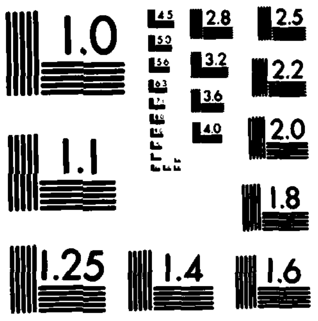
AD-A171 287 A COMMUNICATION BANDWIDTH MODEL FOR SHUFFLE-EXCHANGE AND AUGMENTED SHUFFL..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH C R BISBEE 1986 1/2

UNCLASSIFIED AFIT/CI/NR-86-115D F/G 17/2 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

AD-A171 287

1

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/CI/NR 86-115D	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Communications Bandwidth Model for Shuffle-Exchange and Augmented Shuffle-Exchange Interprocessor Communication Networks		5. TYPE OF REPORT & PERIOD COVERED ~~THESIS~~/DISSERTATION
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Charles Robiou Bisbee III		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: Auburn University		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433-6583		12. REPORT DATE 1986
		13. NUMBER OF PAGES 118
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLAS
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED		DTIC ELECTE AUG 28 1986 S D
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		B
18. SUPPLEMENTARY NOTES APPROVED FOR PUBLIC RELEASE: IAW AFR 190-1		[illegible] LYNN E. WOLAVER 8 Aug 86 Dean for Research and Professional Development AFIT/NR
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) ATTACHED.		

DTIC FILE COPY

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

5

DISSERTATION ABSTRACT

A COMMUNICATIONS BANDWIDTH MODEL FOR SHUFFLE-EXCHANGE AND AUGMENTED SHUFFLE-EXCHANGE INTERPROCESSOR COMMUNICATION NETWORKS

Charles Robiou Bisbee III

Doctor of Philosophy, June 6, 1986
(M. S., Stanford University 1971)
(B. S., U. S. Air Force Academy 1970)

128 Typed Pages

Directed By Victor P. Nelson

A failure dependent bandwidth model for shuffle exchange (S/E) and augmented shuffle exchange (S/E+) interconnection networks is presented. The models are based on probabilities of either data or address mode failures for the individual binary switches which comprise the SE or SE+ network. The model gives the expected bandwidth as a function of the probability of failures in these switches. The model, which is consistent with those previously published when the probability of failure is zero, is first developed for the S/E network. This model is extended to the S/E+ network by developing a special model for the input stage of the S/E+ network and then proving that, to within a close ɛpproximation, the

8 C

conditions necessary for the S/E model hold at the outputs of first stage of the S/E+. The model is verified using a computer simulation. An example is presented which demonstrates use of the model to predict the effects of several fault tolerance schemes on the bandwidth of these networks. The model demonstrates that, when used as a reliability enhancement, the extra stage of the S/E+ causes a reduction in bandwidth as compared to an S/E network. Thus the S/E+ network increases the probability that any single processor-memory connection can be supported at the expense of network throughput. The primary uses of the bandwidth models presented here are as a network design parameter and as a measure to evaluate the cost effectiveness of proposed, switch level, reliability enhancements.

A COMMUNICATIONS BANDWIDTH MODEL FOR SHUFFLE-EXCHANGE AND AUGMENTED SHUFFLE-EXCHANGE INTERPROCESSOR COMMUNICATION NETWORKS

Charles Robiou Bisbee III

Certificate of Approval:

C. C. Carroll

C. C. Carroll
Professor
Electrical Engineering

[illegible]

V. P. Nelson, Chairman
Associate Professor
Electrical Engineering

W. N. Hudson

W. N. Hudson
Professor
Mathematics

[illegible]

M. E. Baginski
Assistant Professor
Electrical Engineering

Warren W. Brandt
Acting Dean
Graduate School

A COMMUNICATIONS BANDWIDTH MODEL FOR SHUFFLE-EXCHANGE AND AUGMENTED SHUFFLE-EXCHANGE INTERPROCESSOR COMMUNICATION NETWORKS

Charles Robiou Bisbee III

A Dissertation

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Doctor of Philosophy

Auburn, Alabama

June 6, 1986

VITA

Charles Robiou Bisbee III, son of Charles R. Bisbee Jr. and Elizabeth (Roberts) Bisbee, was born August 16, 1948 in Jacksonville, Florida. He graduated from Jesuit High School, Tampa, Florida in 1966. In June, 1966, he entered the United States Air Force Academy, where, in June 1970, he received the degree of Bachelor of Science in Electrical Engineering. In August, 1970, he entered Stanford University and in June, 1971, he received the degree of Master of Science (Electrical Engineering). In July, 1971, he entered U. S. Air Force Pilot Training at Williams Air Force Base, Arizona. From June, 1972, until January, 1979 he served as a officer and tactical fighter pilot, U. S. Air Force. In January, 1979, he entered the U. S. Air Force Experimental Test Pilot School, Edwards AFB, California. In December, 1979, he was assigned as an experimental test pilot at Eglin AFB, Florida. He returned to graduate studies in August, 1983. He married Dayna, daughter of Owen C. and Phyllis (Mathews) Davis in December, 1975. They have two sons, Joseph Owen and Charles Robiou, and one daughter, Chinae Roberta.

DISSERTATION ABSTRACT

A COMMUNICATIONS BANDWIDTH MODEL FOR SHUFFLE-EXCHANGE AND AUGMENTED SHUFFLE-EXCHANGE INTERPROCESSOR COMMUNICATION NETWORKS

Charles Robiou Bisbee III

Doctor of Philosophy, June 6, 1986
(M. S., Stanford University 1971)
(B. S., U. S. Air Force Academy 1970)

128 Typed Pages

Directed By Victor P. Nelson

A failure dependent bandwidth model for shuffle exchange (S/E) and augmented shuffle exchange (S/E+) interconnection networks is presented. The models are based on probabilities of either data or address mode failures for the individual binary switches which comprise the SE or SE+ network. The model gives the expected bandwidth as a function of the probability of failures in these switches. The model, which is consistent with those previously published when the probability of failure is zero, is first developed for the S/E network. This model is extended to the S/E+ network by developing a special model for the input stage of the S/E+ network and then proving that, to within a close approximation, the conditions necessary for the S/E model hold at the outputs of first stage of the S/E+. The model is verified using a computer simulation. An example is presented which demonstrates use of the model to predict the effects of several fault tolerance schemes on the bandwidth of these networks. The model demonstrates that, when used as a reliability enhancement, the extra stage of the S/E+ causes a reduction in bandwidth as compared to an S/E network. Thus the S/E+ network increases the probability that any single processor-memory connection can be supported at the expense of network throughput. The primary uses of the bandwidth models presented here are as a network design parameter and as a measure to evaluate the cost effectiveness of proposed, switch level, reliability enhancements.

TABLE OF CONTENTS

LIST OF TABLES

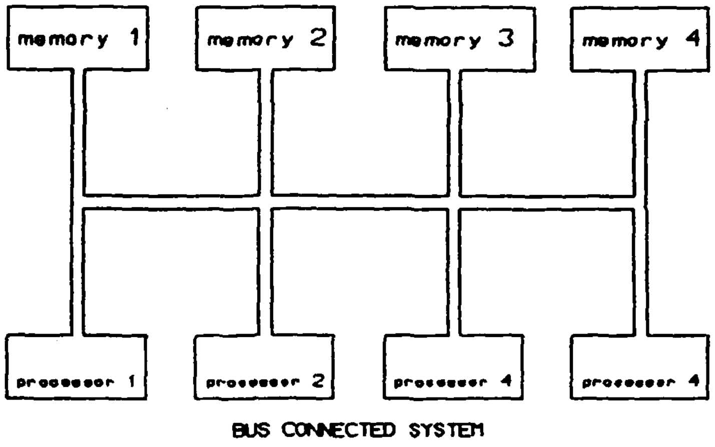
LIST OF FIGURES

I. INTRODUCTION

Connecting Networks

The computational capacity of modern computers, while immense by the standards of even a decade ago, fails to meet the requirements of many currently relevant problems. Until the recent past, advances in computational capacity were gained by increasing the speed and decreasing the size of the components from which computers are constructed. The gains in computational capacity which can be expected as a result of further advances in semiconductor technology, while important, cannot alone meet the growing requirements of modern problems.

The most promising approach for the development of the next generation of computers lies in the development of large parallel processing arrays. Such processors are composed of a large number of individual processing elements which communicate over an interconnection and communication network (ICN). As illustrated in figure 1, the ICN may be as simple as a single bus structure, in which a communication path can only be established between a single pair of elements at a time, or as complicated as a full



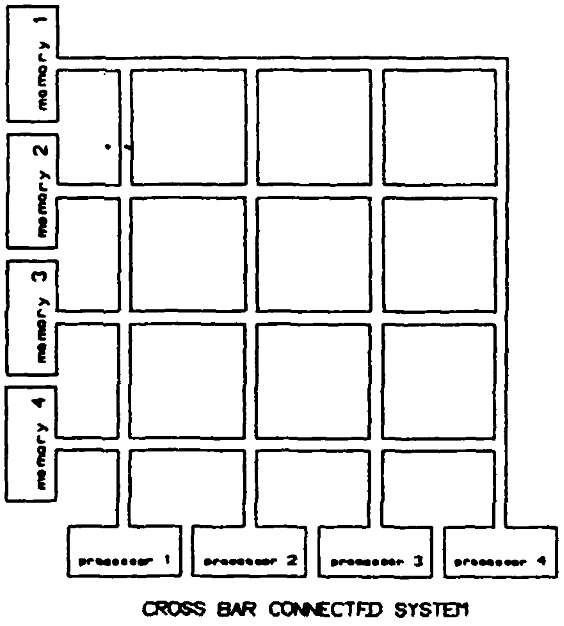


Figure 1 -- Cross Bar and Bus Connected Systems

crossbar network which allows a communication path to be established between any free pair of elements.

The single bus structure is simple and grows in complexity linearly as the number of processing elements attached to it increases. The communication capacity, however, is not sufficient for meaningfully sized processor arrays. The full crossbar on the other hand, provides a high communication capacity but grows as the square of the number of connected elements. For large processor arrays connected by a crossbar system, the cost and complexity of the ICN dominate the system. Simpler ICNs, which provide reasonable communication capacity and whose growth is logarithmic with the number of processing elements have been proposed by several researchers. Among these are the indirect binary n cube[24], the omega network [17], the regular banyan [10], and the shuffle-exchange network[31], some of which are illustrated in figure 2.

Previous Work

The study of connecting networks and their switching capabilities found initial importance in the telephone switching network. An excellent summary of this work was written by Benes[3] in 1965. Stone[31] first proposed the perfect shuffle inteconnection pattern as a useful permutation generator for use in parallel processing applications. In 1975 Lawrie[17] proposed and analyzed the

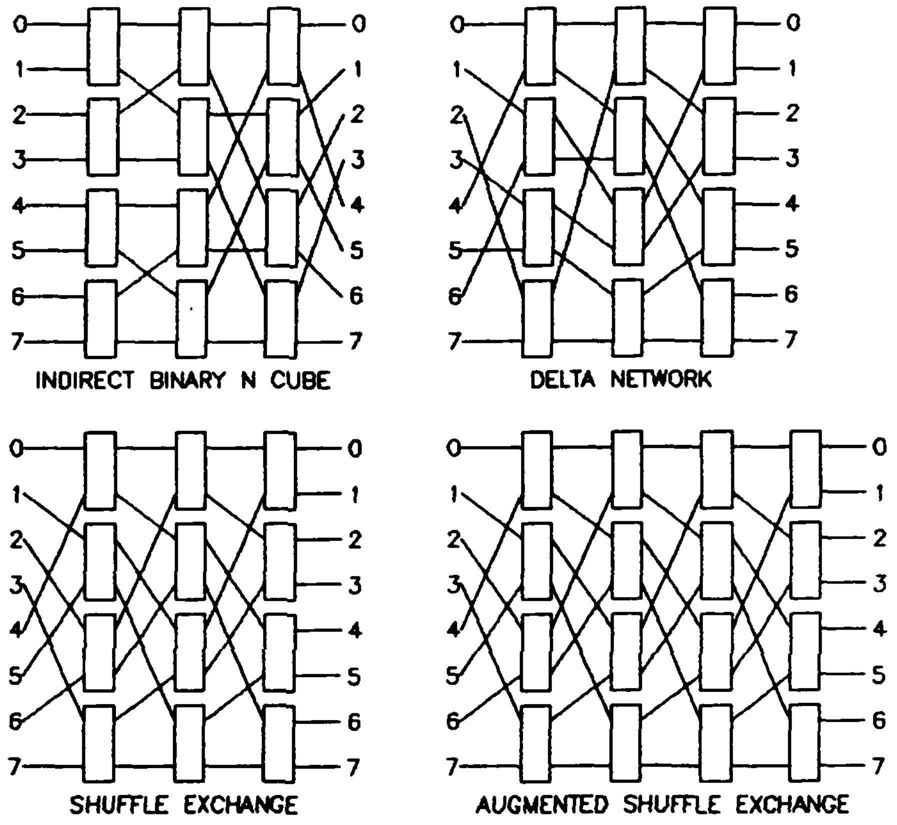


Figure 2 — Examples of Binary Cross Bar ICNs

capabilities of the omega network. This network was designed to access and distribute the inputs and outputs of N processing elements over N separate memories so as to facilitate parallel vector computations. The omega network was made up of $\log_2 N$ stages. Each stage consisted of N/2 binary (2 input - 2 output) crossbar switches. The stages were interconnected using a perfect shuffle wiring pattern. The omega network is topologically equivalent and the name is now synonymous with the shuffle-exchange network which will be discussed in this paper. Lawrie's work was a major contribution in that it not only described the network's topology but also analyzed the network in terms of the allowable permutations as applied to the particular problem of data access in a single instruction multiple data (SIMD) type machine. Subsequently a number of researchers have proposed networks which are similar in structure and topography but which appear to offer advantages for particular applications. Notable among these were the indirect binary n cube by Pease[24] and the delta by Patel[23]. Pease proposed his network for application to multiple instruction multiple data (MIMD) machines. Patel's paper, while important at the time because it presented a new network, is more important in that he developed a performance analysis in terms of the bandwidth of his network for MIMD applications. Pease defined bandwidth as the number of simultaneously active connections the network could support. This is the definition that will be used in this work.

In 1980 and 81 Feng and Wu[37-38] and Parker[22] proved that many of the previously proposed networks, including the omega or shuffle-exchange, the binary n-cube, the data manipulator, the flip network, the delta network, the regular banyan and one form of the Clos network, were topologically equivalent. This result is important in that given that this is true, performance analysis done for any network in the class is generally applicable. In 1983 Bhuyan[5] generalized the theory of these networks by analyzing a network composed of mixed radix crossbar switches as opposed to the fixed radix, binary crossbars generally used to form the networks prior to this. In 1983 Padmanaban[21] studied the addition of an extra stage to these networks. The extra stage provided a reliability enhancement by providing r paths, independent except for the first and last stages, between any pair of ports, where r is the radix of the crossbars used to form the network. This scheme when applied to the S/E ICN is the S/E+ which will be analyzed in this work.

Several researchers have proposed additional enhancements to the basic network which are designed to improve the fault tolerance of the networks. Adams[1] proposes the use of bypass stages for the first and last stage of the network. Tzeng[34] proposes the addition of intra-stage links to reroute misdirected communication links. Kumar[16] suggests minimizing the network by removing inter-stage links not used for required permutations in an SIMD machine.

Many of the above cited references contain performance analyses. The most notable are Patel[23] and Pease[24]. Other papers exist which focus on performance analysis as opposed to network topology. Dias[8] provides a performance analysis of a buffered, packet switched delta network in a fault free state. Thanawastien[33] provides a Markov chain traffic model of a fault free shuffle-exchange network. Kruskal[15] studies the performance of multistage networks in a packet switching mode. Cherhassky[7] provides equations that can be used to calculate the probability that a given pair of elements can communicate using an ICN whose switches fail in a data or broken connection mode. Shen[27-28] provides a method for determining the sets of switch failures which are critical in the sense that they disconnect the network into two or more disjoint sets of processing elements.

Several experimental multiprocessor systems have been built which utilize the above networks for system interconnection. The Auburn Fault-Tolerance Distributed Computing Laboratory currently includes a four processor, four memory parrallel system which utilizes a circuit switched 4x4 shuffle exchange network as the system ICN[20]. This system is designed for experiments in fault tolerance and system software as related to multiprocessor systems. The Texas Reconfigurable Array Processor(TRAC) consists of a 16 processors and 81 memories / IO ports connected by a 4 stage banyan network utilizing switches with 2 inputs and three outputs each[14,25,26]. This ICN operates in a mixed circuit switched, packet switched mode. The TRAC system is designed for experiments in software and hardware integration on complex, multiprocessor systems. The NYU 'Ultracomputer' group has conducted extensive studies on the architectural requirements for a 4096x4096 system utilizing an omega ICN in the packet switched mode[9,11,12]. An experimental 8x8 system is currently being implemented.

The properties of these networks have been investigated by many researchers. Their efforts have concentrated, however, on either analyzing the throughput and permutation capabilities of such networks in the fully functional state or on designing and analyzing the fault tolerant capabilities of the network. Little if any work has been done in the area of reliability measures for these networks. Such measures are needed to evaluate the effectiveness of fault tolerant system designs employing these networks as ICNs. The purpose of this work is to develop a model which predicts the bandwidth of the ICN as a function of failure parameters for the switches which comprise the ICN. Such a model can then be used to evaluate the effects of proposed reliability enhancements on the system bandwidth. Models which relate ICN performance measures to failure characteristics of the ICN switches are critical for the design of fault tolerant systems and for evaluating the cost and performance effects of proposed reliability measures.

Fault Tolerance and Reliability

Fault tolerance is defined as 'the correct execution of a specified algorithm in the presence of defects'[29].

The repeated and regular nature of large scale processing arrays, coupled with the relatively high probability that one or more of the system elements is defective at any time, demands that such systems be designed to tolerate faults. If the overall system is to tolerate faults and continue to operate correctly, the ICN must be capable of functioning in the presence of internal faults. Thus, in terms of the ICN, fault tolerance is defined as the ability to meet system communication demands in the presence of failures internal to the ICN.

Reliability as a function of time is defined as 'the conditional probability that the system has survived the interval [0,t], given that it was operational at time t=0' [29]. In terms of the ICN, reliability can be defined as the probability that the ICN is able to meet the communication requirements of the system at time t, given that the ICN was fault free at time 0. Such a definition requires one to define 'communication requirements' for a system of parallel processing elements and their associated memory units. Two measures of communication capability, bandwidth and connectivity, can be used to specify the requirements of such a system. Cherhassky et. al.[7] have developed models which predict the probability that a communication channel can be established between a randomly selected pair of processing elements for a class of the above ICNs, in the presence of data type faults in the underlying switches that comprise the ICN. Several researchers [23,33] have developed bandwidth models for the ICNs mentioned above. These models are, however, only valid in the fault free case. The purpose of this work is to develop and present a model which can be used to predict the bandwidth of shuffle-exchange (S/E) and augmented shuffle-exchange (S/E+) interconnection networks, given a failure model for the switches that comprise these networks.

S/E and S/E+ Networks

The binary crossbar S/E network is composed of $\log_2 N$ identical stages, where N is the number of processors and memories connected to the network. Each stage consists of N/2 binary crossbar switches. The stages are interconnected by a perfect shuffle wiring pattern. Such a network admits a simple, distributed control algorithm. Each switch within the network is set according to the corresponding digit in the binary number of the memory desired. Figure 3 shows an 8x8 S/E network. The bold lines indicate the switch settings required for processor 5 to access memory 3. At each stage within the network, the corresponding switch is set so that, if the corresponding bit of the desired memory address is 0 then the input is connected to the upper output. If the corresponding bit is 1, the input is connected to the lower output. Figure 4 shows the four possible input output combinations that can

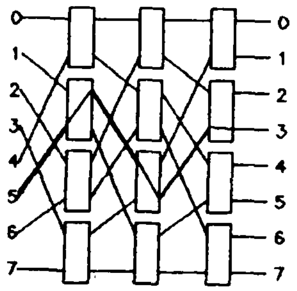


Figure 3 - 8 x 8 SE

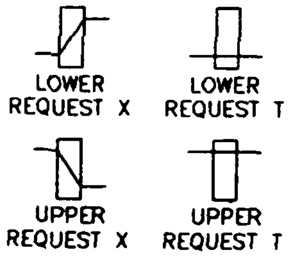


Figure 4 -- Allowable Requests

be requested. In this model the switch itself can only support one of two possible configurations at any one time. These are the X and T states, and are shown in figure 5. In the more general case the switches can also support a broadcast mode in which an input is connected to more than one output. The analysis of faults in a system which utilizes the broadcast mode is beyond the scope of this work.

When two requests for different configurations are present at the inputs, one and only one can be satisfied. The other is blocked and cannot be completed until the switch is released. In this analysis it is assumed that the arbitration between conflicting inputs is random with either of the switch inputs equally likely to have its request satisfied in a given cycle.

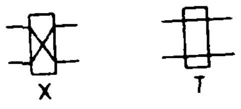


Figure 5 - Allowable Switch States

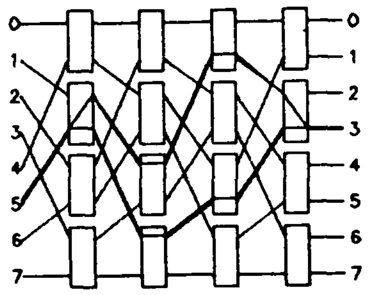


Figure 6 - 8x8 SE+ Processor 5 to Memory 3 Connection

Figure 6 shows an 8x8 S/E+ network. This network is identical to the S/E network except that it has an extra stage[21]. In this model the extra stage is used only for fault tolerance. The base S/E network provides only a single path from any input to any output, while the S/E+ network provides two paths from any input to any output. These paths are disjoint with the exception of the first and last stage switches. The bold paths show the two paths that can be used to connect processor 5 to memory number 3.

A number of different control strategies can be developed for the S/E+ network. These can be designed to provide either performance improvement or increased fault tolerance as compared to the base network. In this study, it is assumed that the extra stage is used only for reliability enhancement. The control strategy considered in this study is as follows: Each first stage switch is controlled with the first bit of the corresponding processor ID until an error is detected. Once any error is detected along the path to a memory, the first bit of the address used to access that memory is inverted for all subsequent access attempts for that processor-memory pair. Thus all memory requests will initially attempt to set the first column switches in the T, or through' configuration. As failures which produce errors in accessing a memory occur, the processors will change the requests going to that memory so as to request an X configuration at the first stage. The remaining columns in the network are controlled in the same manner as the base S/E network. Suca a control strategy requires little or no intelligence in the ICN and limits the need for substaintial logic in each switch.

Importance of a Bandwidth Model

Designers of large scale, fault tolerant, parallel systems must make many design choices regarding the ICN to be used. Currently available design tools are not sufficient to assist and guide those choices. The equations derived by Cherhassky[7] provide a guide to calculating the probability that a random pair of elements can successfully communicate over an ICN which may have experienced data mode failures. The bandwidth model provided by Patel[23] can only be used to estimate the ICN bandwidth prior to the occurrence of faults within the ICN. The model derived by Shen[27,28] can be used to identify the faults which can disconnect the system but cannot determine the effect of these or other faults on communications bandwidth. With the exception of Cherhassky's equation none of these tools estimates the performance of the ICN in the presence of expected failures.

In chapter 2 a bandwidth model for the SE network is presented. This model gives the expected bandwidth as a function of failure parameters of the component switches of the network. In chapter 3 this model is extended to cover the S/E+ network. In chapter 4, in order to verify the model, comparison of the results of the model and a simulation of an 8x8 S/E+ network are presented. In chapter 5 the model is used to investigate reliability enhancements as they effect the ICN bandwidth. In chapter 6 a summary is presented and some topics for further research in this area are discussed.

II. FAILURE DEPENDENT BANDWIDTH IN SHUFFLE EXCHANGE NETWORKS

A model, which can be used to predict the failure dependent bandwidth of S/E networks, composed of binary crossbar switches, in the presence of either address or data mode faults in these switches, is now presented. For the purposes of this paper, bandwidth is defined as the average number of active connections which are simultaneously supported by the network. The purpose of this model is to provide reasonable estimates of the expected bandwith as a function of failure parameters associated with the component switches of the ICN. The model can then be used to estimate the cost effectiveness of reliability enhancements as related to the ICN bandwidth.

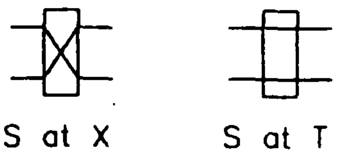


Figure 7 Stuck at Faults

Binary Crossbar Fault Model

A static fault model, which classifies all faults in the ICN as either address mode or data mode faults in the binary crossbar switches, will be used. The address mode fault model has been used by other researchers[3,23] and can model a large number, though certainly not all, of the faults possible in an S/E network. The address mode failure model states that the switch fails in one of two possible configurations. These are shown in figure 7 and represent a condition in which the switch is frozen in one of its two possible configurations. As a result, the switch no longer responds to input requests but rather routes inputs to outputs in a fixed pattern. Given that an address mode failure has occurred, it is assumed that either failure configuration, S at T or S at X, is equally likely. Data mode faults result in the switch being unable to correctly transmit any information. It is certainly possible for a switch to undergo a data mode failure subsequent to an address mode failure. In this case it is classified as a data mode failure. Thus, at all times, the total probability of failure is equal to the probabilty of an address mode failure plus the probability of a data mode failure. Further it is assumed that the failure probabilities for the switches are known or can be calculated.

Processor Input Assumptions

Analysis of these networks is based on the following assumptions:

1) The system contains $N = 2^k$ processors and $N = 2^k$ memory modules that are statistically identical in each group.

2) The processors and memories are connected by either a k column S/E network or a k+1 column S/E network as described previously.

3) All two input, two output routing switches are statistically identical. Only the internal path configurations described in fig. 5 are allowed.

4) Circuit switching is used. A processor is held in a wait state if a requested access cannot be completed.

5) Conflict resolution at each routing switch is unbiased. Given that two conflicting requests are present at the inputs to a switch, each request has probability 1/2 of being satisfied.

6) Memory requests issued by each processor are independent and are uniformly distributed over the N memory modules.

7) In the case of the S/E+ network, each processor maintains a single bit history of the accesses made to each memory module. This history specifies whether an error has occurred during an attempted

access to that memory. If no error has occurred then the processor requests a T setting of the first column switch for all accesses to that memory. If an error has occurred during an access to that memory then the processor will request an X setting in the first column switch for all subsequent accesses to that memory.

8) During each cycle each processor will submit a request with probability designated by m_0.

9) Error detection is perfect. Only those memory accesses that are correctly routed reach the desired memory and thus, only these are considered in the bandwidth computation.

10) Failures among the switches that comprise the network occur at a very low frequency with respect to processor memory requests.

These assumptions are necessary to make the model tractable. The limitations they impose in relation to actual systems should, however, be understood. Assumptions 6 and 8 imply that blocked requests are not resubmitted during the next cycle but rather are ignored. Simulations performed by others[22,23,37,38] indicate that, for fault free systems, this assumption does not significantly alter the results. These assumptions also imply that processors continue to attempt to access memories that cannot be reached by reason of multiple failures. It is reasonable to assume that some system reconfiguration will occur after a failure is detected and that this reconfiguration will restrict the set of memories that may be accessed by a processor to those that can be reached. The amount of error induced by this assumption is not known but should be small for times where the probability of failure is reasonably low. For times where the probability of failure for an individual switch is high, the bandwidth derived here should be considered a lower bound.

Assumption 9 states that error detection is perfect. It is not reasonable to expect perfect error detection in a faulty network. The probability that an undetected error occurs can be reduced by employing both hardware error detection and periodic software testing. Undetected errors will decrease the effective bandwidth. The model treats undetected errors as successful accesses and thus overestimates slightly the effective bandwidth.

Assumption 10 implies that, for the S/E+ network, the processor can be assumed to know whether an error has occurred on the primary path (T connection of first stage switch) prior to making the request. This is equivalent to ignoring the requests that actually discover the error in the bandwidth calculations. If, on average, many accesses occur between failures this will have a negligible effect on the bandwidth. This should be true for all practical systems.

Analysis of The S/E Network

The basic S/E network has only a single possible path for each processor memory pair. As a result of this, the independence assumption for the processor requests and the statistical independence of the component switches, the events that the two inputs to any given switch are active, are statistically independent. This follows since the probability that an input is active is a function of the set of processors that could have generated the input and the set of switches that could have processed the input prior to the switch. For the two inputs of any given switch in the S/E network the sets of processors and switches that can effect one input are disjoint from the sets that can effect the other. This is illustrated in

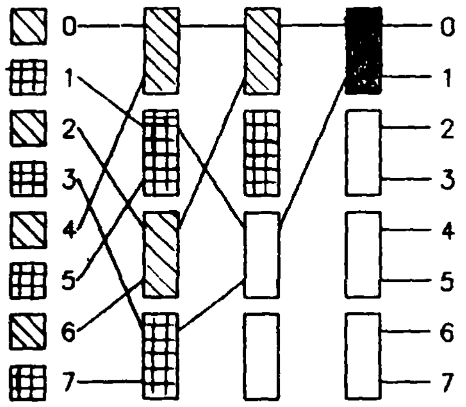


Figure 8 - Sets of Processors and Switches Effecting an Input

figure 8 which shows the sets of processors and switches that can effect the inputs to the first switch in the third column.

The bandwidth of the system can be determined by calculating the probability that a single output line is active. By symmetry this probability is the same for all output lines. The expected number of active connections then is the bandwidth and is equal to N times the probability that a single line is active. The probability that an output line is active can be be determined by k repeated evaluations of the probability that a stage output line is active given the probability that its input lines are active.

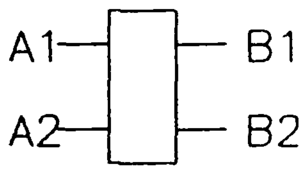


Figure 9 - Switch I/O Notation

Figure 9 illustrates the notation that will be used in the development of these equations. The inputs to a switch are labeled A and the outputs are labeled B. The event that the upper input is active is denoted by A_1 and the event that the lower input is active by A_2. The event that the upper input is active and requests the upper output is denoted by A_{11}. The event that the upper output is active is denoted by B_1. Thus A_{22} is the event that the lower input is active and is requesting the lower output(address bit is a 1).

The probability that an output line from a given switch is active can be expressed as follows:

$$P(B_1) = P(B_1 | \text{no fail})\, P(\text{no fail})$$
$$+ P(B_1 | \text{address fail})\, P(\text{address fail})$$
$$+ P(B_1 | \text{data fail})\, P(\text{data fail})$$

where the failure event probabilities are:

$$P(\text{address fail}) = p_a$$
$$P(\text{data fail}) = p_d$$
$$P(\text{fail}) = p_a + p_d = p_f$$

Now the probability that an output is active, given that the switch has failed in address mode, can be expressed as

$$P(B_1 | \text{address fail}) = P(B_1 | S \text{ at } X)\, P(S \text{ at } X | \text{addr fail})$$
$$+ P(B_1 | S \text{ at } T)\, P(S \text{ at } T | \text{addr fail})$$

and since, given a failure either mode is equally likely, then

$$P(S \text{ at } X | \text{address fail}) = P(S \text{ at } T | \text{address fail}) = 0.5$$

For the purposes of the model it is assumed that any request that is misrouted as a result of a failure of a switch is blocked in that switch. In reality this would probably be detected in the next stage. Since perfect error coverage has been assummed, the request will be blocked in the network. In addition it will, with probability one, be detected at the next unfailed stage to which it is incorrectly routed and then discarded. As a result it will not effect arbitration and therefore blocking at an operational switch. Thus there is no loss in generality by assuming that the request is blocked at the stage in which the error occurs. Given this, then

$$P(B_1 | S \text{ at } T) = 0.5P(A_1)$$

$$P(B_1 | S \text{ at } X) = 0.5P(A_2)$$

$$P(B_1 | \text{data fail}) = 0$$

Let us represent the probability that an input is active by m_{in}. And by symmetry

$$P(A_1) = P(A_2) = m_{in}$$

Then the probability that the upper output is active can be written as

$$P(B_1) = P(B_1 | \text{no fail})(1-p_f) + 0.5m_{in}p_s$$

Now

$$P(B_1 | \text{no fail}) = P(A_{11} \cup A_{21})$$

That is, the output will be active if either input is active and requests that output. Then

$$P(A_{11} \cup A_{21}) = P(A_{11}) + P(A_{21}) - P(A_{11} \cap A_{21})$$

Now, because the requested memories are independent and uniformly distributed, so also are the control bits at each switch and

$$P(A_{11}) = P(A_{11} | A_1)P(A_1) = 0.5m_{in}$$

$$P(A_{21}) = P(A_{21} | A_2)P(A_2) = 0.5m_{in}$$

and

$$P(A_{11} \cap A_{21}) = P(A_{11} \cap A_{21} | A_1 \cap A_2)P(A_1 \cap A_2) = 0.25m_{in}^2$$

Combining

$$P(B_1) = (m_{in}-0.25m_{in}^2)(1-p_f) + 0.5m_{in}p_a$$

This formula can be evaluated k times using m_0 as m_{in} in the first evaluation and P(B) for stage i as m_{in} for stage i+1. This will give the probability that a last stage output line is active. Notice that this formula reduces to that derived by Patel[23] when the p_f is zero. Figure 10 is a plot of the bandwidth, normalized by N, of an S/E network as a function of the probability of address mode failure for several values of the probability of data mode failure.

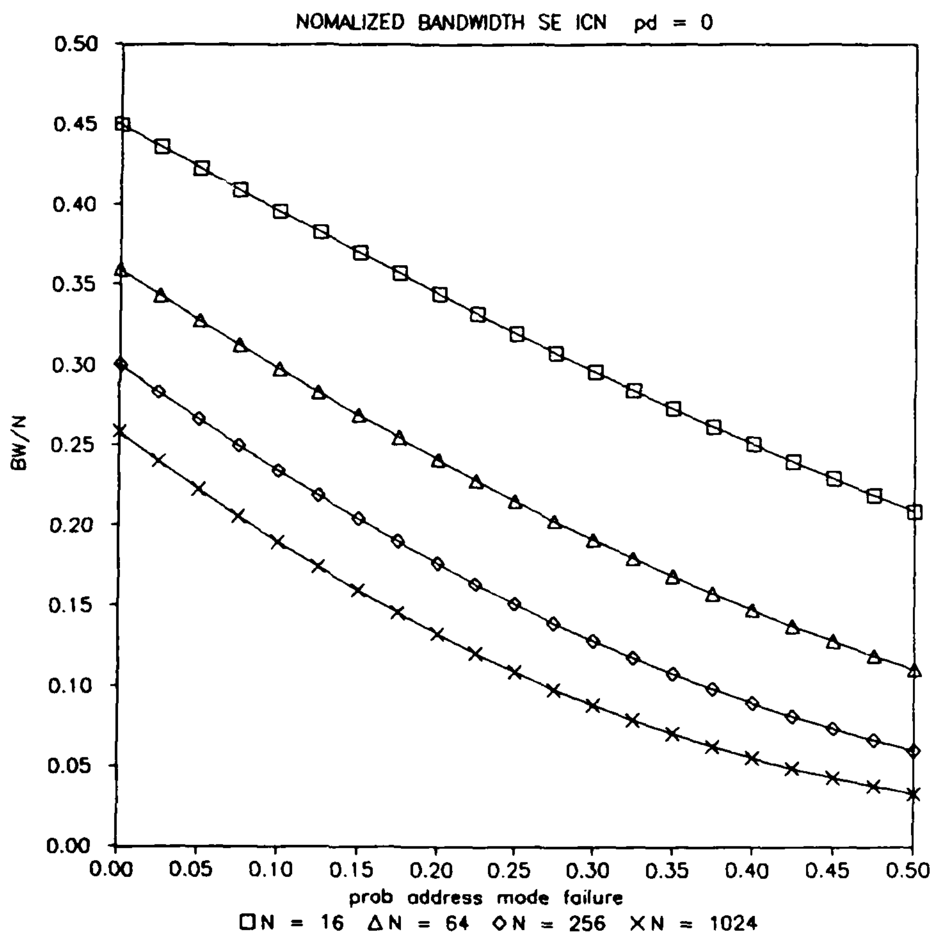


Figure 10a – Normalized Bandwidth vs P_a for P_d=0

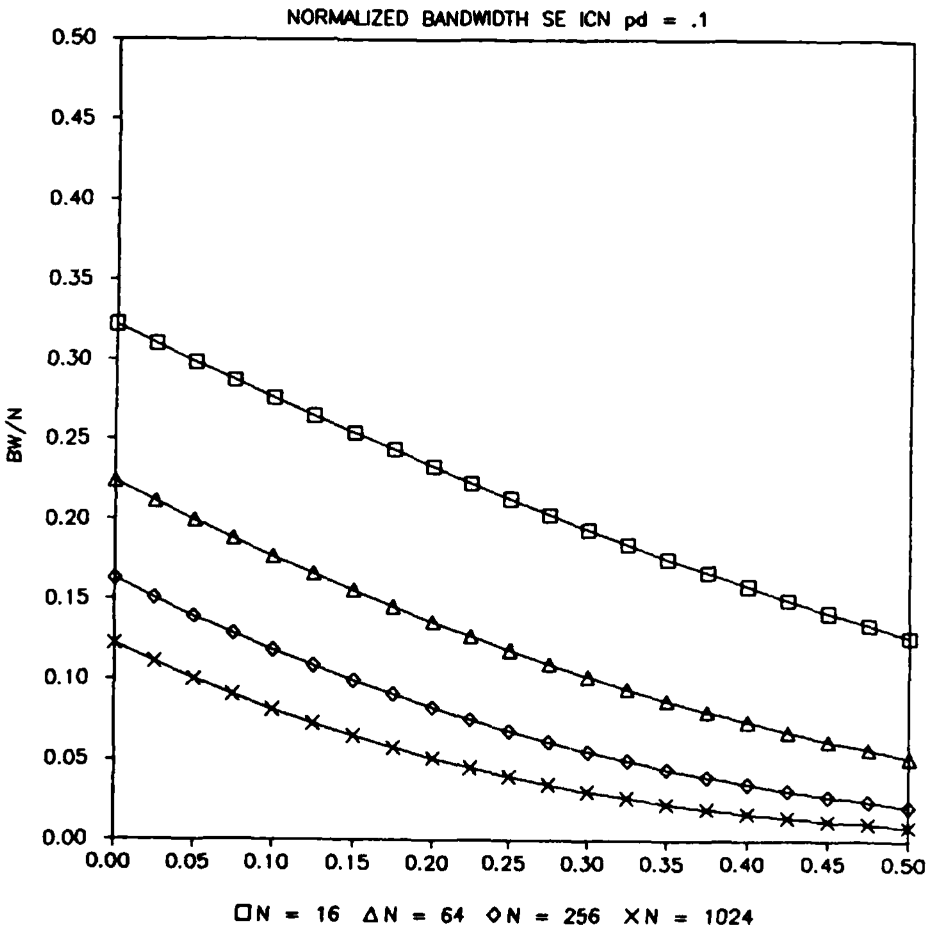


Figure 10b - Normalized Bandwidth vs P_a for P_d=0.1

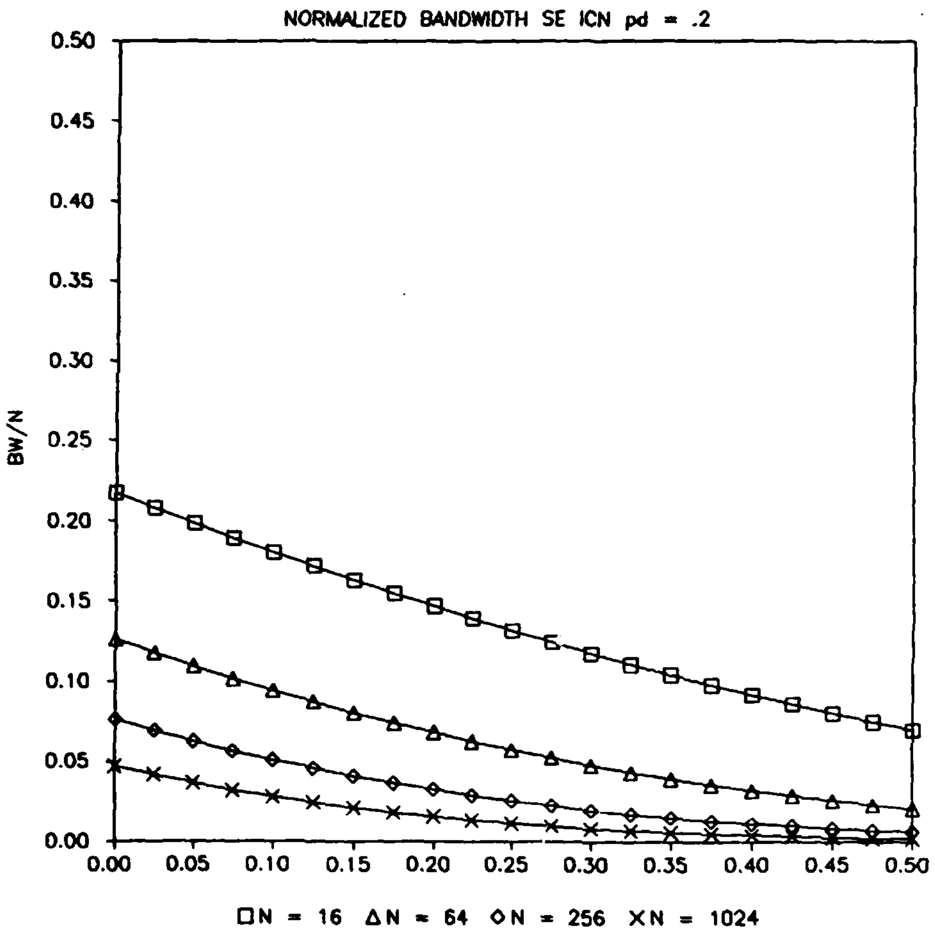


Figure 10c - Normalized Bandwidth vs P_a for P_d=0.2

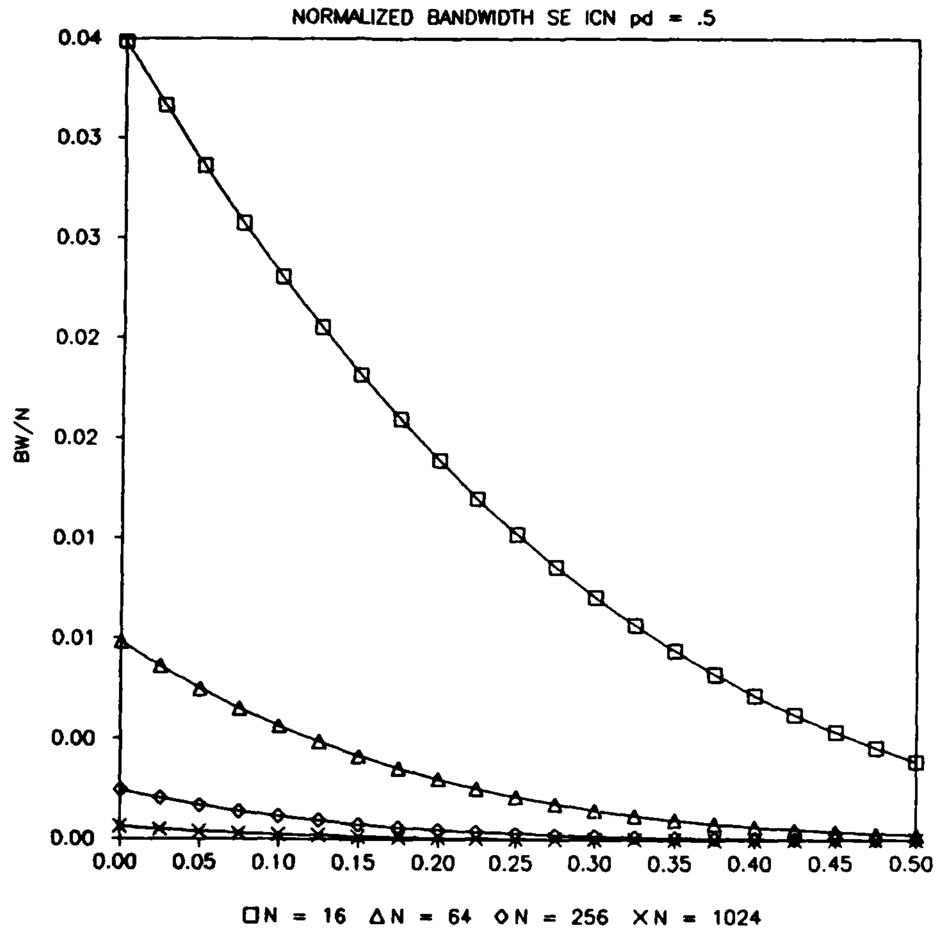


Figure 10d - Normalized Bandwidth vs P_a for P_d=0.5

III. ANALYSIS OF THE S/E+ NETWORK

The analysis of the S/E+ network provides a great deal more challenge. The situation is complicated by two facts. First the extra stage, which is refered to as stage 0, behaves differently than the stages of the S/E network. Its control bits are not independent and uniformly distributed, as are the control bits in the S/E network, but rather are dependent on the state of the network. Next, since the setting of the first stage switches depends on the failure state of the remainder of the network, it cannot be assumed that the events that the two input lines to a switch in column 1 to k are active are independent. The approach that will be used is to first calculate the probability that the output lines from stage zero are active and then show that the inputs to any first stage switch (the stage following the added stage) are essentially independent. Once this has been established, much of the above analysis can be used on the last k stages of the S/E+ network.

Probability of Active Stage 0 Outputs

Let B_{10} represent the event that the upper output of a column 0 switch (the added column) is active. The probability that this line is active will now be calculated. As in the case of the S/E network, first condition on whether the column 0 switch has failed or is operational. Thus

$$P(B_{10}) = P(B_{10}|\text{no fail})\ P(\text{no fail})$$
$$+P(B_{10}|\text{address fail})\ P(\text{address fail})$$
$$+P(B_{10}|\text{data fail})\ P(\text{data fail})$$

As before

$$P(B_{10}|\text{data fail}) = 0$$

Now, given that the column 0 switch has failed in the address mode, again condition on the configuration of the failure. This time, however, there is no basis on which to delete any requests which pass through a column 0 switch. This follows since, if there are no other failures along the path from that column 0 output to the requested memory, the memory can be accessed from either output of the column 0 switch. Thus, address mode failures in column 0 do not effect addressability unless they are coupled with other failures. Requests, which are blocked due to other failures, will be accounted for when those failures are treated. Thus

$$\begin{aligned} P(B_{10} | \text{address fail}) &= P(B_{10} | S \text{ at } X)\ P(S \text{ at } X | \text{address fail}) \\ &\quad + P(B_{10} | S \text{ at } T)\ P(S \text{ at } T | \text{address fail}) \\ &= 0.5P(A_2) + 0.5P(A_1) \\ &= m_0 \end{aligned}$$

Where m_0 is the probability that a processor submits a request in a given cycle. Thus

$$P(B_{10}) = P(B_{10} | \text{no fail column } 0)(1-p_f) + m_0 p_a$$

Now let

$$P'(\text{event}) = P(\text{event} | \text{ no fail column } 0)$$

Then

$$P(B_{10} | \text{no fail column } 0) = P'(B_{10}) = P'(A_{11} \cup A_{21})$$

and

$$P'(A_{11} \cup A_{21}) = P'(A_{11}) + P'(A_{21}) - P'(A_{11} \cap A_{21})$$

$$P'(A_{11}) = P'(A_{11} | A_1)P(A_1) = P'(A_{11} | A_1)m_0$$

$$P'(A_{21}) = P'(A_{21} | A_2)P(A_2) = P'(A_{21} | A_2)m_0$$

$$P'(A_{11} \cap A_{21}) = P'(A_{11} \cap A_{21} | A_1 \cap A_2)m_0^2$$

as before. Now let

$$P''(\text{event}) = P'(\text{event} | A_1 \cap A_2)$$

That is P''of an event is the probability of that event given that the two inputs of interest are active and that the column 0 switch has not failed.

Now given that A_1 is active, the event A_{11} will occur only if there is no error along the primary path (the one selected by the T setting of the column 0 switch) for the A_1 input and

$P'(A_{11}|A_1) =$

$P''(A_{11}) = P(\text{no error exists in the selected primary path columns 1 to k})$

$= (1-0.5p_a-p_d)^k$

Given that A_2 is active, the event A_{21} will occur only if there is at least one error on the primary path for the A_2 input. Thus

$P'(A_{21}|A_2) =$

$P''(A_{21}) =$

$P(\text{at least one error exists in the selected primary path columns 1 to k})$

$= 1-(1-0.5p_a-p_d)^k$

To calculate the probability of the intersection of events A_{11} and A_{21}, condition on whether the A_1 and the A_2 primary paths share a last column switch. Note that they cannot share any switches in columns 1 to k-1 or there would be more than two paths from any input to output of the S/E+ network. This fact is easily proved. Thus

$$P''(A_{11}\cap A_{21}) = P''(A_{11}\cap A_{21}|\text{share last column switch})(2/N)$$
$$+P''(A_{11}\cap A_{21}|\text{last column switch not shared})(1-2/N)$$

Where 2/N is the probability that the two inputs request a pair of memories that must be accessed through the same last column switch.

Now if the two input requests are not for memories which require the sharing of a last column switch, the two paths are independent and

$$P''(A_{11} \cap A_{21} \mid \text{last col switch not shared}) = P''(A_{11})P''(A_{21})$$
$$= (1-0.5p_a-p_d)^k[1-(1-0.5p_a-p_d)^k]$$

In order to calculate the required probabilities given that a last column switch is shared condition on whether the shared switch is failed. Thus

$$P''(A_{11} \cap A_{21} \mid \text{share last col switch}) =$$
$$P''(A_{11} \cap A_{21} \mid \text{share, shared switch unfailed})(1-p_f)$$
$$+ P''(A_{11} \cap A_{21} \mid \text{share, shared switch addr fail})p_a$$
$$+ P''(A_{11} \cap A_{21} \mid \text{share, shared switch data fail})p_d$$

and, if the switch has not failed, it cannot cause an error therefore

$$P''(A_{11} \cap A_{21} \mid \text{share unfailed}) = (1-0.5p_a-p_d)^{k-1}[1-(1-0.5p_a-p_d)^{k-1}]$$

Given that the shared switch has failed in the address mode, again condition on whether the failure causes an error for the A_2 primary path. Note again that it must not cause an error for the A_1 path as the probability of the event is 0 if there is an error on that primary path. Thus

$$P''(A_{11} \cap A_{21} \mid \text{share, address fail}) =$$
$$P''(A_{11} \cap A_{21} \mid \text{share, addr fail, error } A_2)\ P''(\text{error} \mid \text{share, addr fail})$$
$$+ P''(A_{11} \cap A_{21} \mid \text{share, addr fail, no error})\ P''(\text{no error} \mid \text{share, addr fail})$$

Now given that a switch has failed in the address mode and has two active inputs, one of four equally likely conditions holds with respect to errors for those inputs. Of interest are the two of these that do not produce an error for the A_1 input. Thus

$$P''(\text{error } A_2 \cap \text{ no error } A_1 | \text{addr fail}) = 0.25$$

$$P''(\text{no error } A_2 \cap \text{ no error } A_1 | \text{addr fail}) = 0.25$$

Now, if a switch is failed in the address mode and is shared in the last column, but produces no error, then it is the same as if it were functioning and

$$P''(A_{11} \cap A_{21} | \text{share, addr fail, no error}) = (1-0.5p_a-p_d)^{k-1}[1-(1-0.5p_a-p_d)^{k-1}]$$

If it produces an error for the A_2 path but not for the A_1 path then the probability that at least one error occurs in the A_2 path is one and

$$P''(A_{11} \cap A_{21}) | \text{shared, failed, error}) = (1-0.5p_a-p_d)^{k-1}$$

Combining these equations will allow the calculation of the probability that one of the output lines from a column 0 switch is active. By symmetry this value is the same for all lines.

Joint Probability of Active Stage 0 Outputs

The above derived probability is not sufficient. It would be convenient to apply the results of the calculations done for the S/E network by using the probability that a column 0 output line is active as the m_0 in the equations derived for the S/E network, using that to represent the remainder of the S/E+ network. Recall, however, that these equations were derived under the condition that requests arriving at the two input lines are independent. This is not the case for the switches in column 1 of the S/E+ network. It is the contention of this work, however, that they are approximately independent. Given this, the assumption that they are independent will induce only small errors in the model. To prove that the dependence is indeed small, first calculate the joint probability that the two inputs to any column 1 switch are active and then calculate the covariance of indicator random variables for each of these lines. The covariance is given by

$$\mathrm{COVAR} = P(B^U_{10} \cap B^L_{10}) - P^2(B_{10})$$

where B^U_{10} and B^L_{10} represent the events that the two outputs from stage 0 switches, which are the inputs to a particular column 1 switch, are active. Now $P(B^U_{10} \cap B^L_{10})$ is calculated

First examine figure 11. The highlighted switches illustrate an important relationship. Notice that if any column 1 switch is picked and its input lines traced back to their respective column 0 switches, the other output lines from the column 0 switches both terminate at the same column 1 switch. Thus, to calculate the probabilities that two inputs to the same column 1 switch are jointly active, two sets of paths must be considered. These sets of

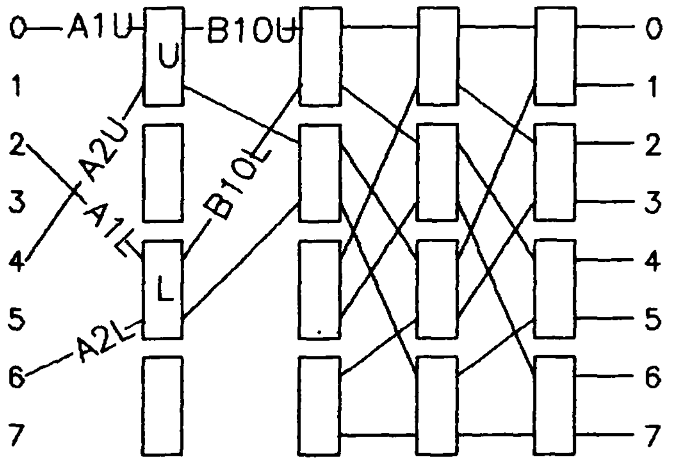


Figure 11 — Column 0 Switch Notation

paths enter column 1 through the two highlighted switches. In one set the probability that no errors exist must be determined. In the other set, the probability that at least one error exists in each of the paths in the set must be calculated. As shown in figure 11 the upper column 0 switch is marked with a U. All quantities relating to that switch will be superscripted with a U, while all quantities relating to the lower switch will be superscripted with an L. Thus $P(B^L_{10} \cap B^U_{10})$ must be calculated. Now

$$\begin{aligned}
P(B^L_{10} \cap B^U_{10}) = {} & P(B^L_{10} \cap B^U_{10} \mid \text{upper lower col 0 not failed})(1-p_f)^2 \\
& + P(B^L_{10} \cap B^U_{10} \mid \text{upper addr fail lower not failed})(1-p_f)p_a \\
& + P(B^L_{10} \cap B^U_{10} \mid \text{upper not failed lower addr fail})(1-p_f)p_a \\
& + P(B^L_{10} \cap B^U_{10} \mid \text{upper and lower addr fail})p_a^2 \\
& + P(B^L_{10} \cap B^U_{10} \mid \text{upper data fail lower not failed})(1-p_f)p_d \\
& + P(B^L_{10} \cap B^U_{10} \mid \text{upper not failed lower data fail})(1-p_f)p_d \\
& + P(B^L_{10} \cap B^U_{10} \mid \text{upper and lower data fail})p_d^2 \\
& + P(B^L_{10} \cap B^U_{10} \mid \text{upper data fail lower addr failed})p_a p_d \\
& + P(B^L_{10} \cap B^U_{10} \mid \text{upper addr failed lower data fail})p_a p_d
\end{aligned}$$

The last five terms in the above equation represent one or more data mode failures in the column 0 switches which form the input the selected column 1 switch. Given that a data mode failure has occurred in one of these switches the probability of $B^L_{10} \cap B^U_{10}$ is 0. Therefore these terms will be disregarded in the following dicussion.

Now redefine P' of an event as the probability of the event given that the stage 0 switches that affect the event are functioning. This retains the previous definition and includes, for events that depend on two column 0 switches, the following

$$P'(\text{event}) = P(\text{event} \mid \text{upper, lower col 0 not failed})$$

Then

$$\begin{aligned} P(B^L_{10} \cap B^U_{10}) = {} & P'(B^L_{10} \cap B^U_{10})(1-p_f)^2 \\ & + P'(B^L_{10}) m_0 p_a (1-p_f) \\ & + P'(B^U_{10}) m_0 p_a (1-p_f) \\ & + m_0{}^2 p_a{}^2 \end{aligned}$$

Now by symmetry

$$P'(B^U_{10}) = P'(B^L_{10}) = P'(B_{10})$$

which was derived previously. Thus the first term of the above equation is the only new value which must be evaluated. It can be expressed as follows:

$$P'(B^L_{10} \cap B^U_{10}) = P'(B^L_{10}) + P'(B^U_{10}) - P'(B^L_{10} \cup B^U_{10})$$

Again the first two terms of this equation have been calculated. The last can be expressed as

$$\begin{aligned} P'(B^L_{10} \cup B^U_{10}) & = P'([A^U_{11} \cup A^U_{21}] \cup [A^L_{11} \cup A^L_{21}]) \\ & = 1 - P'(\sim[A^U_{11} \cup A^U_{21} \cup A^L_{11} \cup A^L_{21}]) \\ & = 1 - P'([A^U_{12} \cup \sim A^U_1] \cap [A^U_{22} \cup \sim A^U_2] \cap [A^L_{12} \cup \sim A^L_1] \cap [A^L_{22} \cup \sim A^L_2]) \end{aligned}$$

where the final form follows from the repeated application of DeMorgan's law and the fact that the complement of the event A_{22} is the event that the request is for the other output or that there is no request at all. Expansion of this term will result in the union of 16 mutually exclusive events. The probability is then just the sum of the probabilities of each of the events. Now the probability of each of these 16 events must be evaluated.

The simplest is the case where no inputs are active. This term is given by

$$P'(^{-}A_1^U \cap {}^{-}A_2^U \cap {}^{-}A_1^L \cap {}^{-}A_2^L) = (1-m_0)^4$$

Next the four terms in which only one input is active must be considered. These are easily evaluated in terms of previously calculated quantities. They are

$$\begin{aligned} P'(^{-}A_1^U \cap {}^{-}A_2^U \cap {}^{-}A_1^L \cap A_{22}^L) &= P'(A_{12}^L \mid A_2^L) m_0 (1-m_0)^3 \\ &= (1-0.5p_a-p_d)^k m_0 (1-m_0)^3 \end{aligned}$$

$$\begin{aligned} P'(^{-}A_1^U \cap {}^{-}A_2^U \cap A_{12}^L \cap {}^{-}A_2^L) &= P'(A_{12}^L \mid A_2^L) m_0 (1-m_0)^3 \\ &= [1-(1-0.5p_a-p_d)^k] m_0 (1-m_0)^3 \end{aligned}$$

$$P'(^{-}A_1^U \cap A_{22}^U \cap {}^{-}A_1^L \cap {}^{-}A_2^L) = (1-0.5p_a-p_d)^k m_0 (1-m_0)^3$$

$$P'(A_{12}^U \cap {}^{-}A_2^U \cap {}^{-}A_1^L \cap {}^{-}A_2^L) = [1-(1-0.5p_a-p_d)^k] m_0 (1-m_0)^3$$

Next the six terms that involve two active inputs must be evaluated. The first of these is

$$\begin{aligned} P'(^{-}A_1^U \cap {}^{-}A_2^U \cap A_{12}^L \cap A_{22}^L) &= P'(A_{22}^L \cap A_{12}^L \mid A_2^L \cap A_1^L) m_0^2 (1-m_0)^2 \\ &= P''(A_{22}^L \cap A_{12}^L) m_0^2 (1-m_0)^2 \end{aligned}$$

where P'' of an event has been redefined as the probability of that event given that all switches in column 0 which effect the event are functional and that all inputs that are necessary for the event are active. Thus

P''(event) =

P'(event | all inputs required for the event are active)

Then by symmetry

$$P''(A_{12}^{L} \cap A_{22}^{L}) = P''(A_{11} \cap A_{21})$$

which has been evaluated.

The next term to be evaluated is

$$P'(\text{~}A_{1}^{U} \cap A_{22}^{U} \cap \text{~}A_{1}^{L} \cap A_{22}^{L}) = P''(A_{22}^{U} \cap A_{22}^{L}) m_0{}^2 (1-m_0)^2$$

This term has not been seen before. Here the two primary paths both of which use the same column 1 switch must be considered. Since they both pass through the same column 1 switch they can share any number of switches from 1 to k. Here, as before, though not explicitly stated previously, share is in the sense that both attempt to use the same switches. Obviously they may not be able to do this simultaneously. Only the probability that errors do not occur on either of these paths is of interest. In other words, given no conflicts, that both accesses can be made. To evaluate this term first condition on the number of switches shared between the two paths. Thus

$$P''(A_{22}^{U} \cap A_{22}^{L}) =$$

$$\Sigma_{j=1}^{k} P''(A_{22}^{U} \cap A_{22}^{L} \mid \text{paths share } j \text{ switches}) P(\text{share } j \text{ switches})$$

The probability that j switches are shared is

$$P(\text{share } j \text{ switches}) = \begin{cases} 1/2^{j} & j<k \\ 2/N & j=k \end{cases}$$

This can easily be seen by realizing that the ID of the memory requested determines the path and therefore the number of shared switches. The memory ID is a k bit binary number. The two paths will share exactly j switches if and only if these two numbers match in exactly j-1 places. Since the individual binary digits in each number are independent and uniformly distributed the probability is as given above.

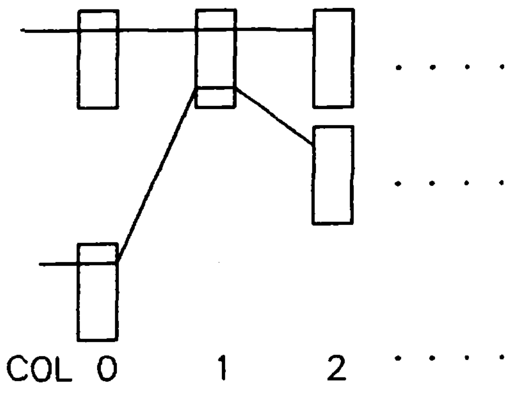


Figure 12 -- Exactly One Switch Shared

Figure 12 shows an example of the case where exactly one switch is shared by the two paths. Notice that if this occurs, this event can occur only if that shared switch has not failed, or ,if it has failed in an address mode, the failure matches both request given that the request are known to be different; and then if no other errors are caused by the remaining k-1 switches on each of the two paths. Thus

$$P''(A_{22}^{U} \cap A_{22}^{L} | \ j=1) = (1-0.5p_a-p_d)\ (1-0.5p_a-p_d)^{k-1}\ (1-0.5p_a-p_d)^{k-1}$$
$$= (1-0.5p_a-p_d)^{2k-1}$$

Figure 13 shows an example of the case where there are 3 shared switches and k>3. Here, for the event to occur, both the first and last shared switch must be functional. The control digits for the switches between the first and last must be the same and therefore these switches will not produce an error for either path if they do not produce an error for one of the paths. Then the remaining switches along both paths must not produce any errors. Thus

$$P''(A^U_{22} \cap A^L_{22} \mid 1<j<k) =$$

$$= (1-p_f)\,(1-0.5p_a-p_d)^{j-2}\,(1-p_f)\,(1-0.5p_a-p_d)^{k-j}\,(1-0.5p_a-p_d)^{k-j}$$

$$= (1-p_f)^2\,(1-0.5p_a-p_d)^{2k-j-2}$$

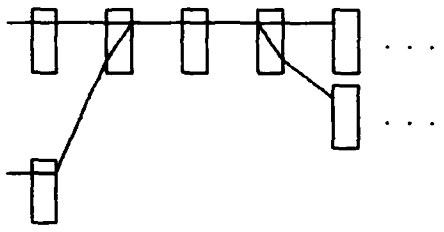


Figure 13 Exactly 3 Shared Switches

Finally, the case where k switches are shared must be evaluated. Here again the first shared switch must be functional. Then the next k-2 switches must not produce an error for the paths, given that the control digits are the same for both paths. *Finally the last switch must not* produce an error for either path. If the switch has failed in the address mode this can only occur if both paths request the same memory and the switch has failed so that the connection is possible. Thus

$$P''(A_{22}^{U} \cap A_{22}^{L} | \ j=k) = (1-p_f)\ (1-0.5p_a-p_d)^{k-2}\ (1-0.75p_a-p_d)$$

The next term to be evaluated contains two secondary path requests. This implies that at least one error exists in each of the primary paths. The desired term is

$$P'(A_{12}^{U} \cap {}^{-}A_{2}^{U} \cap A_{12}^{L} \cap {}^{-}A_{2}^{L}) = P''(A_{12}^{U} \cap A_{12}^{L})\ m_0^{2}(1-m_0)^{2}$$

The primary paths associated with these requests enter the same column 1 switch. The desired probability then can be derived from the last calculated term as follows:

$$\begin{aligned} P''(A_{12}^{U} \cap A_{12}^{L}) &= 1 - P''(A_{11}^{U} \cup A_{11}^{L}) \\ &= 1 - P''(A_{11}^{U}) - P''(A_{11}^{L}) + P''(A_{11}^{U} \cap A_{11}^{L}) \\ &= 1 - 2(1-0.5p_a-p_d)^{k} + P''(A_{22}^{U} \cap A_{22}^{U}) \end{aligned}$$

where the last step follows from the symmetry involved.

The final three terms with two active inputs can be shown by symmetry to be equal to terms previously derived. Thus

$$\begin{aligned} P'({}^{-}A_{1}^{U} \cap {}^{-}A_{2}^{U} \cap A_{12}^{L} \cap A_{22}^{L}) &= P'({}^{-}A_{1}^{U} \cap A_{22}^{U} \cap A_{12}^{L} \cap {}^{-}A_{2}^{L}) \\ &= P'(A_{12}^{U} \cap {}^{-}A_{2}^{U} \cap {}^{-}A_{1}^{L} \cap A_{22}^{L}) \\ &= P'(A_{12}^{U} \cap A_{22}^{U} \cap {}^{-}A_{1}^{L} \cap {}^{-}A_{2}^{L}) \end{aligned}$$

Next, the equations for the four terms with three active inputs must be derived. The first of these is

$$P'({}^{-}A_{1}^{U} \cap A_{22}^{U} \cap A_{12}^{L} \cap A_{22}^{L}) = P''(A_{22}^{U} \cap A_{12}^{L} \cap A_{22}^{L})\ m_0^{3}(1-m_0)$$

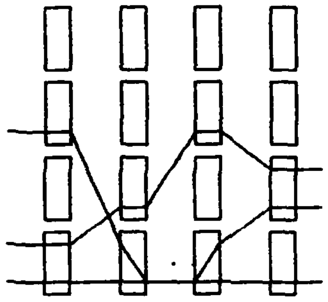


Figure 14 Example of a Three Input Term

Figure 14 shows a typical example of one of the sets of paths which are the events in this term. Notice that there are two primary paths which come from the A_2 inputs in which there can be no errors, and one primary path which comes from the A_1 input, in which at least one error must occur. The two paths from the A_2 inputs may share 1 to k switches with each other and finally may share a column k switch with the path from the A_1 input. The probability of this event can be calculated by conditioning on the number of shared switches in the A_2 paths and then on whether a column k switch is shared with the A_1 path.

First define the following:

$$E = [A_{22}^{U} \cap A_{12}^{L} \cap A_{22}^{L}]$$

Then, conditioning on the number of shared switches in the A_2 paths,

$$P''(A_{22}^{U} \cap A_{12}^{L} \cap A_{22}^{L}) = \Sigma_{j=1}^{k} P''(E \mid A_{22}^{U}, A_{22}^{L} \text{ share j switches.})\ P(\text{share j.})$$

Where the probability that the A_2 paths share j switches was previously derived.

Now define

$$P''^{j}(E) = P''(E|A_{22}^{U},A_{22}^{L} \text{ share } j \text{ switches.})$$

Finally, condition on the sharing of a column k switch.

$$P''^{j}(E) = P''^{j}(E|A_{22} \text{ paths disjoint from } A_1 \text{ primary path})\ P(\text{disjoint})$$
$$+\ P''^{j}(E|A_{22} \text{ share switch with } A_1 \text{ primary path})\ P(\text{share})$$

If the A_2 paths share fewer than k switches they access two of four possible memories. If the A_1 path accesses one of those four memories, it will share a last column switch with the A_2 paths. Thus

$$P(\text{share}) = 4/N \quad j<k$$
$$= 2/N \quad j=k$$

$$P(\text{disjoint}) = 1-4/N \quad j<k$$
$$= 1-2/N \quad j=k$$

and, if a last column switch is shared by the A_1 and A_2 paths, condition on whether that switch has failed.

$$P''^{j}(E|\text{share.}) = P''^{j}(E|\text{share, shared not failed.})(1-p_f)$$
$$+\ P''^{j}(E|\text{share, shared addr fail.})p_a$$
$$+\ P''^{j}(E|\text{share, shared data fail.})p_d$$

If it has failed in the address mode condition on whether it produces an error for the A_1 primary path. Thus

$P''^{j}(E|\text{share, addr fail})$

$= P''^{j}(E|\text{shr addr failed, no err } A_1 \text{ primary path})\ P(\text{no err}|\text{addr fail shr})$

$+ P''^{j}(E|\text{shr addr failed, err } A_1 \text{ primary path})\ P(\text{err}|\text{addr fail shr})$

Now given that the A_2 paths and the A_1 path do not share a final column switch, they are independent and, for the case where the A_2 paths share 1 switch,

$P''^{1}(E|\text{no shr})$

$= (1-0.5p_a-p_d)(1-0.5p_a-p_d)^{k-1}(1-0.5p_a-p_d)^{k-1}[1-(1-0.5p_a-p_d)^{k}]$

$= (1-0.5p_a-p_d)^{2k-1}[1-(1-0.5p_a-p_d)^{k}]$

For $1<j<k$

$P''^{j}(E|\text{no shr}, 1<j<k)$

$=(1-p_f)^2(1-0.5p_a-p_d)^{2k-j-2}[1-(1-0.5p_a-p_d)^{k}]$

and for $j=k$

$P''^{k}(E|\text{no shr})=(1-p_f)(1-0.5p_a-p_d)^{k-2}(1-0.75p_a-p_d)[1-(1-0.5p_a-p_d)^{k}]$

Now if they share a switch but that switch is functional, it cannot produce an error and thus, for the case where there is one shared switch in the A_2 paths,

$P''^{1}(E|\text{shr, no fail})$

$= (1-0.5p_a-p_d)(1-0.5p_a-p_d)^{k-2}(1-0.5p_a-p_d)^{k-1}[1-(1-0.5p_a-p_d)^{k-1}]$

$= (1-0.5p_a-p_d)^{2k-2}[1-(1-0.5p_a-p_d)^{k-1}]$

$P''^{j}(E|\text{shr, no fail}, 1<j<k)$

$=(1-p_f)^2(1-0.5p_a-p_d)^{2k-j-3}[1-(1-0.5p_a-p_d)^{k-1}]$

$P''^{k}(E|\text{shr, no fail}) = (1-p_f)(1-0.5p_a-p_d)^{k-2}[1-(1-0.5p_a-p_d)^{k-1}]$

If a column k switch is shared and has failed but produces no errors then it is as if the switch had not failed and

$$P''^{j}(E|\text{share, addr fail, no error}) = P''^{j}(E|\text{share, not failed})$$

Now if the shared switch has failed in the address mode and produces an error for the A_1 path then the probability of at least one error on that path is one and

$$P''^{1}(E|\text{shr, addr fail, err } A_1)$$
$$= (1-0.5p_a-p_d)(1-0.5p_a-p_d)^{k-1}(1-0.5p_a-p_d)^{k-2}$$
$$= (1-0.5p_a-p_d)^{2k-2}$$

$$P''^{j}(E|\text{shr, addr fail, err } A_1,\ 1<j<k) = (1-p_f)^2(1-0.5p_a-p_d)^{2k-j-3}$$

$$P''^{k}(E|\text{shr, addr fail, err } A_1) = (1-p_f)(1-0.5p_a-p_d)^{k-2}$$

Now if a column k switch is failed in the address mode the probability that it produces errors depends on the number of paths using the switch. If $j<k$ then two paths use the switch one from the pair of A_2 paths and the A_1 path. Thus

$$P(\text{no error shared addr failed switch}) = 1/4 \quad j<k$$
$$= 1/8 \quad j=k$$

$$P(\text{err } A_1, \text{ no error } A_2 \text{ paths shared addr failed switch}) = 1/4 \quad j<k$$
$$= 1/8 \quad j=k$$

This completes the evaluation of this term. The other three terms that involve three active inputs can be derived from this one as follows:

$$P'(A_{12}^{U}\cap{\sim}A_{2}^{U}\cap A_{12}^{L}\cap A_{22}^{L}) = P''(A_{12}^{U}\cap A_{12}^{L}\cap A_{22}^{L})m_0^{3}(1-m_0)$$

and

$$\begin{aligned}P''(A_{12}^{U}\cap A_{12}^{L}\cap A_{22}^{L}) &= 1 - P''({\sim}[A_{12}^{U}\cap A_{12}^{L}\cap A_{22}^{L}])\\ &= 1 - P''(A_{11}^{U}\cap A_{11}^{L}\cap A_{21}^{L})\\ &= 1 - P''(A_{11}) - P''(A_{11}) - P''(A_{21})\\ &\quad + P''(A_{11}^{U}\cap A_{11}^{L}) + P''(A_{11}^{L}\cap A_{21}^{L}) + P''(A_{11}^{U}\cap A_{21}^{L})\\ &\quad - P''(A_{11}^{U}\cap A_{11}^{L}\cap A_{21}^{L})\end{aligned}$$

Now

$$P''(A_{11}^{U}\cap A_{11}^{L}\cap A_{21}^{L}) = P''(A_{22}^{U}\cap A_{12}^{L}\cap A_{22}^{L})$$

by symmetry and all the other terms in the equation have been previously evaluated. Finally by symmetry

$$P'(A_{12}^{U}\cap A_{22}^{U}\cap{\sim}A_{1}^{L}\cap A_{22}^{L}) = P'({\sim}A_{1}^{U}\cap A_{22}^{U}\cap A_{12}^{L}\cap A_{22}^{L})$$

and

$$P'(A_{12}^{U}\cap A_{22}^{U}\cap A_{12}^{L}\cap{\sim}A_{2}^{L}) = P'(A_{12}^{U}\cap{\sim}A_{2}^{U}\cap A_{12}^{L}\cap A_{22}^{L})$$

This completes the evaluation of the terms with three active inputs. Now the term which has all four inputs active must be evaluated. This term is

$$P'(A_{12}^{U}\cap A_{22}^{U}\cap A_{12}^{L}\cap A_{22}^{L}) = P''(A_{12}^{U}\cap A_{22}^{U}\cap A_{12}^{L}\cap A_{22}^{L})m_0^{4}$$

Let

$$E = [A_{12}^{U}\cap A_{22}^{U}\cap A_{12}^{L}\cap A_{22}^{L}]$$

To evaluate this term, first condition on the number of switches shared by the two paths which are primary for the A_1 pair of paths and the number of shared switches in the A_2 pair of paths. Then condition on the number of column k switches which are shared between the two paths. Note that for this event to occur it must be that there are no errors in either of the A_2 primary paths and at least one error in each of the A_1 primary paths. Thus

$$P''(E) =$$

$$\Sigma_{i=1}^{k}\Sigma_{j=1}^{k}\Sigma_{l=1}^{2}P''(E \mid A_1^U A_1^L \text{ shr } i,\ A_2^U A_2^L \text{ shr } j,\ \text{sets shr } l)P(\text{shr } l \mid i,j)P(\text{shr } i,j)$$

Now the number of shared switches in each set of paths is determined by the memories requested. These are independent by assumption and therefore the probability of sharing i switches in the A_1 paths is independent of the probability of sharing j switches in the A_2 paths. Thus have

$$P(\text{share } i,j) = \begin{cases} (1/2)^{i+j} & i,j<k \\ (2/N)(1/2)^{i} & i<k,j=k \\ (2/N)(1/2)^{j} & j<k,i=k \\ (4/N) & i,j=k \end{cases}$$

The probability of sharing l column k switches is more difficult to determine. Let X_m represent the binary digits of the memory addressed by one of the A_1 primary paths and let Y_m represent the digits of the other. Similarly let W_m and Z_m represent the digits of the memories addressed by the A_2 paths. Now notice that the switch used in column k is determined by the first k-1 digits of the address of the memory accessed. Also notice that the since the A_i paths share exactly i switches, the first i-1 digits of their memory requests are identical and the ith digits are complements of each other. Thus the addresses can be expressed as

$$X_1X_2\cdots\cdots X_{i-1}X_iX_{i+1}\cdots\cdots X_{k-1}\ X_k$$
$$X_1X_2\cdots\cdots X_{i-1}\bar{X}_iY_{i+1}\cdots\cdots Y_{k-1}\ Y_k$$

$$W_1W_2\cdots\cdots W_{j-1}W_jW_{j+1}\cdots\cdots W_{k-1}\ W_k$$
$$W_1W_2\cdots\cdots W_{j-1}\bar{W}_jZ_{j+1}\cdots\cdots Z_{k-1}\ Z_k$$

Now, since the memory requests are all independent, the probability of sharing l column k switches can be determined by determining the probability that l memory IDs in the first pair match l IDs in the second pair. The result is as follows:

$i=j=k$ P(share 0) = $1-0.5^{k-1}$

P(share 1) = 0.5^{k-1}

P(share 2) = 0

$i=k,\ j<k$ P(share 0) = $1-0.5^{k-2}$

P(share 1) = 0.5^{k-2}

P(share 2) = 0

$i=j<k$ P(share 0) = $1-0.5^{k-3}+0.5^{2k-4-i}-0.5^{i-1}\ 0.25^{k-1-i}$

P(share 1) = $0.5^{k-3} - 0.5^{2k-4-i}$

P(share 2) = $0.5^{i-1}\ 0.25^{k-1-i}$

$i<j<=k$ $P(\text{share } 0) = 1-0.5^{k-j}$

$j<i<=k$ $P(\text{share } 1) = 0.5^{k-j}$

$P(\text{share } 2) = 0$

Now define the following:

$P^{\prime\prime ijl}(E) =$

$P^{\prime\prime}(E \mid A_1^U, A_1^L \text{ paths shr } i,\ A_2^U, A_2^L \text{ paths shr } j, \text{ sets share } l \text{ in column } k)$

The objective is to evaluate this probability for all possible values of i, j and *l*.

Consider the case of *l*=0. In this case the two path sets are disjoint and therefore independent. As a result

$$P^{\prime\prime ij0}(E) = P^{\prime\prime}(A_{12}^U \cap A_{12}^L)\ P^{\prime\prime}(A_{22}^U \cap A_{22}^L)$$

$$\begin{array}{lll} P^{\prime\prime j}(A_{22}^U \cap A_{22}^L) = & (1-0.5p_a-p_d)^{2k-1} & j=1 \\ & (1-p_f)^2(1-0.5p_a-p_d)^{2k-j-2} & 1<j<k \\ & (1-p_f)(1-0.5p_a-p_d)^{k-2}(1-0.75p_a-p_d) & j=k \end{array}$$

$$\begin{aligned} P^{\prime\prime i}(A_{12}^U \cap A_{12}^L) &= 1 - P^{\prime\prime i}(A_{11}^U \cap A_{11}^L) \\ &= 1 - P^{\prime\prime i}(A_{11}^U) - P^{\prime\prime i}(A_{11}^L) + P^{\prime\prime i}(A_{11}^U \cap A_{11}^L) \\ &= 1 - 2(1-0.5p_a-p_d)^k + P^{\prime\prime i}(A_{22}^U \cap A_{22}^L) \end{aligned}$$

Now consider the case of *l*=1. Here as before, condition on whether the shared switch is functional or failed and then, given that it has failed, condition on the failure mode and then, for address mode failures, on whether the failure produces an error for the A_1 path or paths that go through the failed switch. Note that the probability of the event is 0 if the failure causes an error for the A_2 paths that pass through it. Thus

$$P''^{ij1}(E) = P''^{ij1}(E|\text{shared last column switch not failed})(1-p_f)$$
$$+ P''^{ij1}(E|\text{shared last column switch addr failed})p_a$$
$$+ P''^{ij1}(E|\text{shared last column switch data failed})p_d$$

Now if the shared switch has not failed it cannot cause errors for either of the sets of paths. If this is true the probability of errors in one set of paths exclusive of that switch, is independent of the same probability in the other set. Thus

$$P''^{ij1}(E|\text{shr ok}) = P''^{i}(A_{12}^{U}\cap A_{12}^{L}|\text{shr ok})\ P''^{j}(A_{22}^{U}\cap A_{22}^{L}|\text{shr ok})$$

The probabilities in the above equation are easily evaluated and are given by

$$P''^{j}(A_{22}^{U}\cap A_{22}^{L}|\text{shr ok}) = \begin{cases} (1-0.5p_a-p_d)^{2k-2} & j=1 \\ (1-p_f)^2(1-0.5p_a-p_d)^{2k-j-1} & 1<j<k \\ (1-p_f)(1-0.5p_a-p_d)^{k-2} & j=k \end{cases}$$

and

$$P''^{i}(A_{12}^{U}\cap A_{12}^{L}|\text{shr ok}) = 1 - P''^{i}(A_{11}^{U}\cup A_{11}^{L}|\text{shr ok})$$
$$= 1 - P''^{i}(A_{11}^{U}|\text{ok}) - P''^{i}(A_{11}^{L}) + P''^{i}(A_{11}^{U}\cap A_{11}^{L}|\text{shr ok})$$
$$= 1 - 2(1-0.5p_a-p_d)^{k-1} + P''^{i}(A_{22}^{U}\cap A_{22}^{L}|\text{shr ok}) \qquad i=k$$
$$1 - (1-0.5p_a-p_d)^{k-1} - (1-0.5p_a-p_d)^{k} + P''^{i}(A_{22}^{U}\cap A_{22}^{L}|\text{shr ok}) \qquad i<k$$

Now if the shared switch is failed in the address mode

$$P''^{ij1}(E|\text{shr addr fail}) =$$
$$P''^{ij1}(E|\text{shr addr fail, error})\ P(\text{error}|\ \text{shr addr fail})$$
$$+ P''^{ij1}(E|\text{shr addr fail, no err})\ P(\text{no err}|\ \text{shr addr fail})$$

If the address mode failed switch does not produce errors for either of the paths that pass through it, the situation is identical to the case when the shared switch is functional and

$$P^{\cdot\cdot ij1}(E \mid \text{shr addr fail no error}) = P^{\cdot\cdot ij1}(E \mid \text{shr no fail})$$

Now the probability that no errors are produced depends on the number of paths passing through the switch and thus on i and j. Evaluating gives

$$P^{\cdot\cdot ij1}(\text{no error} \mid \text{shr addr fail}) = \begin{cases} 1/4 & i,j<k \\ 1/8 & i<k, j=k \text{ or } j<k, i=k \\ 1/16 & i=j=k \end{cases}$$

Now if the shared switch produces an error for the A_1 path or paths passing through it, the probability of errors in the other A_1 path and of no errors in the A_2 paths must be determined. Once again this depends on the number of paths passing through the switch. If both A_1 paths pass through the address mode failed switch, the possibility of an error in only one of the A_1 paths or in both of the A_1 paths must be determined. Evaluating

$$P'^{\cdot ij1}(E \mid \text{shared addr fail, error})\ P'^{\cdot ij1}(\text{error} \mid \text{shared addr fail})$$

$$=[1-(1-0.5p_a-p_d)^k]\ P'^{\cdot ij1}(A_{22}^U \cap A_{22}^L \mid \text{shr ok})(1/4) \qquad i,j<k$$

$$=[1-(1-0.5p_a-p_d)^k]\ P'^{\cdot ij1}(A_{22}^U \cap A_{22}^L \mid \text{shr ok})(1/8) \qquad i<j=k$$

$$=[1-(1-0.5p_a-p_d)^{k-1}]\ P'^{\cdot ij1}(A_{22}^U \cap A_{22}^L \mid \text{shr ok})(1/4) \qquad j<i=k$$

$$+\ P'^{\cdot ij1}(A_{22}^U \cap A_{22}^L \mid \text{shr ok})(1/8)$$

$$=[1-(1-0.5p_a-p_d)^{k-1}]\ P'^{\cdot ij1}(A_{22}^U \cap A_{22}^L \mid \text{shr ok})(1/8) \qquad i=j=k$$

$$+\ P'^{\cdot ij1}(A_{22}^U \cap A_{22}^L \mid \text{shr ok})(1/16)$$

This completes the evaluation for the case of l=1. Now evaluate for l=2.

When l=2 condition on the number of failed switches thus

$$P'^{\cdot ij2}(E) = P'^{\cdot ij2}(E \mid \text{no shared failed})\ P'^{\cdot ij2}(\text{no shared failed})$$

$$+\ P'^{\cdot ij2}(E \mid 1 \text{ shared addr fail})\ P'^{\cdot ij2}(1 \text{ shared addr failed})$$

$$+\ P'^{\cdot ij2}(E \mid 2 \text{ shared addr fail})\ P'^{\cdot ij2}(2 \text{ shared addr failed})$$

$$+\ P'^{\cdot ij2}(E \mid 1 \text{ or } 2 \text{ shared data fail})\ P'^{\cdot ij2}(1 \text{ or } 2 \text{ shared addr failed})$$

and

$$P'^{\cdot ij2}(\text{no shared failed}) = (1-p_f)^2$$

$$P'^{\cdot ij2}(1 \text{ shared addr failed}) = 2p_a(1-p_f)$$

$$P'^{\cdot ij2}(2 \text{ shared addr failed}) = p_a^{\,2}$$

Once again, if there are no failures in the shared switches, the events that errors occur in path sets are independent and

$$P'^{\cdot ij2}(E \mid \text{no fail}) = P'^{\cdot i}(A_{12}^U \cap A_{12}^L \mid 2 \text{ ok})\ P'^{\cdot j}(A_{22}^U \cap A_{22}^L \mid 2 \text{ ok})$$

and

$$P^{\cdot\cdot j}(A^U_{22}\cap A^L_{22}|2\text{ ok}) = (1-0.5p_a-p_d)^{2k-j} \qquad j=1$$
$$= (1-p_f)^2(1-0.5p_a-p_d)^{2k-j-4} \qquad j>1$$

and

$$P^{\cdot\cdot i}(A^U_{12}\cap A^L_{12}|2\text{ ok}) = 1 - P^{\cdot\cdot i}(A^U_{11}\cup A^L_{11}|2\text{ ok})$$
$$= 1 - 2P^{\cdot\cdot i}(A_{11}|1\text{ ok}) + P^{\cdot\cdot i}(A^U_{11}\cap A^L_{11}|2\text{ ok})$$
$$= 1 - 2(1-0.5p_a-p_d)^{k-1} + P^{\cdot\cdot i}(A^U_{22}\cap A^L_{22}|2\text{ ok})$$

Now if one shared switch is failed in the address mode that switch may or may not produce an error in the A_1 path that goes through it. Thus

$$P^{\cdot\cdot ij2}(E|1\text{ addr fail}) =$$
$$P^{\cdot\cdot ij2}(E|1\text{ fail, error})\,P^{\cdot\cdot ij2}(\text{error}\mid 1\text{ addr fail})$$
$$+\,P^{\cdot\cdot ij2}(E|1\text{ fail, no err})\,P^{\cdot\cdot ij2}(\text{no err}\mid 1\text{ addr fail})$$
$$=[1-(1-0.5p_a-p_d)^{k-1}]P^{\cdot\cdot j}(A^U_{22}\cap A^L_{22}|2\text{ ok})(1/4)$$
$$+P^{\cdot\cdot ij2}(E|2\text{ ok})(1/4)$$

Next if both of the shared switches fail in the address mode, condition on the number of A_1 path errors produced giving

$$P^{\cdot\cdot ij2}(E|2\text{ addr fail}) =$$
$$P^{\cdot\cdot ij2}(E|2\text{ addr fail, 0 error})\,P^{\cdot\cdot ij2}(0\text{ error}\mid 2\text{ addr fail})$$
$$+\,P^{\cdot\cdot ij2}(E|2\text{ addr fail, 1 error})\,P^{\cdot\cdot ij2}(1\text{ error}\mid 2\text{ addr fail})$$
$$+\,P^{\cdot\cdot ij2}(E|2\text{ addr fail, 2 error})\,P^{\cdot\cdot ij2}(2\text{ error}\mid 2\text{ addr fail})$$

The probabilities of errors given two address mode failures are

$$P^{\cdot\cdot ij2}(\text{no errors} \mid 2 \text{ addr fail}) = 1/16$$

$$P^{\cdot\cdot ij2}(1 \text{ error} A_1 \text{ paths, no err } A_2 \text{ paths} \mid 2 \text{ addr fail}) = 1/8$$

$$P^{\cdot\cdot ij2}(2 \text{ error} A_1 \text{ paths, no err } A_2 \text{ paths} \mid 2 \text{ addr fail}) = 1/16$$

If there are no errors, the probability is equal to the probability given that there are no failures.

$$P^{\cdot\cdot ij2}(E \mid 2 \text{ addr fail, no error}) = P^{\cdot\cdot ij2}(E \mid \text{no fail})$$

With one error, the probability that an error exists in the remainder of the A_1 path for which there was no error in the shared switches must be accounted for. Thus

$$P^{\cdot\cdot ij2}(E \mid 2 \text{ addr fail, 1 error}) = [1-(1-0.5p_a-p_d)^{k-1}]P^{\cdot\cdot j}(A_{22}^{U} \cap A_{22}^{L} \mid 2 \text{ ok})$$

If there are errors in both of the A_1 paths then only the probability that there are no errors in the remainder of the A_2 paths is important and

$$P^{\cdot\cdot ij2}(E \mid 2 \text{ addr fail, 2 error}) = P^{\cdot\cdot j}(A_{22}^{U} \cap A_{22}^{L} \mid 2 \text{ ok})$$

Finally, if the shared switch fails in the data mode the probability of the event is 0.

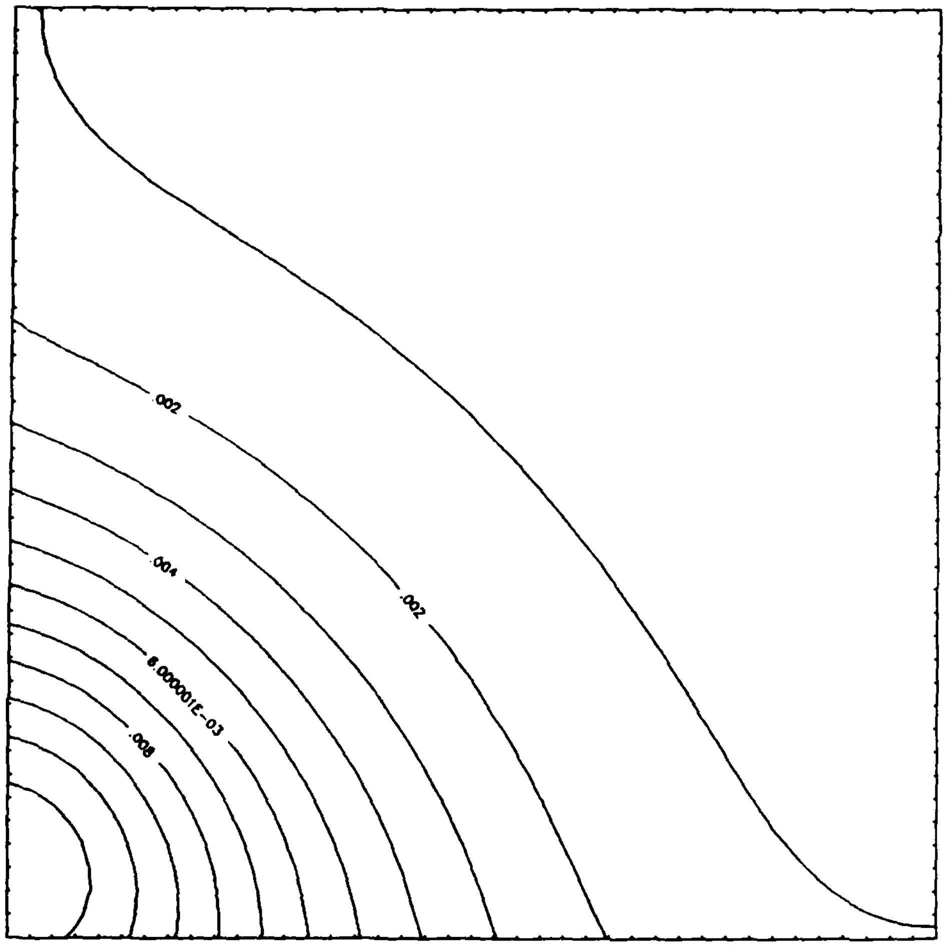
This completes the evaluation of all terms necessary to evaluate the probability that the two lines input to a column one switch of the S/E+ network are jointly active.

As stated previously, if indicator random variables for each of the lines in question are defined such that these random variables are 1 when the line is active and 0 when inactive, the covariance of these random variables is given by

$$COVAR = P(B_{10}^{U} \cap B_{10}^{L}) - P^{2}(B_{10})$$

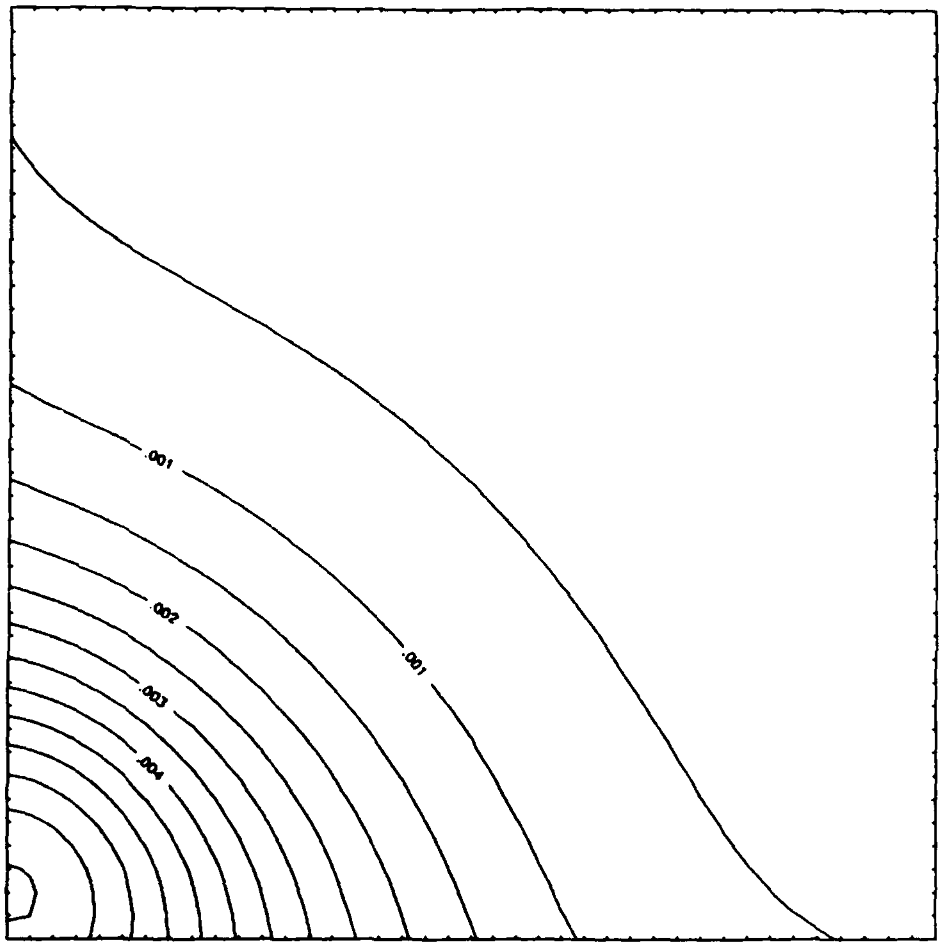
Figure 15 shows plots of the covariance vs p_a for various values of k and p_d. All of the data were calculated with m_0 equal one. The maximum value of the covariance is 0.05 and occurs in the region near values of 0.02 for p_d, 0.00 for p_a and 16 for N. The value of covariance decreases as the number of processors and memories increases. Thus it can be concluded that assuming they are independent for purposes of calculating the expected bandwidth will induce only small errors in the results.

Once the assumption that the inputs to any stage one switch are independent is made, the argument of disjoint sets of independent events can be used to show that the inputs to all switches in stages 1 to k-1 are independent. This being so, the analysis done for the S/E network applies to these stages.



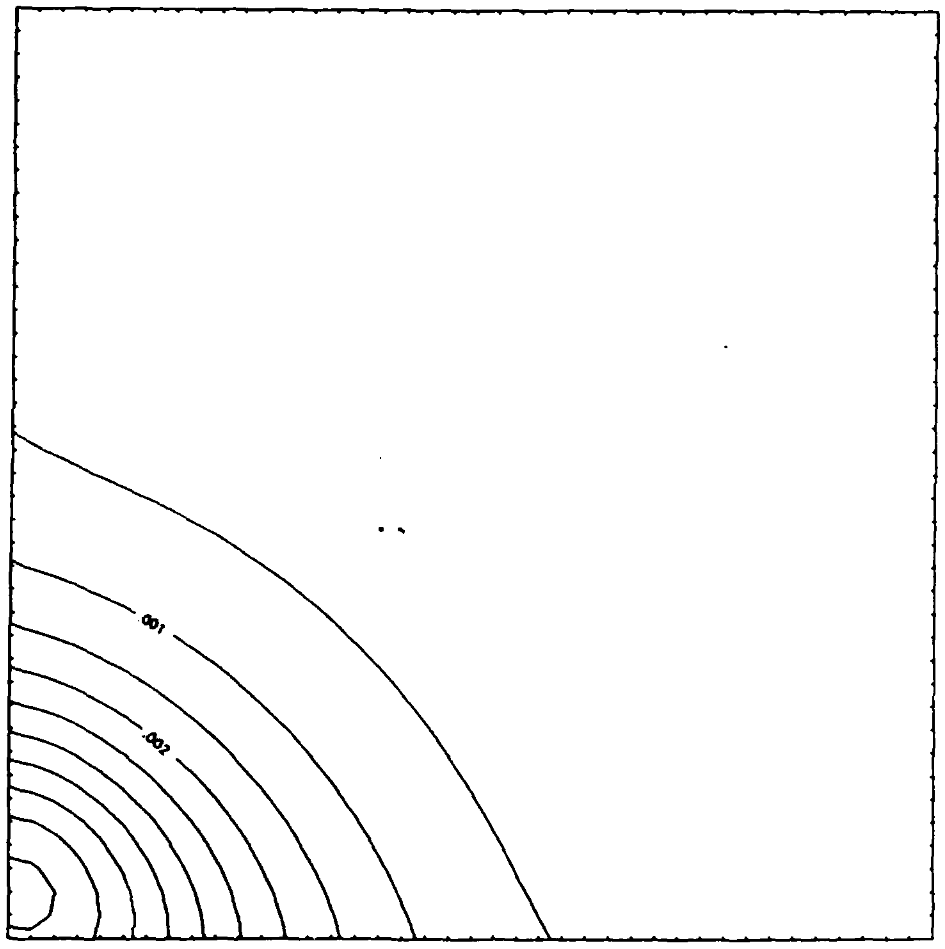
ABS COVARIANCE N=16 MAX=.05 xaxis pa(0-1.0) yaxis pd(0-1.0)

Figure 15a - Covariance as a function of P_a and P_d (N=16)



ABS COVARIANCE N=64 MAX=.035 xaxis pa(0-1.0) yaxis pd(0-1.0)

Figure 15b - Covariance as a function of P_a and P_d (N=64)



ABS COVARIANCE N=256 MAX=.266 xaxis pa(0-1.0) yaxis pd(0-1.0)

Figure 15c - Covariance as a function of P_a and P_d (N=256)

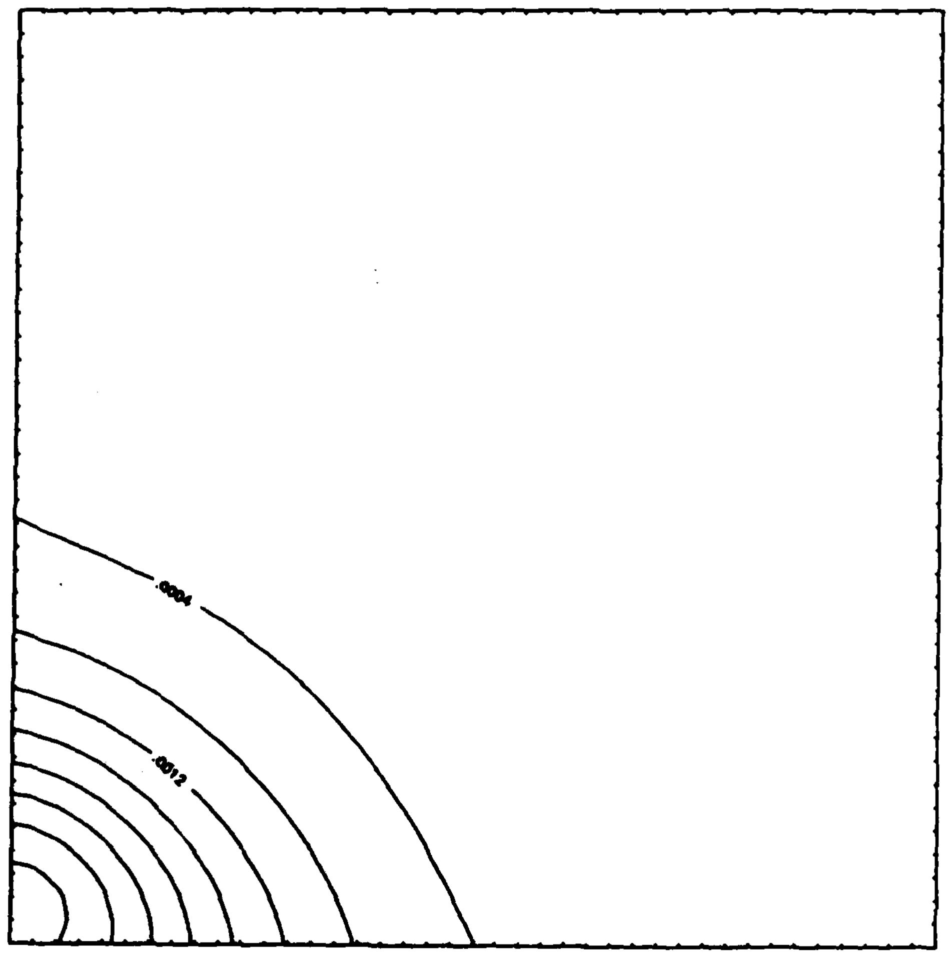


Figure 15d - Covariance as a function of P_a and P_d (N=1024)

The last stage is not so simple. Figure 16 shows a case where the inputs to a final stage switch are not independent. To compute the probability that a final stage output line is active, condition on whether its input lines are independent. thus

$$\begin{aligned}&P(B_{1K})\\ &= P(B_{1K}\mid \text{independent inputs to stage k})P(\text{independent})\\ &\quad + P(B_{1K}\mid \text{dependent inputs to stage k})P(\text{dependent})\end{aligned}$$

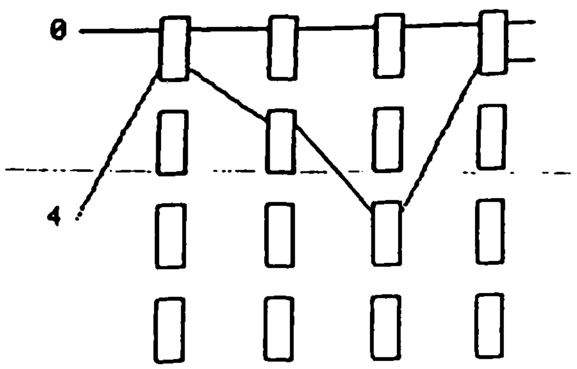


Figure 16 - Dependent Inputs to Stage K

The problem becomes how to calculate the probability that the inputs are independent. The probability that the inputs came from dependent sources clearly requires that some pair of processors, which are the inputs for the same column 0 switch, requested the two memories which are connected to this switch. This probability is 2/N. Then there must be no errors in the paths prior to the last switch. If there were errors in either path the requests would either be blocked when the error was detected or there would be a conflict at the first stage resulting in one of the requests being blocked. Finally neither of the requests could have been blocked by other processor requests prior to the last stage. Clearly, for large values of N this probability must be small and the errors introduced by disregarding the effects of this dependency should be small.

As a result, the bandwidth of the S/E+ network can be approximated by calculating the probability that an output line from stage 1 is active using the derived equation and then using the results for an S/E network to represent the remaining stages.

This provides a simple and useful model for estimating the failure dependent bandwidth of the S/E+ network. Further it demonstrates that the bandwidth of the S/E+ network is strictly less than that for an S/E network of the same size, when the stage 0 control strategy is based only on error detection. The S/E+ network increases the probability that a randomly selected processor to memory connection can be made in the presence of faults at the cost of communication bandwidth. The system designer must determine whether this is a desirable trade. Further this model provides a simple means to evaluate the cost benefit ratio for reliability measures implemented at the switch level.

Figure 17 presents some of the results of this analysis. It displays the bandwidth vs probability of address mode failure for several values of N and p_d for the S/E+ network. All of the calculations used a value of one for m_0.

Comparison of Results

Figures 15 and 17 present the calculated bandwidth for both the S/E and S/E+ networks. The bandwidth of the S/E+ network is strictly less than that for the S/E network. Figure 18 displays the percentage loss in bandwidth suffered by the S/E+ network as compared to the base S/E network. The bandwidth loss is small for small values of p_d. As p_d increases however, the percentage loss becomes a substantial part of the available bandwidth. Thus, the extra column in the S/E+ network serves to increase the probability that any given connection can be made at the cost of a small decrease in bandwidth.

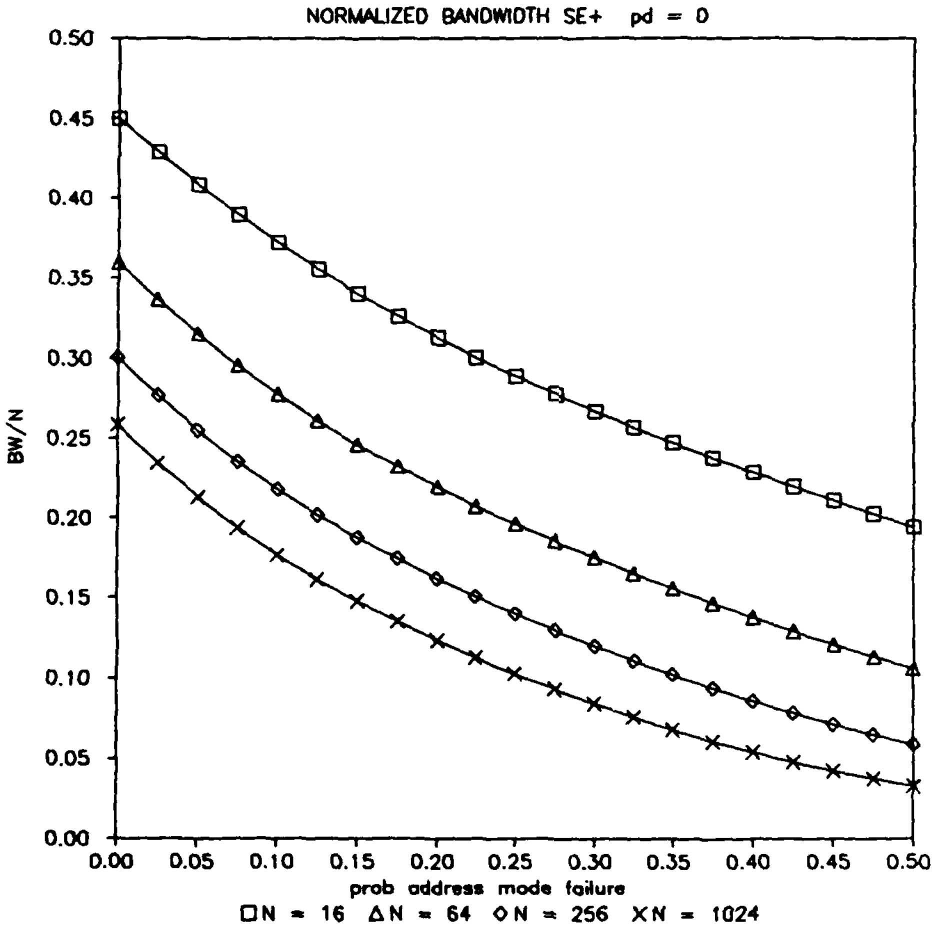


Figure 17a - Normalized Bandwidth SE+ (p_d=0)

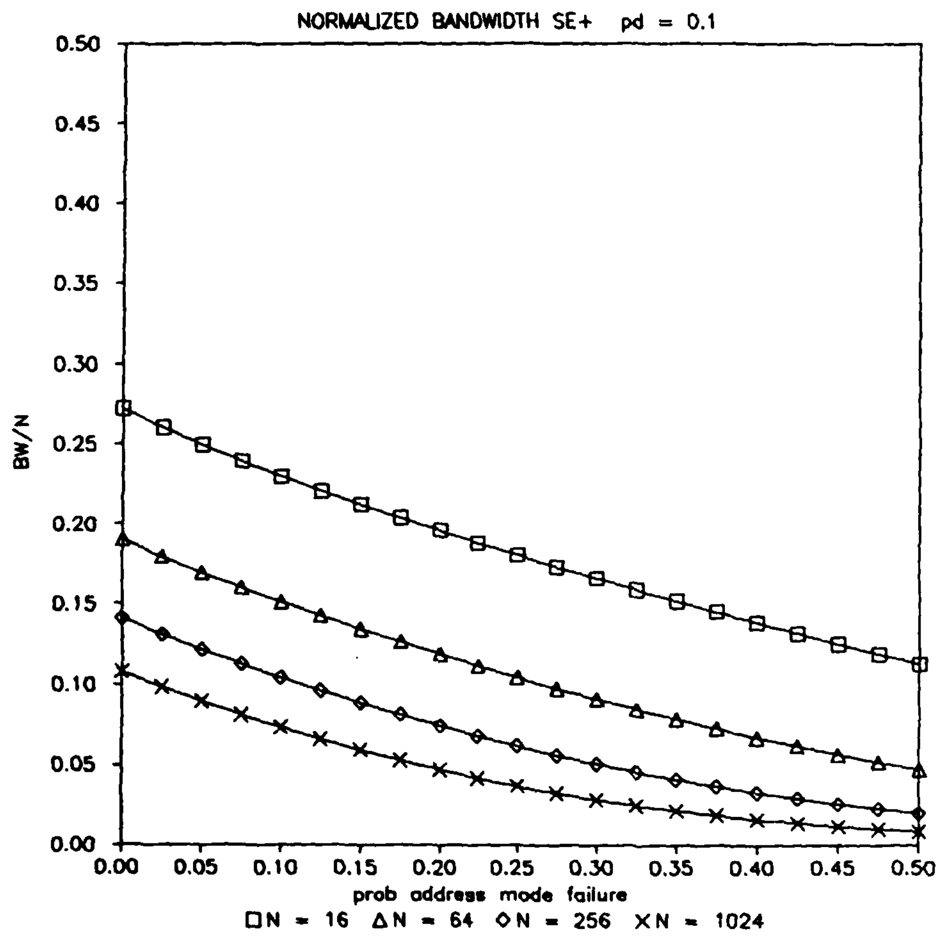


Figure 17b - Normalized Bandwidth SE+ (p_d=0.1)

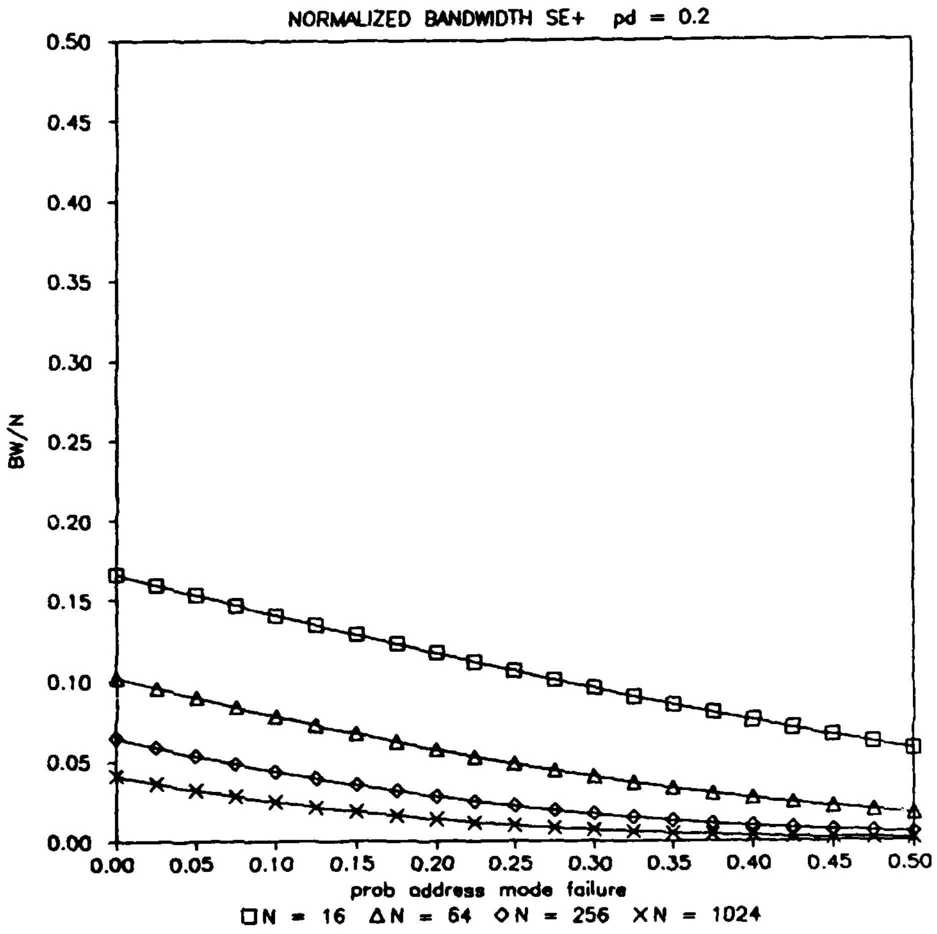


Figure 17c - Normalized Bandwidth SE+ (p_d=0.2)

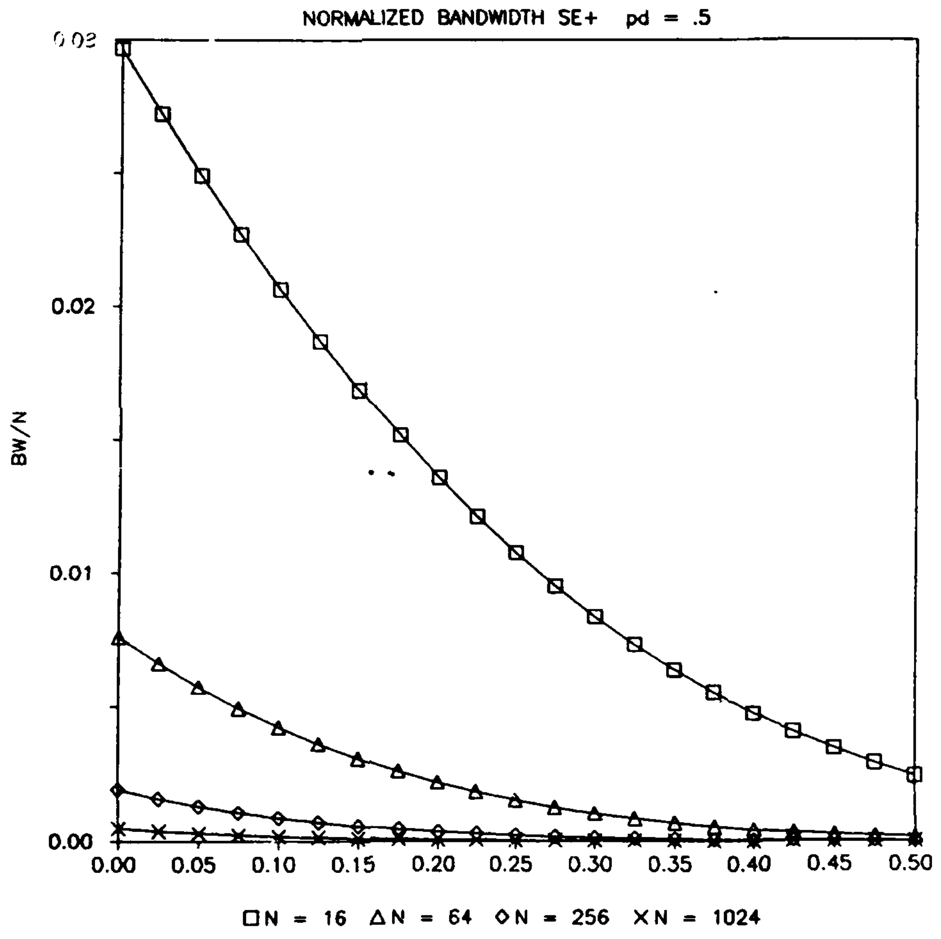


Figure 17d - Normalized Bandwidth SE+ (p_d=0.5)

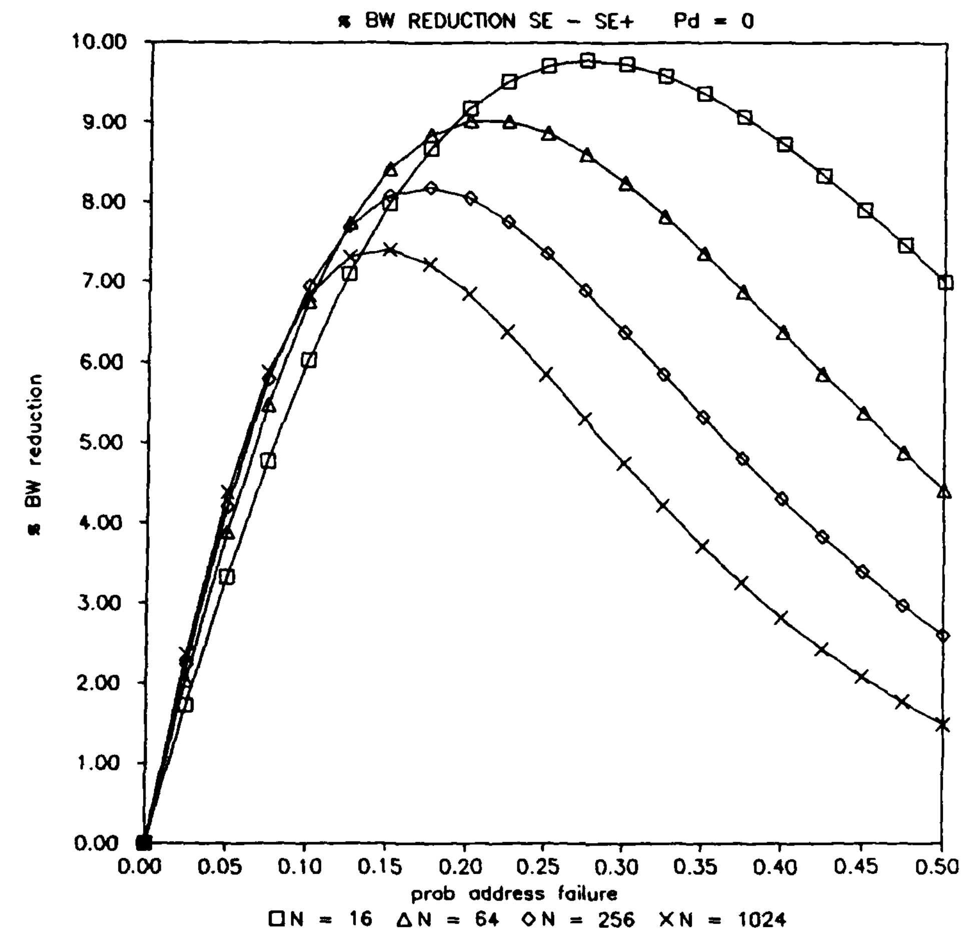


Figure 18a Bandwidth Reduction SE - SE+ (p_d=0)

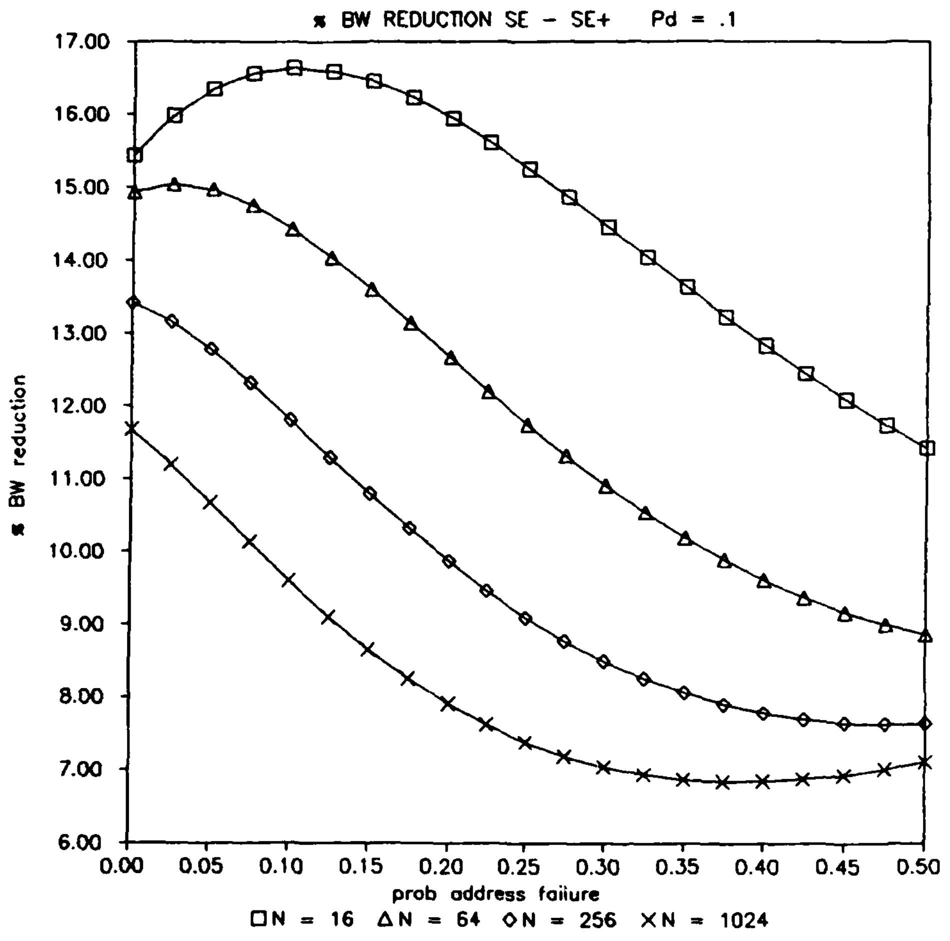


Figure 18b Bandwidth Reduction SE - SE+ (p_d=0.1)

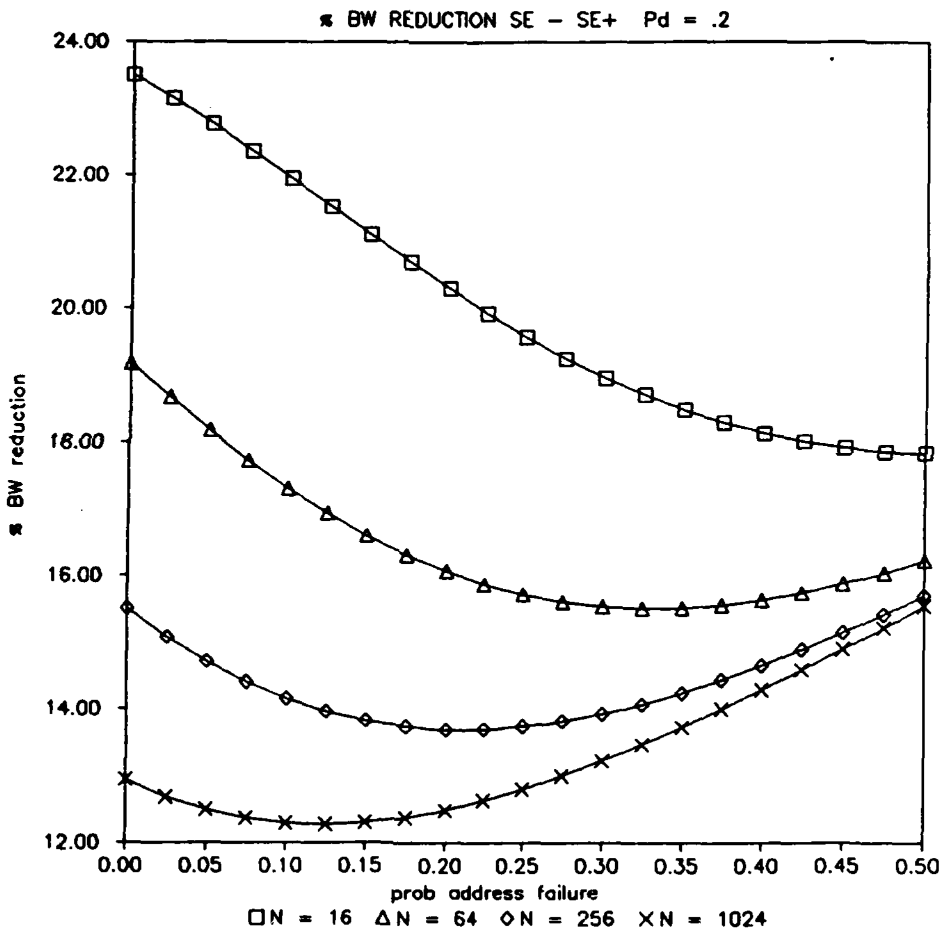


Figure 18c Bandwidth Reduction SE – SE+ (p_d=0.2)

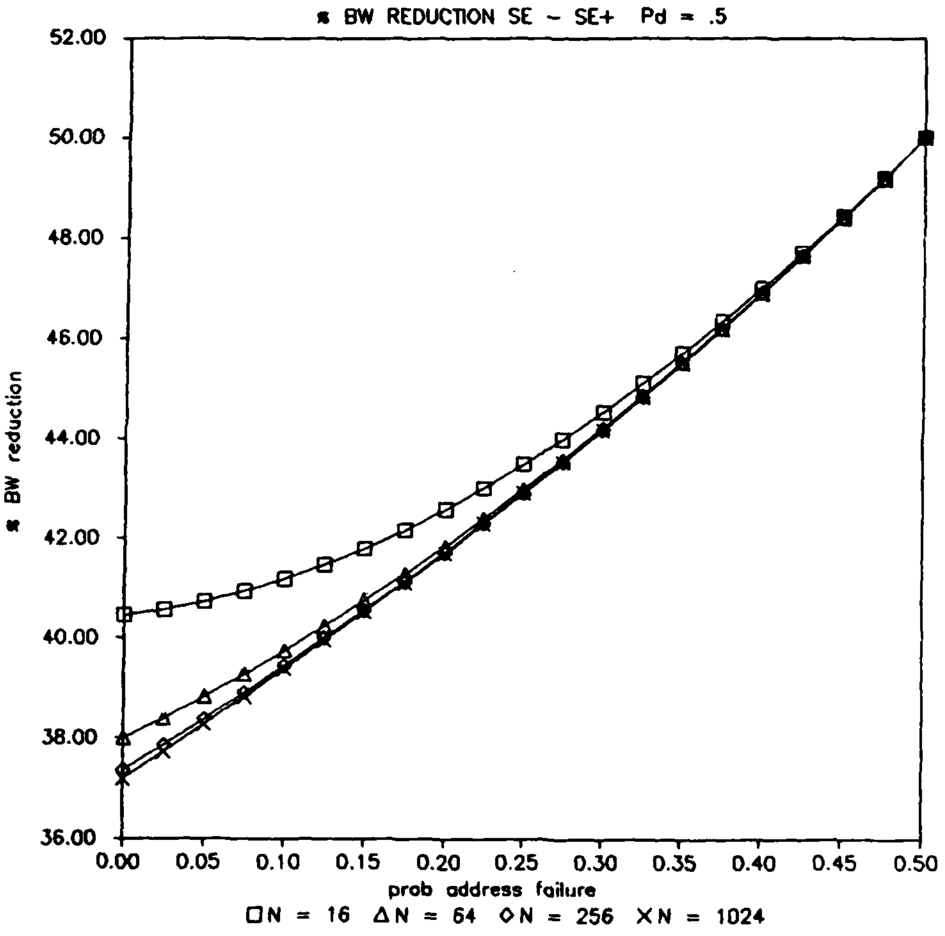


Figure 18d Bandwidth Reduction SE - SE+ (p_d=0.5)

IV. VERIFICATION OF S/E+ MODEL

In chapter 3 a bandwidth model for the S/E+ network was developed. In the development of this model two important assumptions were made. These were:

1.) The small value of covariance allowed us to assume that the events that outputs from the same column 0 are active are independent.

2.) The probability that two active inputs to a final stage switch originate at the same column 0 switch is small and can be ignored.

In order to test the validity of the assumptions a simulation of the S/E+ network was developed. The simulation is based on the following set of equations:

$$E(BW)=\sum_{\text{all fail states}} E(BW|\text{fail state})P(\text{fail state})$$

$$E(BW|\text{fail state})=\sum_{\text{all inputs}} E(BW|\text{fail state, input})P(\text{input})$$

$$E(BW|\text{fail state, input})=$$

$$\sum_{\text{allconflict states}} E(BW|\text{fs , inp, conflict state})P(\text{conflict state})$$

In the first of these equations the expected bandwidth is obtained by conditioning on the failure state of the ICN and then summing over all possible failure states. Here a failure state is defined as one particular failure configuration from the set of all possible ICN switch failure sets. This set includes the case of no failures. Next, in order to calculate the expected bandwidth given a particular failure state, condition on the input state — the set of applied input requests — and sum over all possible input request sets. Finally, given the input and failure states, condition on the resolution of conflicts at the ICN switches and sum over all possible resolutions.

These equations represent a complete calculation of the expected bandwidth. Unfortunately they cannot be completely evaluated in a reasonable amount of time on currently available computer systems. In order to demonstrate this consider an 8x8 S/E+ ICN. The ICN is composed of 16 switches each of which can assume one of four states — operational, stuck at X, stuck at T or data mode failure — this results in 4^{16} or 2^{32} possible failure states. For each of these, even if the evaluation is restricted to the set of inputs for which each processor is active — m_0 is 1 — 8^8 or 2^{24} possible input states must be evaluated. For each of these input states every possible conflict state must be evaluated. Given an input state consisting of 8 requests this will require the evaluation of from 1 to as many as 2^7 conflict states. Thus, a complete calculation for an 8x8 S/E+ ICN would require that a minimum of 2^{54} total network states be evaluated. This is clearly not feasible even on the fastest of modern computers.

The number of states that must be evaluated can be reduced as follows: First observe that the maximum value of the covariance occurs when p_a is 0. Using this, for verification purposes, the calculation can be restricted to the case where p_a is 0. This results in 2^{16} failure states. Next, note that the covariance and the probability that two inputs to a final column switch originate at the same column 0 switch both decrease as N, the number of processors and memories attached to the ICN, increases. Thus, the maximum error induced by the assumptions in the model for S/E+ bandwidth will occur for small values of N. Since a 4x4 S/E+ ICN is trivially equivalent to a 4x4 crossbar, the simulation was performed for an 8x8 ICN. Finally, expected bandwith can be approximated, without inducing large errors, by calculating it for a large number but less than 8^8 random input states.

Restricting the failure modes to the data only mode allows the expected bandwidth to be expressed as a 16th order binomial equation in p_d where, letting f represent the number of failures, the coefficients can be calculated from the equation:

$$C_f = \sum_{\text{all fail states with } f \text{ fails}} E(BW \mid \text{fail state})$$

and the expected bandwidth is given by:

$$E(BW) = \sum_{f=1}^{16} C_f \, p_d^f \, (1-p_d)^{(16-f)}$$

Again this represents a full calculation of the expected bandwidth given that all failures are data mode type failures. Complete evaluation of coefficients (c_f) in this equation is not computationally feasible. An approximation can, however, be made by restricting the number of input requests used in the calculation. Let n represent the number of random input states used to calculate each expected bandwidth given a failure state used in the above equations. Then by the central limit theorem

$$E(BW|\text{fail state})_{\text{calculated}} = \text{Normal}(\mu_i, \sigma_i^2/n)$$

where μ_i is the actual expected bandwidth given a particular failure state and σ_i is the deviation of the sample distribution. Given that this is true the coefficients in the binomial equation are distributed as follows:

$$C_f = \text{Normal}\left(\sum_{i=1}^{\binom{16}{f}} \mu_i, \sum_{i=1}^{\binom{16}{f}} \sigma_i^2/n \right)$$

and the calculated expected bandwidth is distributed as

$$E(BW)_c = \text{Normal}(\mu' \ , \sigma'^2)$$

where

$$\mu' = \sum_{f=0}^{16} p_d^f (1-p_d)^{(16-f)} \sum_{i=1}^{\binom{16}{f}} \mu_{fi}$$

and

$$\sigma'^2 = \sum_{f=0}^{16} p_d^{2f}(1-p_d)^{2(16-f)} \sum_{i=1}^{\binom{16}{f}} \sigma_{fi}^2/n$$

The μ' above is the actual expected bandwidth and σ' is the deviation of the bandwidth calculated using the simulation. In order to determine a bound on the deviation define σ_{max} such that it is greater than σ_{fi} for all values of f and i. Then

$$\sigma'^2 \le \sigma_{max}^2/n \sum_{f=0}^{16} p_d^{2f}(1-p_d)^{2(16-f)} \quad \binom{16}{f}$$
$$\le \sigma_{max}^2/n$$

Using a two point equally probable 0, 8 distribution, it is easily shown that

$$\sigma_{max}^2 \le 16$$

This gives a .99 confidence interval of $\pm$ .09 for n of 20,000 which was used in the simulation.

A comparison of the expected bandwidths calculated using the model and the simulation is presented in figure 19. Figure 20 is a plot of the absolute difference between the model and the simulation. Figure 21 is a plot of the covariance for p_a equal to zero. The maximum value of the error occurs when p_d is 0.17 and represents approximately a 15% error.

Figure 22 shows the model error as a percentage of the simulation bandwidth. The percentage error continues to increase after the absolute error begins to decrease but remains below 15% for values of p_d below 0.20 which is the area of primary interest. Further it is expected that the absolute and percentage errors will be maximums at the conditions presented, that is for p_a zero and for a low value of N, the number of processors and memories connected to the ICN.

Thus the model developed provides a reasonable approximation to the expected bandwidth even in the worst case conditions for the model. The percentage error in the expected bandwidth calculated using the model should decrease as N increases and as p_a increases. It is not, however, computationally feasible with currently available computational resources, to simulate the ICN for either a larger value of N or for various values of p_a.

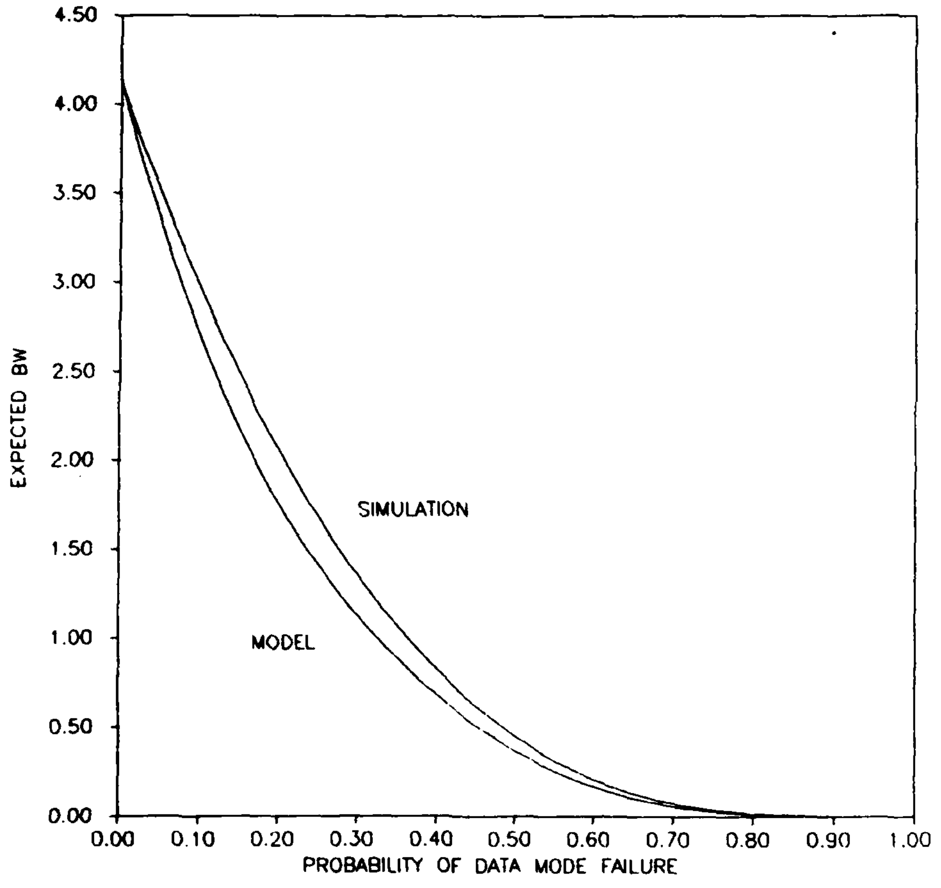


Figure 19 — Expected Bandwidth Model vs Simulation

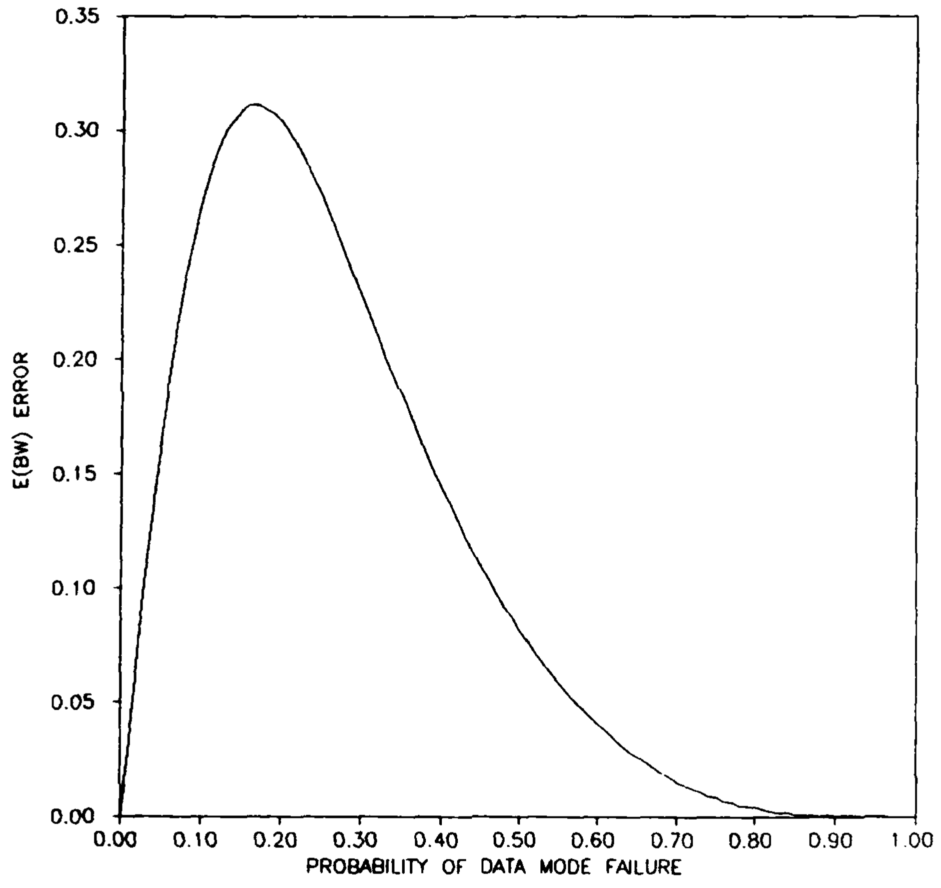


Figure 20 -- Absolute Bandwidth Error

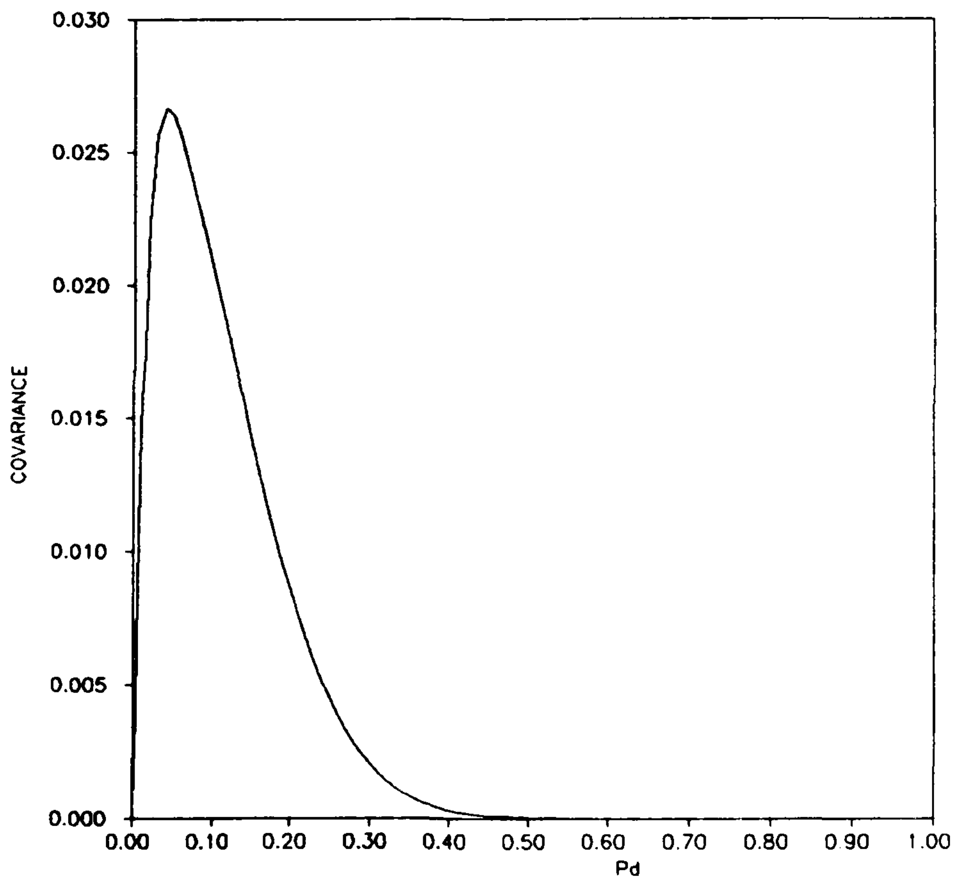
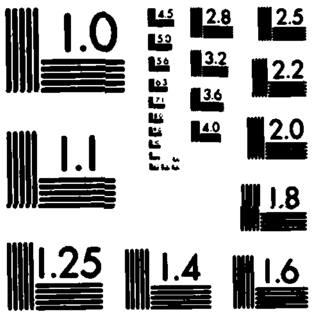
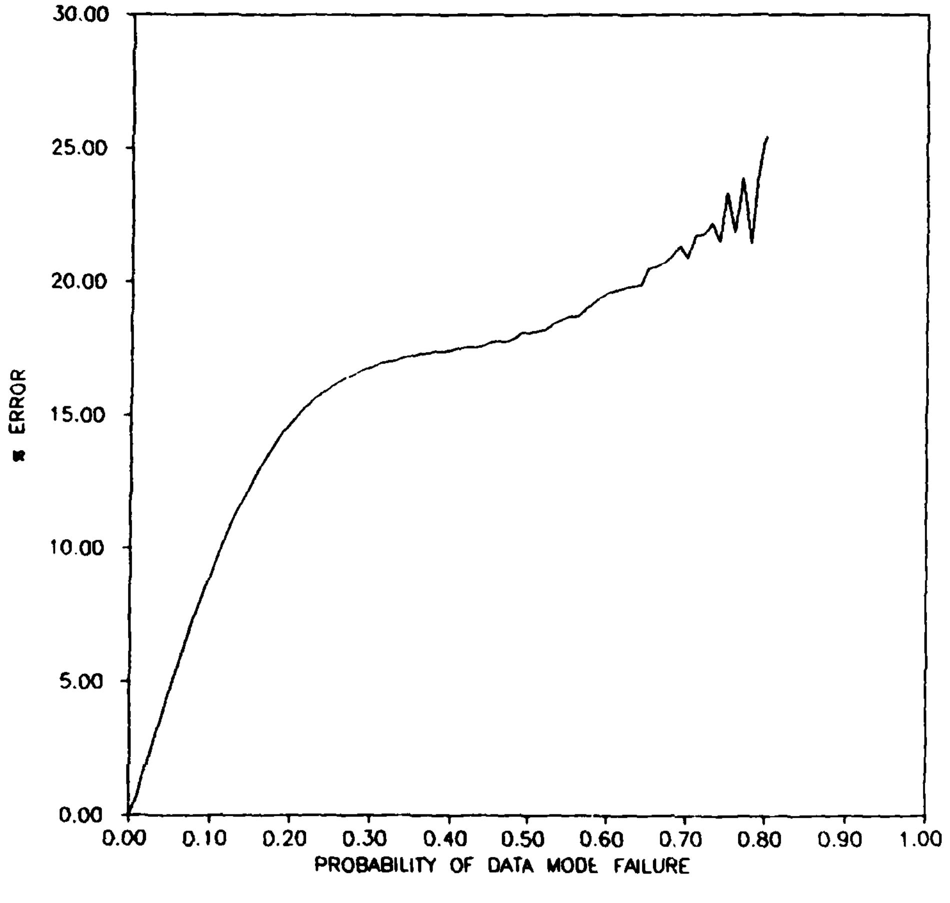


Figure 21 — Covariance vs Pa

AD-A171 287   A COMMUNICATION BANDWIDTH MODEL FOR SHUFFLE-EXCHANGE   2/2
AND AUGMENTED SHUFFL..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH C R BISBEE 1986
UNCLASSIFIED   AFIT/CI/NR-86-115D   F/G 17/2   NL

END
DATE
FILMED
10-86
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0



Figur 22 -- Percentage Bandwidth Error

V. EVALUATION OF RELIABILITY MEASURES — AN EXAMPLE

In this chapter, an example designed to demonstrate the use of the model developed in chapters two and three, to evaluate various reliability enhancement measures as applied to the S/E and S/E+ ICNs, is presented. Before this can be done, however, two tasks must be accomplished. First, as was observed in chapter three, the expected bandwidth of the S/E+ ICN is strictly less than that of the S/E ICN. If these two networks are to be compared directly, connectivity measure must be developed. Such a measure should favor the S/E+ ICN. When combined with the bandwidth analysis, it should allow comparison of the S/E and S/E+ ICNs. Next, p_a and p_d for the switches used in the ICN must be available. For the purposes of this example, a hypothetical switch model which can be used to calculate p_a and p_d will be developed.

Connectivity Equations

Cherhassky[7] has developed equations which can be used to calculate the probability that any pair of terminals can be connected through a tree structured ICN in which data mode faults occur. These equations are not, however, sufficient for this example as they are valid only in the asymptotic sense and they do not extend to the address mode stuck fault model. Thus, complete equations using the more complete fault model must be developed. This is quite simple for the S/E ICN as this ICN contains only a single path from any input to any output. If a connection is to be made, the path between the desired input and output must be fault free or must have only stuck at address mode failures which allow the desired connection. Let P(C) represent the probability that a randomly selected pair of terminals can be connected using the desired ICN. Then

$$P(C)_{s/e} = (1-0.5p_a-p_d)^k$$

In the S/E+ ICN, there are two paths from any input to any output terminal. The desired connection can be made if either of these paths is capable of making the connection. These paths are not, however, independent as they pass through the same first and last stage switches. Thus, in order to calculate $P(C)_{S/E+}$ condition on the failure state of these switches. Thus

$$P(C)_{s/s+} = P(C|\text{stage 0 ok})(1-p_f)+P(C|\text{stage 0 address fail})p_a$$

Now

$$P(C)_{s/s+}(|\text{stage 0 ok})=P'(C|\text{stage k ok})(1-p_f)+P'(C|\text{stage k address fail})p_a$$

Where again P'(C) is used to indicate the conditional probability given that the stage 0 switch in question is functional. If both the stage 0 and stage k switches are functional, the event that a connection can be made occurs if either of the paths, exclusive of the first and last stages, are functional for the desired connection. Thus

$$P'_{S/E+}(C|\text{stage k ok})=1-[1-(1-0.5p_a-p_d)^{k-1}]^2$$

If the stage k switch has failed in the address mode and the stage 0 switch is functioning, the connection can be made if the intermediate path from stage 0 to the appropriate stage k input is error free. Thus

$$P'_{S/E+}(C|\text{stage k address fail})=(1-0.5p_a-p_d)^{k-1}$$

If the stage 0 switch has failed in the address mode then the connection can be made if the remainder of the path from the appropriate output from stage 0 is error free. Thus

$$P_{S/E+}(C|\text{stage 0 address fail})=(1-0.5p_a-p_d)^{k}$$

The above equations can be combined to give the probability that a random pair of terminals can be connected using the S/E+ ICN.

Switch Model

The basic switch model is shown in figure 23. It consists of a data switching circuit and a separate address control and contention logic section. In addition, the ICN designer also has available 1 bit wide, 2 of 3 majority voters packaged in either single voter SSI integrated circuit or a 25 voter integrated circuit contained in a 128 pin JEDEC package as well as a 50 bit wide single error correcting double error detecting unit (ECC). Thus the fault tolerance strategies available are triple modular redundancy (TMR), single error correction (ECC) or any combination of the above.

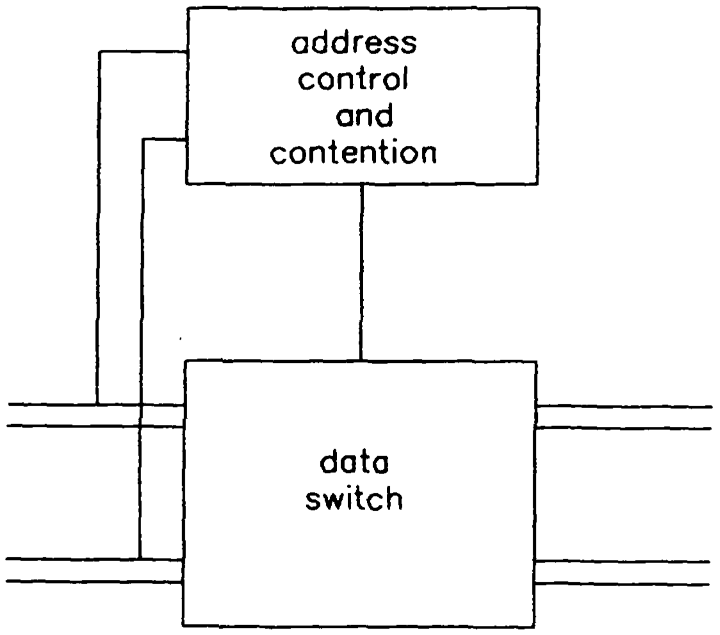


Figure 23 — Switch Model

The primary failure mode for all of components used in the switch design is assumed to be single bit stuck at failures. Further assume that bits within a given circuit are equally likely to fail and have independent Poisson failure distributions. These assumptions were designed to simplify the switch model. It should be realized that the purpose of the switch model is to derive p_a and p_d so as to demonstrate the bandwidth model. The primary purpose of this example is to demonstrate the utility of the bandwidth model in evaluating various reliability enhancement options. An unduly complicated switch model, while perhaps more realistic, complicates and obscures that demonstration.

Table 1 shows the estimated and relative failure rates for the components available to the system designer. The estimated failure rates are based on the gate complexity of the integrated circuits and are taken from Siewiorek[ ] table D-6 which is based on the military handbook 217B reliability model for integrated circuits. The last column of table 1 is the failure rate relative to the single bit failure rate. All data presented for comparison later in this chapter will be presented as a function of time normalized by the bit failure rate. The equations, which will be used to determine p_a and p_d given the assumptions and failure rates for the switch model, will now be developed.

Table 1 — Failure Rates

FUNCTION	#GATES	λ	λ REL	λ REL / BIT
Data switch	300	1.1668	50	1.0
Address	100	0.4839	20	na
Voter(single)	4	0.1207	5	5
Voter(25)	200	0.4935	20	0.8
ECC	200	0.751283	30	na

The data path for the basic system will be 45 bits wide and consists of 32 data/memory control bits 10 ICN address bits, 1 ICN control line and 2 parity lines. When ECC is used the parity bits are replaced with 7 ECC bits resulting in a 50 bit wide data path through the network. The actual data path could decrease by one bit for each stage. This would, however, require different data switches for each stage and would also require a new ECC check bit calculation and insertion at each stage. As a result, assume that the data path width remains constant except at the stage zero of the S/E+. There, the address control bit for stage 0 is removed and not passed to the remainder the network. This is consistent with memory unit address error checking as the stage 0 control bit selects the ICN path but does not effect the memory module addressed.

The ICN models require that the events represented by p_a and p_d be disjoint. If both data mode and address mode failures have occurred in the same switch it must be considered as a data mode failure. Let PD represent the probability that a data mode failure has occurred in a switch and let PA represent the probability that an address mode failure has occurred in the switch. The events represented by PA and PD are independent in the switch model. Thus

$$p_d = PD$$

$$p_s = PA(1-PD)$$

These equations are valid for this switch model regardless of the enhancement features that may be used. Now PA and PD must be derived for the switch model.

For the unenhanced system,

$$PD_{unenhanced} = 1-e^{-45\lambda t}$$

and

$$PA_{unenhanced} = 1-e^{-20\lambda t}$$

For the TMR system, note that in a single bit wide majority voter, the output is correct if the voter operates correctly and either at least two inputs are valid or one is valid and the other two have failed in opposite stuck at levels. Thus

$$PD_{TMR}=1-\{e^{-3\lambda t}+3e^{-2\lambda t}(1-e^{-\lambda t})+3/2e^{-\lambda t}(1-e^{-\lambda t})^2\}^{45}e^{-0.8*45\lambda t}$$

and

$$PA_{TMR}=1-\{e^{-3*20\lambda t}+3e^{-2*20\lambda t}(1-e^{-20\lambda t})+3/2e^{-20\lambda t}(1-e^{-20\lambda t})^2\}e^{-5\lambda t}$$

Finally, for the switch with ECC applied to the data path,

$$PD_{ECC}=[e^{-50\lambda t}+50e^{-49\lambda t}(1-e^{\lambda t})]e^{-30\lambda t}$$

Comparison of Reliability Options

Figure 24 shows the expected bandwidth as a function of λt for a 1024x1024 S/E ICN with several reliability enhancement measures applied to the ICN. The configurations presented are the base or unenhanced ICN, the ICN with TMR applied to both the address logic and the data path outputs, the ICN with ECC applied to the data path and no address redundancy, and the ICN with TMR applied to the address logic and ECC applied to the data path. Figure 25 presents the same data for an S/E+ network. In addition the configuration designated as mixed consists of the S/E+ ICN with ECC applied to the data path and TMR applied to the address logic of the first and last stages only. This is is a resonable configuration for the S/E+ ICN as it is in these stages that the two paths from any input to any output pass through the same switch. Figures 26 and 27 show the probability that a random connection can be made for each of the above configurations. In figures 28 and 29 the probability of connection and the expected bandwidth for the S/E and S/E+ are compared for two configurations.

For values of λt less than 0.006, the maximum expected bandwidth is obtained by applying TMR to the address logic of the ICN and ECC to the data path. For values greater than this the maximum is obtained by applying TMR to both the address logic and to the data path. For values of λt greater than approximately 0.003 the probability of connection is so low that the ICN is, for most practical applications, no longer functional. Thus, in this case, the selection of TMR applied to the address logic and ECC to the data path is clearly superior in terms of ICN bandwidth.

As was noted in chapter 3, the expected bandwidth for the S/E+ network is always lower than than for a similarly configured S/E network. However, the probability that a randomly chosen connection can be made is higher for the S/E+. The is especially significant for low values of λt as is clearly demonstrated in figures 26 and 27.

Table 2 lists the relative costs for each of the above configurations. This cost is based on the total number of gates required, normalized by the number of gates in an *unenhanced* S/E ICN. Once the maximum unrepaired operating time or desired mission time, the minimum acceptable bandwidth and minimum acceptable probability of random connection are specified, the system designer can use the cost model and figures 24-27 to determine the minimum cost ICN meeting these constraints.

The mixed configuration for the S/E+ ICN serves to illustrate another feature of the bandwidth model which was not previously mentioned. The equations developed in chapter 3 assumed that p_a and p_d were constant for all switches in the network. There was, however, nothing in the development which required that this be true. The only requirement is that p_a and p_d be constant for each column in the network. As a result, only minor modifications are required to extend the bandwidth model to such situations.

This was done to compute the expected bandwidth with TMR applied only to the first and last stage address circuits rather than to all address circuits in the ICN.

The relative bandwidths shown in figure 24 only apply for the p_a and p_d derived above and these results should not be considered applicable to all S/E and S/E+ ICNs. Rather the figure demonstrates the use of the model to determine the relative merits of various configurations given that p_a and p_d as a function of time are known.

Table 2 — Relative Cost Based on Gate Count

ICN	ENHANCEMENTS	RELATIVE COST
S/E	NONE	1.00
	TMR	3.55
	ECC	1.62
	TMR/ECC	2.17
S/E+	NONE	1.10
	TMR	3.91
	ECC	1.78
	TMR/ECC	2.39
	MIXED	1.89

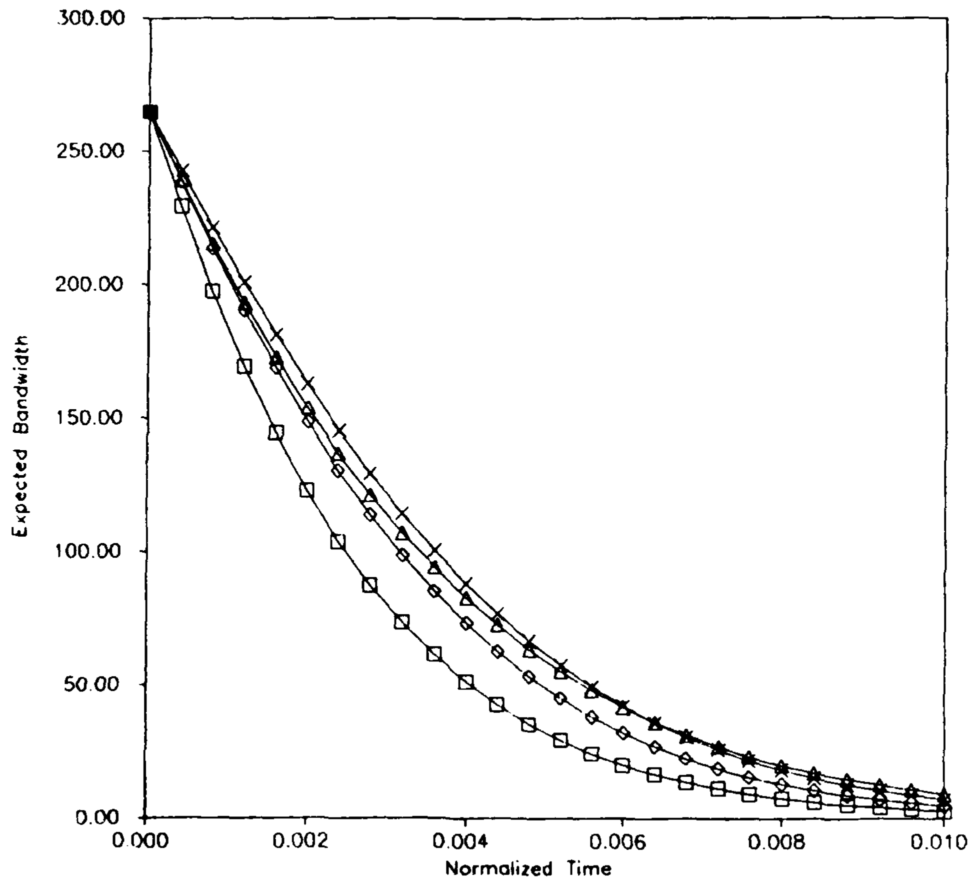


Figure 24 — Expected Bandwidth vs Time S/E ICN

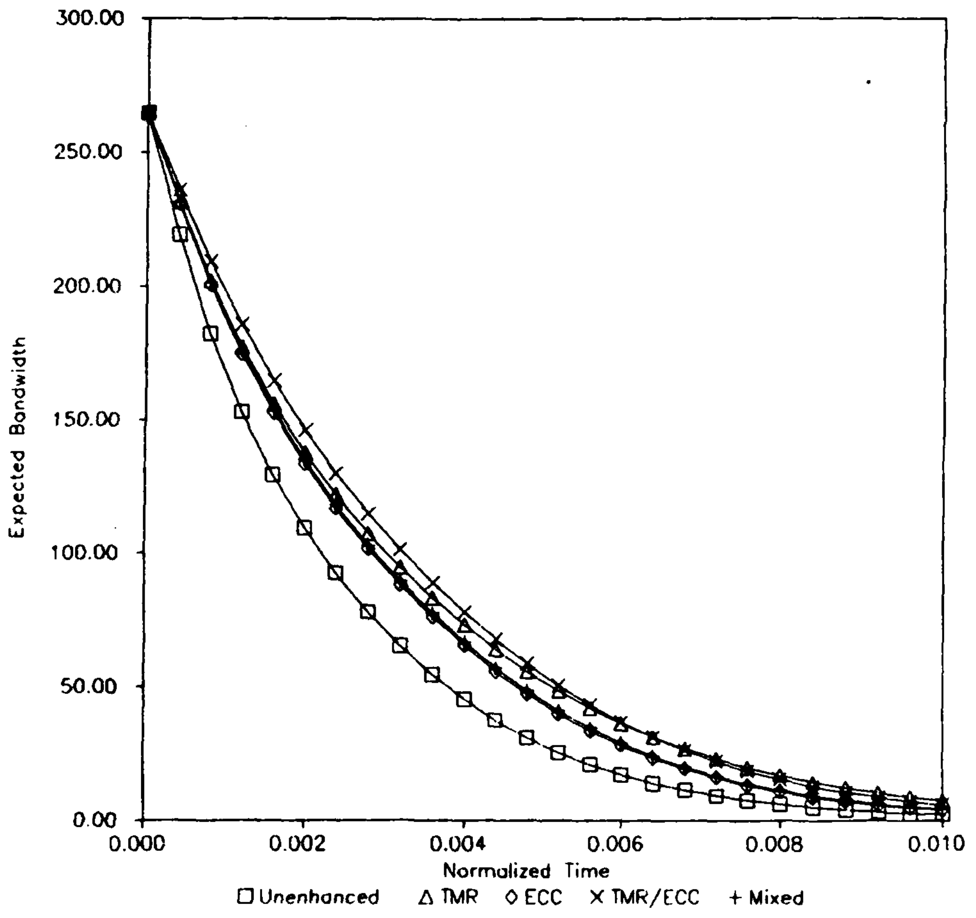


Figure 25 — Expected Bandwidth vs Time S/E+ ICN

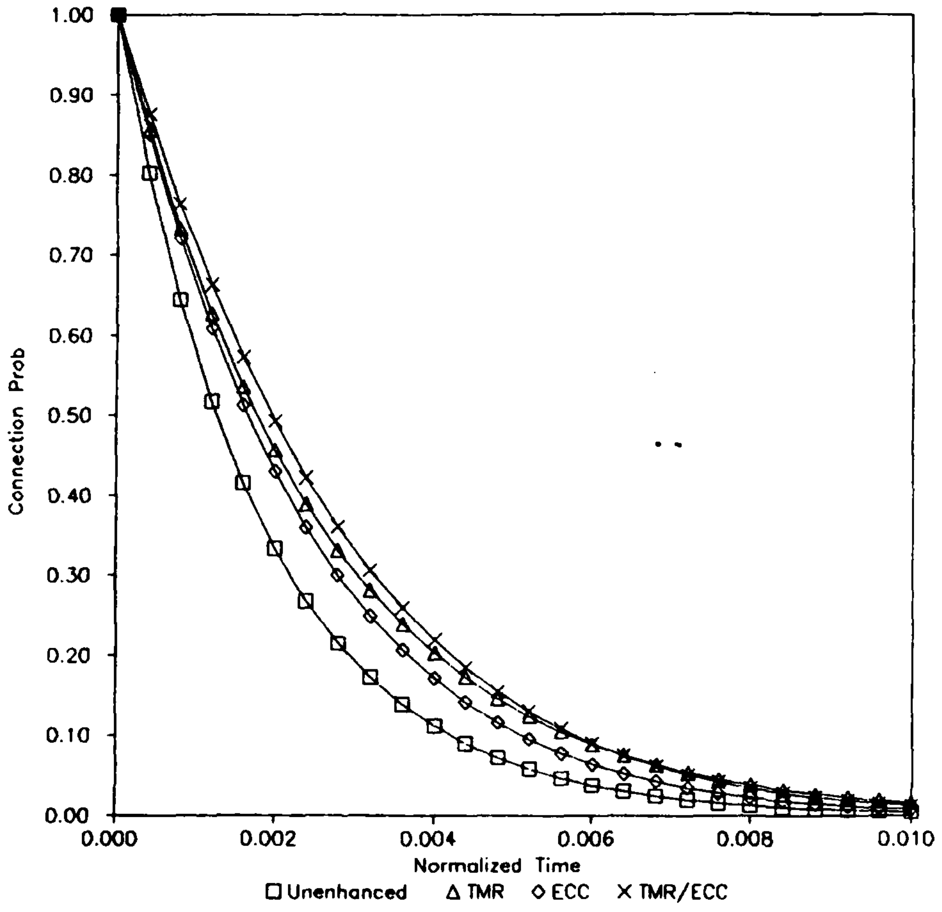


Figure 26 — Probability of Connection S/E ICN

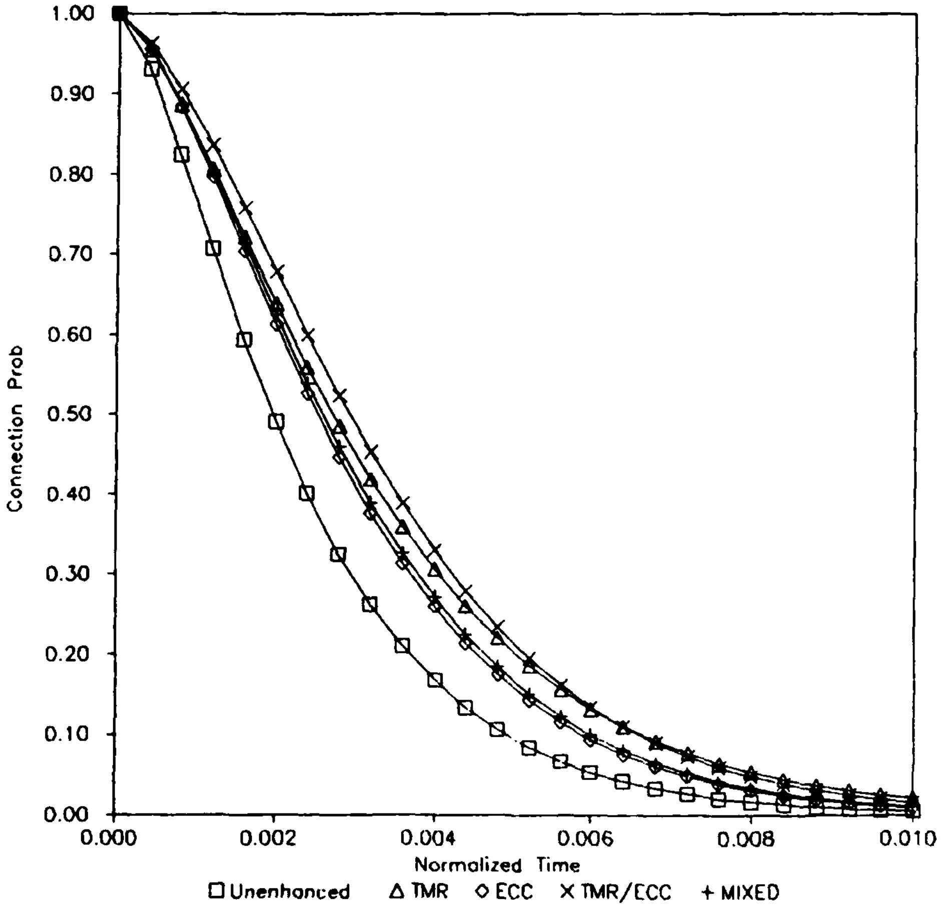


Figure 27 — Probability of Connection S/E+ ICN

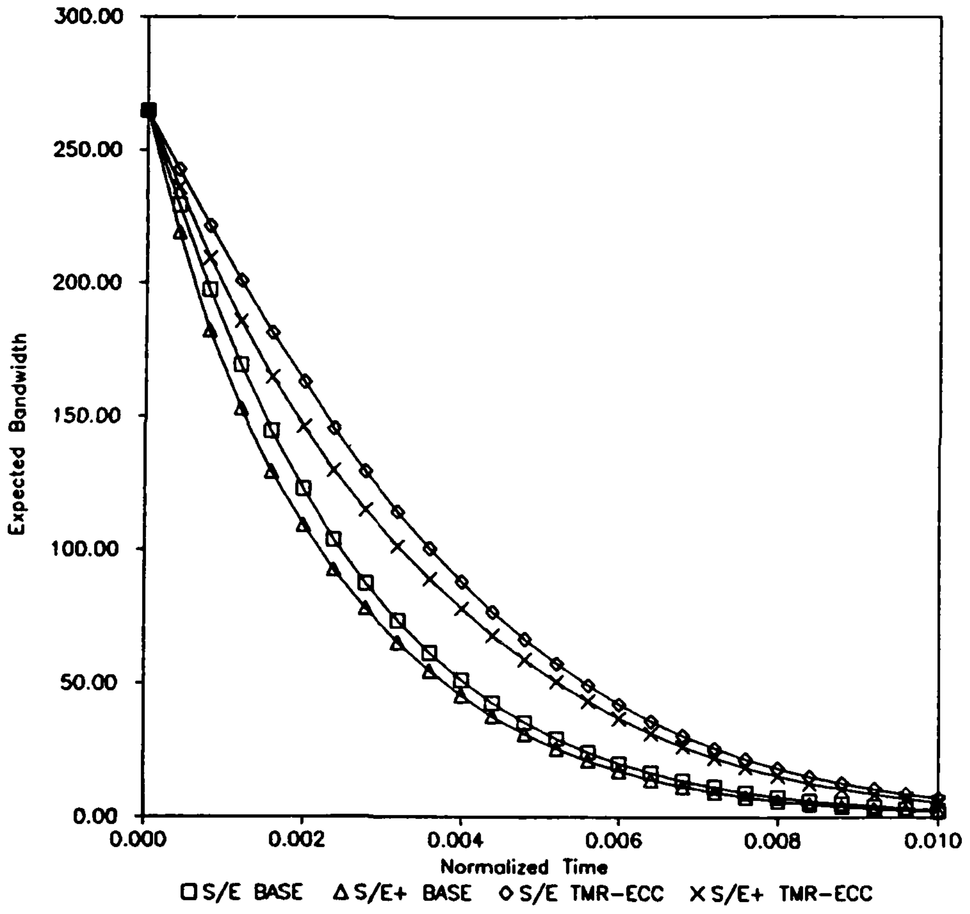


Figure 28 — Bandwidth Comparison S/E vs S/E+

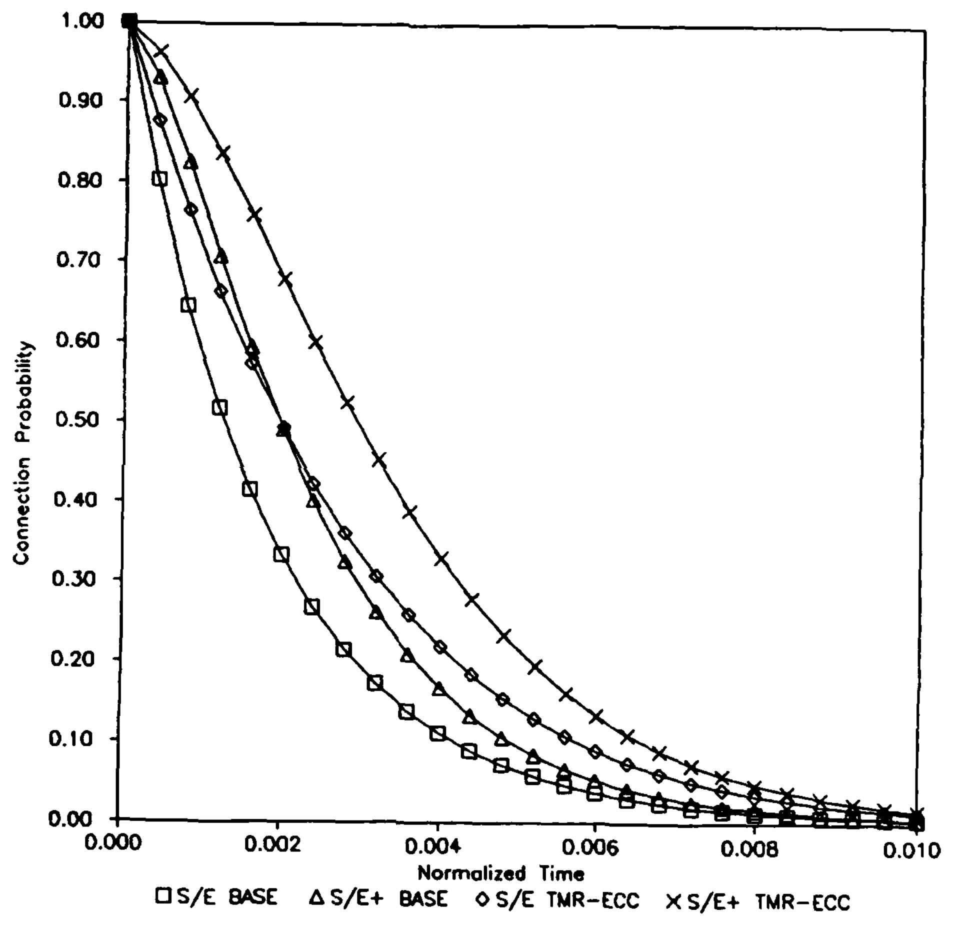


Figure 29 — Connection Probability Comparison S/E vs S/E+

VI. CONCLUSIONS

This dissertation documents a study of a bandwidth analysis of shuffle-exchange (S/E) and augmented shuffle-exchange (S/E+) interconnection networks composed of binary crossbar switches. These networks are intended for use as interconnection and communication networks (ICNs) in large multiprocessor computer systems and are topologically equivalent to a much larger class of interconnection networks. This chapter summarizes the results obtained and suggests some areas for further research.

Summary of Results

The major contributions of this work are summarized as follows:

1. An analysis technique which allows the prediction of ICN bandwidth, in the presence of certain types of failures, has been developed for the S/E and S/E+ ICN.
2. It has been shown that the relatively simple analysis done for the minimal networks composed of $\log_2 N$ stages (S/E) can be extended to networks augmented with an extra stage (S/E+)

provided that the extra stage is appropriately considered.

3. The fault models used previously by other researches have been extended by incorporating both address mode faults and data mode faults into a single model.
4. an example which demonstrates the use of the model to select among various reliability enhancement measures in the design of multiprocessor ICNs was presented.

The binary crossbar shuffle-exchange ICN is composed of $\log_2 N$ stages, each consisting of $N/2$ binary crossbar switches, where N is the *number of input and* output terminals of the network. The stages are interconnected using a perfect shuffle connection pattern. Such networks have been shown to be topologically equivalent to a much larger class of networks which use different stage to stage connection patterns[22,27,28]. Thus results obtained here are applicable to many other proposed interconnection networks.

The augmented shuffle-exchange ICN (S/E+) consists of $\log_2 N+1$ stages. These stages are again interconnected with the perfect shuffle connection pattern. The S/E+ provides a redundant connection path from any input to any output as compared to the S/E network and thus is desirable as a reliability enhancement to the S/E network.

Bandwidth models[7,23,24] have been previously developed for these networks in either the fault free state or in the presence of data path faults — those in which component switches pass no useful data. Additionally, the effect of address mode faults, in which the data is passed unmodified but is incorrectly routed, on the connection capability of the network have been studied[27,28]. Prior to this work, however, these two fault models have not been combined in a single analysis, and no bandwidth analysis has been made using the address mode fault model. The bandwidth model developed in this work is computationally simple and allows the estimation of the bandwidth of these networks given that the probabilities of address and data mode failures for the network's component switches are known. The availability of such a model allows the ICN designer to estimate the effectiveness of switch level reliability measures on network and therefore system performance. This then allows assessment of the cost benefit ratio for various enhancement schemes.

Suggested Further Research

The bandwidth model developed in this work is limited in that it *only* applies to a random distribution of output port requests made at the ICN input ports. Previous research[22,23,37,38] has shown that this assumption has little effect on ICN bandwidth in the fault free state.

However, the control strategy for a fault tolerant computing system would necessarily restrict the accesses of any given input port to those to which connection was possible. The development of a bandwidth model in which non-random input distributions are allowed would improve the ability to estimate system bandwidth in the presence of such reconfiguration.

Several researchers have proposed ICNs composed of crossbar switches of radix greater than two as well as mixed radix switches[5,21]. The development of bandwidth models for such systems would be a natural extension of this work. It would, however, involve substantial work as the number of address mode faults possible at each switch are significantly larger and the analysis thus more complicated.

BIBLIOGRAPHY

[1] Adams and Siegel, 'Extra stage cube: a Fault Tolerant Network for Supercomputers,' IEEE Trans. Comput., vol C-31, May 82.

[2] F. Baskett and A.J. Smith, 'Interference in Multiprocessor Computer Systems with Interleaved Memory,' Commun. Ass. Comput. Mach., vol 19, Jun 76.

[3] V. E. Benes, 'Mathematical Theory of Connecting Networks and Telephone Traffic', Academic Press, New York, 1965.

[4] D.P. Bhandarkar. 'Analysis of Memory Interference in Multiprocessors,' IEEE Trans. Comput., vol. C-24, Sep 75.

[5] L. N. Bhuyan and D. P. Agrawal, 'Design and Performance of Generalized Interconnection Networks,' IEEE Tran. Comput., vol C-32, Dec 83.

[6] D. Y. Chang, D. J. Kuck and D.H. Lawrie, 'On the Effective Bandwidth of Parrallel Memories,' IEEE Tran. Comput., vol C-26, May 77.

[7] V. Cherhassky, E. Opper and M. Malek, 'Reliability and Fault Diagnosis Analysis of Fault Tolerant Multistage Interconnection Networks', Digest of Papers Fourteenth International Conference on Fault Tolerant Computing, Jun 84.

[8] D. M. Dias and J. R. Jump, 'Analysis and Simulation of Buffered Delta Networks,' IEEE Tran. Comput., vol C-30, Apr 81.

[9] J. Elder et al., 'Issues Related to MIMD Shared-Memory Computers: the NYU Ultracomputer Approach', Proceedings 12th Annual International symposium on Computer Architecture, Jun 85.

[10] L. R. Goke and G. J. Lipovski, 'Banyan Networks for Partitioning Multiprocessing Systems,' Proceedings 1st Annual Computer Architecture Conference, Dec 73.

[11] A. Gottleib et al., 'The NYU Ultracomputer -- Designing a MIMD Shared-Memory Parallel Machine', Proceedings 9th Annual Internation Symposium on Computer Architecture, Apr 82.

[12] A. Gottleib et al., 'The NYU Ultracomputer -- Designing an MIMD Shared-Memory Parallel Computer', IEEE Trans. Comput., vol C-32, Feb 83.

[13] R. M. Jenevein and J. C. Browne, 'A Control Processor for a Reconfigurable Array Processor', Proceedings 12th Annual International Symposium on Computer Architecture, Jun 85.

[14] R. N. Kapur, U. V. Premkumar and G. J. Lipovski, 'Organization of the TRAC Processor-Memory Subsystem', National Computer Conference, 1980.

[15] C. D. Kruskal and M. Snir, 'The Performance of Multistage Interconnection Networks of Multiprocessors,' IEEE Tran. Comput., vol C-32, p1091-1098.

[16] V. P. Kumar and S. M. Reddy, 'Design and Analysis of Fault Tolerant Multistage Inteconnection Networks with Low Link Complexity,' Proceedings 12th Annual International Symposium an Computer Architecture, Jun 85.

[17] D. H. Lawrie, 'Access and Alignment of Data on an Array Processor,'IEEE Tran. Comput., vol. C-25, Dec 76.

[18] J. Lenfant, 'Parallel Permutations of Data: A Benes Network Control Algorithm for Frequently Used Permutations,' IEEE Trans. Comput., vol. C-27 78.

[19] D. Nassimi and S. Sahni, 'A Self Routing Benes Network and Parallel Permutation Algorithms,' IEEE Tran. Comput., vol C-30, May 81.

[20] V. P. Nelson and R. L. Fields, 'Hardware and Software Development for the FTDCL Delta-Connected Multi-Microprocessor System,' Proceedings 14th Southeastern Symposium on System Theory, Apr 82.

[21] K. Padmanaban and D. H. Lawrie, 'A Class of Redundant Path Multistage Interconnection Networks,' IEEE Tran. Comput., vol C-32, Dec 83.

[22] D. S. Parker, Jr., 'Notes on Shuffle/Exchange Type Switching Networks' IEEE Trams. Comput., vol. C-29, Mar 80.

[23] J. H. Patel, 'Performance of Processor-Memory Interconnections for Multiprocessors,' IEEE Trans. Comput., vol C-30, Oct 81.

[24] M. C. Pease, ' The Indirect Binary N-cube Microprocessor Array. 'IEEE Trans. Comput., vol. C-26, May 77.

[25] U. V. Premkumar et al. 'Design and Implimentation of the Banyan Interconnection Network in TRAC,' National Computer Conference 1980.

[26] M. C. Sejnowski et al., 'An Overview of the Texas Reconfigurable Array Computer,' National Computer Conference, 1980.

[27] J. P. Shen and J.P. Hayes 'Fault Tolerance of a Class of Connecting Networks,' 7th Annual Symposium on Computer Architecture, May 80.

[28] J. P. Shen 'Fault Tolerance of Beta Networks in Interconnected Multicomputer Systems', Doctoral Dissertation, University of Southern California, Aug 81.

[29] D. P. Siewiorek and R. S. Swarz, 'The Theory and Practice of Reliable System Design,' Digital Press, 1982.

[30] D. Steinberg, 'Invariant Properties of the Shuffle-Exchange and a Simplified Cost-Effective Version of the Omega Network,' IEEE Trans. Comput., vol C-32, May 83.

[31] H. S. Stone. 'Parrallel Processing with the Perfect Shuffle,' IEEE Trans. Comput., vol. C-20. Feb 71.

[32] S. Thanawastien, 'The SE-Plus Network', submitted to ACM Southeastern Regional Conf., 82.

[33] S. Thanawastien and V. P. Nelson, 'Interference Analysis of Shuffle Exchange Networks,' IEEE Trans. Comput., vol C-30, Aug 81.

[34] N. F. Tzeng, P. C. Yew and C. Q. Zhu, 'A Fault Tolerant Scheme for Multistage Interconnection Networks,' Proceedings of the 12th Annual International Conference on Computer Architecture, Jun 85.

[35] K. Y. Wen and D. H. Lawrie, 'Effectiveness of some Processor/Memory Interconnections.' presented at Int. Conf. on Pararallel Processing 76.

[36] C. L. Wu and T. Y. Feng, 'On a Class of Multistage Inteconnection Networks,' IEEE Trans. Comput., vol C-29,

Aug 80.

[37] C. L. Wu and T. Y. Feng, 'The Universality of the Shuffle-Exchange Network,' IEEE Tran. Comput., vol C-30, May 81.

[38] C. l. Wu and T. Y. Feng. 'The Reverse Exchange Interconnection Network.' IEEE Trans. Comput., vol. C-29, Sep 80.

APPENDIX

SIMULATION COMPUTER PROGRAM

```
#include <stdio.h>
unsigned int
      input[8]            /*the input requests for each processor*/
      ,tags[8]            /*source id, redundancy tag, dest id for each
                                   processor*/
      ,failstate[4][4]    /*for each switch 0=ok, 1=SatT, 2=SatX, 3=data fail*/
      ,switchstate[4][4]  /*for each switch 0=not in use 1=set at T,
                                   2=set at X*/
      ,conflicts[4][4]    /*for each switch 1=inputs conflict 0=noconflict*/
      ,redtags[8][8]      /*redundancy tags for each processor memory pair*/
      ,totalconf[4]       /*the number of conflicts in each stage*/
      ,*lines[5][8]       /*pointers to the controling tag for each
                                   stage, line*/
      ,outmap1[8]          /*maps outline at one stage to inline of next*/
        ={0,2,4,6,1,3,5,7}
      ,outmapk[8]          /*map for last stage so it can be treated the*/
        ={0,1,2,3,4,5,6,7} /*same as others*/
      ,numfails[4]         /*used to contain the total number of failures in*/
                           /*a state ie numfails[1]=nuber SatT etc*/
      ,numfail
      ,firstcall
      ,initcall=1          /* when this is 1=true the value in currenstate is*/
      ;                    /* the first state evaled, all others are normal*/
                           /* note well the strong dependence on numerr.  the*/
                           /* nuber of errors must match the initial value in*/
                           /* in currentstate */

char  outfn[80]="se.dat\0",
        ifn[80]="se.dat\0";


unsigned long seed=1234567891,failstat,currentstate;


main()
{  unsigned int stage,i,j,done,count[8];
    double expc,sim();
    long getstate(),temp;
    FILE *ifp;
    srand(seed);
    if ((ifp=fopen(ifn,"r"))==NULL) {
       printf("error opening file %20s\n",ifn);
       currentstate=01;
       }
    else {
       while(fscanf(ifp,"%4X %*d %*d %*d %*lf",&currentstate)!=EOF);
       fclose(ifp);
       }
    numfail=0;
    temp=currentstate;
    for (i=1;i<=16;i++,temp>>=1) if(temp&1) numfail++;
    printf("currentstate=%8x   numfail= %2d",currentstate,numfail);
    if (currentstate==(1<<numfail)-1){
       numfail++;
       initcall=0;
       }
     else {
       firstcall=0;
       getstate(&done);
       initcall=1;
       }
    initlines();
    for (;numfail<=12;numfail++) {
     initstate(numfail);
     done=0;
     while (!done){
       failstat=getstate(&done);
       if (checkblock(failstat));
       else {
          setfailstate1(failstat);
          setredtags();
          expc=0;
          for (j=1;j<=20000;j++){
             getinp();
             settags();
             expc+=sim(0);
             }
          expc=expc/(j-1);
          writeout(failstat,expc);
          }
       }
     }
}

int checkblock(failstat)
unsigned long failstat;
{
   unsigned int i,temp;
   temp=0;
   for (i=0;i<=3;i++) {
      if ((failstat&0xf)==0xf) temp=1;
      failstat>>=4;
      }
   return(temp);
}
```

```
initstate(numerr)
unsigned numerr;
{
   unsigned i;
   if(initcall) initcall=0; else{
      currentstate=01;
      for(i=0;i<numerr;i++) currentstate=currentstate<<1|1;
      currentstate<<=16-i;
      }
   firstcall=1;
}

long getstate(done)
int *done;
{
   unsigned i,num0,num1;
   unsigned long tempstate;
   tempstate=currentstate;
   if(firstcall) {
      firstcall=0;
      if (tempstate==0 || tempstate==0xffff) *done=1; else *done=0;
      return(tempstate);
      }
   num0=num1=0;
   while(tempstate&1 && num1<16) {tempstate>>=1;num1++;}
   while(!(tempstate&1) && num0+num1<16) {tempstate>>=1;num0++;}
   if (num1+num0==16) {
      printf("\nERROR IN GETSTATE num1=%2d  num0=%2d",num1,num0);
      *done=1;
      return(currentstate);
      }
   tempstate>>=1;
   tempstate<<=2;
   tempstate|=1;
   for (i=0;i<num1;i++) tempstate=tempstate<<1|1;
   tempstate<<=num0-1;
   currentstate=tempstate;
            if(currentstate==(1<<numfail)-1) *done=1; else *done=0;
   return(currentstate);
}

getinp()
{
   unsigned int proc;
   for (proc=0;proc<=7;proc++) input[proc]=rand()>>25&7;
}
```

```
setfailstate1(failstat)
unsigned long failstat;
{
   unsigned int stage,swtch,i;
   for(i=0;i<4;i++) numfails[i]=0;
   for (stage=0;stage<4;stage++) for (swtch=0;swtch<4;failstat>>=1,swtch++)
       if (failstat&1) {
          failstate[stage][swtch]=3;
          numfails[3]++;
          }
         else failstate[stage][swtch]=0;
}

setfailstate2(failstat)
unsigned long failstat;
{
   unsigned int stage,swtch,state;
   for (stage=0;stage<4;stage++) for (swtch=0;swtch<4;failstat>>=2,swtch++){
       state=failstat&3;
       failstate[stage][swtch]=state;
       numfails[state]++;
       }
}

settags()
{
   unsigned int proc,mem;
   for(proc=0;proc<=7;proc++){
      mem=input(proc);
      tags[proc]=(proc<<4)+(redtags[proc][mem]<<3)+mem;
      }
}

initlines()
{
    int line;
    for (line=0;line<=3;line++) {
       lines[0][2*line]=(&tags[line]);
       lines[0][2*line+1]=(&tags[line+4]);
       }
    nullines();
}
```

```
nullines()
{
   unsigned int stage,line;
   for(stage=1;stage<5;stage++)
    for(line=0;line<=7;line++) lines[stage][line]=NULL;
}
 writeout(failst,val)
double val;
long failst;
{ FILE *fp;
   fp=fopen(outfn,"a");
   fprintf(fp,"%91x %3d %3d %3d %13.10f\n",failst,numfails[1],numfails[2],
                                                     numfails[3],val);
   fclose(fp);
}

setredtags()
{
   unsigned int proc,mem,tag,stage,s0;
   for(proc=0;proc<=7;proc++){
   s0=proc >> 2 & 1;
      for(mem=0;mem<=7;mem++) {
         tag=(proc << 4)+(s0 << 3)+mem;
         redtags[proc][mem]=s0;
         stage=1;
               while((stage<=3) && (redtags[proc][mem]==s0)) {
            switch(failstate[stage][(tag >>4-stage)&3]) {
               case 0 : break;
               case 1 : if ((tag>> 3-stage & 1)^(tag>> 6-stage & 1))
                              redtags[proc][mem]=~s0&1;
                        break;
               case 2 : if (!((tag>> 3-stage & 1)^(tag>> 6-stage & 1)))
                              redtags[proc][mem]=~s0&1;
                        break;
               case 3 : redtags[proc][mem]=~s0&1;
               }
            stage++;
            }
         }
      }
}
```

```
setlines(stage)
int stage;
{
   unsigned int swtch,*outmap;
   if (stage==3) outmap=outmapk;else outmap=outmapl;
   if (stage==0)
      for (swtch=0;swtch<=3;swtch++) {
       switch(failstate[0][swtch]) {
          case 0:switch(switchstate[0][swtch]){
                        case 0:lines[1][outmap[swtch*2]]=NULL;
                               lines[1][outmap[2*swtch+1]]=NULL;
                               break;
                        case 1:switchT(0,swtch,outmap);break;
                        case 2:switchX(0,swtch,outmap);
                        }
                 break;
          case 1:lines[1][outmap[swtch*2]]=lines[0][swtch*2];
                 lines[1][outmap[2*swtch+1]]=lines[0][swtch*2+1];
                 break;
          case 2:lines[1][outmap[swtch*2]]=lines[0][swtch*2+1];
                 lines[1][outmap[2*swtch+1]]=lines[0][swtch*2];
                 break;
          case 3:lines[1][outmap[swtch*2]]=NULL;
                 lines[1][outmap[2*swtch+1]]=NULL;
       } }
     else
      for (swtch=0;swtch<=3;swtch++) {
       switch(failstate[stage][swtch]) {
          case 0:switch(switchstate[stage][swtch]){
                        case 0:lines[stage+1][outmap[swtch*2]]=NULL;
                               lines[stage+1][outmap[2*swtch+1]]=NULL;
                               break;
                        case 1:switchT(stage,swtch,outmap);break;
                        case 2:switchX(stage,swtch,outmap);
                        }
                 break;
          case 1:if(switchstate[stage][swtch]==1) switchT(stage,swtch,outmap);
                  else { lines[stage+1][outmap[swtch*2]]=NULL;
                         lines[stage+1][outmap[2*swtch+1]]=NULL;
                         }
                 break;
          case 2:if(switchstate[stage][swtch]==2) switchX(stage,swtch,outmap);
                  else { lines[stage+1][outmap[swtch*2]]=NULL;
                         lines[stage+1][outmap[2*swtch+1]]=NULL;
                         }
                 break;
                case 3:lines[stage+1][outmap[swtch*2]]=NULL;
                 lines[stage+1][outmap[2*swtch+1]]=NULL;
}    } }
```

```
switchX(stage,swtch,outmap)
int stage,swtch,*outmap;
{
   unsigned int *upper,*lower;
   upper=lines[stage][2*swtch];
   lower=lines[stage][2*swtch+1];  /* try upper+1*/
             if(upper==NULL) lines[stage+1][outmap[2*swtch+1]]=NULL;
   else
     if((*upper>>3-stage&1)==1) lines[stage+1][outmap[2*swtch+1]]=upper;
                                else lines[stage+1][outmap[2*swtch+1]]=NULL;
   if(lower==NULL) lines[stage+1][outmap[2*swtch]]=NULL;
   else
     if((*lower>>3-stage&1)==0) lines[stage+1][outmap[2*swtch]]=lower;
                                else lines[stage+1][outmap[2*swtch]]=NULL;
}

switchT(stage,swtch,outmap)
int stage,swtch,*outmap;
{
   unsigned int *upper,*lower;
   upper=lines[stage][2*swtch];
   lower=lines[stage][2*swtch+1];  /* try upper+1*/
              if (upper==NULL) lines[stage+1][outmap[2*swtch]]=NULL;
   else
      if((*upper>>3-stage&1)==0) lines[stage+1][outmap[2*swtch]]=upper;
                                else lines[stage+1][outmap[2*swtch]]=NULL;
            if (lower==NULL) lines[stage+1][outmap[2*swtch+1]]=NULL;
   else
      if((*lower>>3-stage&1)==1) lines[stage+1][outmap[2*swtch+1]]=lower;
                                else lines[stage+1][outmap[2*swtch+1]]=NULL;
}

unsigned int setswitches(stage)
unsigned int stage;
{
   unsigned int count,swtch,uppreq,lowreq;
   count=0;
   uppreq=lowreq=3;
   for(swtch=0;swtch<=3;swtch++) {
      conflicts[stage][swtch]=0;
      uppreq=lowreq=3;
      if(lines[stage][swtch*2]) uppreq=(*lines[stage][swtch*2]>>3-stage&1);
      if(lines[stage][swtch*2+1])
         lowreq=(*lines[stage][swtch*2+1]>>3-stage&1);
      if (failstate[stage][swtch]==0) {
                 if(uppreq==3 && lowreq==3) switchstate[stage][swtch]=0;
                  else if(uppreq==3) switchstate[stage][swtch]=(lowreq)?1:2;
                  else if(lowreq==3) switchstate[stage][swtch]=(uppreq)?2:1;
                  else if(lowreq==1 && uppreq==0) switchstate[stage][swtch]=1;
```

```
                    else if(lowreq==0 && uppreq==1) switchstate[stage][swtch]=2;
                    else {switchstate[stage][swtch]=1;
                          conflicts[stage][swtch]=1;
                          count++;}
                    }
          else switchstate[stage][swtch]=failstate[stage][swtch];
          }
   return(count);
  }

initswitchstate()
{
   int stage,swtch;
   for(stage=0;stage<=3;stage++) for(swtch=0;swtch<=3;swtch++){
       switchstate[stage][swtch]=0;
       conflicts[stage][swtch]=0;
       }
}

double sim(stage)
unsigned int stage;
{
   unsigned int line,conflictstate,lastconfstate,stge;
   float count;
   if(stage==3) {
     setswitches(3);
     setlines(3);
     count=0;
     for(line=0;line<=7;line++) if(lines[4][line]!=NULL) count+=1.0;
/*dd
printinp(5,10);
printall(5,18);
getchar();
*/
     return(count/(1<<totalconf[0]+totalconf[1]+totalconf[2]));
     }
    else {
     totalconf[stage]=setswitches(stage);
     setlines(stage);
     count=sim(stage+1);
     lastconfstate=1<<totalconf[stage];
     for(conflictstate=1;conflictstate<lastconfstate;conflictstate++){
       setconflict(stage,conflictstate);
       setlines(stage);
       count+=sim(stage+1);
       }
     return(count);
     }
}
```

```
setconflict(stage,confstate)
unsigned int stage,confstate;
{
   unsigned int swtch;
   for(swtch=0;swtch<=3;swtch++)
      if(conflicts[stage][swtch]) {
       if(confstate&1) switchstate[stage][swtch]=2;
          else switchstate[stage][swtch]=1;
       confstate>>=1;
       }
}
```

DATE
FILMED
0-8