of the R-inputs (i.e., top / bottom / least / highest). In other words, the R-bits of the two inputs of any switch will be complementary to each other for a permutation belonging to RSR.

Example 7.2 : The permutation P = (2 5 7 4 6 1 0 3) is R-invariant self-routable in a Benes network, i.e., P $\in$ RSR.

Theorem 7.1 : For an N×N Benes network, the cardinality of RSR is $2^{Nn/2}$.

Proof: Let RSR_k = the set of R-invariant self-routable permutations in a $2^k \times 2^k$ Benes network. Let $P = (p_0\ p_1 \cdots p_{N/2-1})$ and $Q = (q_0\ q_1 \cdots q_{N/2-1})$ be two elements of RSR_{n-1}, where $N = 2^n$. Take a permutation $\pi = (2p_0\ \ 2q_0{+}1\ \ 2p_1\ \ 2q_1{+}1\ \ \ldots\ \ 2p_{N/2-1}\ \ 2q_{N/2-1}{+}1)$ of N elements. Refering to Fig. 7.1, we can see that if we route π in an N×N Benes network by R-invariant self-routing scheme, the required permutations (after dropping the LSB) to the top and bottom half of the stage 1 would be P and Q respectively. Since $P, Q \in RSR_{n-1}$, they are R-invariant self-routable through the top and bottom $N/2 \times N/2$ Benes networks respectively. At the last stage, the two inputs with the same destination tag (with respect to B_{n-1}), will combine at a switch, as shown in Fig. 7.1. Append a '0' ('1') at the LSB position of the destination tag of the line coming from the top half (bottom half). They are again R-invarinat self-routable in the last stage.

Now if we interchange 2i and (2i+1)-th positions of π, $0 \le i \le N/2-1$, it still remains RSR. For any two given $P, Q \in RSR_{n-1}$, we get $2^{N/2}$ such permutations in RSR_n. Let us take two different pairs (P, Q) and (P′, Q′), where P, Q, P′, $Q' \in RSR_{n-1}$. Say, P and P′ differ at the i-th position. Let the elements at the i-th position of P and P′ be x and y, respectively. Then, in the permutations π of size N generated by (P, Q), the positions 2i and 2i+1 will contain 2x and an odd number. Similarly, the positions 2i and 2i+1 of π', generated by (P′, Q′), will contain 2y and an odd number. Thus π and π' will always be different. Hence for each distinct pair of elements in RSR_{n-1}, we can generate $2^{N/2}$ distinct permutations of size N.

Let $\pi = (r_0, r_1, \ldots, r_{N-1})$ be a permutation in RSR_n. Since π is in RSR, for every i, $0 \le i \le N/2$, one of r_{2i} and r_{2i+1} is even and the other is odd. Let the even value be x_i and the odd value be y_i. Take $p_i = x_i/2$ and $q_i = (y_i-1)/2$. Then, $P = (p_0\ p_1 \cdots p_{N/2-1})$ and $Q = (q_0\ q_1 \cdots q_{N/2-1})$ must be two permutations in RSR_{n-1}. Now, π can be constructed from P and Q, by our technique. That is, our method of constructing permutations in RSR_n from those in RSR_{n-1} covers the whole set RSR_n.

So,
$$\begin{aligned}
|RSR_n| &= 2^{N/2}\,(|RSR_{n-1}|)^2 \\
&= 2^{N/2}\,2^{N/2}\,(|RSR_{n-2}|)^4 \\
&= \ldots \\
&= 2^{N/2\cdot(n-1)}\,(|RSR_1|)^{N/2} \\
&= 2^{N/2\cdot(n-1)}\,.\,2^{N/2} \\
&= 2^{N/2\cdot n}.
\end{aligned}$$

Hence, $|P_n| = 2^{Nn/2}$. ◆

Theorem 7.2 : In an N×N Benes network, the set of R-invarinat self-routable permutations is actually the IΩ set of permutations.

Proof : In an inverse-omega network, for any permutation P let the destination tags of two inputs incident on a switch at stage 0 be $x_{n-1}\, x_{n-2} \cdots x_1\, x_0$ and $y_{n-1}\, y_{n-2} \cdots y_1\, y_0$. For P to be realizable in a single pass, x_0 and y_0 must be distinct. After routing through stage 0, the inputs incident on a switch at stage 1 should differ in the second least significant bit. In general, for P to be realized in a single pass, at any stage i, $0 \leq i \leq n-1$, the destination tags of the lines incident on a switch at that stage must differ in the i-th least significant bit. This condition is identical to that for any R-invariant self-routable permutation in a Benes network. Therefore, it is obvious that the set of RSR permutations in a Benes network is identical to the set of permutations realizable by an inverse-omega network in a single pass, i.e., the IΩ set of permutations. ◆

Corollary 7.1 : In an N×N Benes network, $|FSR| \geq 2^{Nn/2}$.

Proof : By definition, FSR $\supseteq$ RSR. Hence the corollary follows immediately from theorem 7.1. ◆

Theorem 7.3 : The BPC class of permutations is contained in FSR.

Proof : It has already been proved that BPC $\subseteq$ TCR [NS81]. Now let us prove that BPC $\subseteq$ BCR.

We shall prove the result by induction on n. The result can easily be verified for n = 1. Let the result be true for n−1.

Let us consider a permutation P $\in$ BPC, described by the bit-permute complement rule $P : (x_{n-1} \cdots x_0) \rightarrow (y_{n-1} \cdots y_0)$, where $(y_{n-1} \cdots y_0)$ is a permutation of $(x_{n-1} \cdots x_0)$, followed possibly by complementation of some bits. Since $(y_{n-1} \cdots y_0)$ without the complementations is a permutation of $(x_{n-1} \cdots x_0)$, there exists k, $0 \leq k \leq n-1$, such that $y_k = x_0$ or $\overline{x_0}$. Construct P′ from P such that $P' : (x_{n-1} \cdots x_0) \rightarrow (y_{n-1} \cdots \overline{y_k} \cdots y_0)$.

Since P′ $\in$ BPC, we have P′ $\in$ TCR. From the mapping rule of P and P′ it can be shown that if $P = (p_0\, p_1\, p_2\, p_3 \cdots p_{n-2}\, p_{n-1})$, then $P' = (p_1\, p_0\, p_3\, p_2 \cdots p_{n-1}\, p_{n-2})$. In other words, at the input of switches at stage 0, each switch will have the same pair of inputs (destination tags) for P and P′, but their order (top or bottom) will be reversed. Hence, if we route P according to BCR and P′ according to TCR, we will have the same destination tags at the input of stage 1. It has been shown [NS81] that, when P' is routed by the top inputs, the two halves of inputs at stage 1 will again be two BPC's ignoring the LSB. Therefore, when P is routed by bottom inputs, stage 1 will similarly have two BPC's at the two halves of the inputs. Therefore, by induction, P is Bottom Control Routable.

It has already been shown that LCR $\supseteq$ BPC [BR88]. In order to show that BPC is also in HCR, we observe that for a BPC permutation, the larger of the two destination values at the inputs of every switch at stage 0 will consistently be always at the top or always at the bottom input. Hence, so far as stage 0 is concerned, routing a BPC by HCR amounts to routing it by either TCR or BCR. Again, routing a BPC by top or bottom input in stage 0 generates two BPC's at stage 1. Hence, it can be shown by induction that, HCR $\supseteq$ BPC. ◆

Lemma 7.1 : A BPC permutation P, generated from a BP permutation P*, belongs to the IΩ set of permutations, if and only if P* $\in$ IΩ.

Proof: For a permutation to be realizable in a single pass in an inverse-omega network, two destination tags at the input of a switch at stage 0 should differ in the LSB. The destination tags at the input of a switch at stage 1 should differ in at least one of the last two bits and so on. For example, in an 8×8 inverse-omega network, inputs to a switch at stage 0 should have one even and one odd destination tag. Similarly, the destination tags of the inputs to any switch at stage 1 (i.e., their input numbers matching at their MSB only) should differ in at least one of the the last two bits. We can generalize this observation as follows. In other words, if the binary representations of two input numbers are identical in the two most significant bits (i.e., they are incident on the same switch at stage 0), then their destination tags must be such that they differ in the LSB.

In an inverse-omega network, let us consider a source-destination path S $\rightarrow$ D. Let the binary representation of S and D be $s_{n-1} \dots s_k \dots s_0$ and $d_{n-1} \dots d_k \dots d_0$ respectively. For every k, $0 \leq k \leq n-1$, construct a string $s_{n-1} \dots s_{k+1} d_k d_{k-1} \dots d_0$. Two paths conflict at stage k, $0 \leq k \leq n-1$ (i.e., they share an output link of some switch at stage k), if and only if, the two binary strings $s_{n-1} \dots s_{k+1} d_k d_{k-1} \dots d_0$ are same for the two paths. Therefore, it is evident that if a BP-permutation P* $\in$ IΩ, i.e., all input-output paths are passable by in inverse-omega network, any BPC-permutation P, generated from P*, by complementing some definite bits of each output, will also be conflict-free in that network. Similarly, if a BPC permutation P $\in$ IΩ, by the same logic, the corresponding BP permutation P* will also be contained in IΩ. ◆

Remark : It has already been shown that $BP \cap I\Omega$ = identity permutation [NS81].

Corollary 7.2 : $|FSR| \geq 2^n (2^{N/2} + n! - 1)$.

Proof : From definition, FSR contains RSR. So, from theorem 7.2 and theorem 7.3 it follows that FSR contains $BPC \cup I\Omega$. Now, $| BP \cap I\Omega | = 1$ (the identity permutation). Therefore, by lemma 7.1, $| BPC \cap I\Omega | = 2^n$. Since $| BPC | = 2^n n!$ and $| I\Omega | = 2^{n.N/2}$, hence $| BPC \cup I\Omega | = 2^n (2^{N/2} + n! - 1)$. ◆

Remark : By definition, FSR is the intersection of four classes of self-routable permutations. Therefore, corollary 7.2 gives a lower bound of the size of each class.

We now state some general characteristics for any permutation $P \in XCR$, (X = T/ B/L/H).

The recursive structure of a Benes network B(n), is shown in figure 7.1. It is evident that at any stage i, there exists 2^i disjoint sets of switching elements (each set forms the 0-th stage of a B(n−i). The upper (lower) outputs of the switching elements belonging to the same set are the inputs to the upper (lower) B(n−i−1) at the next stage.

Definition : In a Benes network B(n), for any permutation $P \in XCR$ (X = T/B/L/ H), at any stage i, $1 \leq i < n-1$, destination numbers $a = a_{n-1} \cdots a_0$ and $b = b_{n-1} \cdots b_0$ of two inputs of a B(n−i), are said to form a *conjugate pair*, if $a_j = b_j$, for all j's $n-1 \geq j > i$.

Remark : In a B(n), for any permutation $P \in XCR$, (X = T/B/L/H), the routing strategy ensures that at any stage i, $0 \leq i < n$, there exist exactly two destination numbers which form a conjugate pair at the input of a B(n−i).

Theorem 7.4 : In a B(n), for any permutation $P \in XCR$ (X = T / B / L / H), at each stage i, $0 \leq i \leq n-1$, the two elements a and b of any conjugate pair will satisfy any one of the following three conditions :

i) both are the R-inputs

ii) one element (say a) is the R-input but the other (b) is not; say x is the R-input of the switch corresponding to b; then R-bit(a) = R-bit(x), where R-bit(m) is the R-bit of the R-input m.

iii) none of a and b is the R-input; say x and y are the R-inputs of the switches where other inputs are a and b respectively; then R-bit(x) = R-bit(y).

Proof : Whatever be the self-routing strategy, to make P passable through the network, we are to route the two elements of each conjugate pair of any B(n−i), at stage i, $0 \leq i < n-1$, to two different B(n−i−1)'s at the next stage. It will be possible, if and only if any one of the three conditions stated in theorem 7.4 is satisfied. ◆

7.3 Algorithm for Class Identification and Routing

Given any N×N permutation P, our algorithm will check if P is routable by any of the self-routing techniques TCR, BCR, LCR or HCR as defined in section 7.2. If there is a success, i.e., P is routable by one of the techniques, the algorithm will also generate the corresponding controls for switch setting.

We assume a multi-processor system with N/2 processing elements (PE) numbered as PE-0, PE-1, ... Each PE-j, $0 \leq j < N/2$, will have two registers I_{2j} and I_{2j+1} containing the destination tags for the two inputs of the switch S_{ij}. The control-bit for the switch S_{ij}, $0 \leq i \leq n-2$, will be computed at the i-th step of execution of the algorithm and will be stored in a single-bit register C_{ji} in PE-j. Before starting the execution of the i-th step, the registers I_{2j} and I_{2j+1} of PE-j will be loaded by the destination tags of the two inputs of the switch S_{ij}. Suppose the input lines of the switches at stage (i+1) are numbered 0, 1, 2, ... N − 1 sequentially from top to bottom. Hence the output of the switch S_{ij} will go to the two input lines j′ and $j'+2^{n-i-1}$ at stage (i+1) where $j' = (2j \text{ DIV } 2^{n-i-1})\, 2^{n-i-1} + (2j \text{ MOD } 2^{n-i-1}) \text{ DIV } 2$.

In fact j′ (or, $j' + 2^{n-i-1}$) is obtained from 2j (or, 2j+1), by circular right shift over the least significant n−i+1 bits, i.e., if $k = x_n x_{n-1} \ldots x_2 x_1$ then $k' = x_n x_{n-1} \ldots x_{n-i+2}\, x_1\, x_{n-i+1} \ldots x_2$, where k = 2j (or, 2j+1) and $k' = j'$ (or, $j' + 2^{n-i-1}$). We will represent this operation on 2j by a function Φ, i.e., $\Phi(2j) = j'$. This function Φ, in fact is the mathematical representation of the interconnection pattern between stage i and stage (i+1) of the Benes network for $0 \leq i < n-1$.

At each step i, $0 \leq i < n-1$, each PE performs at most one comparison, one exchange and two data transfers (which can be done in parallel). Considering all these steps as a unit computation, the algorithm would have to perform (n−1) computations. The results from all PE's should finally be accumulated into a master processor to decide if there is a success in routing. If the routing is successful, then all the PE's will feed the necessary control vectors to the respective switching elements. Therefore the total complexity of the algorithm is $O(n)$ only. In case of failure, other self-routing startegies may be tried. Moreover, the algorithm may be modified a little to accommodate the trials for all classes of self-routable permutations sequentially one after another until it achieves a success, or fails in all the cases when we are to apply the general looping algorithm [Wa68] for routing. The algorithm for self-routing, algorithm SR, is now described below.

Algorithm SR

Assume M is the specified self-routing strategy (TCR, BCR, LCR or HCR) and x(i) denotes the i-th bit of the variable x. $Get\_Bit_i(x)$ is a function which returns the the value of the i-th bit of x.

```
begin
for any processor j, $0 \leq j \leq N/2-1$ do
begin
  Input (M);
  for i := 0 to n-2 do
  begin
     if M=TCR then $C_{ji} := Get\_Bit_i(I_{2j})$
     else if M=BCR then $C_{ji} := Get\_Bit_i(I_{2j+1})$
          else begin
                    if M=LCR then k := least $\{I_{2j}, I_{2j+1}\}$ else k := highest $\{I_{2j}, I_{2j+1}\}$;
                    if $k=I_{2j}$ then $C_{ji} := Get\_Bit_i(I_{2j})$ else $C_{ji} := Get\_Bit_i(I_{2j+1})$;
               end;
     if $C_i = 1$ then exchange($I_{2j}, I_{2j+1}$);
     $j' := \Phi(2j)$ ; $j'' := \Phi(2j+1)$ ; $I_{j'} := I_{2j}$ ; $I_{j''} := I_{2j+1}$ ;
  end;
  if $Get\_Bit_n(I_{2j}) = Get\_Bit_n(I_{2j+1})$ then success := 1 and terminate;
end;
end.
```

This algorithm can easily be modified to incorporate some more features to tolerate certain classes of faults. We will consider the fault model discussed in [Na89]. We would assume that the links are fault-free and a switching element may have only control stuck faults. Further, at each stage i, $0 \leq i < n-1$, there can be a maximum of 2^i faults, with at most one in every B(n–i), i.e., at stage 0, there can be at most one fault; at stage 1, there can be at most two faults, one in each B(n–1), and so on upto stage (n–2). In each processor PE-j, $0 \leq j < N/2$, the information about these faults in S_{ij} will be stored. In case the control-bit, as computed during the i-th iteration of the loop in algorithm SR, is in conflict with the state of the faulty switch, the computed control bits for each switch in the same B(n–i) are complemented. Since at stage i, there will be at most one fault in each B(n–i), we will get new control bits that will successfully route P through the faulty network.

Example 7.3 : Let us consider the permutation P = (0 4 2 5 6 3 7 1). Its highest-control routing is shown in figure 7.6. Now, let the switch S_{10} be stuck at the exchange mode and S_{13} at the straight mode. With these faults, the control for switch S_{10} should be reversed. By our technique that will reverse the control for the switch S_{11} as well, since both belong to the same B(n–1). However, a stuck-at-straight fault in S_{13} will not affect the routing, since the control generated by algorithm SR for this switch also corresponds to the straight mode. The routing of P, in this faulty situation is shown in figure 7.7.

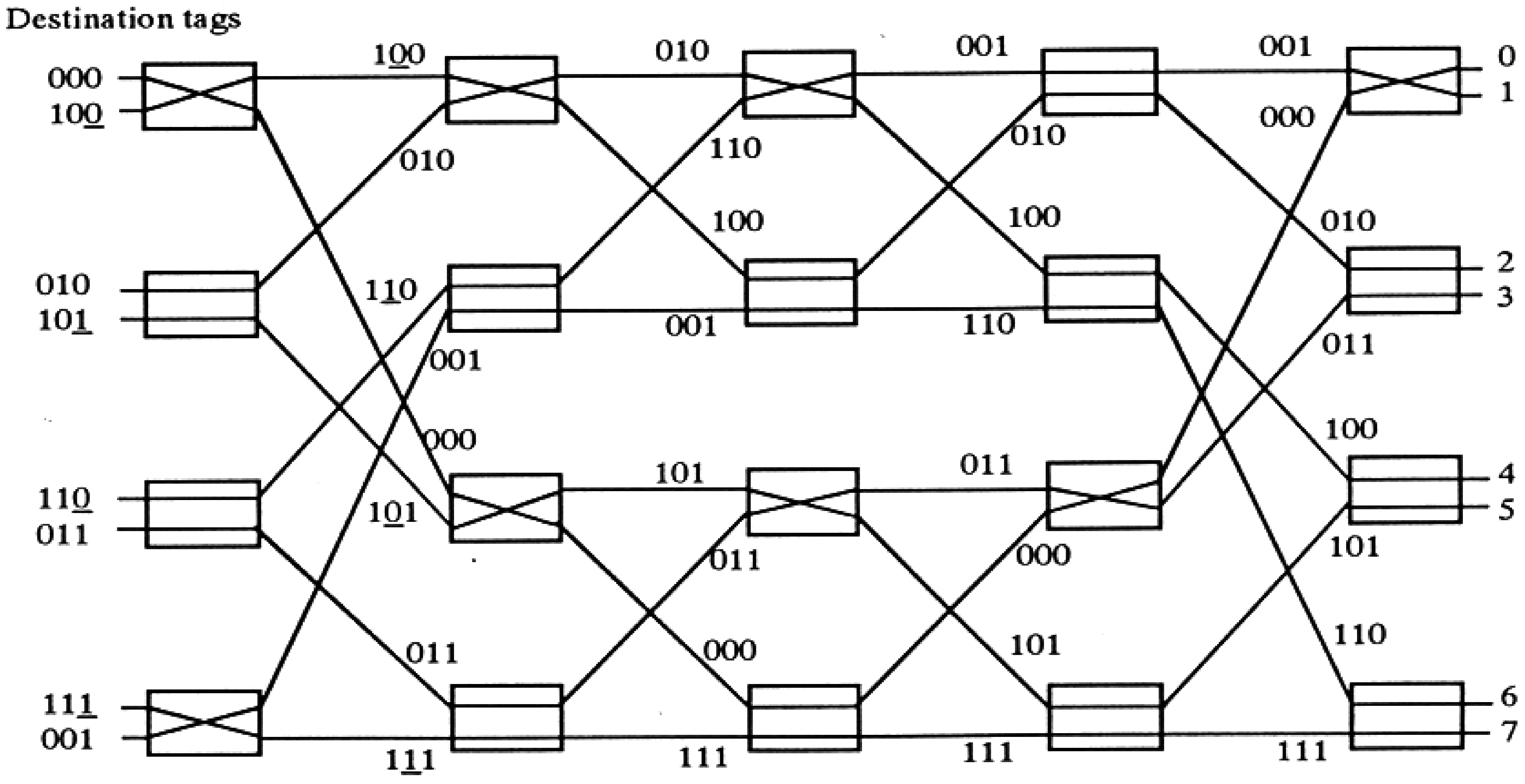


Figure 7.7 : Routing of Permutation P = (0 4 2 5 6 3 7 1) by Highest-Control-Routing Under Faults (Example 7.3).

7.4 Concepts of Equivalence and Closure Sets

As the definitions suggest, the TCR and BCR classes of permutations have similar characteristics and the same is true for LCR and HCR classes of permutations also. Actually, the classes TCR and BCR or HCR and LCR exihibit a one-to-one correspondence with each other. The idea of group transformations developed in [DBD90] finds some excellent applications in the analysis of the relations between TCR (HCR) and BCR (LCR). For better understanding, the idea of group transformations are described in brief in the following subsection.

Definition : For an N×N Benes network, the inputs (outputs) are grouped in different levels, as shown in figure 7.8. The size of a group at level i is 2^i. Two groups at level i are said to be *adjacent,* if both have the same parent at level (i+1).

Definition : Let $a \leftrightarrow b$ denote : interchange a, b. A *group-interchange* tX(j:x), (where X = I stands for input and X = O refers to output) applied on a permutation P, interchanges elements of two adjacent groups of inputs (outputs) at level j, $0 \leq j < n$, following the rule $k \leftrightarrow k+2^j$, $x \leq k < x+2^j$, where x is the least element of the two groups. This process generates another permutation P′ and is denoted by : tX(j:x) [P] → P′.

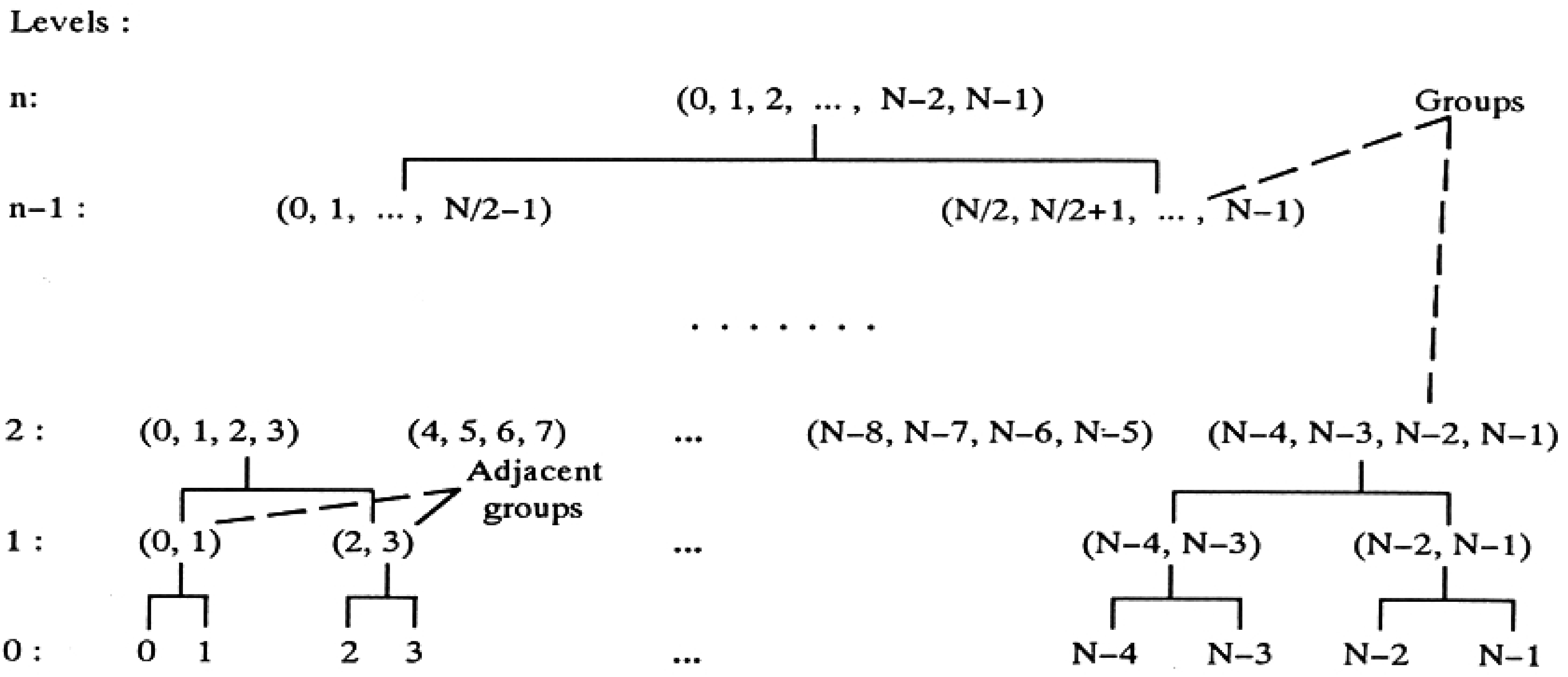


Figure 7.8 : The Input (Output) Groups at Different Levels

Example 7.4 : Consider a permutation $P = (7624\ 0315)$, and the group-interchange tI(1:4) such that $tI(1{:}4)[P] \rightarrow P'$; the interchanging input-pairs are : $4 \leftrightarrow 6$ and $5 \leftrightarrow 7$. Hence $P' = (7\,6\,2\,4\ \ 1\,5\,0\,3)$.

Similarly, a group-interchange on output tO(2:0) applied on P generates a permutation P'' given by $P'' = (3\,2\,6\,0\ \ 4\,7\,5\,1)$. (The output-pairs interchanged are : $0 \leftrightarrow 4$, $1 \leftrightarrow 5$, $2 \leftrightarrow 6$, $3 \leftrightarrow 7$).

Definition : A *sequence of input (output) group-interchanges* $\{tX(l_1{:}x_1);\ tX(l_2{:}x_2);\ \ldots;\ tX(l_k{:}x_k)\}$ is said to be ordered, if for $i < j$, $l_i \leq l_j$ and if $l_i = l_j \Rightarrow x_i < x_j$.

Example 7.5 : The sequence {tI(0:0); tI(0:2); tI(1:4)} applied on P transforms it through the following steps:

$$\begin{aligned} P = (7\,6\,2\,4\ \ 0\,3\,1\,5) &\xrightarrow{tI(0:0)} (6\,7\,2\,4\ \ 0\,3\,1\,5) \\ &\xrightarrow{tI(0:2)} (6\,7\,4\,2\ \ 0\,3\,1\,5) \\ &\xrightarrow{tI(1:4)} (6\,7\,4\,2\ \ 1\,5\,0\,3). \end{aligned}$$

Definition : Two sequences of input (output) group-interchanges RX_1 and RX_2 are said to be *equivalent* if for every permutation P, $RX_1[P] = RX_2[P]$.

Lemma 7.2 : An ordered sequence of input (output) group interchanges with repitition is equivalent to a shorter ordered sequence of group interchanges.

Proof **:** As the sequence is ordered, any repitition would be consecutive. Also for any permutation P, $tX(x{:}y), tX(x{:}y)\,[P] = P$, for X = I or O.

Definition : An input (output) group transformation GX, X = I or O is an ordered sequence of input (output) group interchanges without repitition .

Remark : Input group transformation defines an equivalence relation that partitions the set of all permutations into some equivalence classes. If a permutation P' is derivable from another permutation P by the application of some GI, i.e., $GI[P] \rightarrow P'$, we will say that P and P' belong to the same partition defined by input group transformation.

Obviously, the same is also true for output group transformations but the partitions in the two cases may be different i.e., if $GO[P] \rightarrow P'$, we will say that P and P' belong to the same equivalence class defined by output group transformations.

Lemma 7.3 : For any sequence of input (output) group-interchanges $RX = \{tX(l_1:x_1); tX(l_2:x_2); \dots ; tX(l_k:x_k)\}$ there exists an equivalent input (output) group transformation $GX = \{tX(w_1:z_1); tX(w_2:z_2); \dots ; tX(w_k:z_k)\}$.

Proof : The lemma can be proved by induction on k. The result is trivially true for $k = 1$ (basis). Let it be true for $k-1$ (induction hypothesis).

Without loss of generality let us consider a sequence of input group-interchanges $RI = \{tI(l_1:x_1); tI(l_2:x_2); \dots ; tI(l_{k-1}:x_{k-1}); tI(l_k:x_k)\} = \{RI_1 ; tI(l_k:x_k)\}$, where $RI_1 = \{tI(l_1:x_1); tI(l_2:x_2); \dots ; tI(l_{k-1}:x_{k-1})\}$, i.e., the subsequence of group-interchanges upto $(k-1)$-th term. By induction hypothesis, RI_1 can be replaced by an equivalent ordered sequence : $GI_1 = \{tI(w_1:z_1); tI(w_2:z_2); \dots ; tI(w_{k-1}:z_{k-1})\}$.

If $l_k > w_{k-1}$, $\{GI_1; tI(l_k:x_k)\}$ will be the required ordered sequence of input group-interchanges. If $l_k = w_{k-1}$, construct $\{GI_1; tI(l_k:x_k)\}$ as above; find all group-interchanges at level l_k and reorder them according to increasing values of x_k's. If $l_k < w_{k-1}$, choose $i = \min. \{j \mid w_j > l_k\}$. Let $GI_2 = \{tI\{(w_i:z_i); tI(w_{i+1}:z_{i+1}); \dots ; tI(w_{k-1}:z_{k-1})\}$. Let p be the output corresponding to the input x_k, in the permutation $GI_2[I]$, I being the identity permutation. Then $\{tI(w_1:z_1); \dots ; tI(w_{i-1}:z_{i-1}); tI(l_k:p); GI_2\}$ will be the equivalent ordered sequence. If $tI(l_k:p)$ is preceeded by some other group-interchanges at the same level, they have to be ordered from left to right with increasing values of p's. Thus an equivalent ordered sequence upto the k-th term always exists.

The same will be true for any sequence of output group-interchanges as well. ◆

Example 7.6 : Consider a sequence of input group-interchanges : $\{tI(0:4); tI(1:0); tI(0:2)\}$. The equivalent input group transformation can be generated as follows : first we create a sequence $\{tI(0:4); tI(0:p); tI(1:0)\}$, where p is the output corresponding to the input 2 in $tI(1:0)[I]$, i.e., $p = 0$; thus we get $\{tI(0:4); tI(0:0); tI(1:0)\}$. Rearranging the two tI's at level 0, we finally get, $\{tI(0:0); tI(0:4); tI(1:0)\}$ which is the equivalent input group transformation.

To transform a given permutation into another, a suitable ordered sequence of group-interchanges may or may not exist; but whenever it exists, it is unique. To prove this, we introduce the notion of input (output) clusters.

Definition : Given a permutation P, an *input (output) cluster* $CX(j,x)$, ($X = I$ for input cluster and O for output cluster), is defined as the sequence of inputs (outputs) corresponding to the outputs (inputs) of the group at level j, whose least output (input) is x.

The *set of input (output) clusters* at level j, denoted by SCX(j) is the collection of all input (output) clusters at level j.

Example 7.7 : For the permutation P = (6 7 4 2 1 5 0 3), CO(2,0) = (6,7,4,2), CI(2,4) = (2,5,0,1); SCO(1)= {(6,7), (4,2), (1,5), (0,3)}.

Lemma 7.4 : A group-interchange on inputs (outputs) tX(j:x), for all i's, $0 \leq i \leq j$, keeps the sets of output (input) clusters SCX(i) of a permutation unaltered; for $i > j$, the elements of any output (input) cluster are preserved, but the sequence may change.

Proof : Without loss of generality, consider only input group-interchanges. Let an input group interchange tI(j:x), applied on a permutation P, result some other permutation P′. From the definition of group-interchange, it follows that the output clusters CO(j, x) and CO(j, x+2^j) for P (P′) will be identical to output clusters CO(j, x+2^j) and CO(j, x) for P′ (P) respectively. But the set SCO(j) will remain same for both P and P′. Thus the lemma is proved for all i's, $0 \leq i \leq j$. For $i > j$, the claim trivially follows. ◆

Theorem 7.5 : For two different input (output) group transformations GX and GX′, GX[P] ≠ GX′[P], for any P.

Proof : Let us consider two different input group transformations GI = {tI{(w_1:z_1); tI(w_2:z_2); ... ; tI(w_k:z_k)} and GI′ = {tI{(l_1:x_1); tI(l_2:x_2); ... ; tI(l_m:x_m)}. Since GI and GI′ are different, for some p, tI(w_p:z_p) ≠ tI(l_p:x_p). Without loss of generality, we may assume that tI(w_1:z_1) ≠ tI(l_1:x_1). Let $w_1 < l_1$. Since in GI′, all group interchanges are in levels $j \geq l_1$, by lemma 7.4, the set of output clusters at level w_1+1 in GI′[P] will remain same as that in P. But in GI[P] it will be different. If $w_1 > l_1$, similar argument works. Lastly, if $w_1 = l_1$, $z_1 \neq x_1$; this also results GI[P] ≠ GI′[P]. The claim similarly follows for output group-interchanges. ◆

Definition : Given any permutation P, let SX(P), X = I or O, denote the set of all permutations derivable from P by the application of all possible input (output) group transformations. Then SX(P) is said to be the *input (output) equivalence set* of P.

Lemma 7.5 : Given any permutation P, the cardinality of its input (output) equivalence set SX(P) is 2^{N-1}, X = I or O.

Proof: For an N×N system, at any level j we can apply input group interchanges independently on $2^{n-(j+1)}$ groups. Hence the total number of distinct sequences of input group interchanges at that level is $2^{n-(j+1)}$. By lemma 7.3 and theorem 7.5, we get the total number of distinct input group transformations for this system as $\prod_{j=0,n-1} 2^{n-(j+1)} = 2^{N-1}$.

Now by theorem 7.4, we get that from any permutation P, by the application of all possible input group transformations we can generate a set of 2^{N-1} permutations, which is the input equivalence set SI(P).

The same is true for output equivalence set SO(P) also. ◆

So far we considered input (output) group-interchanges in isolation. Simultaneous application of both in appropriate order now leads to the concept of group-transformation that explores some new guidelines of routing in interconnection networks.

Definition : A *group-transformation T* is defined as a sequence of an output group transformation followed by an input group transformation (any one may be a null sequence also).

Example 7.8 : A group-transformation T = {tO(0:0), tI(1:4)} applied on P will transform P in the following way :

$$P = (7\,2\,6\,4 \quad 0\,3\,1\,5) \xrightarrow{tO(0:0)} P' = (7\,2\,6\,4 \quad 1\,3\,0\,5) \xrightarrow{tI(1:4)} P'' = (7\,2\,6\,4 \quad 0\,5\,1\,3)$$

It is easy to show that for any random sequence of input and output group-interchanges, there exists a group-transformation which imparts same effect on any permutation. This follows from a similar argument as in lemma 7.3. Thus it is sufficient to consider group-transformations as defined above.

Group-transformation induces an *equivalence partition* on the set of all permutations. If a permutation P′ is derivable from another permutation P by applying some group-transformation T, i.e., if T [P] → P′, we say that P′ ^ P (P′ is related to P) and it is easy to see that '^' is an *equivalence relationship.*

Definition : Given a permutation P, let C(P) denote the set of permutations derivable from P by the application of all possible group-transformations. Then C(P) is said to be the *closure set* of P.

Note that C(P) is a superset of SI(P) as well as of SO(P).

Theorem 7.6 : The same optimal routing algorithm may be applied for routing all the permutations belonging to the same closure set.

Proof : See [DBD90]. ◆

These ideas about group transformations, are already introduced in [DBD90], mainly to resolve the problem of optimal routing in blocking $\log_2 N$ stage, full-access and unique-path MIN's, such as baseline, omega, inverse-omega, etc. The group-transformation rules described above are actually applicable for baseline network only. But it has been shown in [DBD90], that these ideas are general and can be extended to any full-access unique-path MIN.

Now, it is interesting to note that these concepts of equivalence classes and closure sets find an application in self-routing of permutations in Benes network. The following section gives the details.

7.5 Group Transformations and Self-Routable Permutations

The following lemmas state some results on the cardinalities of the four self-routable classes of permutations.

Lemma 7.6 : The cardinality of the set of TCR permutations is exactly equal to the cardinality of the set of BCR permutations.

Proof: Let us consider a permutation $P \in$ TCR and route it by top control routing technique. Now, let us apply input group interchanges tI(0:x), for all possible values of x, on P that transforms it into P′. It will essentially exchange the two inputs i.e., the destination tags T_{0j} and B_{0j} of all the switches S_{0j}, for $0 \leq j < N/2$. Since P was routable by top control, if we route P′ by bottom control technique, the switches S_{1j}, for $0 \leq j < N/2$, will all have the same values as they have for P under top control routing.

Therefore, it is easy to see that if we apply all the input group interchanges tI(j:x), for $0 \leq j < n$ and for each j, with all possible values of x, on P to transform it into P*, say, then if P is top control routable, P* will be bottom control routable. Therefore by theorem 7.5, for each $P \in$ TCR, there exists a unique P* in the set BCR. It proves the lemma. ◆

Example 7.9 : Let P = (0 2 1 6 7 3 4 5), P∈TCR, its routing is shown in figure 7.3. Now let us generate P* from P by the input group transformation, {tI(2:0), tI(1:0), tI(1:4), tI(0:0), tI(0:2), tI(0:4), tI(0:6)}[P] → P*, i.e., P* = (5 4 3 7 6 1 2 0). Note that P*∈BCR.

Lemma 7.7 : The cardinality of the set of LCR permutations is exactly equal to that of the set of HCR permutations.

Proof: Let us consider a permutation P∈LCR and route it through the Benes network by least control routing technique.

Now let us apply output group interchanges {tO(n−1:0)}[P], to generate P′, it will just complement the MSB of each destination tag of P in P′, with all other bits unchanged. To route P′, we will find that for all the switches S_{0j}, for $0 \leq j < N/2$, the highest and least inputs will interchange their positions (top and bottom). Since in P′, the LSB's of the destination tags remain the same as they were in P, if we route P′ by highest control technique, the switches S_{1j} for $0 \leq j < N/2$, will have the same input (top or bottom) as their least input.

Therefore, if we apply output group interchanges {tO(i:x)}, for $0 \leq i < n$ and for each i, we include all possible values of x, P will generate P*, such that if P∈ LCR, P* ∈HCR.

Again by theorem 7.5, for a given P, P* is unique. Hence follows the lemma. ◆

Example 7.10 : Let us consider a permutation P = (0 4 2 7 5 3 6 1), where P∈LCR, the routing is shown in figure 7.5.

Now let us transform P to P* in the following way :

{tO(2:0), tO(1:0), tO(1:4), tO(0:0), tI(0:2), tO(0:4), tO(0:6)}[P] → P*, where P* = (7 3 5 0 2 4 1 6). Note that P* ∈HCR.

In [BR88], it has been mentioned that the least-control self-routing technique is applicable to LC (Linear-Complement) class of permutations as well as IΩ permutations. Here, it has been shown that the IΩ permutations are routable by all the four classes of self-routing strategies discussed above. The following theorem states an interesting property of LCR and HCR classes.

Theorem 7.7 : If a permutation $P \in$ LCR (HCR), then any permutation $P' \in SI(P)$ also belong to LCR (HCR), where SI(P) is the input equivalence set of P.

Proof: Let us route both P and P′ by least control routing technique in the first n stages. After that P is obviously routable by destination tag. What we have to show is that P' is also routable by destination tag.

We note certain similarities in the sets of inputs to the different stages of the network. By lemma 7.2, the output clusters of P and P′ are the same. In particular, in both the cases, the set of output clusters at level 1 are the same. The routing algorithms are the same and so also are the pairings at all the switches. So an input which is routed through the upper link in P, will also be routed through the upper link in P′. The set of inputs which are routed through the upper link in stage 0 forms the input set for the top half of stage 1. So, both P and P′ will have the same set of inputs for the top half (and also the bottom half) of stage 1.

Let us think of the inputs to stage 1, also as a permutation. Let the permutations corresponding to P and P′ be $P_{(1)}$ and $P'_{(1)}$ respectively. We observe that the sets of output clusters of $P_{(1)}$ and $P'_{(1)}$ are the same. Take the output clusters of size 2^k in $P_{(1)}$ (say, the top half). This is formed by the upper links of an output cluster of size 2^{k+1} of P. This output cluster of P is also present somewhere in P′ (by lemma 7.1). Now consider the inputs, routed through the upper links of that output cluster of P′. It forms an output cluster of size 2^k in $P'_{(1)}$. This output cluster of $P'_{(1)}$ is the same as the output cluster of $P_{(1)}$ we started with.

If we repeat the above arguments, it is clear that at stage i, the output clusters at level i, are not only the same, but also in same position with respect to the inputs. Also, in general the set of output clusters of $P_{(i)}$ and $P'_{(i)}$ are the same. From this, we see that, the input pairs to all the switches of stage (n−1) are the same for $P_{(n-1)}$ and $P'_{(n-1)}$. Since $P_{(n-1)}$ is passable, so is $P'_{(n-1)}$.

It is evident that the same will be true for any permutation routable by highest control technique. ◆

Definition : Let C represent any class of permutations, then CE is defined as the union of all input equivalence classes generated by the permutations $P \in C$, and is denoted by, $CE = \bigcup_{\forall P \in C} SI(P)$.

Corollary 7.3 : Any permutation $P \in CE$ where $C = LE \cup I\Omega$, is routable by least-control routing technique.

Proof: Follows directly from theorem 7.6, since LC and $I\Omega$ classes are routable by least-control routing technique. ◆

The following subsection presents the analysis of a prticular subset of CE, namely the BPE class.

7.5.1 BPE (Bit-Permute Equivalent) Class of Permutation

BP (bit-permute) class of permutations [Le78] is a typical subset of L (linear) class of permutations [DBD90]. For any permutation $P \in BP$, for all input I (binary representation $I_n I_{n-1} \dots I_1$) and output O (binary representation $O_n O_{n-1} \dots O_1$), the non-singular binary matrix Q_{nxn} that satisfies $O^T = Q \times I^T$, is essentially the identity matrix, with some rows interchanged. The efffect is that for P, for all input-output pair I & O, $O_j = I_k$, for $1 \leq j, k \leq n$ and $O_i = O_j$ for $i = j$.

Definition : The BPE (bit-permute equivalent) class of permutations is the union of all the permutations generated by each $P \in BP$, by the application of all possible input group transformations.

Remark : Since BP is a subset of L, BPE is a subset of CE.

Example 7.11 : Given a permutation $P^* \in BP$, determined by the rule, $I_3 I_2 I_1 \rightarrow I_2 I_1 I_3$, i.e., $P^* = (0\,2\,4\,6\;\;1\,3\,5\,7)$, generate P by the application of $T = \{tI(1{:}0), tI(0{:}6), tI(0{:}4), tI(0{:}2)\}$ on P^*, i.e., $T[P^*] \rightarrow P$. Hence $P = (6\,4\,0\,2\;\;3\,1\,7\,5)$, whose routing is shown in figure 7.3. Note that, $P \in BPE$.

Lemma 7.8 : Let P_1 and P_2 denote two distinct BP-permutations. Then there does not exist any input group transformation GI such that $GI[P_1] = P_2$.

Proof : Let $P_1 : I_{n-1} I_{n-2} \dots I_1 \rightarrow O_{n-1,1} O_{n-2,1} \dots O_{1,1}$ and $P_2 : I_{n-1} I_{n-2} \dots I_1 \rightarrow O_{n-1,2} O_{n-2,2} \dots O_{1,2}$ where $y_{ik} = x_j$ for $1 \leq i, j < n$ and $y_{ik} = y_{jk}$ for $k = 1, 2$.

Now say $y_{i1} = y_{i2} = x_j$ for $1 \leq j < m$, $1 \leq i < n$. Then it is easy to see that both P_1 and P_2 will have the same sets of output clusters at all levels $l \leq m$. But, at level (m+1) the output clusters are different and by lemma 7.1, no input group transformation can make them equal. Hence follows the lemma. ◆

Lemma 7.9 : | BPE | = n! 2^{N-1}, as | BP | = n!.

Proof : Follows from lemma 7.7. ◆

Given any permutation P, we present an algorithm BPE-ID, that will decide whether P∈BPE or not and will output the BP that generates P by input group transformation. If it is successful, we can immediately route it by LCR or HCR technique.

Lemma 7.10 : A permutation P is a BP permutation iff :

a) the outputs (inputs) X(k) and X($k+2^i$) corresponding to the least inputs (outputs) k and $k+2^i$ of two adjacent input (output) groups at level i, differ in a unique bit position for all possible values of k and i.

b) the output (input) X(k) corresponding to the least element k of an input (output) group at level i, is the least element of the output (input) cluster CX(i,k) for all possible values of i and k.

Proof : Since a BP permutation is defined by a definite bit-permute rule, any BP will satisfy the above conditions.

If any permutation P satisfies the above rules, we can easily construct a unique BP-rule to describe P. Since, a permutation P∈ BPE is the outcome of an input group transformation on a BP P*, by lemma 7.4, the elements of the output clusters are same in P* and P. Our algorithm checks P for condition b) of lemma 7.10, first at level 1 and rearranges the output clusters CO(1,x), if necessary, by applying tI(0:x) and for all clusters it also checks condition a). If it succeeds, it proceeds to higher levels. Ultimately it results a BP at level n or fails at an intermediate level. ◆

Algorithm BPE-ID

The permutation P is stored in the form of an array OUT(.), where OUT(i) stores the output corresponding to the input i.

```
1. For k=1 to n, repeat
        1.1 d=|OUT(0) – OUT(2k-1)|
            If d ≠2j, $0 \leq j < n$, OUTPUT "P is not in BPE" and terminate.
        1.2 For p=0 to 2n-k –1
                1.2.1 Find the least element l of OUT(p.2k) and OUT(p.2k+2k-1)
                1.2.2 If l=OUT(p.2k+2k-1) apply tI(k–1: p.2k).
        1.2.3 If OUT(p.2k+2k-1) - OUT(p.2k) $\neq$ d, then OUTPUT "P is not in BPE"
              and terminate.
        1.2.4 Next p.
    1.3 Next k
2. OUTPUT the string OUT that represents the generator BP and terminate.
```

Complexity Analysis of the Algorithm BPE-ID :

At any stage k, $1 \leq k \leq n$, our algorithm compares two outputs OUT(x) and OUT(x + 2^{k-1}) and if necessary, it applies tI(k–1:x), where $x = p.2^k$ for $0 \leq p \leq 2^{n-k} - 1$. For each p, an extra comparison is needed to check condition a) of lemma 7.6. Therefore, at stage k, it will require 2^{n-k+1} comparisons and N/2 input interchanges in the worst case.

The total number of comparisons involved is $\sum_{k=1}^{n} (2^{n-k+1} + N/2)$.

Hence the complexity of the algorithm is O(N logN) on a uniprocessor system. But the comparisons and interchanges at each level can be performed in parallel using a system with N/2 PE's. This will reduce the complexity of the algorithm to O(log N) only.

Remark : The BPE (bit-permute-equivalent) class of permutations are routable by both least-control and highest-control techniques, since BP's are routable by both (theorem 7.6).

Lemma 7.11 : $|LCR \cap HCR| \geq n!\, 2^{N-1}$.

Proof : Clear from the remark above, since $|BPE| = n!\, 2^{N-1}$. ◆

7.5.2 BPCL (Bit-Permute Closure) Class of Permutations

The idea of closure sets have already been introduced in [DBD90]. Now, with the self-routing of BPE class of permutations in Benes network, we are interested to know, whether any self-routing strategy can be developed for a much larger class of permutations, namely the BPCL class [DBD90], defined below.

Definition : The *BPCL (bit-permute closure) set* of permutations is defined as :

$\text{BPCL} = \bigcup_{\forall P \in BP} \text{Closure}(P)$.

Example 7.12: Let us consider a permutation P = (2 0 6 4 7 5 1 3). Note that {tO(1:4), tI(0:0), tI(1:4)} [P] → P*, where P* = (0 2 4 6 1 3 5 7) is a BP, defined by the bit-permute rule, $x_2\ x_1\ x_0 \rightarrow x_1\ x_0\ x_2$. Therefore P^P*, i.e., P ∈ BPCL.

It has been established in [DBD90], that the BP generated from a given P is unique, or in other words, the closure sets generated from the BP's are all disjoint .

Routing of BPCL Permutations in Benes Network

Benes network, as mentioned earlier, comprises of a baseline network, followed by a reverse baseline network. Now, these two networks are not only topologically equivalent but also the interconnection between stages of the two are such that they realize the same set of permutations. It results in same group interchange rules for both [DBD90], i.e., the same group interchange rules developed for baseline network are applicable for reverse baseline network.

Theorem 7.8 : Given any permutation P ∈ BPCL, there exists a P′ ∈ SO(P), such that P′ ∈ BPE. If we route P′ by LCR (HCR) through the first (n−1) stages of the Benes network and next at the input of n-th stage, we recover P by the application of output group interchanges GO′, such that GO′[P′] → P, on the destination tags and route it by normal destination tag routing through the rest of the network, it will ultimately route P through the Benes network,

Proof : For any permutation P ∈ BPCL, there exists a group transformation T, such that T[P] → P*, where P* ∈ BP. Now, by definition, T is a sequence of an output group transformation GO, followed by an input group transformation GI, i.e., GO[P] → P′ and GI[P′] → P*. As, the input (output) group transformations result an equivalence relation, it is obvious that there exist input (output) group transformations, to generate P from P* in a reverse way- GI′[P*] → P' and GO′[P′] → P. Since P* ∈ BP, P* ∈ LCR (HCR), by theorem 7.2. Now P′ ∈ SI(P*) and therefore, P′ ∈ LCR (HCR), by theorem 7.7.

Let us route P′ by LCR *through the first (n−1) stages of the Benes network. That* will result a permutation, say P″ at the input of n-th stage of the Benes network. The rest part of the network is essentially a reverse baseline network and P″ is passable through it without conflict, since P′ ∈ LCR. Now let us transform P″ by,

$GO'[P''] \rightarrow P'''$ and route it through the remaining stages. Since P'' is routable through the reverse baseline network, so is P''', by theorem 7.6.

Next, we are to prove that this routing technique, essentially realizes P on the Benes network.

Note that if we would not apply GO′ at the input of n-th stage, we will simply realize P′ on Benes network. Now, since $GO'[P'] \rightarrow P$, the routing technique will ultimately realize P.

Therefore, to route a permutation $P \in BPCL$, we may transform it into $P' \in BPE$ and route it through the first (n−1) stages of the Benes network by least (highest) control algorithm and next at the input of the reverse baseline network, we apply GO′ on the destination tags to recover P and route it through the following stages. The switch settings obtained by this technique, essentially will route P through the Benes network. ◆

Next, we develop an algorithm that will check if a given permutation $P \in BPCL$ and if yes, it will generate a $P' \in BPE$. It will also result GO′, where $GO'[P'] \rightarrow P$, in terms of an output node correspondece between P′ and P.

Algorithm BPCL-Routing:

The given permutation P is represented as an array IN(.) and OUT(.), where IN(i) is the input corresponding to the output i in P and OUT(i) stores the output corresponding to input i. N(.) is another array that stores the output node correspondence between P′ and P. $N(i)=j$, indicates that the output i in P′ is replaced by output j in P. Initially $N(i) = i$, for $0 \leq i < N$.

```
1. For $k = 1$ to n do
   1.1 For $p = 0$ to $(2^{n-k} - 1)$ do
       1.1.1 If $IN(p.2^k) > IN(p.2^k + 2^{k-1})$ then apply $tO(k-1: p.2^k)$ and modify IN(.),
             OUT(.) and N(i) accordingly.
       1.2.2 Next p.
   1.2 Next k
2. For $k = 1$ to n do
   2.1 $d = |OUT(0) - OUT(2^{k-1})|$
       If $(d \neq 2^j, \forall\ 0 \leq j < n)$ then OUTPUT "P is not in BPCL" and terminate.
```

```
    2.2 For p = 0 to $2^{n-k} - 1$ do
        2.2.1 If $|OUT(p.2^k + 2^{k-1}) - OUT(p.2^k)| \neq d$, then OUTPUT "P is not in
              BPCL" and terminate.
        2.2.2 Next p
    2.3 Next k
3. OUTPUT the arrays OUT(.), (which is the BPE) and N(.) (the output node
   correspondence) and terminate.
```

In step 1, we are applying necessary output group interchanges on P, so that the transformed permutation P′ may satisfy condition b) of lemma 7.10 for input clusters. Step 2 is necessary to check if $P' \in$ BPE. The computations involved are very much similar to those in algorithm BPE-ID and it is easy to see that the time complexity of the algorithm is $O(N \log N)$ in a uniprocessor system and can be implemented in a parallel system with time complexity $O(\log N)$ only.

Example 7.13 : Let us consider a permutation P → BPCL, given by, P = (5 3 2 4 0 6 1 7), such that, {tO(1:0), tI(0:0), tI(1:0)}[P] → P*, where, P* = (0 4 1 5 2 6 3 7), is a BP, defined by the BP-rule, $x_2 x_1 x_0 \rightarrow x_0 x_2 x_1$. Our algorithm will result P′ = (0 4 5 1 6 2 7 3) and N(.) = (5 4 6 7 3 2 0 1).

Note that $P' \in$ BPE. Therefore it is least control (highest control) routable. To route P, we present P′ at the input of Benes network and route it by least control routing technique through the first (n−1) stages, namely stages 0 to (n−2). Now at the input of stage (n−1), the destination tags are changed according to N(.) and route it through the remaining stages by normal destination tag routing. It will finally realize P in Benes network. Figure 7.9 shows the complete routing of P.

7.6 Conclusion

In this chapter, we propose four simple self-routing strategies for N × N Benes network. It is shown that the union of the BPC and IΩ class of permutations with cardinality $2^n (2^{N/2} + n! - 1)$ is a subset of the intersection of all the four classes of permutations routable by the proposed self-routing strategies. It implies that $2^n (2^{N/2} + n! - 1)$ is the lower bound for the cardinality of any class of self-routable permutations, considered here. The enumeration of the exact cardinality of each class is still an open problem. But by the application of the theory of equivalence classes as presented in [DBD90], we establish the facts that $|TCR| = |BCR|$, $|LCR| = |HCR|$ and that $|LCR \cap HCR| \geq n!\, 2^{N-1}$. Next we develop an algorithm, with time complexity

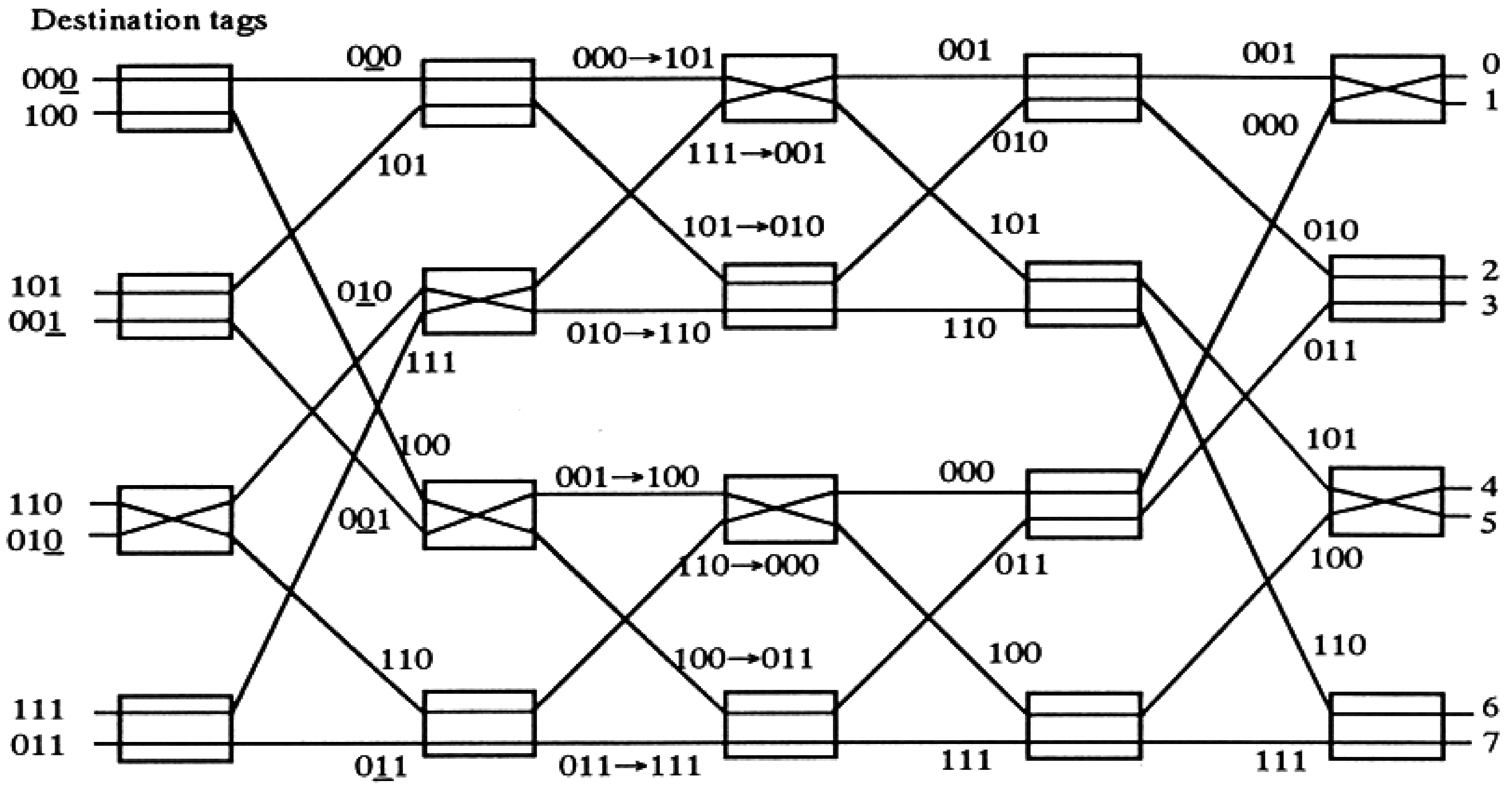


Figure 7.9 : Routing of Permutation P = (5 3 2 4 0 6 1 7) of Example 7.13.

O(n) that will identify if any given permutation P ∈ S_i, where S_i is a self-routable class of permutations mentioned here and also generates the necessary controls for routing of P. We also mention how easily the algorithm can be modified to generate the controls even in presence of some faults in the network.

Finally, we outline an *O*(n) routing technique for BPCL class of permutations [DBD90]. This class actually covers a large number of permutations. It includes the BPC, BPE, LC and LE classes of permutations.

If we allow combinations of S_i's for routing in different stages i, $0 \leq i \leq n-2$, the set of self-routable permutations can be extended further. Moreover, we can generalize our results to any other (2n−1)-stage MIN's, such as (2n−1)-stage shuffle-exchange network etc.

Chapter 8
Multi-Layered Non-Blocking MIN's

8.1 Introduction

Depending upon the capability of a dynamic interconnection network (IN) to achieve different permutations of input-output combinations, dynamic IN's may be classified into a) *blocking* and b) *non-blocking*. In some IN's a particular pattern of connection between input and output lines may not be possible. Such IN's are called blocking networks, the others are called non-blocking networks. If we assume that input-output connection requirements are coming one by one, in some non-blocking networks it may so happen that a particular input-output connection request can not be satisfied till some of the existing connections finish their communications. But by rearranging the existing connections, it is possible to accommodate all the existing input-output connections as well as the new request. In other words, the network can realize all possible permutations of input-output connections, but in a partially connected state it may not be possible to incorporate a new request with disturbing the existing connections. Such IN's are called the *rearrangeable* IN's. Benes network is a well-known rearrangeable MIN. Another possible classification of non-blocking IN's is due to Benes [Be62]. If a network is always non-blocking, it is called *non-blocking in the strict sense*. If a network has blocking states but the blocking states can be avoided by some technique (say, by suitable routing) the network is called *non-blocking in the wide sense*. Rearrangeable networks are non-blocking in the wide sense.

Though non-blocking networks are better from the point of view of performances, they are more expensive too. Two examples of networks non-blocking in the strict sense, are crossbar [Fe81] and Clos networks [Cl53]. In a crossbar network, the hardware cost is $O(N^2)$ for N inputs and N outputs. In a 3-stage Clos network C(n, m, r), the number of crosspoints for $n = \sqrt{N}$ is, $6N^{3/2} - 3N$, i.e., $O(N^{3/2})$. Compared to this, an N×N Benes network, which is rearrangeable, has cost $O(N \log_2 N)$. Another network non-blocking in the strict sense, is the Cantór network [Ca71]. Cantor network consists of several layers of N×N Benes networks. Each input line is connected to all the corresponding positions in the different layers. Similarly each output line is connected to all the multiple layers.

In this chapter, we propose a model for analyzing the blocking / non-blocking behavior of a Multistage Interconnection Network (MIN). We also introduce a concept of *routing-dependent* non-blocking behavior of networks and show that a single-layer routing-dependent MIN with 2×2 switches needs at least $(N-1)$ stages, resulting into $\Omega(N^2)$ hardware cost. We then show that $\log_2 N$ such layers of Benes network are necessary and sufficient to make the IN non-blocking in the strict sense, irrespective of the routing strategy. The non-blocking layered Benes network requires $N(\log_2 N)^2$ number of 2×2 switches as opposed to $O(N^2)$ hardware complexity of crossbar network or $O(N^{3/2})$ cost of Clos network. The fan-out / fan-in at each input output line is however, $\log_2 N$. We also propose a routing hardware for the above network. Routing through the layered Benes network is performed by a combinatorial circuit, whose different stages are suitably pipelined. This routing hardware requires additional $O(N)$ 2 to 1 multiplexers, 1 to 2 demultiplexers and a bank of circulating shift registers with a total of $N \log_2 N$ bits of information. Due to the pipelined operation, the routing time is limited to be of the same order as the delay through $2 \log_2 N$ stages of 2×2 switches.

We also find the minimum number of layers for making a multi-layered Baseline network and a $(2 \log_2 N - 1)$-stage shuffle-exchange network non-blocking, independent of the routing strategy.

8.2 The Underlying Model and Some Basic Results

In this work, we shall concentrate on MINs with the following properties :

i) Only 2×2 switching elements are used.
ii) Connections are only among switches in consecutive stages.
iii) Switches can only be in straight mode or in exchange mode.
iv) Requests for input-output connections arrive one by one and they are serviced immediately without any queuing.

The number of input lines (source) and output-lines (destination) will generally be denoted by N. The sources will be denoted by S_i, $0 \leq i \leq N-1$, and destinations T_i, $0 \leq i \leq N-1$. A *state* of a network is a description of the existing paths. We shall call a state *blocking* if a request for connection between any free input line and a free output line cannot be served. A *non-blocking* state is one which is not blocking. Here we note that in a non-blocking state only the next request for connection is guaranteed to be served. But from a non-blocking state, the network can go into a blocking state. A state will be called *leading to blocking* if there exists a sequence of requests for connection set-up and interleaved with requests for connection release, such that starting from that state, this sequence leads the network into a blocking state. We define an *all-busy* state to be one in which all the input lines are busy in communicating with the output lines. Two paths of a network are said to *meet at a switch,* if the two input lines to the switch belong to these two paths.

Lemma 8.1 : In an all-busy state, if any two paths do not meet at some switch, the state is leading to blocking.

Proof : Let P_1 and P_2 be the two paths which do not meet at any switch. Let the sources and destinations of P_1 and P_2 be (S_i, T_j) and (S_x, T_y) respectively. Consider the transitions in which the paths P_1 and P_2 are released after communication. We claim that the source S_i cannot be connected to destination T_y. Since P_1 and P_2 do not meet at any switch, for every switch at least one of the lines will belong to an existing path (note that we start from an all-busy state). Thus, none of the switch settings can be changed, without disturbing the existing paths. But in the existing switch setting S_i can only be connected to T_j. ◆

Lemma 8.2 : In any state of a network, if two paths do not meet at some switch, the state is leading to blocking.

Proof : Take any sequence of requests for setting-up connections (without any request for release) among all the currently free sources and the currently free destinations. If this sequence cannot be satisfied, our claim is vindicated. If the sequence of requests is satisfied, we have an all-busy state with two paths not meeting at any switch. Hence, the result follows from lemma 8.1. ◆

8.2.1 Routing-Dependent Non-Blocking MINs

The existing concept of the non-blocking behavior of networks do not concern about any particular routing strategy. We call a network non-blocking when no input-output connection request can ever be blocked, irrespective of the specific routing strategy used for the network. In contrast to these networks, one can design a network which has blocking states, but the blocking states can be avoided through proper routing. In other words, if a particular routing discipline is followed, it can be shown that one would never reach a blocking state.

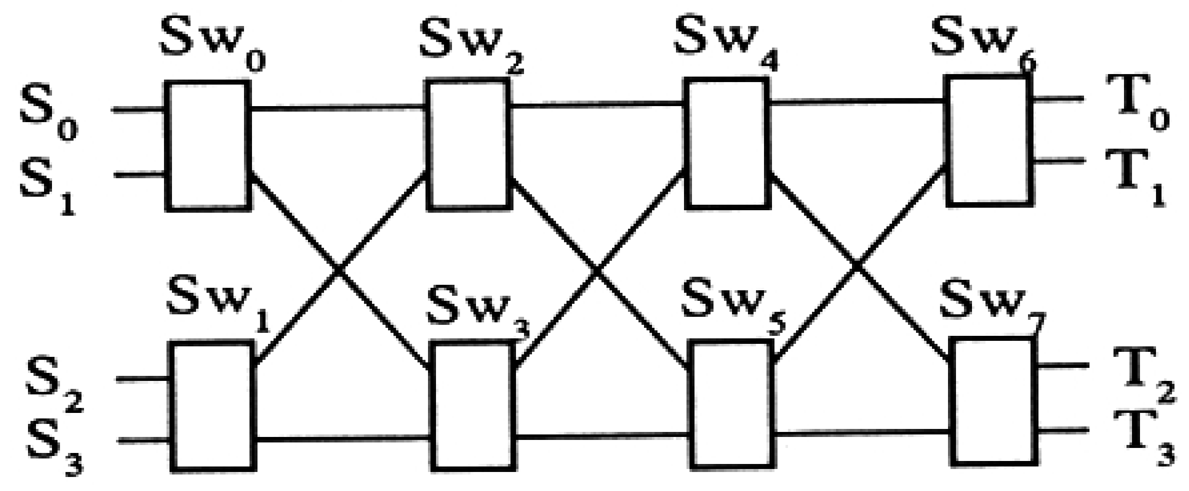


Figure 8.1 : An example of a routing dependent MIN

Example 8.1 : Consider the network in figure 8.1, with the switches numbered as shown in the figure. We note that the network becomes blocking, only if all of the intermediate switches, i.e., Sw_2, Sw_3, Sw_4 and Sw_5 are set to the same mode (straight or exchange). One can design a routing strategy such that blocking states are avoided. So, the network in figure 8.1 is routing-dependent non-blocking but not routing-independent non-blocking.

Such a network can be grouped into the general category of "non-blocking in the wide-sense" [Be62]. However, we designate these networks as *routing-dependent non-blocking networks* (to distinguish them from the non-blocking in the strict-sense networks) so that the behavior of such networks can be truly reflected through this classification.

Theorem 8.1 : For N inputs (N > 2), it is not possible to design a single-layered strictly non-blocking MIN.

Proof: Take any single-layered MIN. We claim that there exists a state of the network in which two of the paths do not meet at any switch. We can construct these two paths as follows. First we choose two input lines which are not incident on the same switch. We can always do this, as we have more than one switch at each stage (as N > 2). After constructing the paths non-intersecting upto i stages ($i \geq 1$), we have two paths leading upto two different switches at stage i. From these two switches there are 4 outgoing lines. Hence, they must go to at least two different switches. So we can always extend the path upto stage (i+1). Once we have two paths which do not meet at any switch, we can set the rest of the switches arbitrarily, to make the state all-busy. Now, it follows from lemma 8.1, that this state of the network is leading to blocking. In other words, the network can be blocked. ◆

In a network, when a particular routing strategy is followed, starting from an initial state the network may not go to all possible states, but only a subset of the total set of states. We shall call these states as the *valid* states of the network, *with respect to the routing strategy and the initial state.* We observe that for a network to be routing-dependent non-blocking, the valid states can neither be blocking nor leading to blocking.

Theorem 8.2 : A single-layered routing-dependent non-blocking MIN with N inputs has at least $(N - 1)$ stages.

Proof: Take any valid all-busy state of the network. Consider the path from source S_0. Since the state is not leading to blocking, this path must meet all $(N - 1)$ other paths at some switch or the other. Since at any stage the path goes through a single switch, it can meet at most one new path at each stage. So there must be at least $(N - 1)$ stages. ◆

Corollary : A single-layered routing dependent non-blocking MIN needs $\Omega(N^2)$ hardware cost. In other words, the cost is at least that of a crossbar network.

8.3 Analysis of Blockage in Benes and Shuffle-Exchange Networks

In this section, first we discuss the blocking properties of a Benes network. In the sub-sections 8.3.1 and 8.3.2, unless otherwise mentioned, by a network we shall mean a Benes network.

8.3.1 Blocking Properties of Single-Layered Benes Network

We assume that the stages of the network are numbered as 0 onwards from the input side.

Definition : By 'Middle Boxes' we shall mean the switches in stage $n-1$ $(= \log_2 N - 1)$. The middle boxes will be denoted by M_i, $1 \leq i \leq N/2 - 1$. Figure 8.2 shows the middle boxes for the Benes network with $N = 8$.

Notation : $S_i \leftrightarrow T_j$ will denote a path connecting source S_i to destination T_j.

Definition : Let P be a path already set up in a network. For a new path from S_i to T_j we define the *blockage due to P* as,

$bl(P, i \leftrightarrow j) =$ The number of middle boxes which can not be reached from S_i and /or those from which T_j can not be reached because of P.

Definition : For a new path between S_i and T_j to be set up in network, we define the *total blockage* in the network as

$BL(i \leftrightarrow j)$ = The number of middle boxes which cannot be reached from S_i and /or those from which T_j can not be reached because of the existing paths in the network.

Lemma 8.3 : Let $P_1, P_2, \ldots, P_k$ be the existing paths in the network. Then

$$BL(i \leftrightarrow j) \le \sum_{t=1}^{k} bl(P_t, i \leftrightarrow j).$$

Proof: Let S_t be the set of middle boxes which are blocked by the path P_t. Then,

$$BL(i \leftrightarrow j) = \left|\bigcup_{t=1}^{k} S_t\right| \le \sum_{t=1}^{k} |S_t| = \sum_{t=1}^{k} bl(P_t, i \leftrightarrow j).$$ ◆

Theorem 8.3 : A path from S_i to T_j exists if and only if $BL(i \leftrightarrow j) < N/2$.

Proof : (**<u>If</u>** part) If $BL(i \leftrightarrow j) < N/2$, then at least one middle box can be reached from S_i and from that middle box T_j can also be reached. So we can set up the required path between S_i and T_j through that middle box.

(**<u>Only if</u>** part) If a path between S_i and T_j exists, it must be passing through one of the middle boxes. That middle box can be reached from S_i and from that middle box T_j can also be reached. So,

$$BL(i \leftrightarrow j) \le N/2 - 1 < N/2.$$ ◆

Definition : Let P and Q be two non-negative integers, with their n-bit binary representations as $P = p_{n-1}\, p_{n-2} \cdots p_0$ and $Q = q_{n-1}\, q_{n-2} \cdots q_0$. Then the distance between P and Q is defined as

$$d(P, Q) = i \text{ so that } p_i \ne q_i \text{ and } p_j = q_j \text{ for all } i+1 \le j \le n-1.$$

Example 8.2 :

$$d(3, 4) = d(011, 100) = 2$$
$$d(6, 7) = d(110, 111) = 0.$$

Definition : Among the middle boxes, we define a group of 2^k boxes, $k = 0, 1, \ldots, n-1$, as the boxes $\{M_t \mid (i-1)2^k \le t < i\, 2^k\}$ for $i = 1, 2, \ldots, 2^{n-k-1}$.

Example 8.3 : For $N = 16$, (M_0, M_1), (M_2, M_3), (M_4, M_5), (M_6, M_7) form the groups of size 2, (M_0, M_1, M_2, M_3) and (M_4, M_5, M_6, M_7) are the groups of size 4.

Remark : It is clear from the definition that different groups of same size are non-overlapping and taken together they span the complete set of permutations.

Definition : By a group of size 2^k around M_m, we shall refer to only that group of middle boxes of size 2^k which contains M_m.

Lemma 8.4 : The group of size 2^k around M_m is $\{M_t \mid \lfloor m/2^k \rfloor 2^k \leq t < (\lfloor m/2^k \rfloor + 1)2^k\}$.

Proof : The result follows from the fact that $\lfloor m/2^k \rfloor 2^k \leq m < (\lfloor m/2^k \rfloor + 1)2^k$. ◆

Lemma 8.5 : M_i and M_j belong to the same group of size 2^k iff (if and only if) $d(i, j) < k$.

Proof : The group of 2^k around M_i and M_j are $\{M_t \mid \lfloor i/2^k \rfloor 2^k \leq t < (\lfloor i/2^k \rfloor + 1)2^k\}$ and $\{M_t \mid \lfloor j/2^k \rfloor 2^k \leq t < (\lfloor j/2^k \rfloor + 1)2^k\}$. M_i and M_j belong to the same group of size 2^k

iff the groups of size 2^k around M_i and M_j are the same,
i.e., iff $\lfloor i/2^k \rfloor = \lfloor j/2^k \rfloor$,
i.e., iff the most significant $(n - k)$ bits of i and j are identical,
i.e., iff $d(i, j) < k$. ◆

Lemma 8.6 : For $x \leq y$, the group of size 2^x around M_m is contained in the group of size 2^y around M_m.

Proof : The result follows from lemma 8.5. ◆

Definition : By *Reachability Tree* of an input S_i, we shall mean the complete binary tree formed by the all possible lines going out of S_i and the relevant switches, upto the middle stage. The root of the tree is a switch in the first stage and the switches in the middle stage are the leaves. The different levels of the tree are numbered from 0 to $n - 1$ with the root at level 0. The subtrees of any node are called *top* and *bottom subtrees* as they appear in the standard drawing of a Benes network. Similarly, we can define the reachability tree of an output line.

Example 8.4 : Figure 8.2 shows an example of the reachability trees of the input lines S_2 and S_3 for $N = 8$.

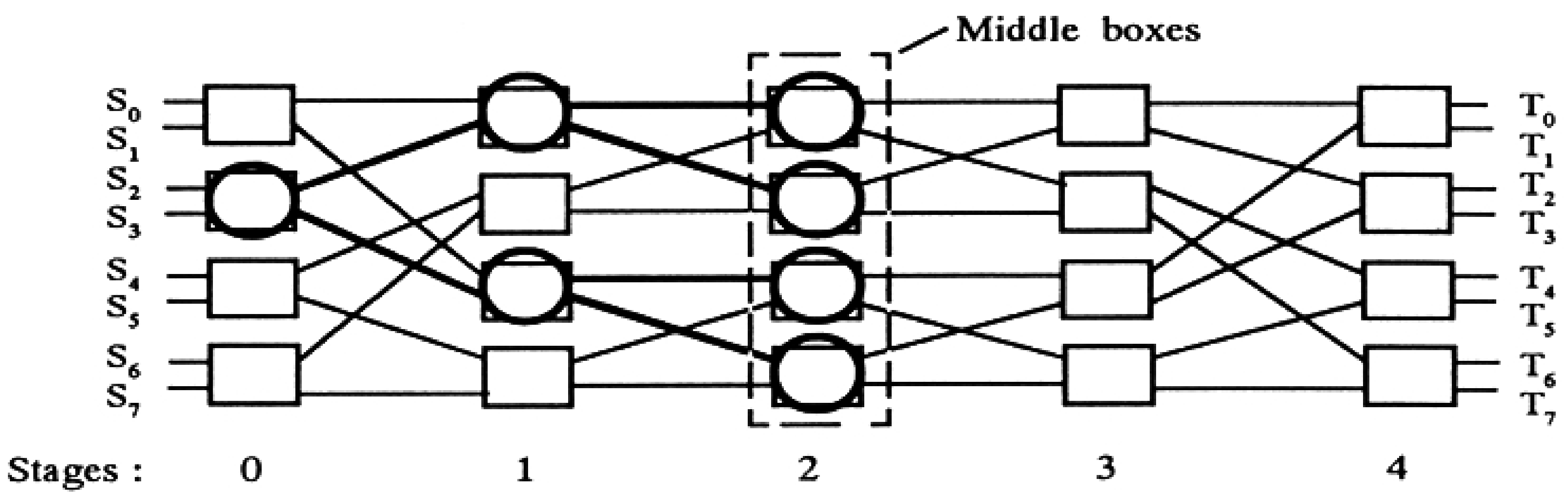


Figure 8.2 : The *Reachability Tree* of S_2 / S_3

Lemma 8.7 : The subtrees of the reachability trees of S_i and S_k are identical from level d(i, k) onwards.

Proof : It is easy to see from the structure of the network that the switches reachable in the level d(i, k) from S_i and S_k are same. Hence the result. ◆

Lemma 8.8 : The leaves of a subtree of a node at level k ($0 \leq k \leq n-2$) of reachability tree form a group of size $N/2^{k+2}$.

Proof: The proof follows from the structure of the reachability tree. ◆

Lemma 8.9 : Let P be a path between S_k and T_r and passing through M_m. Then the middle boxes which can not be reached from S_i because of P are

i) the group of size $N/2^{d(i, k)+2}$ if $d(i, k) < n-1$

ii) $\varnothing$ if $d(i, k) = n-1$.

Proof : The reachability trees of S_i and S_k come together at level d(i, k). So the path between S_k and T_r intersects the reachability tree of S_i at level d(i, k) and proceeds through one of the children of that node. Thus, that particular subtree goes beyond reach from S_i because of P. Now, the result follows from lemma 8.8. ◆

Lemma 8.10 : Let P be a path between S_k and T_r and passing through M_m. Then the middle boxes from which T_j can not be reached because of P are

i) the group of size $N/2^{d(j, r)+2}$ if $d(j, r) < n-1$

ii) $\varnothing$ if $d(j, r) = n-1$.

Proof: Similar to that of lemma 8.9. ◆

Now, combinining lemmas 8.9 and 8.10 we get,

Theorem 8.4 : Let P be a path from S_k to T_r. Then

$$bl(P, i \leftrightarrow j) = \max(\lfloor N/2^{d(i,k)+2} \rfloor, \lfloor N/2^{d(j,r)+2} \rfloor). \qquad \blacklozenge$$

Without loss of generality, let us assume that we are trying to set a path between S_0 and T_0. Then the number of middle boxes that may be blocked for S_0 are as follows :

Existing path from	Number of middle boxes blocked
S_1	2^{n-2}
S_2 / S_3	2^{n-3}
$S_4 / S_5 / S_6 / S_7$	2^{n-4}
$S_{N/2} / \ldots / S_{N/2-1}$	1
$S_{N/2} / \ldots / S_{N-1}$	0

Existing path to	Number of middle boxes blocked
T_1	2^{n-2}
T_2 / T_3	2^{n-3}
$T_4 / T_5 / T_6 / T_7$	2^{n-4}
$T_{N/4} / \ldots / T_{N/2-1}$	1
$T_{N/2} / \ldots / T_{N-1}$	0

8.3.2 Multi-Layered Benes Network

Now suppose we have multiple layers (say, p) of Benes network. In this section we try to find out the minimum number of layers that will make the network non-blocking in the strict sense.

Notation : Let $BL_i(0 \leftrightarrow 0)$ denote the blockage to the path from S_0 to T_0 in the i-th layer.

The sum of total blockages over all the layers is

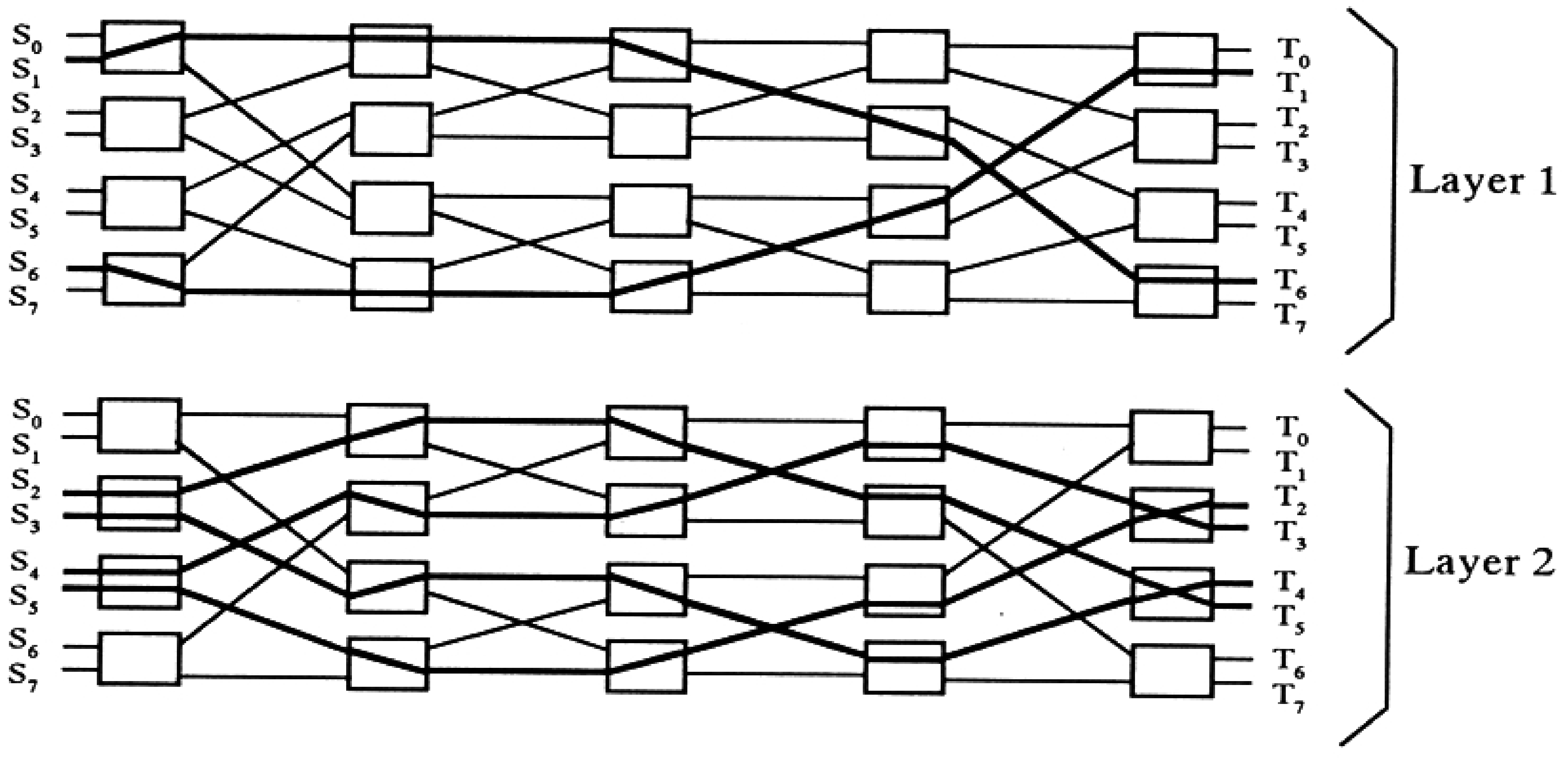


Figure 8.3 : An example showing that (log N – 1) layers of Benes network can be blocking, for N = 8

$$
\begin{aligned}
& BL_1(0 \leftrightarrow 0) + BL_2(0 \leftrightarrow 0) + \ldots + BL_p(0 \leftrightarrow 0) \\
\leq\ & 2\,(1.\,N/4 + 2.\,N/8 + 4.\,N/16 + \ldots + N/4.\,1 + N/2.\,0) \\
=\ & 2\,(n-1)\,N/4 \\
=\ & (n-1)\,N/2
\end{aligned}
$$

Now, if there are at least n layers, then in at least one of the layers, say the t-th layer, $BL_t(0 \leftrightarrow 0) < N/2$. Hence a path between S_0 and T_0 exists in the t-th layer.

Remark : Notice that we have not assumed any specific routing algorithm to route the previous paths. So we can claim that *in an n-layer N×N Benes network a path between a free source and a free destination always exists, irrespective of the routing strategy.*

Now a question that obviously comes to our mind is : *'Can we have the same property for less than n layers?'* The answer is : *'No'*. Let us consider the following example with N = 8.

Example 8.5 : If the existing connections are as shown in figure 8.3, then $S_0 \leftrightarrow T_0$ is blocked in both the layers.

In general, we may construct the example as follows :

Layer 1 : $S_1 \leftrightarrow T_{N-2}$ through upper half in the middle stage.
$S_{N-2} \leftrightarrow T_1$ through lower half in the middle stage.

Layer 2 : $S_2 \leftrightarrow T_{N-3}$ through first quarter in the middle stage.
$S_3 \leftrightarrow T_{N-4}$ through third quarter in the middle stage.
$S_{N-4} \leftrightarrow T_3$ through second quarter in the middle stage.
$S_{N-3} \leftrightarrow T_2$ through fourth quarter in the middle stage.

Layer 3 : $S_4 \leftrightarrow T_{N-5}$ through first octant in the middle stage.
$S_5 \leftrightarrow T_{N-6}$ through fifth octant in the middle stage.
$S_6 \leftrightarrow T_{N-7}$ through third octant in the middle stage.
$S_7 \leftrightarrow T_{N-8}$ through seventh octant in the middle stage.
$S_{N-8} \leftrightarrow T_7$ through second octant in the middle stage.
$S_{N-7} \leftrightarrow T_6$ through sixth octant in the middle stage.
$S_{N-6} \leftrightarrow T_5$ through fourth octant in the middle stage.
$S_{N-4} \leftrightarrow T_4$ through eighth octant in the middle stage.

Similarly we can construct the connections for the remaining layers.

It can easily be verified that $BL_i(0 \leftrightarrow 0) = N/2$ for $i = 1, 2, \dots, n-1$.

Thus we have the following theorem :

Theorem 8.5 : $\log_2 N$ layers are necessary and sufficient to make a multi-layered Benes network non-blocking in the strict sense. ◆

Remark : Here we note that $\log_2 N$ layers of Benes network is nothing but the Cantor network. Thus in effect we have shown that the minimum layer, strictly non-blocking multi-layered Benes network is the Cantor network.

8.3.3 Multi-Layered $(2 \log_2 N - 1)$-Stage Shuffle-Exchange Network

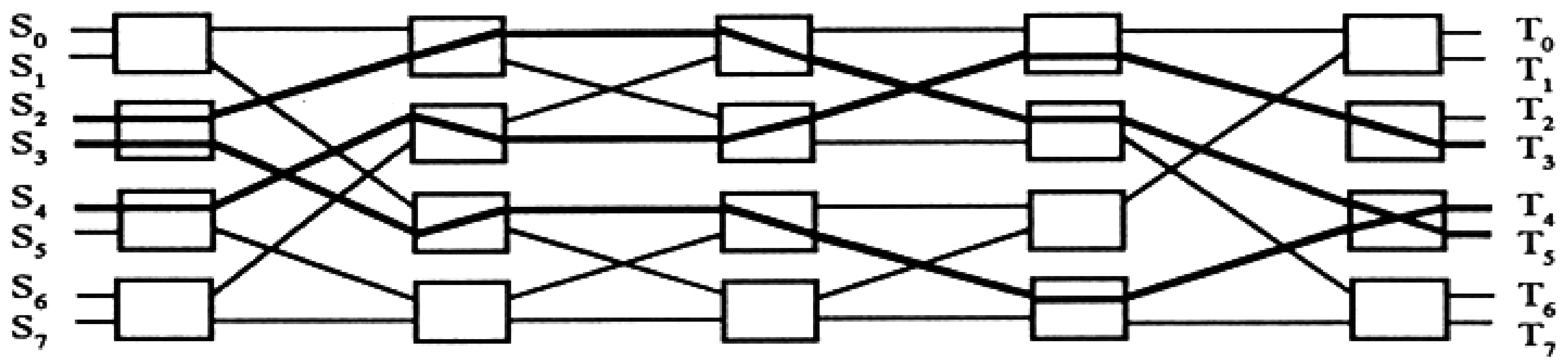
As we have discussed for Benes network, we can similarly define middle boxes and reachability trees for a $(2 \log_2 N - 1)$-stage shuffle-exchange network. The difference will be that the grouping of the middle boxes will be different depending on whether we are looking at them from the input or the output side. But, the logic for the amount of blockage due to different paths will be similar. We state the following theorem without proof.

Theorem 8.6 : $\log_2 N$ layers are necessary and sufficient to make a multi-layered $(2\log_2 N - 1)$-stage shuffle-exchange network routing-independent non-blocking. ◆

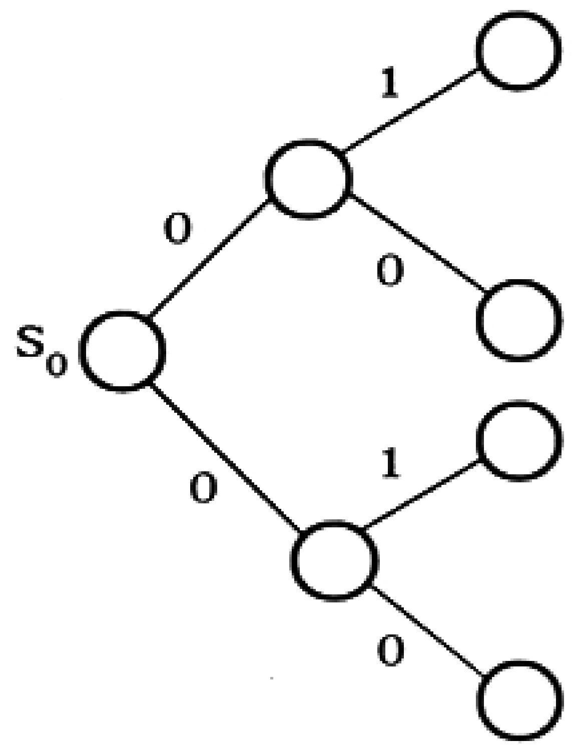
8.4 Routing in Multi-Layered Benes Network

As we have already seen, $\log_2 N$ layers of $N \times N$ Benes network is non-blocking. So the existence of a path between a free source and a free destination is always guaranteed in one of the layers. But how do we find such a path? In each layer we find out whether a path between the required source and destination exists or not. This can be done in parallel over the layers. So, the problem of finding a path in a multi-layered Benes network boils down to that of finding out whether in a particular layer a path between the required source-destination pair exists or not. The information we shall need is about the existing paths in that layer.

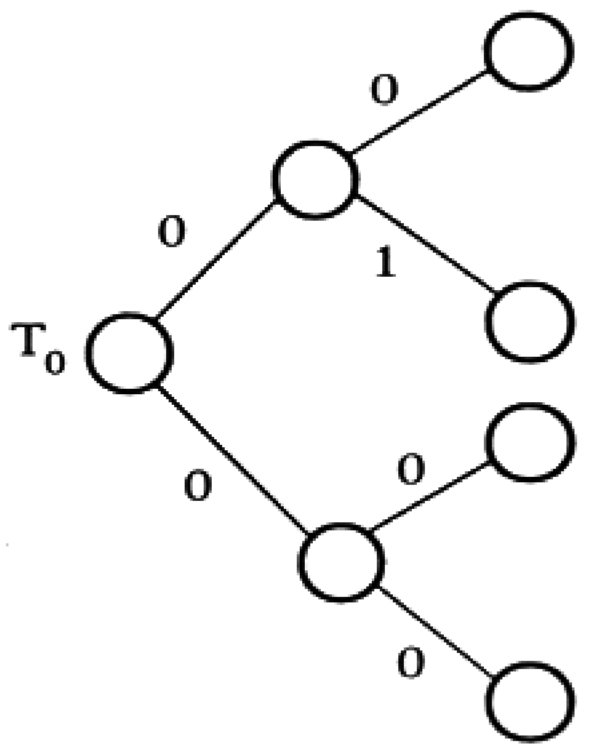
We shall keep the information regarding the blockage in a layer, in terms of the reachability trees of different sources and destinations. Suppose we want to set a path between S_i and T_j. Any existing path will intersect the reachability tree of S_i at certain level and block one of the subtrees under that level. For each node of the reachability tree (excepting the leaves), we shall keep 2 bits of information, about whether the top and the bottom subtrees of that node are blocked or not. If a subtree is blocked, the corresponding bit is set to '1' else to 0. We can get a path between S_i and T_j, if a middle box is reachable from both S_i and T_j. In order to impose this condition that the middle box should be reachable from both the source and the destination, we take the reachability trees of S_i and T_j. The two trees are identical in structure. We generate a new tree by 'OR'ing the informations in the corresponding nodes in the two trees. Now, if we can find a path from the root to a leaf, the middle box corresponding to that leaf gives us the required path. In order to make the searching back-track free, we first the preprocess the resulting tree as follows. Starting from the leaf level, we make the value corresponding to a subtree 1, if both the children of that subtree are blocked. After the preprocessing is done we OR the two bits at the root. If the resultant bit is 1, there is no path available. If the result is 0, at least one bit must be 0. We go to one subtree which is not blocked. Since the value at that the root of that subtree is 0, at at least one subtree should not be blocked. Proceeding in this fashion, we can reach the root. Figure 8.4, shows an example of the above procedure.



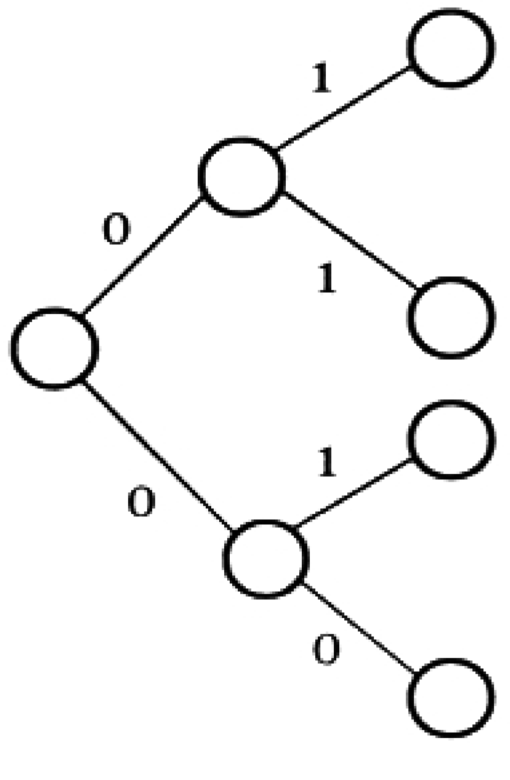
a) Diagram showing the existing paths in one layer of Benes network



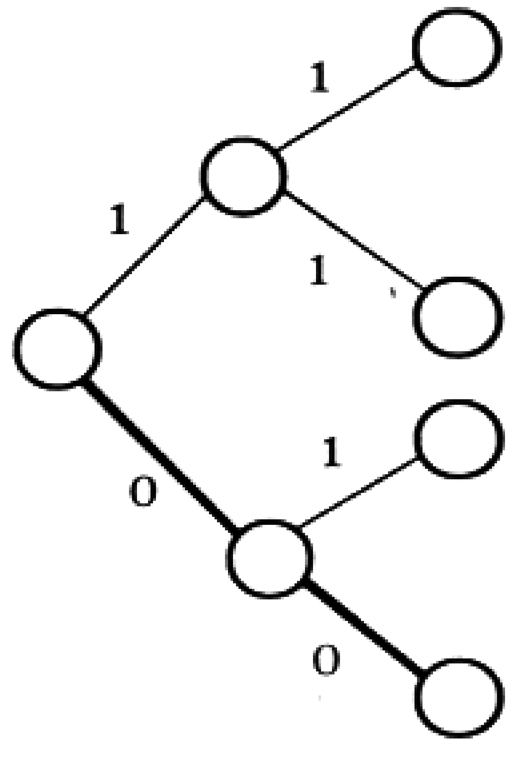
b) Reachability tree of S_0



c) Reachability tree of T_0



d) Resultant tree



e) After preprocessing

Figure 8.4 : An example illustrating the basic algorithm for finding a path in one layer of Benes network

We observe that the reachability trees of different sources intermingle with one another and all taken together topologically they form the first half of the Benes network, i.e., the baseline network. Similarly, the reachability trees of the different destinations form the reverse baseline network. So, instead of keeping the information corresponding to each reachability tree, we may keep, the information corresponding to each node to indicate whether the top and the bottom subtrees of that node are blocked or not. The whole information for the reachability trees for the different sources may be kept in a matrix $[a_{ij}]$, $i = 0, \ldots, n-1$, $j = 0, \ldots, N-1$, such that the i-th row in the matrix corresponds to the information regarding the switches in stage i of the network. The 0-th row contains 2 bits of information corresponding to the root node of every reachability tree, i.e., 2 bits for each group of 2 inputs. The 1-st row will contain 4 bits corresponding to each group of size 4, i.e., the first level of the reachability trees of the two input lines of every switch. In general, we have the information regarding the level j of the reachability tree of S_i, in the elements a_{jk} through a_{jt} where $k = 2^{j+1}[i \text{ DIV } 2^{j+1}]$ and $t = k + 2^{j+1} - 1$. Note that a similar matrix $B = [b_{ij}]$ may be kept corresponding to the reachability trees from the destination side.

In actual implementation, we store the matrices A and B in two banks of circulating shift registers. Figure 8.5 shows a schematic diagram of the routing hardware. The registers R_0, ..., $R_{\log N-2}$ hold the information about the resultant reachability tree after 'OR'ing the reachability trees for the inputs and outputs. R_i is a 2^{i+1}-bit register holding the information about the nodes at level i of the reachability tree. The registers will be set by appropriately gating the clock pulses. C_i's are the outputs of a modulo $(\log_2 N - 1)$ counter. C_i takes value 1 during the interval of counting the i-th pulse and the (i + 1)-th pulse by the counter.

8.5 Layered Baseline Network

We define the groups of input lines and groups of output lines in the same way as the groups of middle boxes defined earlier. The reachability tree is also defined in the same way as in the case of Benes network. From the structure of the network, it is easy to see that two inputs in the same group of 2^1 should be connected to two outputs belonging to two different groups of size N/2. Again, two inputs from the same group of 2^2 (and <u>not</u> same 2^1) should go to outputs in different groups of size N/4. In general we can say,

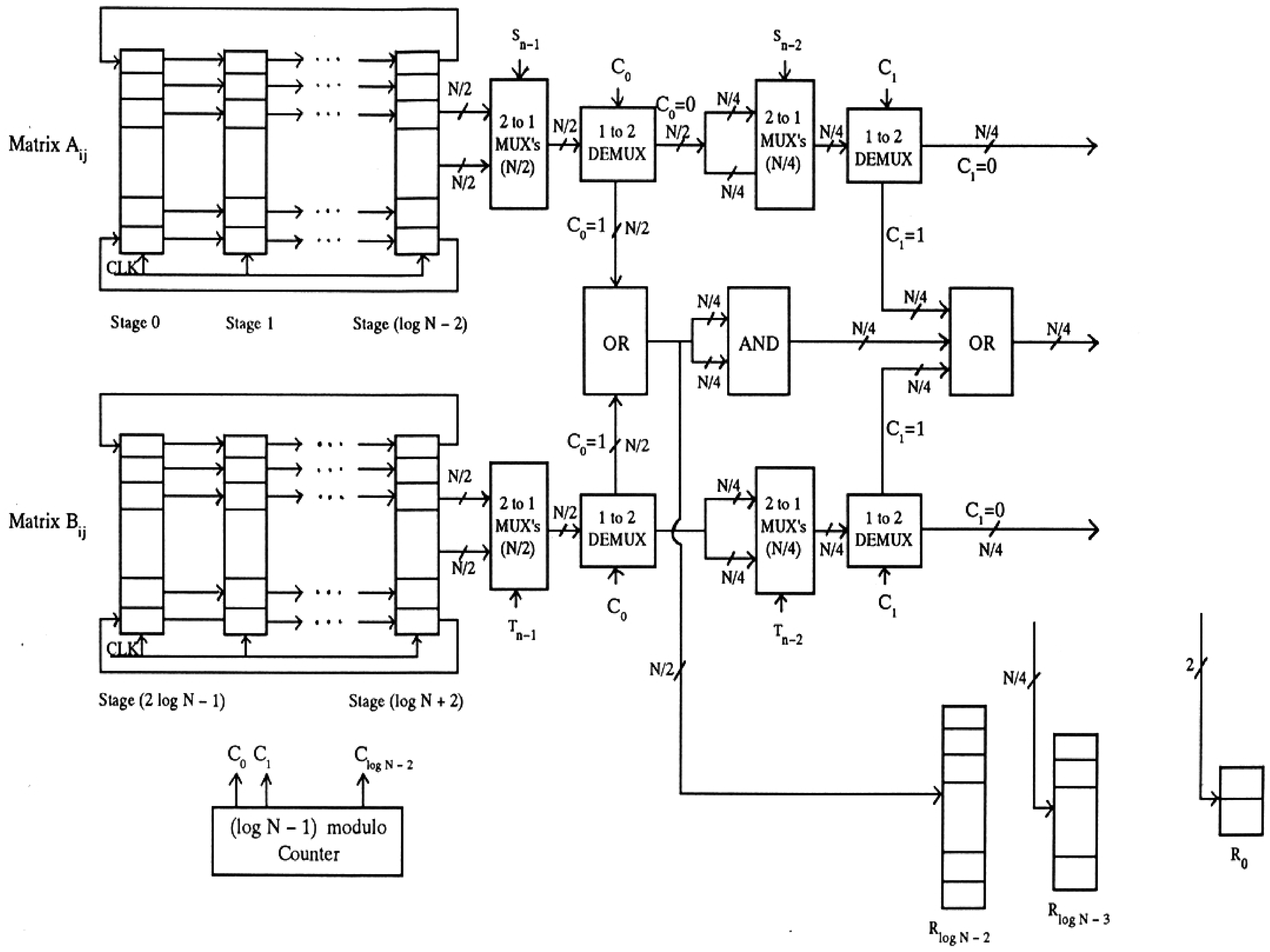


Figure 7.5 : Schematic Diagram of the Routing Hardware for one layer of Cantor Network

Lemma 8.11 : Paths between $S_i \leftrightarrow T_j$ and $S_k \leftrightarrow T_r$ are conflicting if and only if $d(i, k) + d(j, r) < n - 1$.

Proof : The reachability trees from S_i and S_k meet at level $d(i, k)$. So the path from S_k blocks a group of $N/2^{d(i,k)+2}$ switch boxes at the last stage, i.e., $N/2^{d(i,k)+1}$ output-lines for S_i and vice versa. So the paths will be conflicting if and only if T_j and T_r belong to the same group of $N/2^{d(i,k)+1}$ $(= 2^{n-d(i,k)-1})$
i.e., $d(j, r) < n - d(i, k) - 1$
i.e., $d(i, k) + d(j, r) < n - 1$. ◆

Without loss of generality, let us assume that we are trying to set up a path between S_0 and T_0. An existing path between S_i and T_j will block the required path if $d(i, 0) + d(j, 0) < n - 1$. Now we try to maximize the number of conflicting paths. So we try to sets of S_x and T_y such that $d(x, 0) + d(y, 0) = n - 2$. Now, if $d(x, 0) = k$, then there are 2^k many possible values of x. Again we have $d(j, 0) = n - k - 2$. So number of possible values of y is 2^{n-k-2}. Since we have to choose one x and one y for a path, the number of conflicting paths with $d(i, 0) = k$ is $\min(2^k, 2^{n-k-1})$. Varying over different possible values of k, we find the number of paths conflicting with $S_0 \leftrightarrow T_0$ as

$$\sum_{k=0}^{n-2} \min(2^k, 2^{n-k-2}).$$

The above sum can be simplified as follows :

CASE I : n is even (n = 2m).

$$\begin{aligned}\sum_{k=0}^{n-2} \min(2^k, 2^{n-k-2}) &= 1 + 2 + \ldots + 2^{m-2} + 2^{m-1} + 2^{m-2} + \ldots + 1\\ &= 2\,(2^m - 1) - 2^{m-1}\\ &= 3.\,2^{m-1} - 2.\end{aligned}$$

CASE II : n is odd (n = 2m + 1)

$$\begin{aligned}\sum_{k=0}^{n-2} \min(2^k, 2^{n-k-2}) &= 1 + 2 + \ldots + 2^{m-2} + 2^{m-1} + 2^{m-1} + 2^{m-2} + \ldots + 1\\ &= 2^{m+1} - 2.\end{aligned}$$

This gives us the maximum number of paths that may be conflicting with $S_0 \leftrightarrow T_0$, which, in turn, is the maximum number of layers that may be blocked in a multi-layered baseline network. So if we have one more layer, we can always guarantee the availability of at least one path. Hence, we have the following theorem :

Theorem 8.7 : L layers are sufficient to make a multi-layered N×N ($N = 2^n$) baseline network non-blocking in the strict sense, where

$$L = \begin{cases} 3 . 2^{m-1} - 1 & \text{if } n = 2m \\ 2^{m+1} - 1 & \text{if } n = 2m+1. \end{cases}$$ ◆

Remark: It is also quite clear from the above discussion that L layers are necessary, if we want to make the network non-blocking independent of the routing strategy.

8.6 Conclusion

In this chapter, we have analyzed blocking / non-blocking behavior of MIN's. We have introduced the concept of routing-dependent MIN's. We have shown that in single-layered MIN's, it is not possible to construct networks non-blocking in the strict sense, and a single-layered routing-dependent non-blocking MIN has cost $\Omega(N^2)$. In multi-layered strictly non-blocking MIN's, we have found out for general values of N, the minimum number of layers required for making multi-layered Benes, ($2 \log_2 N - 1$)-stage shuffle-exchange and baseline network non-blocking in the strict sense. Further work in this line may be carried out on the following problems : 1) finding routing-dependent multi-layered networks, and 2) finding the minimum number of layers required to make a general k-stage shuffle-exchange network non-blocking in the strict sense and optimizing the cost over the values of k.

Chapter 9
Conclusion

An interconnection network plays a very vital role in the recent age of fast and efficient computing. The speed and efficiency of the network depend heavily on the interconnection network chosen. In this thesis, we have discussed many aspects of the interconnection networks. For both static and dynamic interconnection networks, we have addressed some of the problems of recent research interests among the present-day computer scientists.

In the area of static interconnection networks, we have first discussed some problems related to the distributed loop networks. For a given value of the number of nodes, we have suggested some criteria for the design of loop networks, having minimum diameter over the different choices of jump-size. We have also proposed an algorithm for routing a message from one node to another, along a shortest path. We have also shown that our proposed algorithm gives a path of length not more than one over the optimal, in the event of a single node or link failure. Further research in this area may be carried out in the following directions :

i) Deriving an analytical formula for the diameter of $G(N; 1, s)$,
ii) Design of optimal loop networks for all values of N,
iii) Optimal routing under single as well as multiple faults,
iv) Analysis of generalized loop networks $G(N; 1, s_1, s_2, \dots, s_k)$ etc.

One of the most important and popular static network topologies is the binary hypercube. In this thesis, we have proposed some modifications over the hypercube connection pattern, so as to decrease the diameter but maintaining the other desirable properties. We have shown that by addition of some extra links (bridges) or by exchanging some of the link pairs (twists), we can reduce the diameter of the hypercube upto half of its original value. The number of bridges added or the number of link pairs exchanged is pretty small, compared to the total number of links in the graph. We have also proposed algorithms for node-to-node routing in bridged and twisted hypercubes. Other types of routing, viz., a) broadcast, b) scattering, and c) all-pair-broadcast, in the bridged and twisted hypercubes, constitute interesting topics of further studies. Design of efficient algorithms for solving various real-life problems on the bridged or twisted hypercubes will also be an interesting area of research.

For some applications it may be needed that a hamiltonian cycle be present in the network topology. For such purposes, ring network is an ideal topology. But, in case of a node or a link failure, the ring ceases to be hamiltonian. We have proposed a new network topology, that is hamiltonian, and remains hamiltonian even in the presense of a single node or single link failure. Among all networks which are 1-node-deleted-hamiltonian and 1-link-deleted-hamiltonian, our proposed topology has the minimum number of links. In this direction, one may try to find network graphs, which have certain pre-specified topologies (say, hypercube or complete binary tree) in the presence of a single fault or, in general, k faults.

For the analysis of a network topology, measure of its fault-tolerance is of vital interest. Fault-tolerance of a network graph is measured either by the connectivity or by using a stochastic fault model. We have found a recursive formla for the evaluation of the network reliability under a stochastic model. We have also derived analytical expressions for the reliabilities of some standard network graphs.

In the area of dynamic interconnection networks, Benes network is a very popular multistage interconnection network. It has a propagation delay of $O(\log_2 N)$. Compared to this, the standard looping algorithm for routing a permutation in the Benes network takes $O(N \log_2 N)$ time. But some permutations are self-routable, i.e., their switch settings can be achieved in a distributed manner, with every switch being set independently depending only on its two inputs. We have considered four classes of self-routing algorithms and have discussed some of their properties. We have classified, to some extent, the permutations routable by them. We have also extended the self-routing strategy to cover a large set of permutations called the BPCL class of permutations. The complete classification of the self-routable permutations is an important direction for further work. Finding self-routing algorithms, for permutations thus far not covered, is another topic for future research.

Finally, for building networks which are non-blocking in the strict sense, we have proposed multiple layers of different standard MIN's (Benes, $(2n-1)$-stage shuffle-exchange, baseline). We have calculated the minimum number of layers that will make them non-blocking in the strict sense. Multilayered Benes network, optimized over the number of layers, turns out to be the existing Cantor network. We have proposed a routing hardware for the routing in the Cantor network. Minimizing the number of layers for k-stage shuffle-exchange network and then minimizing the total number of switches over k is an interesting unsolved problem. One may also try to design routing-dependent multilayered MIN's.

References

[AA80] M. A. Abidi and D. P. Agrawal, ''On conflict-free permutations in multistage interconnection networks,'' *J. Digital System,* vol. V, pp. 115-136, Summer 1980.

[AC90] H. M. AboElFotoh and C. J. Colbourn, ''Efficient algorithms for computing the reliability of permutation and interval graphs,'' *Networks*, vol. 20, pp. 883 - 898, 1990.

[AF92] A. Abdennadher and T. Feng, ''On rearrangeability of omega-omega networks,'' in *Proc. 1992 International Conference on Parallel Processing*, pp. I-159 - I-165, 1992.

[AG81] J. R. Armstrong and F. G. Gray, ''Fault diagnosis in a boolean cube of microprocessors,'' *IEEE Trans. Comput.,* vol. C-30, no. 8, pp. 587 - 590, August 1981.

[Ag82] D. P. Agrawal, ''Testing and fault tolerance of multistage interconnection networks,'' *Computer,* vol. 15, no. 4, pp. 41-53, April 1982.

[Ag83] D. P. Agrawal, ''Graph theoretical analysis and design of multistage interconnection networks,'' *IEEE Trans. on Computers*, vol. C-32, pp. 637-648, July 1983.

[AKS88] D. P. Agrawal, S.C. Kim and N.K. Swain, ''Analysis and design of non-equivalent multistage interconnection networks,'' *IEEE Trans. on Computers*, vol. C-37, pp. 232-237, Feb. 1988.

[AL81] B. W. Arden and H. Lee, ''Analysis of chordal ring,'' *IEEE Trans. Comput.*, vol. C-30, pp. 291 - 292, 1981.

[AL90] A. E. Amawy and S. Latifi, ''Bridged hypercube networks,'' *Journal of Parallel and Distributed Computing.,* pp. 90-95, September 1990.

[AP89] S. Abraham and K. Padmanabhan, ''Performance of direct binary n-cube network for multiprocessors,'' *IEEE Trans. Comput.,* vol. C - 38, no. 7, pp. 1000 - 1011, July 1989.

[AP91] S. Abraham and K. Padmanabhan, ''Twisted cube : a study in asymmetry,'' *Journal of Parallel and Distributed Computing.* vol. 13, pp. 104 - 110, November 1991.

[AR81] K. K. Aggarwal and S. Rai, ''Reliability evaluation in computer communication networks,'' *IEEE Trans. Reliability,* vol. R-30, pp. 32-35, Jan. 1981.

[AS82] G. B. Adams III and H. J. Siegel, ''On the number of permutations performable by the ADM network,'' *IEEE Trans. on Computers*, vol. C-31, pp. 270-277, Apr. 1982.

[Ba68] K. E. Batcher, ''Sorting networks and their applications,'' *1968 Spring Joint Computer Conf., AFIPS Proc.*, vol. 32, Washington D. C. : Thompson, pp. 307-314, 1968.

[Ba76] K. E. Batcher, ''The flip network in STARAN,'' in *Proc. 1976 International Conference on Parallel Processing,* pp. 65-71, 1976.

[Ba80] M. O. Ball, ''Complexity of network reliability computations,'' *Networks,* vol. 10, pp. 153 - 165, 1980.

[BC89] T. B. Brecht and C. J. Colbourn, ''Multiplicative improvements in network reliability bounds,'' *Networks,* vol. 19, pp. 521-529, 1989.

[Be62] V. E. Benes, ''On rearrangeable three-stage connecting networks,'' *The Bell Systems Technical Journal,* vol. 41, no. 5, pp. 1481 – 1492, September 1962.

[Be65] V. E. Benes, *Mathematical Theory of Connecting Networks,* Academic Press, New York, 1980

[BOS⁺91] D.P. Bertsekas, C. Ozveren, G.D. Stamoulis, P. Tseng and J.N. Tsitsiklis, ''Optimal communication algorithms for hypercubes,'' *Journal of Parallel and Distributed Computing,* vol. 11, pp. 263 - 275, April 1991.

[BP82] M. O. Ball and J. S. Provan, ''Bounds on the reliability polynomial for shellable independence systems,'' *SIAM J. Alg. Disc. Methods*, vol. 3, pp. 166 - 181, 1982.

[Br46] N. G. de Bruijn, ''A combinatorial problem,'' *Koninklijke Nederlande Academie van Watenschapen Proc.,* vol. A49, pp. 758 - 764, 1946.

[BR88] R. Boppana and C. S. Raghavendra, ''On self-routing in Benes and shuffle exchange networks,'' in *Proc. International Conference on Parallel Processing*, pp. 196-200, Aug. 1988.

[BR91] P. J. Bernherd and D. J. Rosenkrantz, ''An efficient method for representing and transmitting message patterns on multiprocessor interconnection networks,'' *J. Parallel and Distributed Comput.*, vol. 11, pp. 72-85, 1991.

[BS84] R. A. Boyles and F. J. Samaniego, ''Modeling and inference for multivariate binary data with positive dependence,'' *J. Am. Stat. Assoc.* vol. 79, pp. 188-193, 1984.

[BT84] F. T. Boesch and R. Tindell, ''Circulants and their connectivities,'' *J. Graph Theory*, vol. 8, pp. 487-499, 1984.

[BT91] J. -C. Bermond, D. Tzvieli, ''Minimal diameter double-loop networks. II. dense optimal families,'' *Networks*, vol. 21, no. 1, pp. 1 - 10, January, 1991.

[BW85] F.T. Boesch and J.-F. Wang, ''Reliable circulant networks with minimum transmission delay,'' *IEEE Trans. Circuits and Systems*, vol. CAS-32, pp. 1286 - 1291, 1985.

[Ca71] D. G. Cantor, ''On non-blocking switching networks,'' *Networks*, vol. 1, no. 4, pp. 367-377, winter 1971.

[CH88] C. J. Colbourn and D. D. Harms, ''Bounding all-terminal reliability in computer networks,'' *Networks*, vol. 18, pp. 1-12, 1988.

[Cl53] C. Clos, ''A study of non-blocking switching networks,'' *The Bell Systems Technical Journal*, vol. 32, pp. 406 – 424, March 1953.

[Co87a] C. J. Colbourn, *The Combinatorics of Network Reliability*, Oxford University Press, 1987.

[Co87b] C. J. Colbourn, ''Network resilience,'' *SIAM J. Alg. Disc. Methods*, vol. 8, pp. 404 - 409, 1987.

[Da79] P. J. Davis, *Circulant Matrices*, John Wiley, New York (1979).

[DBD90] N. Das, B. B. Bhattacharya and J. Dattagupta, ''Analysis of conflict graphs in multistage interconnection networks,'' in *Proc. IEEE Region 10 Conference on Computer and Communication Systems*, pp. 175-179, Sept. 1990.

[DF87] J. S. Deogun and Z. Fang, ''A heuristic algorithm for conflict resolution problem in multistage interconnection networks,'' *Proc. 1987 International Conference on Parallel Processing*, pp. 475-478, Aug. 1987.

[DHL⁺90] D.-Z. Du, D.-F. Hsu, Q. Li and J. Xu, ''A combinatorial problem related to distributed loop networks,'' *Networks*, vol. 20, pp. 173 - 180, 1990.

[EH91] T. Egeland and A. B. Huseby, ''On dependence and reliability computations,'' *Networks*, vol 21, no. 5, pp. 521-546, Aug 1991.

[El92] E. S. Elmallah, ''Algorithms for K-terminal réliability problems with node failures,'' *Networks*, vol. 22, pp. 369 - 384, 1992.

[ENS91] A. Esfahanian, M. L. Ni and B. E. Sagan, ''The twisted *N*- cube with application to microprocessing,'' *IEEE Trans. on Comput.*, vol. C-40, no. 1, Jan. 1991, pp. 88-93.

[Fe74] T. Feng, ''Data manipulating functions in parallel processors and their implementations,'' *IEEE Trans. on Computers,* vol. C-23, pp. 309-318, March 1974.

[Fe81] T. Feng, ''A survey of interconnection networks,'' *Computer*, vol 14, no. 12, pp. 12 – 27, Dec. 1981.

[FW81] T. Feng and C. Wu, ''Fault diagnosis for a class of multistage interconnection networks,'' *IEEE Trans. Comput.,* vol. C-30, no. 10, pp. 743-758, Oct. 1981.

[GG81] A. Grnarov and M. Grela, ''Multiteminal reliability analysis of distributed processing systems,'' in *Proc. 1981 International Conference on Parallel Processing*, pp. 79-86, Aug, 1981.

[GL73] L. R. Goke and G. J. Lipovoski, ''Banyan networks for partitioning multiprocessor systems,'' *Proc. of 1st. Annual Symp. on Comp. Architecture*, pp. 21-28, 1973.

[Ha69] F. Harary, *Graph Theory*, New York : Addison-Wesley, 1969.

[HKS87] P. A. J. Hilberts, M. R. J. Koopman and J. L. A. van de Snepscheut, ''The twisted cube,'' *Parallel Architectures and Languages Europe,* Lecture notes in Computer Science, June 1987, pp. 152 -159.

[HT86] S. T. Huang and S. K. Tripathy, ''Finite state model and compatibility theory : New analysis tools for permutation networks,'' *IEEE Trans. on Computers,* vol. C-35, pp. 591-601, July 1986.

[Hu86] A. B. Huseby, ''A unified theory of domination and signed domination with application to reliability theory,'' *Research Report, Center for Industrial Research, Oslo, Norway,* 1986.

[Já92] J. JáJá, *An Introduction to Parallel Algorithm.* Reading, Massachusetts : Addision-Wesley Publishing Company, 1992.

[Ka88] H. P. Katseff, ''Incomplete hypercubes,'' *IEEE Trans. on Computers,* vol. C-37, no. 5, pp. 604 - 608, May 1988.

[KA90] A. Kumar and D. P. Agrawal, ''Computer network reliability evaluation from application's point of view,'' in *Proc. 1990 International Conference on Parallel Processing,* pp. 552-555, August 1990.

[La75] D. H. Lawrie, ''Access and alignment of data in an array processor,'' *IEEE Trans. on Comput.*, vol. C-24, no. 12, pp. 1145-1155, Dec. 1975.

[Le78] J. Lenfant, ''Parallel permutations of data : A Benes network control algorithm for frequently used permutation,'' *IEEE Trans. Comput.*, vol. C-27, no. 7, pp. 637-647, July 1978.

[Le93] F. T. Leighton, *Parallel Architectures and Algorithms : Arraya, Trees, and Hypercubes.* San Mateo, California : Morgan Kaufmann Publishers, 1993.

[LP71] M. V. Lomonosov and V. P. Polesskii, ''An upper bound for the reliability of information networks'', *Prob. Inf. Trans.*, 7, pp. 337-339, 1971.

[LPV81] G. P. Lev, N. Pippenger and L. G. Valiant, ''A fast parallel algorithm for routing in permutation networks,'' *IEEE Trans. on Computers,* vol. C-30, no. 2, pp. 93-100, Feb. 1981.

[LS82] W. E. Leland and M. H. Solomon, ''Dense trivalent graphs for processor interconnection,'' *IEEE Trans. on Comput.*, vol. C-31, no. 3, pp. 219 - 222, March 1982.

[Ma91] E. Mata-Montero, ''Resilience of partial k-tree networks with edge and node failures,'' *Networks*, vol. 21, pp. 321 - 344, 1991.

[Na89] D. Nassimi, ''A fault-tolerant routing algorithm for BPC permutations on multistage interconnection networks,'' in *Proc. 1989 International Conference on Parallel Processing*, pp. I-278 - I-287, 1989.

[NG90] W. Najjar and J. L. Gaudiot, ''Network resilience : a measure of network fault-tolerance,'' *IEEE Trans. Comput.*, vol. 39, no. 2, pp. 174 - 181, Feb. 1990.

[NS81] D. Nassimi and S. Sahni, ''Data broadcasting in SIMD computers,'' *IEEE Trans. on Computers,* vol. C-30, no. 2, pp. 101-107, Feb. 1981.

[NS81a] D. Nassimi and S. Sahni, ''A self-routing Benes network and parallel permutation algorithms,'' *IEEE Trans. Comput.*, vol. C-30, no. 5, pp. 332-340, May 1981.

[NS82] D. Nassimi and S. Sahni, ''Parallel algorithm to set up the Benes permutation network,'' *IEEE Trans. on Comput.*, vol. C-31, no. 2, pp. 148-154, Feb. 1982.

[OO85] Y. A. Oruc and M. Y. Oruc, ''Equivalence relations among interconnection networks,'' *J. Parallel and Distributed Comput.,* vol. 2, pp. 30-45, Feb. 1985.

[OO85a] Y.A. Oruc and M.Y. Oruc, ''On testing isomorphism of permutation networks,'' *IEEE Trans. on Computers,* vol. C-34, pp. 958-962, Oct. 1985.

[OOB85] Y. A. Oruc, M. Y. Oruc and N. Balabanian, ''Reconfiguration algorithm for interconnection networks,'' *IEEE Trans. on Comput.*, vol. C-34, no. 8, pp. 773-776, Aug, 1985.

[Pa80] D. S. Parker, ''Notes on shuffle/exchange type switching networks,'' *IEEE Trans. on Computers,* vol. C-29, no. 3, pp. 213-221, March 1980.

[Pa81] J. H. Patel, ''Performance of processor-memory interconnection for multiprocessors,'' *IEEE Trans. on Computers,* vol. C-30, no. 10, pp. 771-780, Oct. 1981.

[PB83] J. S. Provan and M. O. Ball, ''The complexity of counting cuts and of computing the probability that a graph is connected,'' *SIAM J. Comput.*, vol. 12, pp. 777 - 788, 1983.

[Pe77] M.C. Pease, ''The indirect binary n-cube microprocessor array,'' *IEEE Trans. on Computers,* vol. C-26, pp. 458-473, May 1977.

[PK80] D.K. Pradhan and K.L. Kodandapani, ''A uniform representation of single and multistage interconnection networks used in SIMD machines,'' *IEEE Trans. on Computers,* vol. C-29, no. 9, pp. 777-790, Sept. 1980.

[Po71] V. P. Polesskii, ''A lower boundary for the reliability of information networks,'' *Prob. Inf. Trans.*, 7, pp. 165-171, 1971.

[PR82] D.K. Pradhan and S.M. Reddy, ''A fault-tolerant communication architecture for distributed systems,'' *IEEE Trans. Comput.*, vol. C-31, pp. 863 - 870, Sept. 1982.

[Pr86] J. S. Provan, ''The complexity of reliability computations on planar and acyclic graphs,'' *SIAM J. Comput.*, vol. 15, pp. 694 - 702, 1986.

[PS87] J. L. Peterson and A. Silberschatz, *Operating System Concepts*, Reading, Massachusetts : Addison-Wesley, 1987.

[PV81] F. P. Preparata and J. Vuillemin, ''The cube-connected-cycles : A versatile network for parallel computation,'' *Communication of the Association for Computing Machinery,* vol. 24, pp. 300 - 309, May 1981.

[Qu87] M. J. Quinn, *Designing Efficient Algorithms for Parallel Computers.* New York : McGraw-Hill, 1987.

[RB91] C.S. Raghavendra and R.V. Boppana, "On self-routing in Benes and shuffle-exchange networks," *IEEE Trans. on Computers,* vol. 40, pp. 1057-1064, Sept. 1991.

[RRK83] S.M. Reddy, P. Raghavan and J.G. Kuhl, ''A class of graphs for processor interconnection,'' in *Proc. International Conference on Parallel Processing,* pp. 154-157, August 1983.

[RV86] C. S. Raghavendra and A. Varma, "Fault-tolerant multiprocessors with redundant path interconnection networks," *IEEE Trans. on Computers,* vol. C-35, pp. 307-316, April 1986.

[RV87] C. S. Raghavendra and A. Varma, "Rearrangeability of the 5-stage shuffle/ exchange network for N=8," *IEEE Trans. on Communication,* vol. COM-35, pp. 808-812, Aug. 1987.

[Se87] A. Seznec, "A new interconnection network for SIMD computers : the sigma network," *IEEE Trans. on Computers,* vol. C-36, pp. 794-801, July 1987.

[SF72] R. M. van Slyke and H. Frank, "Network reliability analysis - I" *Networks,* vol. 1, pp. 279-290, 1972.

[SH87] T. H. Szymanski and V. C. Hamacher, "On the permutation capability of multistage interconnection networks," *IEEE Trans. on Computers,* vol. C-36, pp. 810-822, July 1987.

[Si79] H. J. Siegel, "A model of SIMD machines and a comparison of various interconnection networks," *IEEE Trans. on Computers,* vol. C-28, pp. 907-917, Dec. 1979.

[Si80] H. J. Siegel, "The theory underlying the partitioning of permutation networks," *IEEE Trans. on Computers,* vol. C-29, pp. 791-801, Sept. 1980.

[Sl87] P. J. Slater, "A summary of results on pair-connected reliability," *Technical Report 556, Department of Mathematical Sciences*, Clemson University, 1987.

[SM81] H. J. Siegel and R. J. Mcmillen, "The multistage cube : A versatile interconnection network," *Computer*, pp. 65-76, Dec. 1981.

[Sr89] M. A. Sridhar, "A fast algorithm for testing isomorphism of permutation networks," *IEEE Trans. on Computers,* vol. 38, pp. 903-909, June 1989.

[SS78] H. J. Siegel and S. D. Smith, ''Study of multistage SIMD interconnection networks,'' in *Proc. of 5th. Symp. of Computer Architecture*, pp. 223-229, Apr. 1978.

[SSB+91] P. K. Srimani, B. P. Sinha, B. B. Bhattacharya and S. Ghose, ''Properties of a class of trivalent network graphs and optimal routing,'' *Computers Math. Applic.*, vol. 22, no. 2, pp. 39 - 47, 1991.

[SSS91] K. Sutner, A. Satyanarayana and C. Suffel, ''The complexity of the resudual node connectedness reliability problem,'' *SIAM J. Comput.*, vol. 20, pp. 149 - 155, 1991.

[SSS92] A. Satyanarayana, L. Schoppmann and C.L. Suffel, ''A reliability-improving graph transformation with applications to network reliability,'' *Network*, vol. 22, pp. 209 - 216, 1992.

[St71] H. S. Stone, ''Parallel processing with the perfect shuffle,'' *IEEE Trans. on Computers,* vol. C-20, no. 2, pp. 153-161, Feb. 1971.

[Th78] C. D. Thompson, ''Generalized connection networks for parallel processor intercommunication,'' *IEEE Trans. on Computers,* vol. C-24, no. 12, pp. 1119-112, Dec. 1978.

[TW91] N. -F. Tzeng and S. Wei, ''Enhanced hypercubes,'' *IEEE Trans. Comput.*, vol. C-40, no. 1, March. 1991, pp 284-293.

[Tz90] N. -F. Tzeng, ''Structural properties of incomplete hypercube computers,'' in *Proc. 10th International Conference on Distributed Computing Systems*, pp. 262 - 269, May 1990.

[Tz91] D. Tzvieli, ''Minimal diameter double-loop networks. I. large infinite optimal classes,'' *Networks*, vol. 21, pp. 387 - 415, 1991.

[Va79] L. G. Valiant, ''The complexity of enumeration and reliability problems,'' *SIAM J. Comput.*, vol. 8, pp. 410 - 421, 1979.

[Wa68] A. Waksman, ''A permutation network,'' *J. Assoc. Comput. Mach.*, vol. 15, no. 1, pp. 159-163, Jan. 1968.

[WC74] C. K. Wang and D. Coppersmith, ''A combinatorial problem related to multimodule memory organizations,'' *J. ACM* 21, pp. 392-402, 1974.

[WF80] C. Wu and T. Feng, ''On a class of multistage interconnection networks,'' *IEEE Trans. Comput.*, vol. C-29, no. 8, pp. 694 - 702, Aug. 1980.

[WF80a] C. Wu and T. Feng, ''The reverse-exchange interconnection network,'' *IEEE Trans. on Computers,* vol. C-29, pp. 801-811, Sept. 1980.

[WF81] C. Wu and T. Feng, ''The universality of the shuffle-exchange network,'' *IEEE Trans. on Computers,* vol. C-30, pp. 324-332, May 1981.

List of Publications of the Author on some of which this thesis is based

Published in Journals :

[1] K. Mukhopadhyaya and B. P. Sinha, ''Reliability analysis of networks using stochastic model,'' *Information Sciences,* vol 65, no., 3, pp. 225-237, Nov. 1992.

[2] K. Mukhopadhyaya and B. P. Sinha, ''Hamiltonian graphs with minimum number of edges for fault-tolerant topologies,'' *Information Processing Letters,* vol. 44, pp. 95 - 99, 19 Nov. 1992.

[3] R. K. Das, K. Mukhopadhyaya and B. P. Sinha, ''A new family of bridged and twisted hypercubes,'' to appear in *IEEE Transactions on Computers.*

[4] S. Sengupta, K. Mukhopadhyaya, B. B. Bhattacharya and B. P. Sinha, ''Geometric classification of triangulations and their enumerations in a convex polygon,'' to appear in *Computers and Mathematics with Applications.*

[5] D. Das, K. Mukhopadhyaya and B. P. Sinha, ''Implementations of four common functions on LNS Co-processor,'' to appear in *IEEE Transactions on Computers.*

Published in Conference Proceedings :

[1] K. Mukhopadhyaya and B. P. Sinha, ''Optimal design and routing of distributed loop networks'', *in Proc. IEEE International Symposium on Circuits and Systems,* pp. 1021 - 1024, Aug. 11-14, 1991.

[2] N. Das, K. Mukhopadhyaya and J. Dattagupta, ''On self-routable permutations in Benes network'', *in Proc. International Conference on Parallel Processing,* pp. I-270 - I-273, Aug. 12-16, 1991.

[3] P. Biswas, K. Mukhopadhyaya and B. P. Sinha, ''Real-time FFT processing by overlapping data acquisition and computation phases,'' *Frontiers in Parallel Computing,* pp. 271-277, Narosa Publishing House, 1991.

[4] R. K. Das, K. Mukhopadhyaya and B. P. Sinha, ''Bridged and twisted hypercubes with reduced diameters,'' in *Proc. International Conference on Parallel Processing,*, pp. I-72 - I-75, Aug 17-21, 1992.

[5] R. K. Das and K. Mukhopadhyaya ''Bridged and twisted hypercubes with reduced diameters,'' in *Proc. Second National Seminar on Theoretical Computer Science,* pp. 114 - 128, Calcutta, June 17 - 19, 1992.

[6] N. Das, K. Mukhopadhyaya and J. Dattagupta, ''A versatile external control method for self-routable permutations in Benes network'', in *Proc. International Conference on Parallel Processing,*, Aug 17-21, pp. I-123 - I-126, 1992.

[7] N. Das, K. Mukhopadhyaya and J. Dattagupta, ''External-control method for self-routable permutations in Benes network,'' in *Proc. International Conference on Computer Communication,*, Genova, Italy, Sept. 28 - Oct. 2, 1992.

[8] K. Mukhopadhyaya and B. P. Sinha, ''On multi-layered non-blocking networks,'' in *Proc. Third National Seminar on Theoretical Computer Science,* pp. 91 - 102, June 16 - 18, 1993.

[9] S. C. Nandy, K. Mukhopadhyaya and B. B. Bhattacharya, ''On a new class of firing problems,'' in *Proc. Third National Seminar on Theoretical Computer Science,* pp. 211 - 222, June 16 - 18, 1993.

[10] S. Sen Gupta, K. Mukhopadhyaya, B. B. Bhattacharya and B. P. Sinha, ''Geometric classification of triangulations and their enumeration in a convex polygon,'' in *Proc. Third National Seminar on Theoretical Computer Science,* pp. 91 - 102, June 16 - 18, 1993.