1-31-1991

Fault tolerant clos network

Preet Mohan S. Ahluwalia
New Jersey Institute of Technology

Fault Tolerant Clos Network

by

Preet Mohan S. Ahluwalia

Thesis submitted to the Faculty of the Graduate School of
the New Jersey Institute of Technology in partial fulfillment of
the requirements for the degree of
Master of Science in Electrical Engineering
1991

APPROVAL SHEET

Title of Thesis: Fault-Tolerant Clos Network

Candidate: Preet Mohan S. Ahluwalia
Master of Science in Electrical Engineering, 1991

Thesis and Abstract Approved by the Examining Committee:

Dr. John D. Carpinelli, Advisor
Assistant Professor
Department of Electrical and Computer Engineering

Date

Dr. Anthony Robbi
Associate Professor
Department of Electrical and Computer Engineering

Date

Dr. Sotirios Ziavaras
Assistant Professor
Department of Electrical and Computer Engineering

Date

ABSTRACT

Thesis Title: Fault Tolerant Clos Network

Candidate's Name: Preet Mohan S. Ahluwalia

Thesis directed by : Dr. John D. Carpinelli, Assistant Professor

Multistage interconnection networks, or MINs, provide paths between functional modules in multiprocessor systems. The MINs are usually segmented into several stages. Each stage connects inputs to appropriate links of the next stage so that the cumulative effect of all the stages satisfies input-output connection requirements.

This thesis deals with a fault tolerant Clos network. The fault tolerance technique involves addition of extra switches per stage to compensate for any switch failure The reliability analysis of both ordinary and fault tolerant Clos networks is presented. The optimal number of extra switches required to get the best reliability results has been analyzed.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

In recent years the demand for high-performance computers has increased due to their indispensability in the fields of weather forecasting, structural analysis, medical diagnosis, military defense, genetic engineering and many other scientific and engineering applications. Computer architecture of advanced machines is centered around the concept of parallel processing. A parallel computer system can be characterized into three structural classes: *array processors, pipelined computers and multiprocessor systems.*

The advances in VLSI technology have made large scale multiprocessor systems feasible. These systems may have hundreds or even thousands of processors to carry out computations of a program concurrently, thereby speeding up the execution of a program. The research and development of a multiprocessor system is aimed at improving throughput, reliability, flexibility and availability. A basic multiprocessor architecture is shown in figure 1.1.

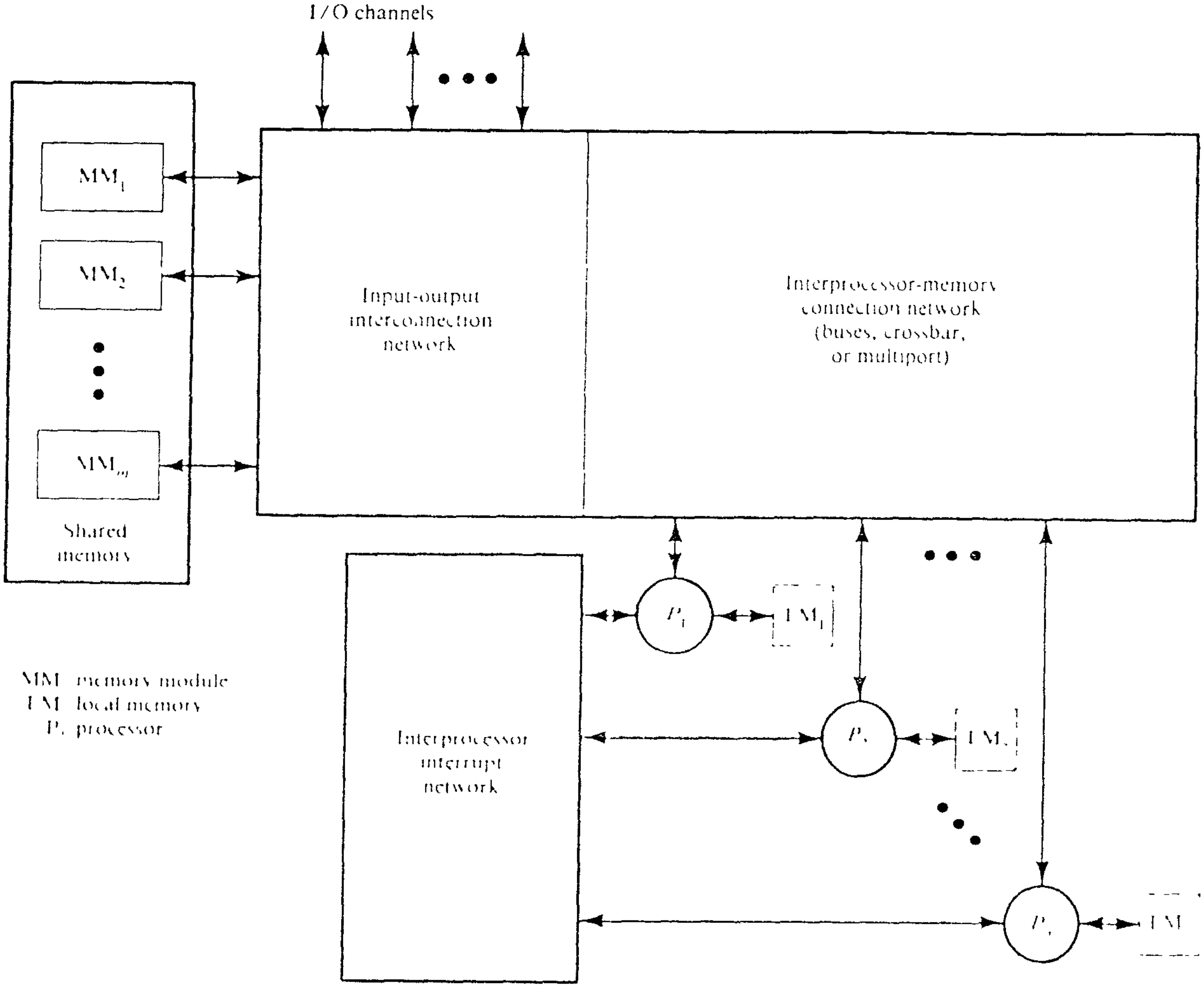
I/O channels
MM$_1$
MM$_2$
MM$_m$
Shared memory
Input-output interconnection network
Interprocessor-memory connection network (buses, crossbar, or multiport)
Interprocessor interrupt network
P_1
LM$_1$
P_2
LM$_2$
P_n
LM
MM memory module
LM local memory
P_i processor

Figure 1.1: Functional Design of a Multiprocessor System

All the processors share access to common sets of memory modules, I/O channels and peripheral devices. A single integrated operating system provides interaction between the processors and their programs at all levels. In addition each processor is supplied with its own local memory and private devices. Interprocessor communication is achieved with the help of an *interconnection network*. The efficiency of the system depends on the efficiency of the interconnection network since it establishes the path between two interacting modules. The interconnection structure which is used between the memories and processors determines primarily the organization of a multiprocessor hardware system. There have been three different kinds of networks:

1. Time-shared common bus
2. Crossbar switch
3. Multistage Interconnection Network (MIN)

A multiprocessor system that uses a shared bus as a means of communication is shown in fig. 1.2. All the processors and memory modules in the multiprocessor system are connected to the same bus. A control unit limits the access to the bus to one processor at a time. If a processor wants to communicate with a particular module it puts the address of that memory location on the bus. The address is decoded by the relevant memory location and a link is established. However, this mechanism has proven to be inefficient when the number of processors is large. This is because only one processor can use the bus at a time. If the number of processors is small, the shared bus can be used as a simple and inexpensive communication mechanism. The overall system capacity is limited by the bus transfer rate. Failure of the bus leads to the failure of the system.

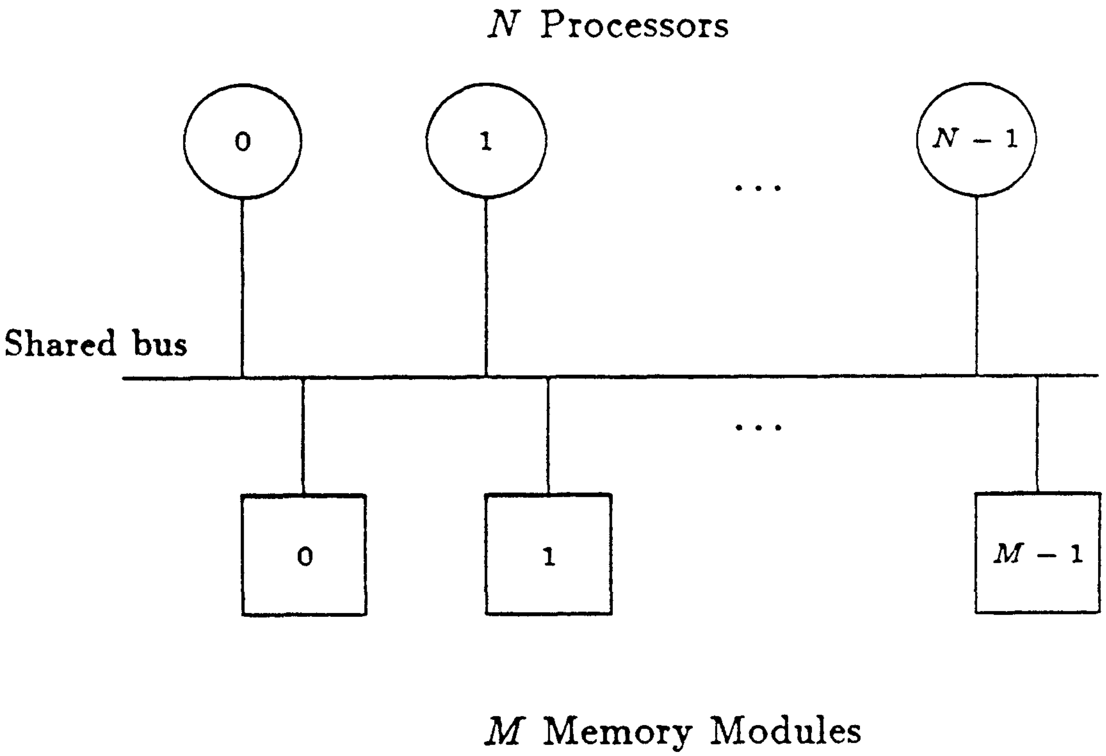

Figure 1.2: Multiprocessor system with a shared bus

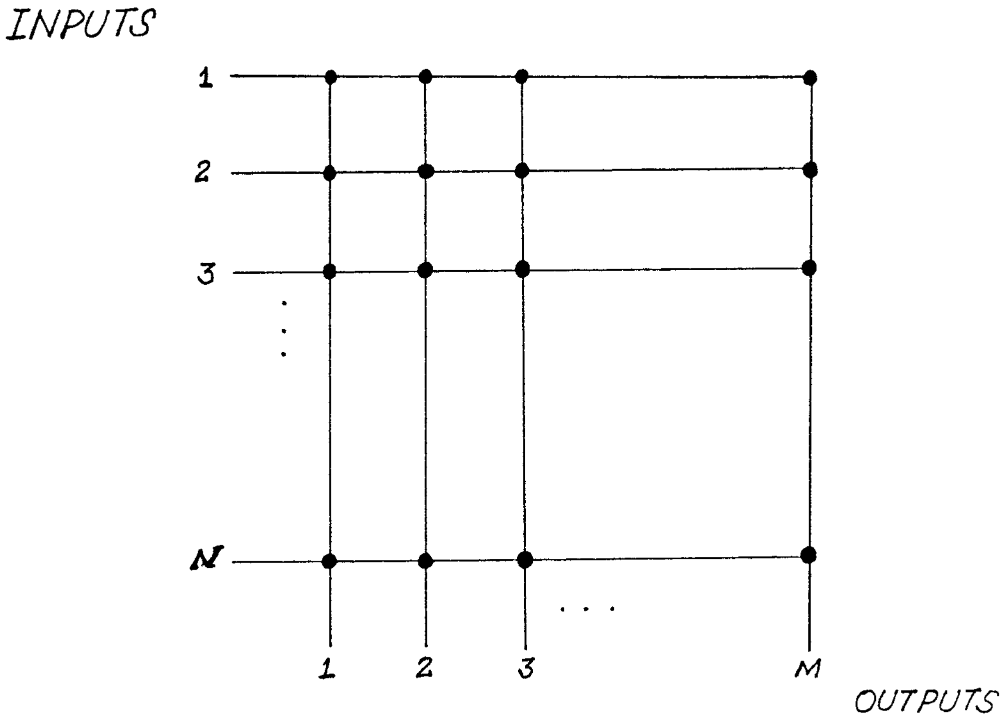

Figure 1.3: $N \times M$ Crossbar switch

of the bus leads to the failure of the system.

A crossbar switch is shown in figure 1.3. It is represented as an $N \times M$ switch because it has N inputs and M outputs. The point of intersection of these input and output lines provides the necessary connection between a particular input and a particular output. To disconnect a link the connection at the point of interconnection is broken. This connection and disconnection is implemented by a control unit attached to the switch. This is the most complex interconnection network and has the potential for the highest total transfer rate. System expansion i.e.addition of functional units, improves overall performance.

The design of a multistage interconnection network has evolved from the basic

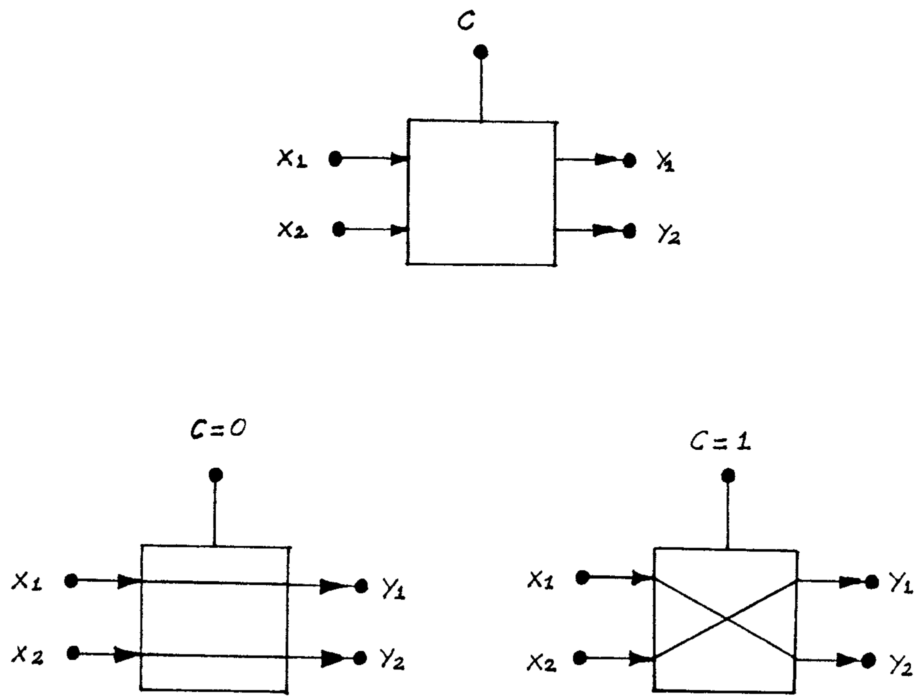

Figure 1.4: 2 × 2 Crossbar switch

principle of the crossbar switch. Consider a 2 × 2 crossbar switch as shown in fig. 1.4. This switch can connect the input to either output 0 or output 1, depending on the value of the control bit c of the input. If $c = 0$, the upper input is connected to the upper output and if $c = 1$, the connection is made to the lower output. A MIN is built from crossbar switches which are arranged in stages, with each stage connected to the adjoining one via links.

In general a multistage network consists of n stages where $N = 2^n$, gives the number of input and output lines. However, there are exceptions to this rule. For example Beneš networks have $2n - 1$ stages. Each stage may use $N/2$ switching boxes. At least N paths are required to connect one stage to another. The propa-

In order to select the architecture of an interconnection network, four design features have been identified.

1. Operation Mode
2. Control Strategy
3. Switching Method
4. Network Topology

1.1.1 Operation Mode

Typically interconnection networks have been classified into three catagories of operation mode.

Synchronous: Communication paths are established synchronously whenever there is a data or instruction broadcast.

Asynchronous: Whenever connection requests are issued dynamically in multiprocessing systems asynchronous communication is needed.

Combined: Systems that facilitates both synchronous as well as asynchronous communication are included in this category.

1.1.2 Control Strategy

An interconnection network consists of a number of switching elements connected via interconnecting links. To make the entire network operate successfully control of the switching elements is performed in either of two ways.

Distributed control: The control setting function manages the individual switching elements.

Centralized control: A centralized controller manages the control setting.

The state of the switch boxes is determined by the switching methodology. These can be identified as:

Circuit Switching: In circuit switching a dedicated end to end path ıs set up before data transfer takes place. Circuit switched interconnection networks have a well established path from the processor to the memory module, which is kept for the entire duration of the memory cycle.

Packet Switching: In operation, packet switching deals with the transmission of addressed data packets in which a channel is occupied only for the duration of transmission of the packet. A message that exceeds the packet size is broken up into several packets.

1.1.3 Network Topology

The topologies are grouped into two categories:

1. **Static:** In static topologies links between two processors are passive and dedicated buses which cannot be reconfigured for direct connection to other processors. Classifications of the various topologies are according to the dimensions required for layout. For example, the linear array is a one dimensional topology, the tree is a two dimensional topology and the 3-cube is a three dimensional topology.

2. **Dynamic:** The dynamic topology is reconfigurable by setting the system's active switching elements. Dynamic networks can be further classified as:

   Single-stage networks: A single-stage network is a network having N input selectors and N output selectors. An input selector is primarily a 1-to-D demultiplexer and an output selector is an M-to-1 multiplexer. Different control signals are required for each selector in order to establish a desired path. This type of network is also called a recirculating network, since data may recirculate several times through the single stage before reaching the final destination.

   Multistage Networks: Multistage interconnection networks, which were originally used for telephony, are networks that are capable of connecting an arbitrary input to an arbitrary output terminal. These netwoiks can be one-sided or two-sided. The former type has both the input and output on the same side. The two-sided network has different input and output sides and can be classified as:

   (a) **Blocking:** In the blocking type of network connections of more than one terminal pair simultaneously may result in a conflict in the communication links.

   (b) **Rearrangeable:** A rearrangeable network can establish all possible connections between its inputs and outputs. This is made possible by rearranging the existing connections so that a connection path between an input and an output pair is always established. The Benes network belongs to this class.

(c) **Strictly Nonblocking:** This type of network can handle all possible connections without modifying any existing connections. Some Clos networks are strictly nonblocking.

1.2 Outline

The rest of the thesis is organized as follows. Chapter 2 presents some basic concepts and terminology required to understand the work mentioned in the thesis. In particular, a fault tolerance model is presented. There is also a brief description of the fundamentals of probability. Furthermore an introduction to the theory of reliability is dealt with. In Chapter 3 mathematical background of the reliability analysis is explained. Later, the technique used in order to make a fault tolerant Clos network is described. The last chapter has some concluding remarks on the work done in this thesis and proposed future research.

Chapter 2

Basic Concepts

2.1 Clos non-blocking networks

The Clos network is a 3-stage network which was initial developed for telephone traffic routing. The 3-stage Clos network is shown in figure 2.1. The first stage consists of k switches each one of which has m inputs and n outputs. The inputs to the first stage switch i $(1 \leq \imath \leq k)$ are numbered from $(i-1) \cdot m + 1$ to $i \cdot m$. The second stage contains n $k \times k$ cells, each of which receives one input from each first stage switch. The final stage has k $n \times m$ switches, each of which gets an input from each second stage switch. Each cell can realize any one-to-one mapping of its inputs onto its outputs. A Clos network as described above is referred to as an (n, m, k) Clos network. If $n \geq m$, the network is rearrangeable and if $n \geq 2m-1$, then the network is it strictly non-blocking. The total number of inputs to the network is $N = m \cdot k$. The Clos network is not a single network; it is a family of networks derived by varying the values of n, m, and k.

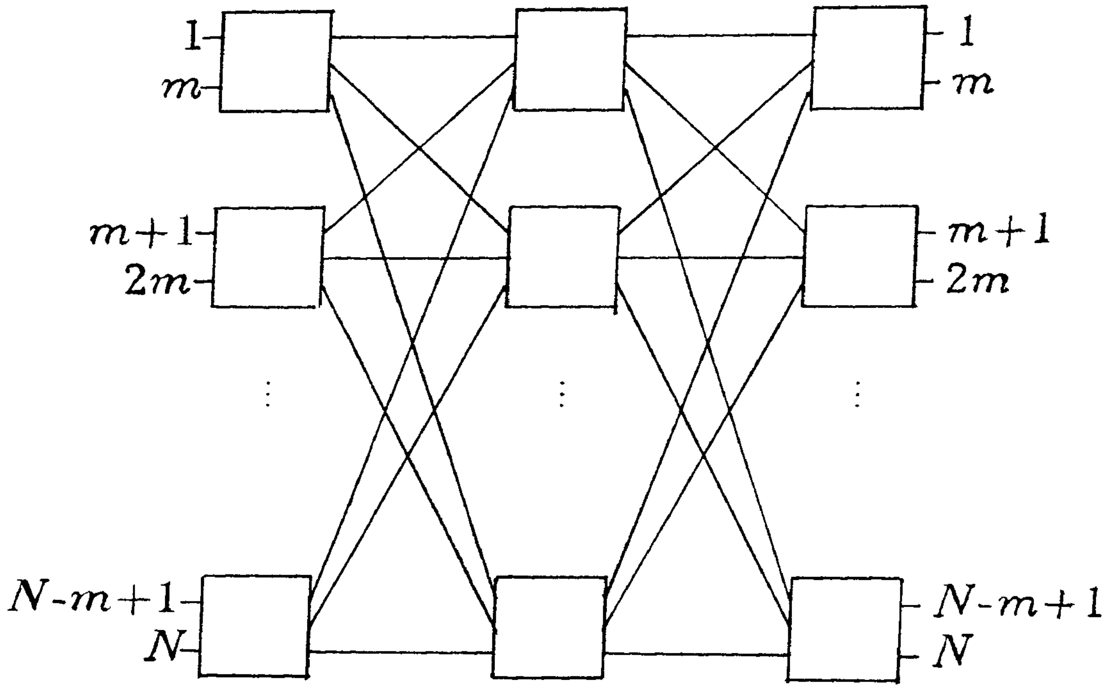

Figure 2.1: 3-Stage Clos network

2.2 Need for Fault Tolerance

Real time applications which require enormous computing power are stimulating significant advances in the area of parallel processing. Multiprocessor systems using many processors executing in parallel have the ability to execute instructions at rates exceeding one billion instructions per second. As the number of processors used in these systems increases, so does the need to insure that the communication network between the system components becomes more reliable.

One way of classifying MINs is according to their *fault tolerance* capabilities. Fault tolerance has evolved as an important issue while designing MINs for multiprocessor systems, the most important reason being that these systems may run important tasks where interruption may have a fatal consequence. Since an interconnection network acts as a communication link between the system components, a fault in the network can seriously affect the performance of the system. Consider the case of a shuffle network, where the paths between the inputs and outputs are unique. If a switch needed to establish a path is faulty then the path cannot be established until the faulty switch has been replaced.

Since a computer program has instructions in a sequential and dependent manner, a memory word can be vital to the completion of a program. If that word is not accessed due to a fault then the program cannot be completed. Hence in a multiprocessor system it is very important that the communication links between processors and memory modules be maintained at all times. This implies that the interconnection network should be fault tolerant.

2.3 Fault Tolerance Model

In order to design a fault tolerant multiprocessor interconnection network a fault tolerance model should be defined. The fault tolerance model contains three elements:

1. Fault Model

2. Fault Tolerance Criterion

3. Fault Tolerance Size

2.3.1 Fault Model

The types of faults that can occur in a network constitute the fault model. In other words the fault model specifies the type of faults that can be recovered from by using the fault tolerance design. A typical fault model is as follows.

1. **Any network component can fail:** Switches and links are the two types of components that make up a MIN. It is therefore possible that the switches and links can fail. The fault tolerance design should be able to recover from any such fault. A link fails if it is open or short circuited. Bridge faults may also occur at the inputs of any switch. A switch fails due to some internal. malfunction. The multiplexers and demultiplexers can also fail. However, it has been assumed througout that the links, multiplexers and demultiplexers do not fail.

2. **The extra hardware can fail:** The extra hardware used to provide fault tolerance to the network can fail. This assumption is made keeping in mind the fact that all switches have equal reliability. Also, every switch functions

independently of every other, i.e. failure of a switch does not in any way effect the failure of another.

2.3.2 Fault Tolerance Criterion

The fault tolerance criterion is the condition that must be fulfilled in order for the system to be called fault tolerant. Two broad categories are defined.

1. **Full-access retention:** Full-access retention implies that after a fault has occurred each processor must still be able to communicate with any one of the memory modules.

2. **Full recovery:** Full recovery is the ability of a network to regain its pre-fault connectivity after the occurrence of a fault.

2.3.3 Fault Size

The total number of faults that a system can recover from gives the fault size. If a network can tolerate x *specific* faults, $x \geq 1$, it is called x-robust. In a fault tolerant system having n stages, one fault per stage makes it n-fault robust.

2.4 Fundamentals of Probability

In any random experiment there is always uncertainty as to whether a particular event will or will not occur. As a measure of the *chance* or *probability* with which we can expect the event to occur it is convenient to assign a number between 0 and 1. If we are sure that the event will occur we say that the probability is 100% or 1, but if we are sure that the event will not occur we say that its probability is zero. If for example the probability is $\frac{1}{5}$, we would say that there is a 20% chance it will occur and a 80% chance that it will not occur. Equivalently we can say that the odds against its occurrence are 80% to 20%.

2.4.1 Theorems On Probability

Theorem 2.1 *For every event A, $0 \leq P(A) \leq 1$, i.e. the probability of the occurrence of an event A is between 0 and 1.*

Theorem 2.2 *$P(O) = 0$, i.e. the impossible event has zero probability.*

Theorem 2.3 *If A' is the complement of A then, $P(A') = 1 - P(A)$.*

2.4.2 Independent Events

Conditional probability states that $P(A_j/A_k)$ is the probability of occurrence of A_j given that A_k has already occurred. If $P(A/B) = P(A)$, i.e.the probability of the occurrence of B is not affected by the occurrence or nonoccurrence of A, then we say that A and B are independent events. This is equivalent to $P(AandB) = P(A) \cdot P(B)$. If A_1, A_2 and A_3 are to be independent then they must be pairwise independent i.e.$P(A_j andA_k) = P(A_j) \cdot P(A_k)$, where $j \neq k$ where $j, k = 1, 2, 3$.

2.4.3 Bernoulli Distribution

Suppose that we have an experiment such as tossing a coin repeatedly. Each toss is called a *trial*. In any single trial there will be a probability associated with a particular event such as head on the coin. In some cases this probability will not change from one trial to the next. Such trials are then said to be *independent* and are called *Bernoulli trials*.

Let p be the probability that an event will happen in any single Bernoulli trial (called the probability of *success*). Then, $q = 1-p$ is the probability that the event will fail to happen in any single trial (called the probability of *failure*). The probability that the event will happen exactly x times in n trials (i.e. x successes and $n - x$ failures will occur) is given by the probability function

$$f(x) \;=\; \binom{n}{x}\; p^{x}\; q^{n-x} \qquad (2.1)$$

2.5 Reliability Analysis

The probability that a system performs a given task under a set of conditions and for a certain period of time is called its *reliability*. Mathematically the reliability of a system is defined as

$$R = e^{-\lambda t} \qquad (2.2)$$

where R is the reliability, λ is the failure rate and t gives a stated period of time. Figure 2.2 shows a generalized MIN. This multistage interconnection network has N inlets and M outlets, p stages, and x_j switches in each stage j, $0 \leq j \leq p-1$. It should be noted that stage j derives all of its inputs from stage $j-1$, stage 0 derives all of its inputs from the inlets of the MIN and the outlets are derived from stage $p-1$.

As seen from figure 2.2, for an input to be mapped to an output all the stages of a MIN must be operational, i.e. free of any fault. In addition, the links between any two stages should also be functioning normally. Let F_{ij}, $1 \leq j \leq p-1$ be the function realized by the set of links between stages $j-1$ and j, mapping the outputs of stage $j-1$ onto the inputs of stage j. Also, let F_{sj}, $0 \leq j \leq p-1$, be the function realized by the switches in stage j mapping the inputs of stage j onto its outputs. If the realization of functions F_{ij} and F_{sj} can be made more reliable, the reliability of the system can be enhanced. In other words, by making all the stages of a MIN more reliable the overall reliability of the system can be increased.

Let R_i, $0 \leq i \leq p-1$, be the reliability of stage i, of a system having p stages. When all the stages are in series, failure of any one stage results in system failure. Mathematically, the system reliability is given by,

$$R = \prod_{i=0}^{p-1} R_i \tag{2.3}$$

Moreover with p statistically independent stages in parallel the system reliability is

$$R = (R_0 \cup R_1 \cup R_2 \cup \ldots \cup R_{p-1}) \tag{2.4}$$

which can be written as

$$R = 1 - \prod_{i=0}^{p-1}(1 - R_i) \tag{2.5}$$

The usefulness of the above expression will become evident in Chapter 3, where the reliability of the Clos network is analyzed.

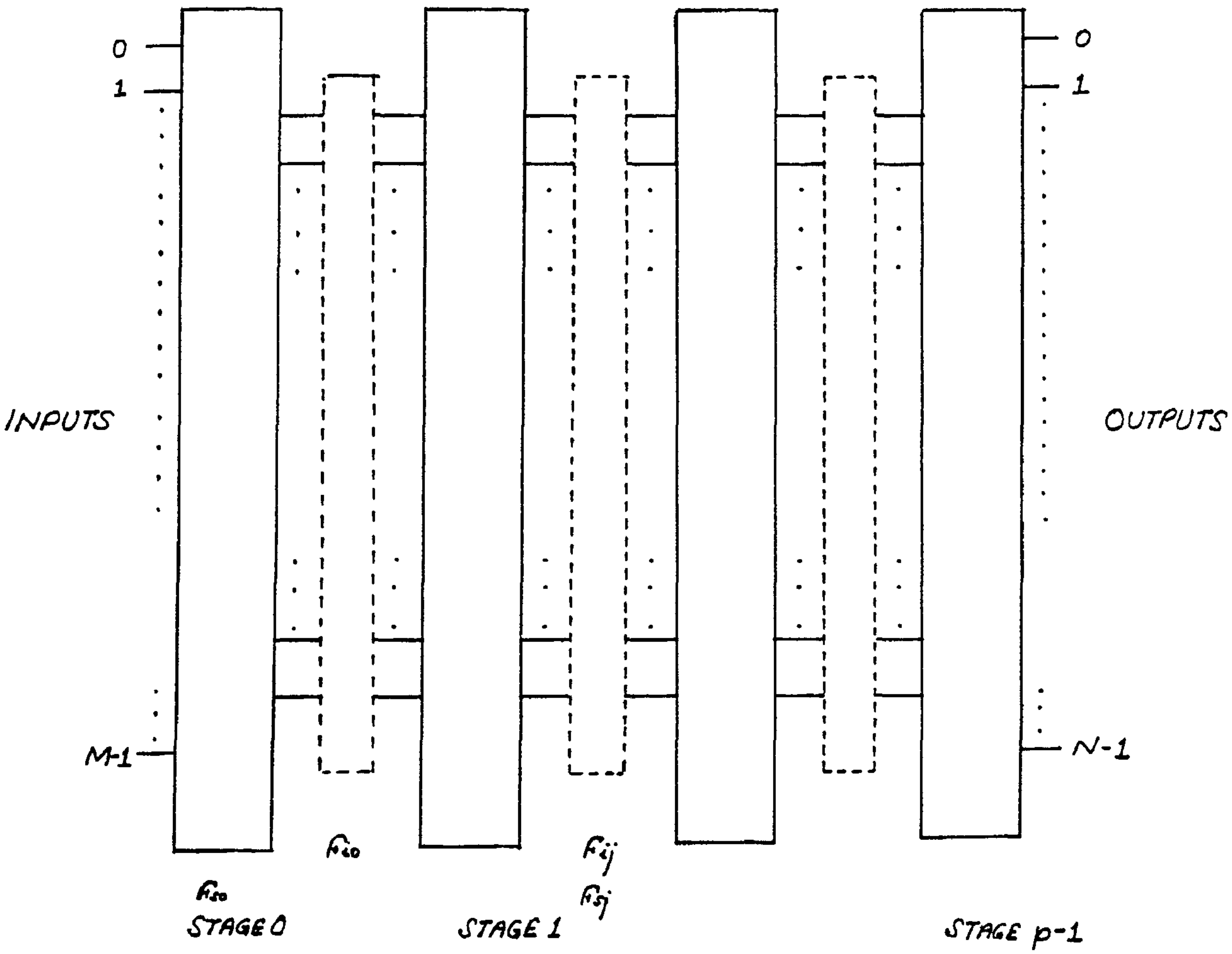

Figure 2.2: A generalized multistage interconnection network

Chapter 3

Fault-Tolerant Clos Network Design

3.1 Reliability Analysis of a Clos Network

The Clos network of figure 2.1 is redrawn in figure 3.1. From the reliability analysis described in Chapter 2, it can be inferred that for maximum reliability of the Clos network all the stages should have maximum reliability independent of each other. Let R be the reliability of the network. Also, let R_p, $1 \leq p \leq 3$, be the reliability of the three stages respectively. Then the overall reliability is

$$R_{oall} = \prod_{p=1}^{3} R_p \tag{3.1}$$

In order to calculate the reliability of any one stage it is important to understand that, in general, for all parallel systems of n independent switches the combined reliability is

$$R = 1 - \prod_{i=1}^{n} (1 - R_i) \tag{3.2}$$

This is identical to equation 2.5. The fault tolerant approach described in this thesis is to add extra switches to each stage. These switches compensate for any

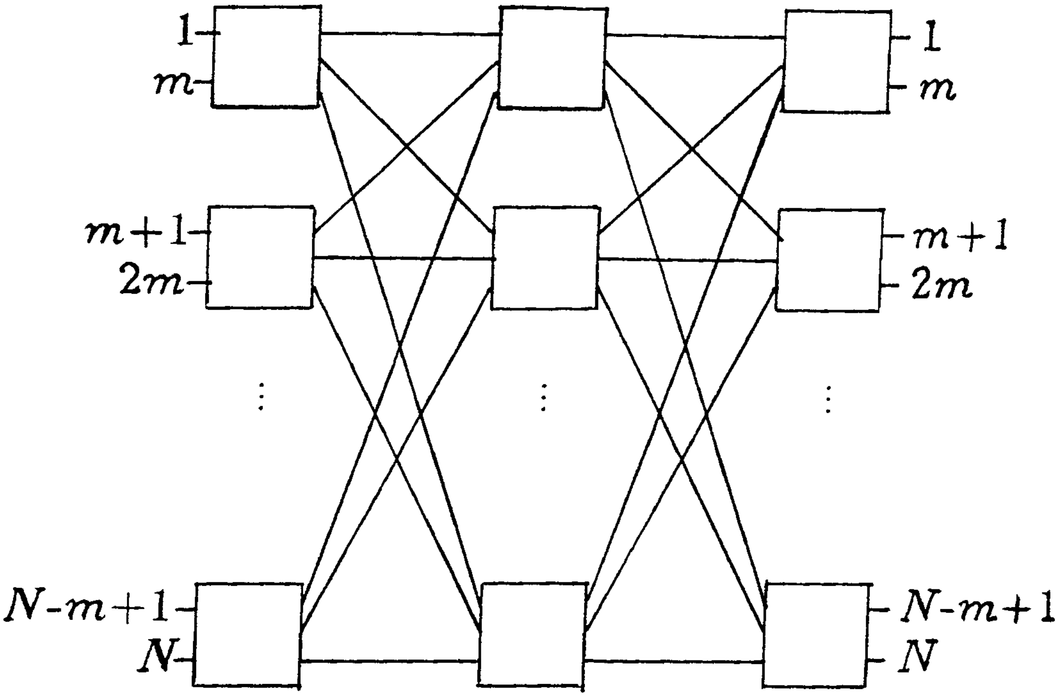

Figure 3.1: 3-Stage Clos network

faulty ones. The optimum number of switches required per stage to give a highly reliable network without encountering additional cost is the endeavor of the thesis.

Consider a simple case where there is one fault per stage in a 9 × 9 Clos network. Assume there is one extra switch per stage in order to make the network fault tolerant as shown in figure 3.2. The shaded blocks represent extra switches. Let $R_\imath$, $0 \leq \imath \leq 2$ be the reliability of the i^{th} stage. Then, the reliability of the network is

$$R_{oall} = R_0 \cdot R_1 \cdot R_2 \tag{3.3}$$

The reliability of the first stage, R_0, is the probability that not more than one switch is faulty. Similarly, R_1 and R_2 give the probabilities that stages 2 and 3 are functioning with or without a single fault.

Let f be the probability that a switch fails. Then $r = (1 - f)$ is the probability that the switch functions normally, i.e.the reliability of the switch. If F_0 is the probability that stage 1 fails it is evident that more than one switch has failed. Therefore,

$$\begin{aligned} F_0 &= \sum_{\imath=2}^{4} \left(\begin{array}{c} 4 \\ \imath \end{array} \right) f^\imath r^{4-\imath} \\ &= \sum_{\imath=2}^{4} \left(\begin{array}{c} 4 \\ i \end{array} \right) f^\imath (1-f)^{4-\imath} \end{aligned} \tag{3.4}$$

Since $R_0 = R_2$ due to symmetry, the probability that stage 2 fails is

$$F_2 = \sum_{\imath=2}^{4} \left(\begin{array}{c} 4 \\ i \end{array} \right) f^\imath (1-f)^{4-\imath} \tag{3.5}$$

For this netwiork this will also be the probability of failure of the middle stage. The reliability of all the three stages in this case is

$$R_0 = R_1 = R_2 = 1 - \sum_{\imath=2}^{4} \left(\begin{array}{c} 4 \\ \imath \end{array} \right) f^\imath (1-f)^{4-\imath} \tag{3.6}$$

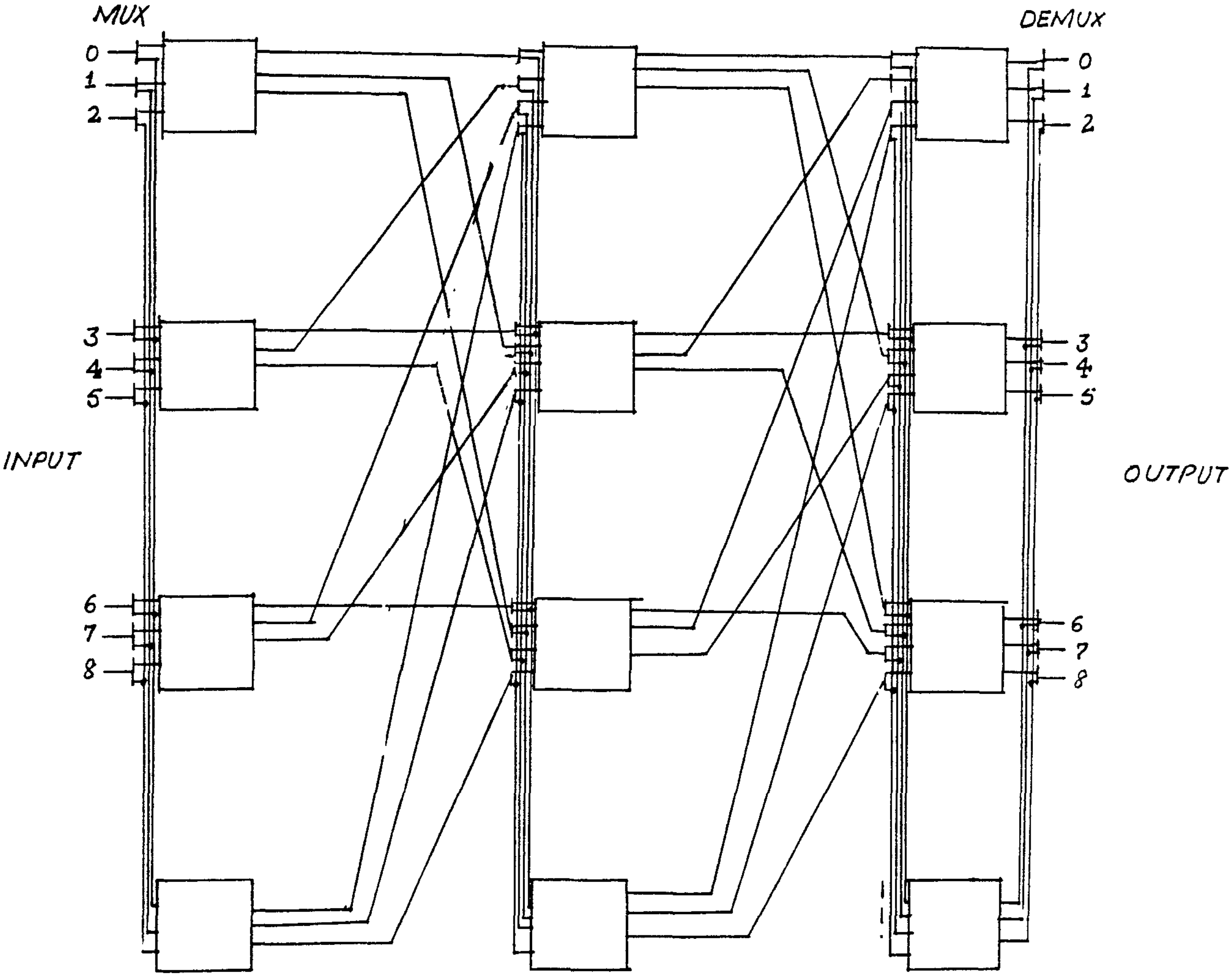

Figure 3.2: 9 × 9 FTC network with one extra switch per stage.

Hence equation 3.6 becomes

$$R_{oall} = \left(1 - \sum_{i=2}^{4} \binom{4}{i} f^{i}(1-f)^{4-i}\right)^{3} \tag{3.7}$$

On the basis of the above discussion a similar relationship can be obtained for a Clos network k switches in the outer stages and m switches in the middle stage. The number of extra switches in the two outer stages and the middle stage are x and y respectively. The overall reliability is then

$$R_{oall} = \left\{1 - \sum_{i=x+1}^{k+x} \binom{k+x}{i} f^{i}(1-f)^{k+x-i}\right\}^{2} \left\{1 - \sum_{i=y+1}^{m+y} \binom{m+y}{i} f^{i}(1-f)^{m+y-i}\right\} \tag{3.8}$$

3.2 Optimization of Extra Switches

Equation 3.8 gives the reliability of a network with $(2x + y)$ extra switches. Two approaches can be followed to find the values of x and y. These are

1. Mathematical

2. Analytical

This thesis is based on the latter approach. However, a slight explanation of the former is dealt with in the next subsection.

3.2.1 Mathematical

This approach involves the solution of equation 3.8 by using *partial differential equations.* For any network the network parameters k and m are assumed to be known. The number of extra switches x and y for which the reliability is maximum can be obtained as follows.

$$\frac{\partial R_{oall}}{\partial x} = 0 \qquad \frac{\partial R_{oall}}{\partial y} = 0 \tag{3.9}$$

$$\frac{\partial^2 R_{oall}}{\partial x^2} < 0 \qquad \frac{\partial^2 R_{oall}}{\partial x^2} < 0 \tag{3.10}$$

$$\frac{\partial^2 R_{oall}}{\partial x \cdot \partial y} < 0 \tag{3.11}$$

Solving the given set of equations, the best value of x and y for which the condititon of maximum reliability results, can be obtained. However, the solution to these equations is beyond the scope of this thesis and is deferred for future research.

3.2.2 Analytical

The analytical method involves analyzing a list of results by varying the network parameters. A C program, shown in the appendix, which simulates equation 3.8, was run for various values of k, m, x and y. The values of x and y were made to vary from 0 to 8, i.e. $0 \leq x, y \leq 8$ for every k and m between 3 and 8. Also, as defined in section 3.1, r is the probability that the switch is functional and $f = (1 - r)$. The networks analyzed in this thesis are square networks, i.e. networks for which $n = m$. Before a fault tolerant Clos network is analyzed it is necessary to define an *ordinary* Clos network. An ordinary Clos network is one which does not have any fault tolerance capabilities or has no extra switches in any of its stages. Clos networks are generally used to realize permutations. Without fault tolerance if a switch fails the entire system will be rendered inoperative until the faulty switch has been physically replaced. This shows the necessity of fault tolerant networks. Table 3.1 gives the reliabilities of an ordinary and a $3-$*robust* fault tolerant (FTC) Clos network for various network configurations.

Network Size	R_o	R_{ftc}
(9×9)	.231617	.706113
(16×16)	.142242	.582622
(25×25)	.087354	.468164
(36×36)	.053646	.367961
(49×49)	.032946	.283830
(64×64)	.020233	.215438

Table 3.1: Network size vs. overall reliability for r=.85

It can be seen from Table 3.1 that as the size of the network increases the overall reliability decreases for both the ordinary and fault tolerant (FTC) Clos networks. Here, a switch reliability of .85 is assumed. This decrease is due to the fact that as the number of switches increases there is a greater chance that some switches fail. Also, for a fault tolerant network the reliability increases manyfold as the number of switches increases.

Figure 3.3 shows the increase in reliability of a $3 - robust$ network and an ordinary Clos network for the same reliability of the switches. In the graph shown in figure 3.3, the overall reliability of both types of networks mentioned is shown for the case where the reliability of the individual switches is .95. Table 3.2 shows the reliability of the network for $r = .95$.

Notice the increase in the reliability of both types of networks when the reliability of the switches is increased. This is because the reliability of the network is the product of the reliability of its components, particularly that of the switches. Figure 3.4 shows the reliability of the various networks for the three cases, $r = .85$, $r = .95$ and $r = .99$. It can be seen that there is not much of a shift in the curves for $r = .95$ and $r = .99$, signifying that as r becomes very large, not many extra switches

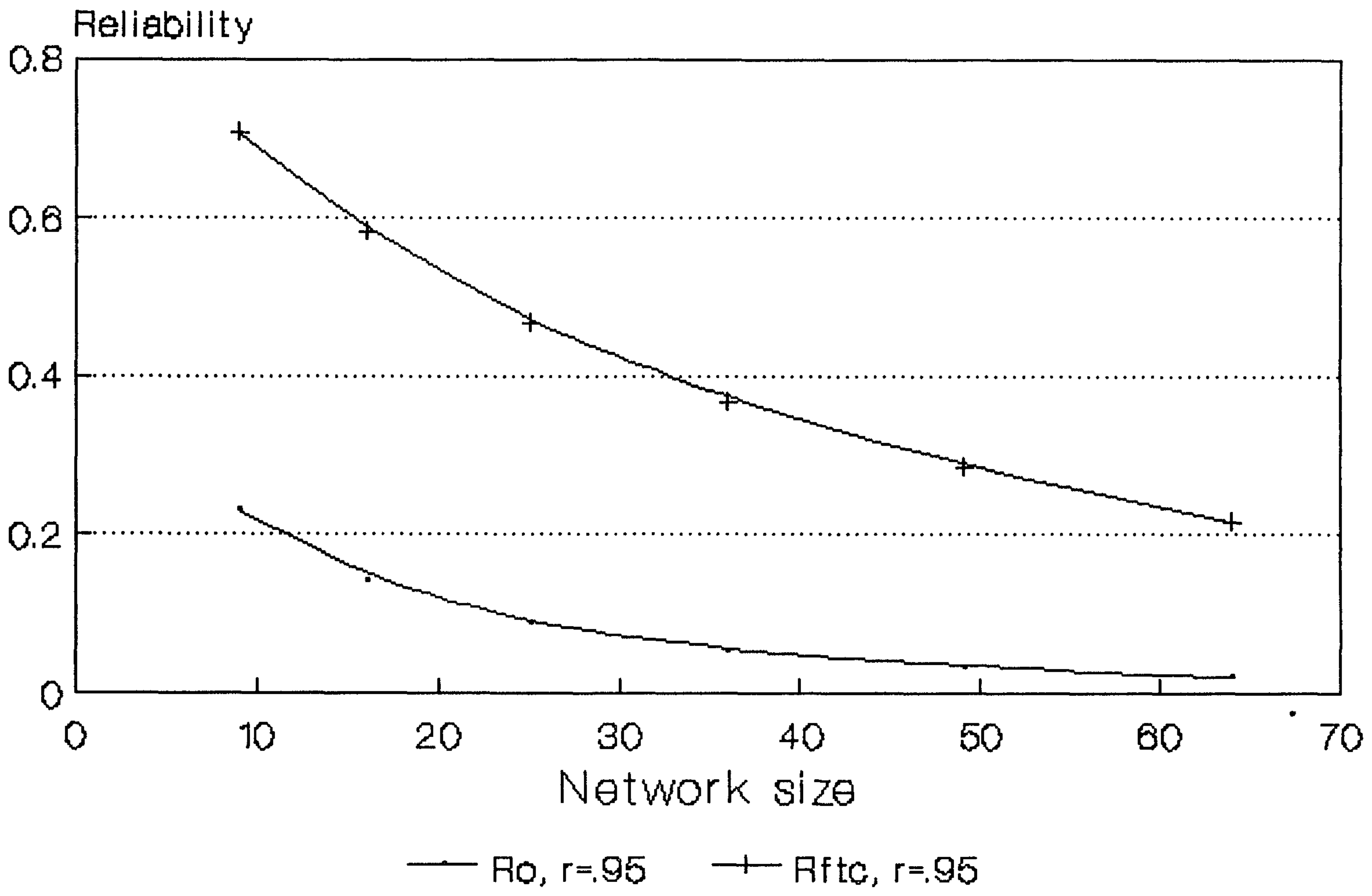

Figure 3.3: Network size vs. overall reliability

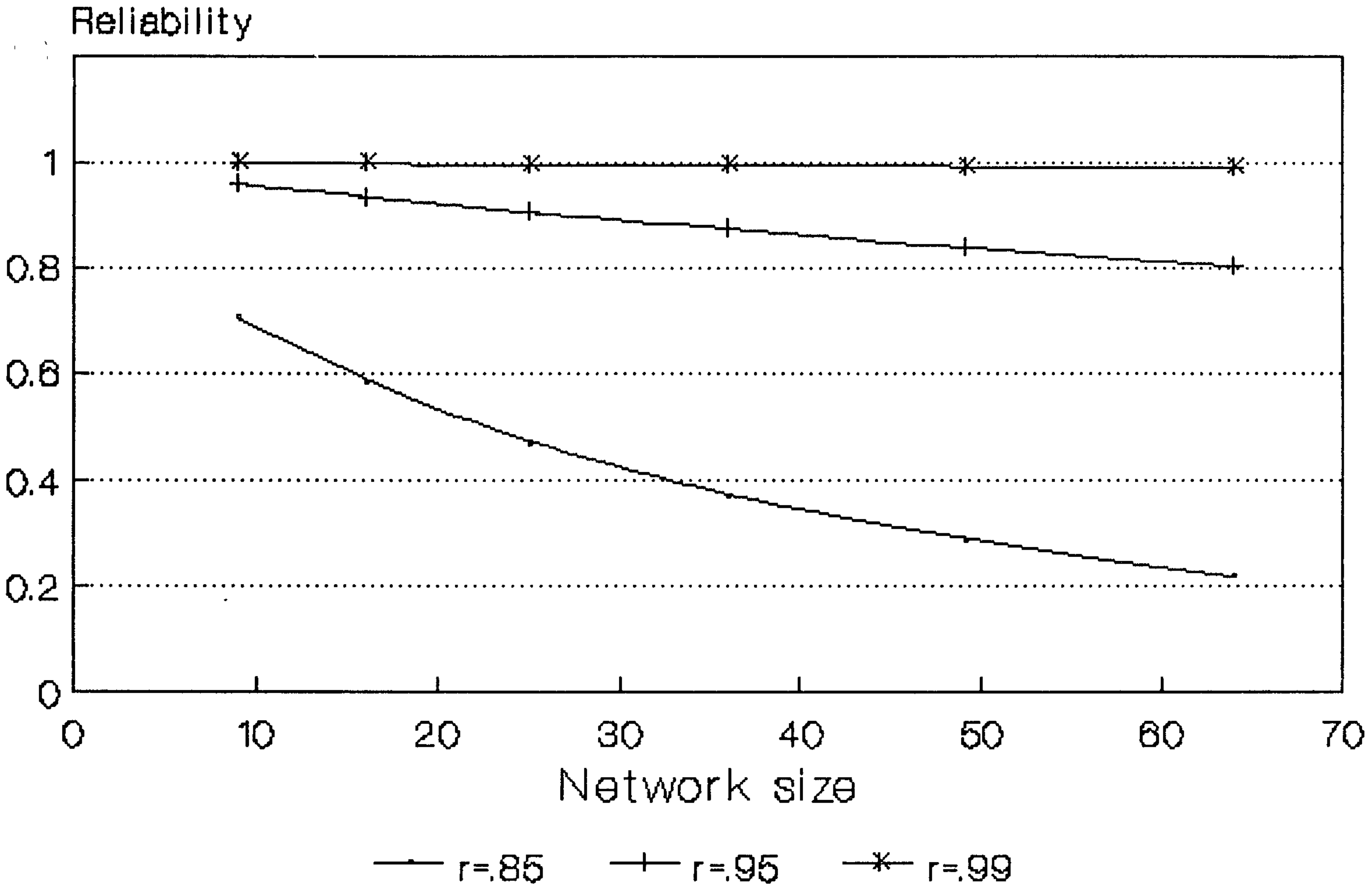

Figure 3.4: Network size vs. switch reliability

Network Size	R_o	R_{ftc}
(9 × 9)	.630249	.958531
(16 × 16)	.540360	.933743
(25 × 25)	.463291	.904866
(36 × 36)	.397214	.872680
(49 × 49)	.340562	.837909
(64 × 64)	.291989	.801218

Table 3.2: Network size vs. overall reliability for r=.95

$Size$	$r = .85$	$r = .95$	$r = .99$
(9 × 9)	.706113	.958531	.998225
(16 × 16)	.582622	.933742	.997062
(25 × 25)	.468164	.904866	.995625
(36 × 36)	.367961	.872680	.993919
(49 × 49)	.283830	.837909	.991951
(64 × 64)	.215438	.801218	.989728

Table 3.3: Network reliability for r=.85, r=.95 and r=.99

will be required to obtain a highly reliable network. Table 3.3 gives the reliabilities of the various networks for a 3-robust network. Notice that for a very high value of r a 3-robust network provides a good fault tolerance.

3.3 Hardware Design of a Clos Network

There are two approaches that are used in order to design a fault tolerant Clos network, both of which follow the analytical model.

1. **Number of extra switches not given:** Table 3.4 gives the values of $(2x + y)$ and R_{oall}. This case is for a 64 × 64 network, i.e. network having $k = m = 8$. Here, the overall reliability is the maximum reliability for a certain value of

$2x+y$	R_{oall}
0	.020233
2	.097926
4	.294758
6	.551764
8	.710235
10	.845218
12	.929937
13	.950501
14	.971520
15	.993004

Table 3.4: Number of extra switches vs. overall reliability

$(2x + y)$, whatever the combination of x and y may be.

Table 3.4 has been graphed in figure 3.5. It is observed that as the number of extra switches in each stage approaches half the number of switches in these stages, in this case $x = y = 4$, a reliability of over 90% is achieved. Similarly, the reliability of a 9×9 network as a function of $(2x + y)$ is plotted in figure 3.5. A reliability of over 90% is achieved for $2x + y \geq 6$. Both networks assume a switch reliability of .85. If however, r is increased to .99 then the reliability of the 9×9 network equals .999970 for a total of 6 extra switches. This justifies our observation: if the number of extra switches per stage is half the number of switches in these stages respectively. a high reliability is achieved. Note that 3 being odd, $\frac{3}{2}$ has been taken as $\lceil 1.5 \rceil$ i.e. 2. Similarly, for a 64×64 network, the network reliability is approximately 100% for a value of $2x+y = 12$. However in practice a switch reliability of .99 may be difficult to obtain. In such cases a few more extra switches may be added beyond $\frac{k}{2}$ and $\frac{m}{2}$ in order to get a reliability

$2x + y$	R_{oall}
2	.267522
4	.558453
6	.794627
7	.839592
8	.887102
9	.937301
10	.952360
11	.967662
12	.983209

Table 3.5: Number of extra switches vs. overall reliability

approximately equal to .99 which may increase the cost and hardware. Consider a 25 × 25 network with the reliability of the individual switches as .85. Table 3.5 gives the network reliability for a different set of extra switches. For this network $\frac{k}{2} = \frac{m}{2} = 3$ and the overall reliability is .937301 for $2x+y = 9$. However, if one more extra switch is added per stage i.e. $2x + y = 12$, network reliability of .983209 is obtained. At this point a tradeoff between cost and reliability results.

2. **Optimal distribution of switches required:** Whenever the total number of extra switches to be used is given, it becomes a design problem to optimally distribute them in the outer and middle stages. Consider again a 64 × 64 network. In order to find the best values of x and y for which a high reliability can be obtained consider Tables 3.6 and 3.7.

   With y fixed at 3, shown in Table 3.6, the reliability of the network increases as x increases. This is due to the fact that as x increases the outer stages become more reliable.

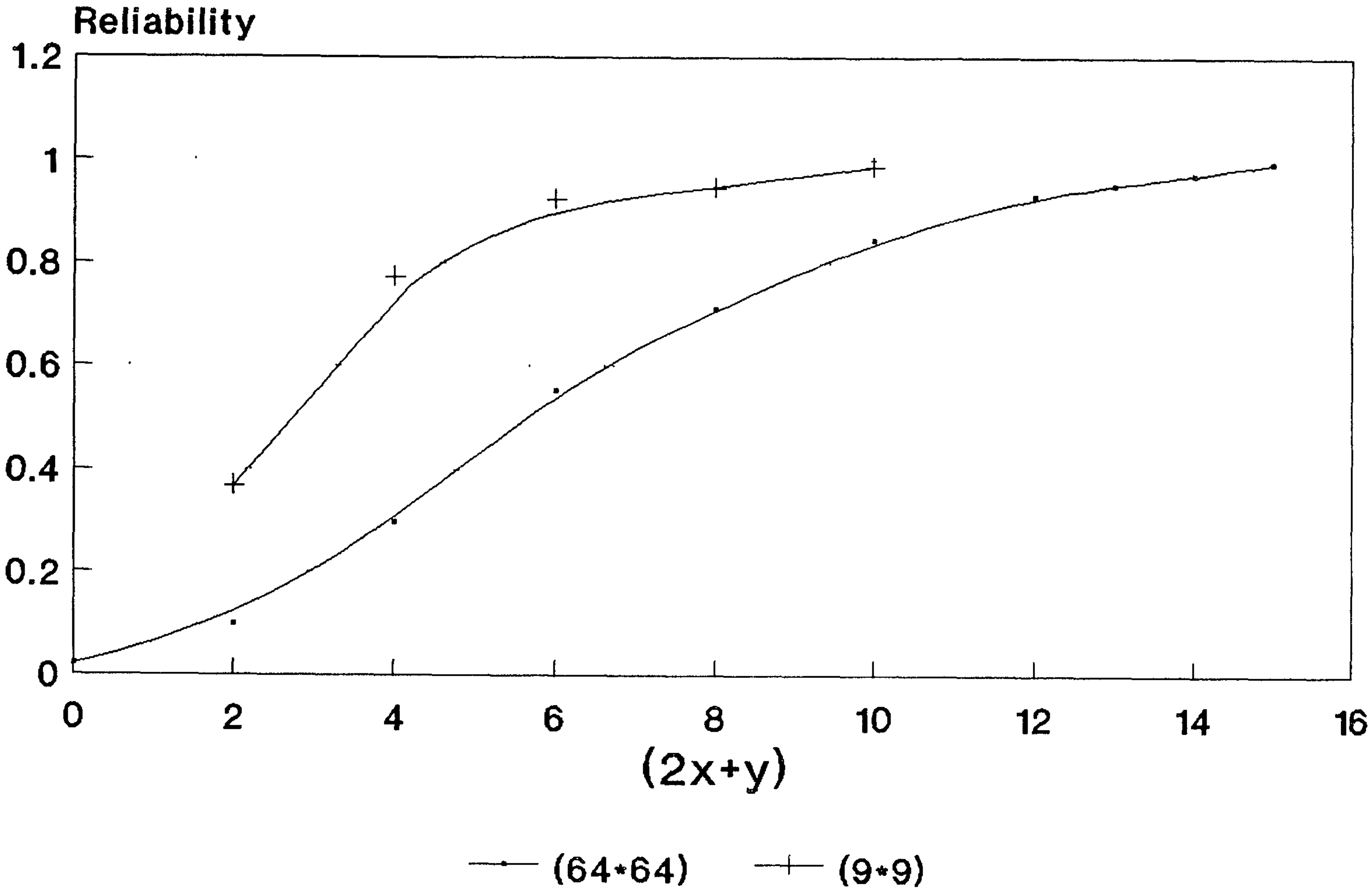

Figure 3.5: Total number of extra switches vs. overall reliability

x	y	R_{oall}
0	3	.069095
1	3	.334418
2	3	.626005
3	3	.805798
4	3	.886566
5	3	.926210
6	3	.930495
7	3	.930553
8	3	.930555

Table 3.6: Overall reliability of a 64 × 64 network with y fixed at 3

x	y	R_{oall}
3	0	.235958
3	1	.519109
3	2	.710235
3	3	.805798
3	4	.845218
3	5	.863909
3	6	.865905
3	7	.865932
3	8	.8659333

Table 3.7: Overall reliability of a 64 × 64 network with x fixed at 3

x,y	R_{oall}
(2,8)	.672722
(3,6)	.865905
(4,4)	.929937
(5,2)	.816367
(6,0)	.272473

Table 3.8: Overall reliability of a 64 × 64 network for $2x + y = 12$

Similarly, for a constant x, shown in Table 3.7, as y is increased the reliability increases as the middle stage becomes more reliable. However, in the former case the increase in reliability is much more than the latter because two stages are affected as compared to one. This result is the first step towards optimal allocation of the extra switches in the network.

Let the number of extra switches specified be 12. The various combinations of x and y and their respective reliabilities are given in Table 3.8

From the discussions above it can be intutively said that the pair (4,4) gives the best reliability. This is because the values of x and y are as close as possible to $\frac{k}{2}$ and $\frac{m}{2}$, in this case they are equal to the midvalue 4. Consider another set of values for a 49 × 49 network, i.e. one in which $k = m = 7$. Also, let the number of extra switches be 15. The various combinations of x and y are given in Table 3.9 For best reliability the *threshold value* of x and y should be as close as possible to $\frac{k}{2}$ and $\frac{m}{2}$, which is $\frac{7}{2}$, i.e. 3.5. Rounding off the fraction by taking the upper ceiling the threshold value becomes 4. It is seen that the pairs (4,7) and (5,5) have the values of x and y greater or equal to 4. The combination (5,5) gives a higher reliability since its outer two stages

x, y	R_{oall}
(4,7)	.968471
(5,5)	.986140
(6,3)	.949283
(7,1)	.657177

Table 3.9: Overall reliability of a 49 × 49 network for $2x + y = 10$

x, y	R_{oall}
(3,7)	.902557
(4,5)	.963980
(5,3)	.941231
(6,1)	.656666

Table 3.10: Overall reliability of a 49 × 49 network for $2x + y = 13$

are more reliable than in (4,7). Also, the values of x and y in (5,5) are closer to the threshold value 4. In general, whenever the number of extra switches is such that the values of x and y are equal, i.e.*symmetric* distribution about the midvalue, the highest reliability is obtained. Consider another case of 13 extra switches for a 49×49 network, given in Table 3.10. The first step towards optimal allocation is to have a symmetric distribution of the extra switches. However, this is not possible since the number of extra switches is 13. Our endevor now should be to distribute the switches in such a way such that the number of switches in every stage is equal to the threshold value, which in this case is $\lceil \frac{7}{2} \rceil$ i.e. 4. The combination $(4, 5)$ gives the required distribution. Hence this is the optimal allocation. Another interesting case is of 16 extra switches in a 36 × 36 network; see Table 3.11. Since a symmetric distribution is not

x,y	R_{oall}
(4,8)	.980349
(5,6)	.994025
(6,4)	.988795
(7,2)	.894697
(8,0)	.377149

Table 3.11: Overall reliability of a 36 × 36 network for $2x + y = 16$

x,y	R_{oall}
(0,4)	.487620
(1,2)	.881356
(2,0)	.686708

Table 3.12: Overall reliability of a 49 × 49 network for $2x + y = 4$

possible here, the pairs (4,8), (5,6) and (6,4) are considered. All these pairs have x and y greater than $\frac{k}{2}$=$\frac{m}{2}$=3. The combination (5,6) gives the highest reliability since the values of x and y, in this *asymmetric* case, are more closely distributed around the midvalue 3.

Consider again a 49×49 network with $r = .95$ and the number of extra switches to be used for designing a FTC network be 4. The various combinations of x and y are given in Table 3.12. None of these combinations have their values of x and y equal to the threshold value of 4. The pair (0,4) provides the least reliability since the outer two stages in this case have no fault tolerance capability. Similarly, (2,0) has its middle stage without any fault tolerance. Hence, the combination (1,2) gives the best reliability. It should be noted that in (1,2) the outer stages are less reliable than in (2,0). However, the presence of two extra switches in the middle stage of (1,2) makes it more reliable than (2,0). This observation is strengthened by the fact that for the same network the pairs (1,0) and (3,0) give reliabilities of .620674 and .696902 respectively which is close to a reliability of .686708 for (2,0). Also a reliability of .837909 is obtained for the pair (1,1) and .881356 for (1,2). Therefore, it is important that a symmetry is maintained while allocating the number of extra switches in a fault tolerant Clos network.

This section provided an insight on how to optimally distribute extra switches in a fault tolerant Clos network. The results were verified for various combinations of x and y for the different networks discussed.

Chapter 4

Conclusions

4.1 Summary

The focus of this thesis was to analyze an optimal number of extra switches to convert an ordinary Clos network into a fault-tolerant one. Fault tolerant computer systems have developed as a major area of research following a massive development in multiprocessor computing. The thesis started by defining a generalized multiprocessor system which was followed by developing a fault model. A rigorous mathematical treatment of the reliability analysis was dealt with in Chapter 2 and Chapter 3. The optimal solution to the distribution of extra switches was analyzed in Chapter 3. Two methods were defined, namely

- **Mathematical Analysis:** A possible solution to the optimal selection of the number of extra switches, to get a maximum reliability, has been defined. Further research following this method has been left open.

- **Analytical Approach:** This method was used to get the results explained in the thesis.

A comparision of an ordinary Clos network was made with a fault tolerant (FTC) network and the necessity of developing the latter was given. The FTC is characterized by ease of operation and requires nominal additional hardware. The FTC model discussed has another advantage, i.e. full recovery. This is important for a Clos network, since these networks are permutation networks that require one-to-one mapping of the inputs with the outputs.

4.2 Future Research

Scope has been left for future research in the optimal distribution of extra switches for non square Clos networks. Also, some new routing algorithms can be developed which may reduce the number of extra switches used. Networks larger than the ones defined in this thesis can also be analyzed. As mentioned earlier the mathematical approach to the best values of x and y has been left open for further research.

Bibliography

[1] G. Adams and H. Siegel. "The Extra Stage Cube: Fault-Tolerant Interconnection Network for Supersystems," *IEEE Transactions on Computers*, vol. C-31, no.5, May 1982, pp. 443-454.

[2] L. Bhuyan. "A Combinatorial Analysis of Multibus Multiprocessors," *Proceedings of 1984 International Conference on Parallel Processing*, August 1984, pp. 225-227.

[3] D. Agarwal. "Testing and Fault Tolerance of Multistage Interconnection Networks," *Computer*, April 1982, pp. 41-53.

[4] G. Adams, D. Agarwal and H. Siegel. "A Survey and Comparision of Fault-tolerant Multistage Interconnection Networks," *Computer*, June 1987, pp. 14-27.

[5] ———. "Modifications to Improve the Fault Tolerance of the Extra Stage Cube Interconnection Network," *Proceedings of the 1984 International Conference on Parallel Processing*, 1984, pp. 169-173.

[6] C. Clos. "A Study of Non-blocking Switching Networks," *Bell Systems Technical Journal* vol.32, no. 2, March 1953, pp. 406-424

[7] J. Carpinelli. *Interconnection Networks: Improved Routing Methods for Clos and Beneš Networks,* Ph. D. Thesis, Rensselaer Polytechnic Institute, Troy, NY, August 1987.

[8] L. Ciminiera and A. Serra. "A Connecting Network with Fault Tolerance Capabilities," *IEEE Transactions on Computers*, vol C-35, no. 6, June 1986, pp. 578-580.

[9] D. Dias. "Packet Switching Interconnection Networks for Modular Systems," *Computer*, December 1981, pp. 43-53.

[10] T. Feng. " A Survey of Interconnection Networks," *Computer*. December 1981, pp. 12-27.

[11] T. Feng and C. Wu. "Fault-Diagnosis for a Class of Multistage Interconnection Networks," *IEEE Transactions on Computers*, vol. C-30, no. 10, October 1981, pp. 743-758

[12] I. Gazit and M. Malek. "Fault Tolerance Capabilities in Multistage Network-Based Multicomputer Systems," *IEEE Transactions on Computers*, vol. 37, no. 7, July 1988, pp. 788-798.

[13] L. Kinny and R. Arnold. "Analysis of a Multiprocessor System with a Shared Bus," *Proceedings of the 5th Annual Symposium on Computer Architecture*, April 1978, pp. 89-95.

[14] A. Pages and M. Gondran. *System Reliability: Evaluation and Prediction in Engineering*, Springer-Verlag, New York, 1986.

[15] H. Nassar *Fault Tolerant Interconnections for Multiprocessor Systems* Ph.D. Thesis, New Jersey Institute of Technology, Newark, New Jersey, May 1989.

[16] S. Reddy and V. Kumar. "On Fault Tolerant Interconnection Networks," *Proceedings of the 1984 International Conference on Parallel Processing*, 1984, pp. 155-164.

[17] J. Shen and J. Hayes. "Fault-Tolerance of Dynamic-Full Access Interconnection Networks," *IEEE Transactions on Computers*, vol C-33, no. 3, March 1984, pp. 241-248.

[18] P. Tobias and D. Trindade. *Applied Reliabilty*, Van Nostrand Reinhold, New York, 1986.

Appendix

```
#ınclude<stdio.h>

double comb();
double power();
int k,m;
maın ()

{
    double R, ft, st, c, ri, r1;
      int x, y, i;
       float ft1;
       float r;
   r = .01;
     /* printf("Enter the value of x:");
      scanf("%d", &x);
            printf("Enter the value of y:");
   scanf("%d", &y);*/
     prıntf("k\t  m\tx\ty\t2*x+y\t   R\n");

for(k =3 ; k <9 ; k++)
 for(m = 3; m <9; m++)
  for(x = 0; x < 9; x++)
   for(y = 0; y < 9; y++)
   {

     ft = 0; ft1 = 0;
       for(ı= x+1;  i<=k+x; ı++)

         {
            c = comb(k+x, i);
              ri = power(r,i);
        r1 = power(1-r, k+x-ı);
     ft1 = c*ri*r1;
    ft  = ft+ft1;
          }
     ft = 1 - ft;
               ft = power(ft,2);
     st = 0; ft1 = 0;
            for(ı = y+1; ı<= m+y; ı++)

          {
            c = comb(m+y, ı);
             ri = power(r,i);
      r1 = power(1-r, m+y-i);
             ft1 = c*ri*r1;
    st = st+ft1;

   }
             st = 1- st;
             R = ft*st;

      prıntf("%d\t %d\t%d\t %d\t %d\t  %f\n", k,m,x,y,(2*x+y),R);
     }
    }

/* end main */

double comb(a,b)
int a,b;

{
   int e, f, g;
   double h;

       e = factorial(a);
       f = factorıal(b);
       g = factorial(a-b);
       h =(double)e/(f*g);

       return(h);
 }

double power(x,y)
float x;
int y;

{
       float j;
        int i;
         j = 1.0;

          for(ı =1; i<=y; i++)
            j =ȷ*x;
             return(ȷ);
 }

factorıal(x)
int x;

{
       int ı,j;
         j =1;
```

```
           for(i =1; i<=x; i++)
              j =j*i;
      return(j);
}
```