NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

THE SYNERGISTICALLY INTEGRATED RELIABILITY ARCHITECTURE: A RELIABILITY ANALYSIS OF AN UNTRA-RELIABLE FAULT TOLERANT COMPUTER DESIGN

by

Ronald J. Nelson

September 1986

Thesis Advisor: L.W. Abbott

Approved for public release; distribution is unlimited.

T232203

ECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
b DECLASSIFICATION/DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School — 6b OFFICE SYMBOL (If applicable) 62	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
c. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
a NAME OF FUNDING/SPONSORING ORGANIZATION — 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

1 TITLE (Include Security Classification)
THE SYNERGISTICALLY INTEGRATED RELIABILITY ARCHITECTURE: A RELIABILITY ANALYSIS OF AN UNTRA-RELIABLE FAULT TOLERANT COMPUTER DESIGN

2 PERSONAL AUTHOR(S)
Ronald J. Nelson

3a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ______ TO ______	1986 September 26	137

6 SUPPLEMENTARY NOTATION

COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Fault Tolerance; Inversion Programming; SIR

9 ABSTRACT (Continue on reverse if necessary and identify by block number)
This thesis develops a Semi-Markov reliability model for the Synergistically Integrated Reliability (SRI) computer architecture. The SIR architecture is an advanced hybrid redundancy scheme that combines several current reliability techniques to achieve hardware and software reliability. These methods include hybrid redundancy, N-Version programming and source congruent data interchange. The architecture is designed to support active control systems in the aircraft avionics industry as well as the bus controller requirements for the Dispersed Sensor Processor Mesh(DSPM) system for ultra-reliable computer communications. The paper also develops high level algorithms for fault detection, location, and configuration management within the SIR system.

The reliability model integrates the hardware design, the hybrid redundancy philosophy, and the operating constraints of an active control system into a single reliability model. Specific models are developed for the 3, 4, and 5 processor cases of the SIR architecture and plots of the system reliability vs mission time are generated using the SURE Reliability Analysis Program

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT] ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL Prof Larry Abbott	22b TELEPHONE (Include Area Code) (408)646-2379	22c OFFICE SYMBOL 62At

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE

The Synergistically Integrated Reliability Architecture: A Reliability Analysis of an Ultra-Reliable Fault Tolerant Computer Design

by

Ronald J. Nelson
Captain, United States Army
B.S.E.E., Virginia Polytechnic and State University, 1977

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1986

ABSTRACT

This thesis develops a Semi-Markov reliability model for the Synergistically Integrated Reliability (SIR) computer architecture. The SIR architecture is an advanced hybrid redundancy scheme that combines several current reliability techniques to acheive hardware and software reliability. These methods include hybrid redundancy, N-Version programming and source congruent data interchange. The architecture is designed to support active control systems in the aircraft avionics industry as well as the bus controller requirements for the Dispersed Sensor Processor Mesh (DSPM) system for ultra-reliable computer communications. The paper also develops high level algorithms for fault detection, location, and configuration management within the SIR system.

The reliability model integrates the hardware design, the hybrid redundancy philosophy, and the operating constraints of an active control system into a single reliability model. Specific models are develoed for the 3, 4, and 5 processor cases of the SIR architecture and plots of the system reliability vs mission time are generated using the SURE Reliability Analysis Program.

Thesis
N36665
c.1

TABLE OF CONTENTS

I. INTRODUCTION

The role that computers play in controlling complex systems has increased dramatically with the advent of low cost microcomputers. Research into fault tolerant computing has also intensified due to the increased cost benefits available when microprocessors are used as redundant system elements.

The combination of fault tolerance and complex system control in a microprocessor based system has made it possible to create cost effective, real time control systems for use in systems in which a failure could have life threatening results. Real time control systems allow design of complex systems at, or near, instability points. Designs of this type offer important economic and performance gains, but there is little tolerance available to account for fluctuations in the operational environment.

Advanced avionics is a category of applications where these concepts can be used. Advanced avionics incompasses the application of real time active control technology to govern a variety of in-flight maneuvers that are designed to enhance cost and performance measures of airplanes.

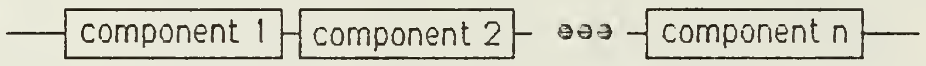
An example using these concepts is the digital fly-by-wire program (DFBW) being researched at the NASA Ames Research Center, Dryden Flight Research Facility. This program uses an F-8 aircraft that is modified to use active controls so that the flight of the airplane can be controlled by a digital computer. The airframe's flight status is updated every 20 ms by a set of sensors with the information being supplied to a computer. The computer analyses the flight data and instructs a set of servo mechanisms to modify the flight pattern to conform to a given set of flight laws. The airframe is designed to be statically stable so the maximum bounds on the length of a control cycle is on the order of 200 ms. The ability of the flight laws the handle possible flight situations is suspect if the upper bound of the control cycle is exceeded. [Refs. 1,2,3]

Another program that is being studied at the Dryden Research Facility uses an X-29 airframe modified to reduce the static stability margin required to fly the aircraft effectively. The avionics control package designed to provide adequate flight control of the aircraft must react within a control cycle that is on the order of 20 to 30 ms in duration. Uncontrolled flight has the possibility of producing oscillatory airframe behavior with the amplitude of the oscillation doubling every 100 ms. Failure to control instability of this type results in the breakup of the aircraft within a very small time period. [Refs. 1,2,3]

The benefits of design with close environment tolerances are very real for the aircraft industry. Active control technology applied to the avionics industry is a field that uses control systems to supply inputs to the effector mechanisms that control the behavior of aircraft flight (independant of specific pilot direction). Estimations on performance increases made possible by using active controls in statically instable designs vary with the design choices made and the area where enhancement is desired. Boeing Aircraft, in a study for the U.S. Air Force, concluded that 25% fuel savings are possible. Boeing also projected a possible 15% increase in payload or a 7% increase in aircraft range for a SST aircraft [Ref. 2,pp. 13]. These figures certainly indicate that implementation of active controls is desirable if the control system can reliably maintain a reasonable margin of operational safety. [Ref. 1]

Aircraft safety is an intensively regulated endeavor. The Federal Aviation Administration (FAA) currently requires reliability figures on aircraft in the range of 10^{-9} catastrophic failures per flight hour for flights with duration of up to 10 hours. Obviously, this reliability requirement would be applied to aircraft designed to the above specifications.

Achieving a system reliability of this magnitude is no trivial task. The system reliability depends on more than the computer itself. The reliability of a series of components degrades as the product of the component reliabilities, even using identical components as is shown in Figure 1-1. In order to meet the FAA requirements for system reliability, the system components must all be ultra reliable. The components of such a system are depicted in Figure 1-2.



$$R_{\text{system}} = \left(R_{\text{component 1}}\right)\left(R_{\text{component 2}}\right)\cdots\left(R_{\text{component n}}\right)$$

Figure 1-1 System Reliability for Components in Series

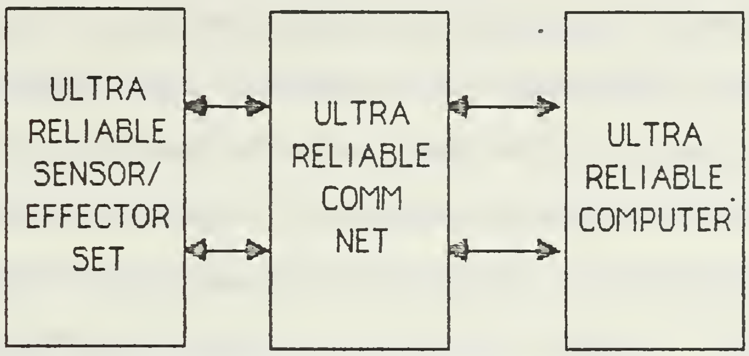


Figure 1-2 Ultra Reliable Computer Control System

A great deal of research on this method of achieving system reliability has already been accomplished. Of concern to this thesis is work conducted at NASA's Dryden Flight Research Facility on the Dispersed Sensor Processor Mesh (DSPM) [Refs. 1,4,5] and current research at the Naval Postgraduate School (NPS) on fault tolerant computers [Refs. 6,7]. The DSPM is an ultra reliable communications network that is to be used in connecting ultra reliable sensor/effector sets with an ultra reliable computer. The DSMP, based on work by Smith [Ref. 8], is an external communications network that monitors and controls the buses providing data to and from the ultra reliable computer and the sensor/effector sets [Refs. 1,4,5].

The current research being conducted by Dr. Abbott at NPS concerns itself with the computer portion of the ultra reliable system. A proposed architecture to satisfy the high reliability requirements is the Synergistically Integrated Reliability (SIR) architecture [Ref. 6]. A hardware design implementing this architecture was recently created at NPS by Captain Virgil Spurlock, US Army [Ref. 7].

The SIR architecture is an advanced hybrid redundancy scheme that combines several of the most current reliability techniques to acheive hardware and software reliability. These methods include hybrid redundancy, N-version programming, and source congruent data interchanges [Refs. 6,9]. The basic architecture is displayed in Figure 1-3. Note that there is no single component that directs the actions of the redundant set of processors. Each active processor is totally independant, makes its own decisions on the correct course of action, and controls some portion of the external sensor/effector set.

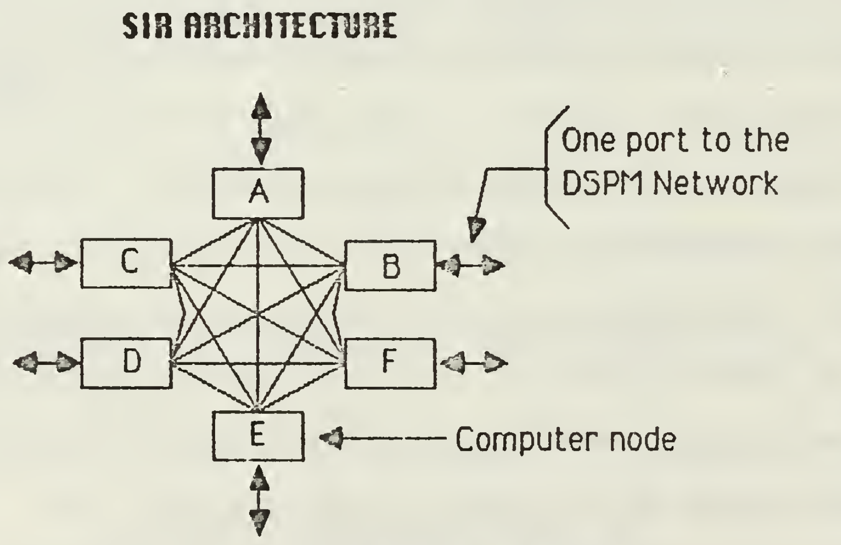


Figure 1-3 SIR Architecture

N-version programming, source congruency algorithms for data supplied to the SIR system, and the lack of a system control point force added complexities in both the hardware implementation and the software that controls task flow in the system. A more thorough discussion of the implications of these techniques on the architecture implementations will be made in Chapters II and V.

The purpose of the SIR architecture is to tolerate a number of faults while still providing results that are judged to be correct within a defined range of confidence. The triple modular redundancy (TMR) model, on which hybrid redundancy is based, specifically judges that a minimum acceptable confidence level can only be obtained when at least two communicating components (processors) agree on the result of a test applied to system data values. The confidence level of correct system operation increases as more components can be used in the verification process. Ideally, a configuration control algorithm will modify the system configuration to one that provides the optimum confidence in system output given the occurance of one, or a series, of specific faults.

Modeling the reliability of such a system, or the Mean Time To Failure (MTTR) for the system taken as a whole, is of course very dependant on the reliability of the components of the system as well as the ability of the system to correctly identify both the occurrence of a fault and its precise location within the present system configuration. The system reliability is thus directly dependant on not only the component hardware reliabilities, but also on an operating protocol that includes a system of fault tests and a system of reconfiguration algorithms. The configuration control algorithm is one of a number of algorithms that will detect, locate, and configure around system faults. The group of algorithms, taken as a whole, will be referred to as operating protocols thoughout the remainder of this paper.

Additional complications are created by the real time nature of the control problem that is to be solved by the SIR computer system. All decisions on correct data values must be made within the context of a 20 to 30 ms control cycle. This implies that any fault detection tests or system configuration management tasks must also conform to this stringent operational cycle.

The set of system configurations consisting of all possible combinations of system components in either good or faulty states is referred to as the set of system states. The size of the set of system states is 2^n, where there are n components in the system each with two states, good or faulty. The subset of all system states that must be controlled by the redundancy management software must be small enough to allow operation within the control window.

This thesis is concerned with developing a reliability model for the SIR architecture that is appropriate to the environment in which it is to operate. A high level specification will be generated for the algorithms that performs fault detection, location, and configuration management tasks. A secondary goal of the thesis is to analyze the model with a semi-Markov analysis tool that was developed at NASA, Langley by Butler based on work by White and Lee [Refs. 10,11,12]. A modification of this program to provide a graphical, event driven, user interface and allow it to operate on an IBM PC-AT microcomputer is being developed as a thesis at NPS by Major John Bordeaux, USMC. The reliability model generated by this thesis will serve as a test vehicle for the program conversion.

II. RELIABILITY ANALYSIS OF THE SIR NODE DESIGN

The hardware of the individual SIR computer nodes must satisfy two equally important design constraints. The nodes must be as simple as possible while still meeting the computational requirements of the problem to be solved. The nodes must also operate with sufficient speed to complete the requisite calculations as well as complete any algorithms that detect errors and determine the correct course of action subsequent to error detection. An additional restraint on the node design is the requirement for the hardware to support N-version programming.

The SIR architecture is based on a variation of basic hybrid redundancy. Basic hybrid redundancy, shown in Figure 2-1, is an organizational scheme proposed by Siewiorek [Ref. 13] that achieves increased reliability by using redundant processors and a voting procedure to decide on a correct answer.

Basic hybrid redundancy utilizes three on-line computer nodes to determine the correct system output value, with the remaining computers being either spare or failed. The rotary multiplexer controls which of the five computers are connected to the voter. The voter performs a bit by bit comparison of the three data streams from the active computer nodes. The correct result is sent to the external interface. The voter rejects any active node values that do not match the other two on-line nodes. A status of the vote is returned to the rotary multiplexer for use in selecting which three processors from the total set will be active.

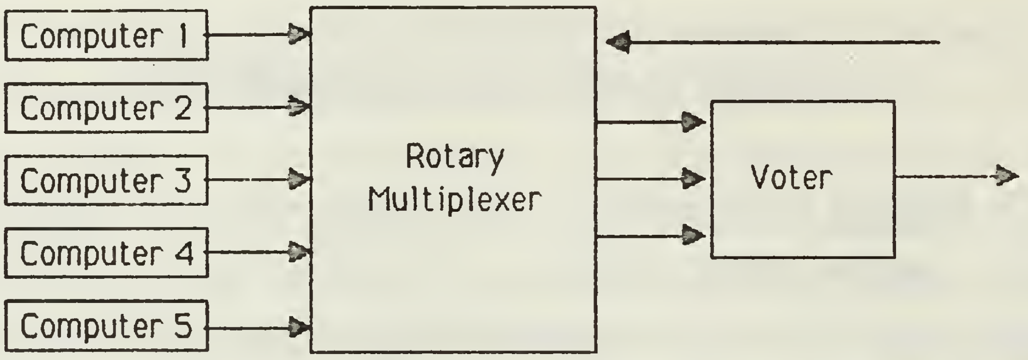


Figure 2-1 Basic Hybrid Redundancy

The N-version concept in software reliability was proposed by Chen and Avizienis [Ref. 14] and is similar to the basic hybrid redundancy technique for hardware. In this case, multiple versions of a computational function are written to the same software specification. The versions may be written in different languages or compiled with different compilers, but the effect proposed is to eliminate a class of software errors that are data driven. This theory states that it is unlikely that the same data driven programming error will surface in all of the programs in exactly the same way. Of course this does not provide proof against design faults but the process could be extended to the software specification also.

N-version programming imposes several constraints on the hybrid redundancy scheme for increased hardware reliability. The different software versions of the function being implemented will behave differently with respect to the roundoff and truncation errors that are inherent in digital computers. This will cause slight variations in the values produced by these routines. Slight variations are quite acceptable if the variations remain within some preset tolerance. If this small expected difference in values is to be tolerated then the voter in the hybrid redundancy scheme cannot be based on an identical match of data values.

There will also be some small differences in the time when specific values are made available to the voters of the three active processors. This variation is due to the differences in the algorithms and the language efficiencies for different N-version programs that generate the data that will be tested by the vote process. Some method and the hardware to support it must be available to synchronize these time skewed values prior to the vote process.

The basic hybrid redundancy system requires lock step sychronization and performs a bit by bit identity comparison on the data being voted. These requirements of basic hybrid redundancy do not support N-version programming.

Regardless of the redundancy and fault tolerant strategy utilized in a fault tolerant system, all fault tolerant systems contain sections, called hardcores, that must work for the system to work. The voter and the rotary multiplexer form a hardcore for the basic hybrid redundancy scheme. The hardcore represents a single point of failure that could result in system catastrophic failure. The voter can vote the wrong computer out; the rotary multiplexer can select the wrong computer node for the voter.

The SIR architecture differs from basic hybrid redundancy in order to reduce the hardcore problems mentioned above. The system still relies on a triad of active computers for detection of error conditions and for deciding the correct value in the presence of an error. A block diagram depicting the design of a SIR node is shown in Figure 2-2.

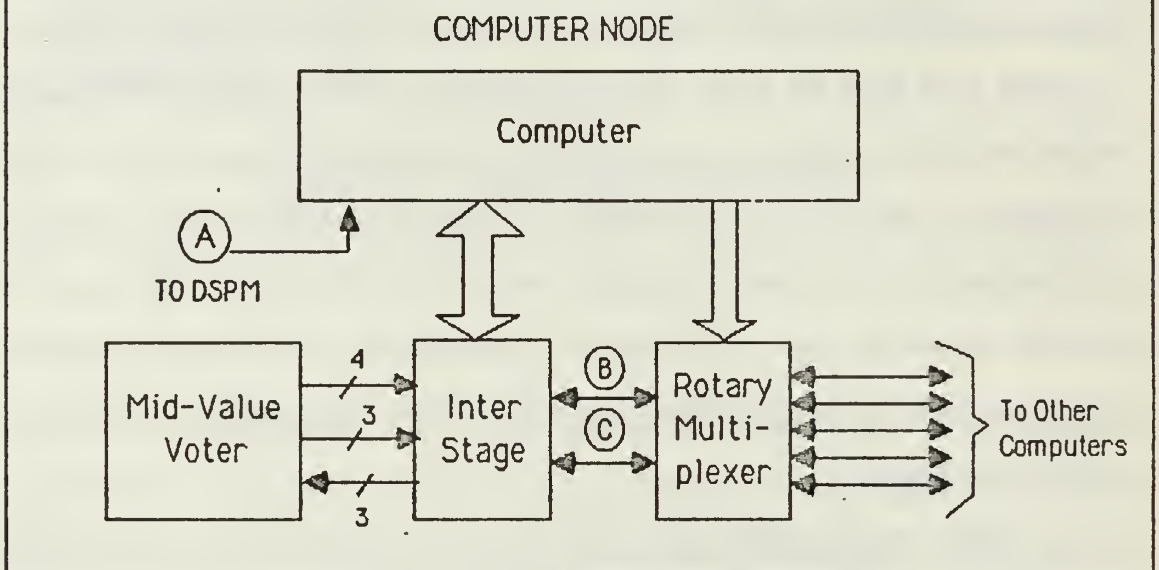


Figure 2-2 SIR computer node

Each of the computer nodes in the SIR system contains a voter and a rotary multiplexer. This allows a great deal more flexibility in configuration management. The strategy also removes the voter and multiplexer from hardcore status. The system can tolerate faults not only in the host computers, but also in a multiplexer or voter and still continue to operate with a high degree of confidence. Recall that the minimum confidence level requirement of the system is two correctly operational host computers and a communications link between them.

The design of the voter is another major difference in the basic hybrid redundancy scheme and the SIR architecture. SIR meets the requirements of

N-version programming and the source congruency algorithms (discussed in Chapter V) by using a mid value voter. The mid value voter concept performs a bit by bit comparison of 3 values and returns those same values sorted in value order (integer values). The mid value of this data triple is taken to be the most correct; it is also the value that will be supplied to the host computer for further processing or communication with SIR's external interface (DSPM). In addition, the state of the voter process is given. The voter status register contains a maximum or minimum indication. If there is no minimum indication then the two smallest values are equal. If there is no maximum indication then the two largest values are equal. If neither indication is given then all three values are equal. The price paid by this increase in functionality is in increased complexity of the voters. Complexity increases equate to decreases in component reliability as will be discussed in section A of this chapter.

The SIR interstage is designed to control the data exchange process between the voter and its sources of data. It is a rather complicated affair because each node is designed to be completely hardware independant of the remaining SIR nodes. The independant clocks used in the SIR nodes are set to operate at the same rate, but there will obviously be some skew between them. To overcome this skew problem, data transfers between nodes require each node to send both data and a clock signal to the remaining interconnected nodes in the SIR system.

A multiple clocking scheme is used in the interstage. Shift registers, controlled by the external node clock signals, are used to interface external data to the remainder of the interstage which is controlled by the host clock signal. A block diagram of the interstage is shown in Figure 2-3.

The externally controlled shift registers (indicated in Figure 2-3 by primes) are interfaced with the internally controlled shift registers (unprimed) through a windowing process based on an expected time margin. The internally controlled registers load the values contained in the externally controlled registers at the completion of a count performed by the watchdog timer (WDT). An indication of receipt of a complete data word is obtained by using modulo 32 counters on the incoming external clock signals. A bit in the slave status register is set when the the proper count is reached and the clock pulses being relayed to the primed registers are terminated. The WDT controls a bit in the slave status register in a like manner.

The rotary multiplexer in the SIR node performs in much the same manner as that proposed by Siewiorek. The rotary multiplexer proposed by Siewiorek implements a particular redundancy management algorithm in hardware as a portion of the design. The SIR architecture performs redundancy management within the host computer nodes. The complexity of the multiplexer is reduced in the SIR concept by performing the connection decision process in the host processor.

The bidirectional nature of the SIR multiplexer is a complexity factor that offsets this advantage somewhat. The design shown in the figure shows only the data communications switch; an identical circuit is necessary to handle the clock signals. The SIR concept also enables a greater flexibility in the decision process. The basic hybrid redundancy scheme imposes a set algorithm in the hardwired logic of the multiplexer. The SIR multiplexer is simply a switching network controlled by a set of flip flops forming a control register. The control register hardware is

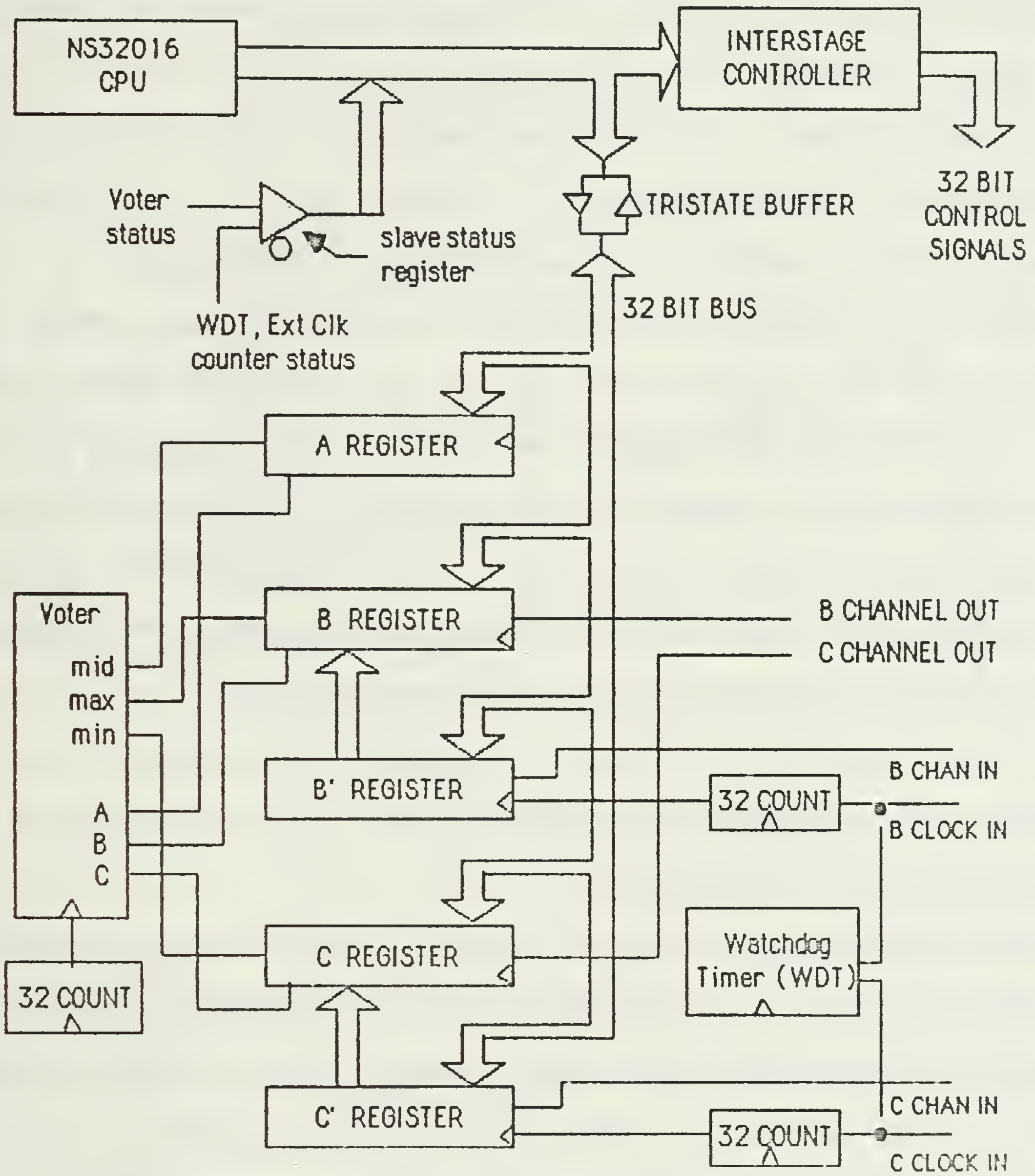


Figure 2-3 SIR Interstage and NS32016-10 CPU

independent of the algorithm that decides how it is set, so any algorithm could be used. Figure 2-4 shows the rotary multiplexer circuit for the case of a 6 node SIR architecture.

The rotary multiplexer is basically a 5 x 2 full duplex switch. The controls that determine which 2 of the 5 external processors are connected to the host interstage are loaded into the flip flops that interface with the host computer (shown in Figure 2-4 as boxes). Any 2 path combination of connections between the two rotary multiplexer interfaces are possible by loading appropiate values into the host controlled flip flop register.

Due to the hardware scheme described above, the SIR nodes do not require lock step synchronization in order for correct operation to take place. Not only can each node can have a unique clock associated with it, but the hardware also supports the design constraints imposed by N-version programming. The system can be said to be loosely coupled, with the degree of coupling being determined by the window size that the WDT imposes on the internode communication within the SIR system (which is variable by an instruction supplied by the host). In practice, the coupling will be comparatively tight due to the constraints of the application problem and the method of detecting faults.

Now that the design of the SIR node has been developed, a systematic reliability analysis of the node design must be performed. Section II A describes the MIL-HDBK-217B reliability model. Section II B partitions the circuitry into appropriate subdivisions that share similiar reliability characteristics. The subdivisions will form the system components that will be used in the system reliability model. The MIL-HNBK-217B reliability model is used to calculate the system component reliabilities.

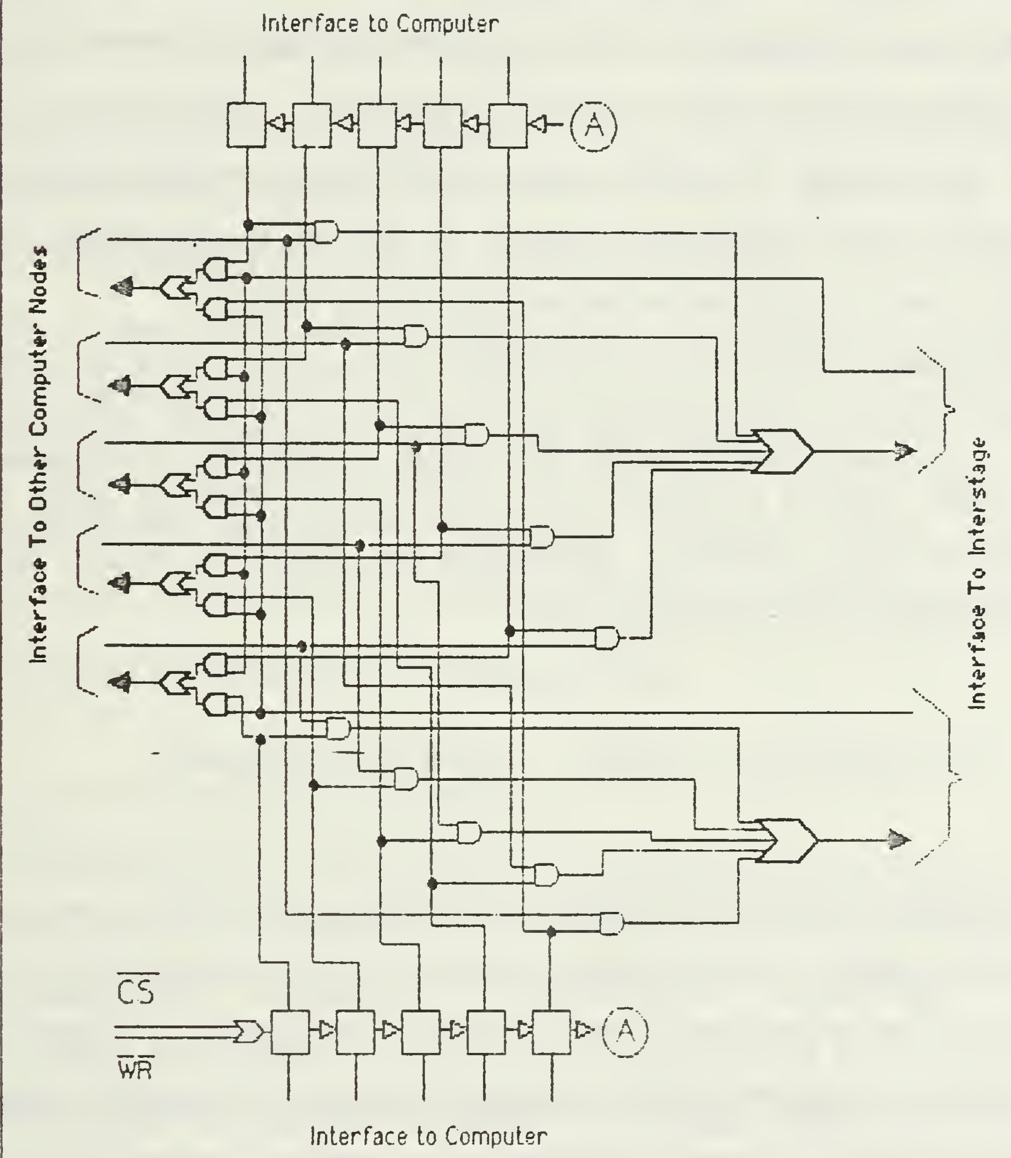


Figure 2-4 SIR Rotary Multiplexer

A. COMPONENT RELIABILITY MODEL

A component fault model for circuit boards composed primarily of integrated chips has been developed by the US Department of Defense. This model is based on exhaustive testing, performed at Rome Air Force Base, NY, on a variety of chips from diverse manufacturers. The conditions of the tests were varied to account for expected environmental conditions in which the circuit boards may operate, as well as the complexity of the circuits that are implemented on the actual chips. [Ref. 15]

This model, designated as MIL-HDBK-217B, was published in 1976 and covers several integrated circuit technologies including TTL, MOS, and ECL. The model predicts a printed circuit failure rate which is based on an exponential fault probability distribution for monolithic bipolar and MOS circuits and has a form shown below:

$$\lambda = \pi_L \pi_Q (C_1 \pi_T + C_2 \pi_E) \pi_P \qquad \text{(failures/million hours)}$$

Experience during the testing process has shown that 90% or more of the faults that occur in printed circuit boards are due to integrated chips. The effects of the printed board itself and such components as resistors and capacitors on board reliability can then be neglected in design studies and are not included in the model. Because an exponential distribution of faults is assumed, the failure rate for an entire printed circuit board is the sum of the failure rates for the chip components that are used in the circuit (a series combination of system components).

The terms in the equation for failure rate each quantify the effects of distinct environmental factors. The π_L term concerns the "learning curve" that is expected with new fabrication processes. The value of the term is set to 1 for established processes and 10 for new processes. π_Q is a function of the amount of screening the chip receives from the manufacturer prior to its release. The model gives a range of values to this variable.

π_T and π_E quantify the impact of environmental factors on the failure rate. The former is a function of temperature while the latter is a function of the mechanical stress (vibration and G forces) that can be expected in environments of interest to the military (flight being one of them).

C_1 and C_2 are factors that quantify the reliability effects of gate complexity on a given chip (or the number of bits for memories). π_P is a function of the number of pins in the chip package.

B. SIR SYSTEM COMPONENT RELIABILITIES

The determination of what portions of the overall SIR architecture are classified as distinct components plays a central role in the algorithms that will manage the redundancy of the computer architecture. The classification scheme must follow a minimal set of rules if it is to be an effective tool in the redundancy management design process as well as the development of an accurate reliability model.

The first rule is that the grouping of circuitry into components should follow functional relations. Division of a circuit into components that are below the functional level should be avoided. This is a logical approach because, for the redundancy management system to properly function, the redundancy management system must be able to recognize the occurrence of a fault with a set of tests of a reasonable size and complexity. There are a number of techniques that are available for diagnosing faults within circuitry, but it must be remembered that the SIR hardware is designed to operate in a real time control system. Tests, and decisions based on the outcome of diagnostic tests, must conform to the limited duration control cycle of the real time application. The tests are also performed by the hardware itself, which increases the complexity of many of the diagnosis techniques. The underlying goal of the architecture is to be ultra reliable; this implies that the hardware and software must be as simple as possible while providing the required functionality.

The second rule for classifying circuitry into components is that the classification scheme should not create components that cannot be effectively managed by the redundancy management system. The lack of this second rule would needlessly complicate the reliability model and perhaps lead to inaccuracies. The redundancy management algorithms would also become more complex for no purpose.

The redundancy management algorithms perform 3 main tasks. First, the redundancy management algorithms perform a set of tests on the system in order to identify any fault in the system of components. A test set must be constructed so that faults in any of the components can be detected . Once the fault is discovered, a fault location process is used that is composed of another set of tests. Once the occurrence of a fault is detected and located, the redundancy management routines must decide on a configuration for the remaining good components (that satisfies the system requirement) in such a way that the selected components interface with the overall system input and output ports and eliminate the faulty component from effecting system operation.

The design of the SIR node must be analyzed keeping the rules outlined above in mind. Recalling the functional block diagrams shown in Figures 2-2 through 2-4, there are several sections of a node that can fail. These functional failure modes are listed in Table 2-1. This grouping of circuits within the overall design is selected so that obvious functionalities will remain within a single component.

There are a variety of ways in which the host processor can fail. The design of the host processor is presumed to be of a generic form for the NS32016 microprocessor. This component category includes the microprocessor, its associated math coprocessor, the mass memory unit and associated memory chips, and the necessary glue chips necessary for binding the components into a system.

This is quite a large component category. The justification for grouping so large a set of subfunctions into one category is that the SIR architecture proposes no reliability enhancement using redundancy within this component. Some management of failures within this grouping of components is possible without using component redundancy (such as a memory chip), but these management techniques are based on software detection and correction algorithms. Of course, the correct execution of software is dependent on some portion of nonfaulty hardware, so the level of confidence of these fault management techniques is questionable.

TABLE 2-1
SIR NODE FAILURE MODES

A. Processor Failure
B. Interstage Failure
   1. Voter failure
      a. False three way equality indication (1 case)
      b. False two way equality indication (6 cases)
      c. False three way inequality indication (1 case)
   2. Timer failure
   3. Controller failure
   4. Slave Status Register Failure
C. Internet Communication Failure
   1. All links fail (Rotary Mux and/or InterStage(B'&C') failure)
   2. Selected links fail (1 - 4)
   3. Single Interstage channel fails (B' or C')

The groupings of components listed in the table within the voter/interstage section of the node design are fairly obvious. There are of course a large number of ways in which single gate level faults can occur within any of these component categories. The result of any of these faults is, however, the same; the component can no longer satisfy the functional requirements for which it is designed.

There is again no redundancy within the voter and interstage sections of the node design. Failure of any one of the components in these sections prevents correct detection of vote errors or the passing of that detection information to the connected host processor component. Recall that each of the nodes is independant and bases it's decisions about fault detection, location, and recovery on the agreement of at least two of the three connected processors. It is evident that any of the voter or interstage failures within the host processor would make this detection process either impossible or suspect. For this reason, the voter and interstage can be classified with the processor as a single component.

The rotary multiplexer consists of a set of 2 paths (full duplex) that connect the host node (processor and voter/interstage) to the external SIR nodes. There are redundant paths inherent in the multiplexer design when spares are included in the basic SIR starting configuration. (Without these spares, there is no need of a rotary multiplexer).

The purpose of advanced hybrid redundancy is to allow the replacement of faulty components with good spare components. This process is physically achieved by managing the redundancy in the rotary multiplexer. The control word that resides in the flip flop set in the multiplexer is changed so that a new path or set of two paths are selected that connect the host with its external nodes. Failure of one of the paths not currently in use is not detectable and does not degrade the confidence level of the decisions currently being made within the host node even though the ability of the node to recover from a detectable fault has been reduced. (This assumes that the failed circuitry does not effect the remaining circuitry by overloading the power supply or injecting noise into the system. The failure of a path will be assumed to be independant from the rest of the circuit, although this assumption may in fact not be true for all cases. The impact of any fault dependance should manifest itself in the an increased rate of failure for the remaining circuitry. Fault independance will be assumed in the model developed in this paper.) The TMR confidence level requirement is only that two connected nodes agree on state values. Therefore, on the occurrence of a failure of one of the selected paths, a new path can be established with another external node without an unacceptable degradation in the confidence level. This of course presumes that the time interval needed to establish the new path is small enough that the probablility of failure of the remaining good path through the multiplexer is vanishingly small.

The establishment of a good path is not solely dependent on one rotary multiplexer path. The link between two SIR nodes is terminated in rotary multiplexers at both ends of the link. The connection path between nodes therefore consists of the paths through two sets of rotary multiplexers and the physical connection between them. For the purposes of this thesis the effects of the physical link between nodes, with respect to overall link reliability, will be neglected (in line with the tenents of the MIL-HDBK-217B model).

Analysis of the circuit used in the rotary multiplexer shows that a distinct subset of component gates are used in each of the paths through the rotary multiplexer. The subset of gates varies between the input and output paths through the multiplexer, with the larger subset controlling the input leg. The subsets for both the output and input legs of a link are shown in Figures 2-5 and 2-6 respectively.

The larger set of gate components in the input leg of the path through the multiplexer is due to the 5 input OR gate. The OR gate does not isolate the output of the path (into the interstage) from stuck-at-1 faults that could arise from the other input paths. This inability to isolate the effects of faults in nonselected links causes a single path failure to propagate to one of the interstage input registers (B' or C') and effectively causes a failure in the interstage. The whole node is thus in a failed state and management of the redundancy in the rotary multiplexer is not possible.

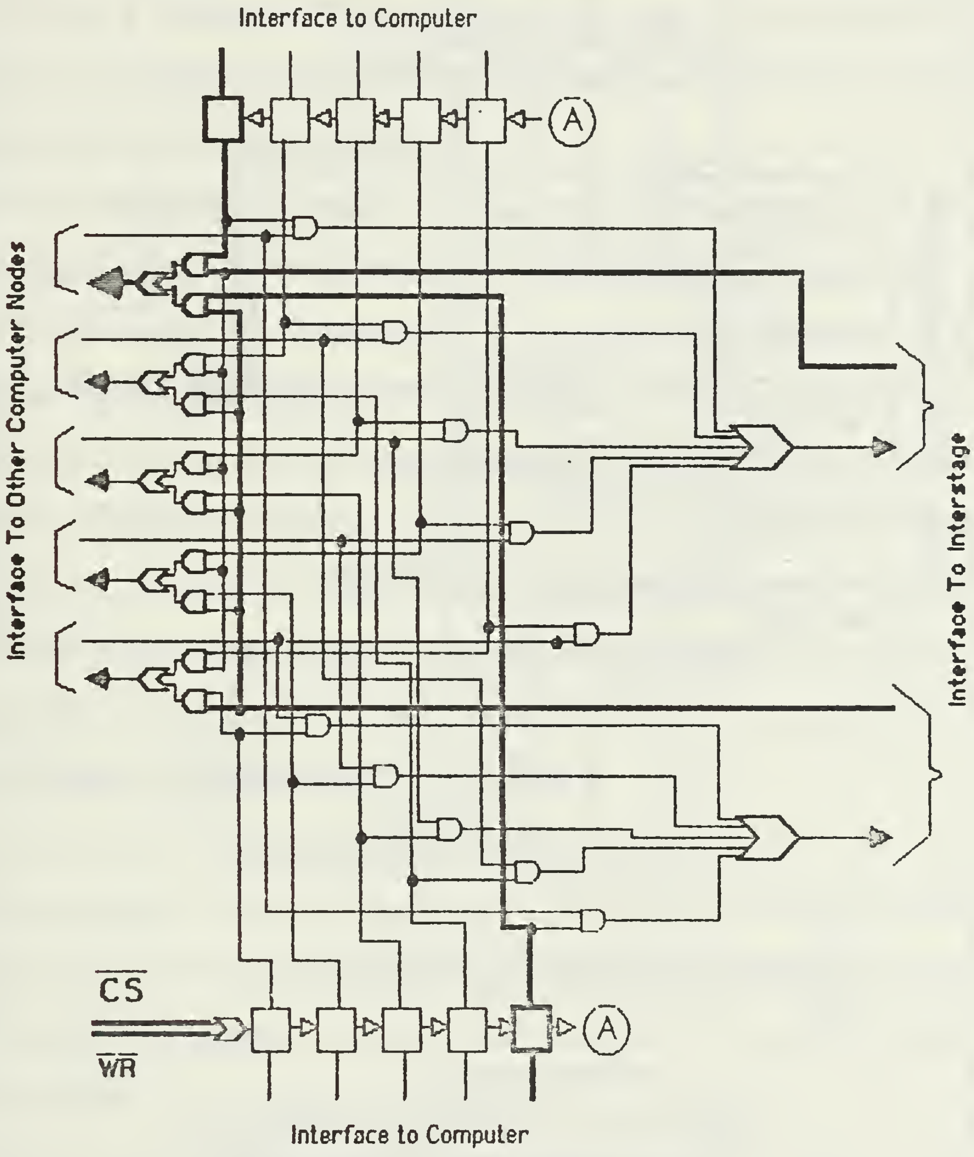


Figure 2-5 SIR Rotary Multiplexer
(Single Output Path Highlighted)

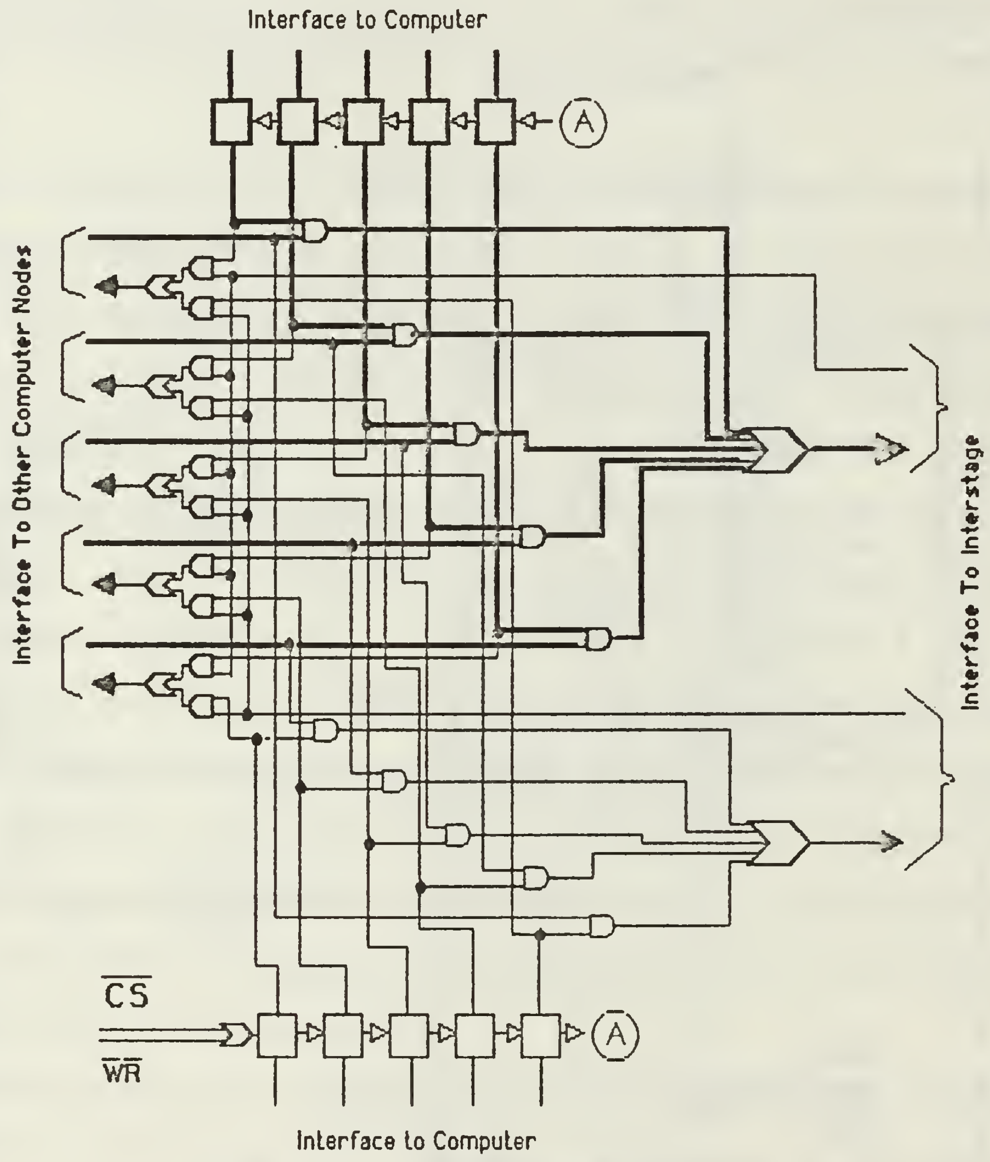


Figure 2-6 SIR Rotary Multiplexer
(Single Input Path Highlighted)

Study of Figure 2-6 reveals that the critical components needed in the isolation of failed paths are the AND gates feeding into the 5 input OR gate. In order for the isolation to take place, the outputs of the nonselected AND gates must be a logical 0. The occurrence of a logic 0 at the output of the AND gate is dependent on the correct operation of the gate as well as a correct set of input values supplied to it.

The SIR architecture is based on a system of cold spares. That is, there is a mechanism that controls the power being supplied to the nodes in the system. The means of powering up a new node and depowering a node determined as a failed node is controlled by the combination of the remaining two active nodes. A single node is unable to affect the power controlling mechanism therefore a single point of failure cannot disrupt the power system. The circuit that performs this power control has not been designed as yet so it will not be included in the probability model that is being generated in this thesis. The implications of the mechanism to the SIR architecture, however, will be included.

The result is that an unpowered spare presents a logic 0 to its rotary multiplexer outputs. This is fortunate in that the isolation of failed paths now relies only on the correct functioning of the AND gates that feed the 5 input OR gate highlighted in Figure 2-6, as well as the 2 flip flops that control the links to the remaining active nodes.

Effective management of the redundancy in the rotary multiplexer requires that isolation of bad components be possible. Since there exists a portion of the input path that cannot support this isolation, that portion must be grouped with the rest of the node for both reliability calculation and redundancy management purposes (the processor, voter, and interstage).

Therefore, the 5 input OR gate, and the 5 AND gates that feed it, will be considered a portion of the node. The AND gate that controls the loading of the flip flops must also be classified as a portion of the node. The 2 flip flops that control the inputs from the powered external nodes will be classified with the link component.

A link component will consist of the output path as shown in Figure 2-5 and an additional 2 flip flops that contribute to the input path. Of course the link is terminated on 2 ends so the component list must be doubled. Each link must also carry the clock signal for use in controlling the interstage B' and C' registers so the component count must be doubled again. All remaining rotary multiplexer gates will be grouped together with the host node for component reliability calculations.

Tables D6 through D10 of Reference 16 contain a breakout of integrated chip failure rates calculated using pessimistic values for the parameters contained in the MIL-HNBK-217B reliability model for printed circuit boards. Tables 2-2 through 2-5 use the data in the referenced tables to calculate the component and subcomponent failure rates of the SIR node. The gate information for the listed chips was extracted from Reference 17. The node's computer is assumed to consist of a NS32016 microprocessor and NS32081 floating point coprocessor along with 64K of memory. The failure rate information for the microprocessor and coprocessor was extracted from References 18 and 19.

TABLE 2-2
INTERSTAGE CONTROLLER HARDWARE REQUIREMENTS

Gates/ chip	Chips	Chip	Description
26	2	LS175	Quad D Flipflops
12	1	LS74	Dual D Flipflops
-	4	27S291	2K x 8 Prom
6	2	LS04	Hex Inverters
4	1	LS11	Quad 3-Input AND
1	1	LS30	8-Input NAND
2	1	LS20	Dual 4-Input NAND
4	1	LS02	Quad 2-Input NOR

TABLE 2-3
VOTER HARDWARE REQUIREMENTS

Gates / chip	Chips	Chip	Description
26	4	LS175	Quad D Flipflops
6	2	LS04	Hex Inverters
1	27	LS30	8-Input NAND
2	14	LS20	Dual 4-Input NAND
1	4	LS133	13-Input NAND

TABLE 2-4
LINK HARDWARE REQUIREMENTS

Gates / Chip	Chips	Chip	Description
12	8	LS74	Dual D Flipflops
2	4	LS30	2-Input AND
2	8	LS20	Dual 2-Input OR

TABLE 2-5
FAILURE RATE BY SUBCOMPONENTS
(IN FAILURES / 10^6 HOURS)

A. Node component list	**Individual failure rate**
Hardcore	
Voter	5.5933
Interstage controller	33.5863
Shift Registers (20 LS299)	17.8429
Buffers (4 LS245)	1.2469
Slave Status Register (2 LS125)	0.2917
Watch Dog Timer (4 LS163)	2.1706
Modulo 32 Counters (2 LS161)	0.9878
Rotary Mux Gates	1.3913
Computer	
NS32016	6.5041
Ns32081	6.5041
Memory	518.1119
Total	594.2309
B. Link **Total**	3.4519

III. System Environment

Before developing the redundancy manangement protocol and a system reliability model, a more complete understanding of the control problem being solved by the SIR architecture is necessary.

Recall that the design of the airframe is such that stability has been reduced to a marginal value. The airplane design allows conditions (a center of gravity aft of center of lift for pitch) that cause the airframe to cross the line between stable and instable operation. Catastrophic instability is avoided by using real time avionic controls to correct the flight pattern before the instability increases to a level that cannot be corrected.

The environment in which the aircraft is flying is of course a major factor in the rate of instability increase. NASA, Ames, has performed tests on airframe stability under a variety of conditions. The analysis showed that the X-29 aircraft, modified to the conditions described in Chapter I, would display an oscillatory instability pattern with the amplitude doubling every 100 ms for environments adverse to the designed airframe characteristics. Breakup of the airframe may occur when the avionics controls are not used to reduce the stress being applied to the aircraft by that adverse environment.

A. DSPM AND THE CONTROL PROBLEM

A 20 ms control cycle is considered by many to be the acceptable frequency of applied controls [Refs. 1,2,20,21]. The cycle consists of gathering flight state information from a set of sensors distributed about the airframe, performing a computation on the sensory data to determine the action necessary to bring the airframe within acceptable tolerances of airframe stress, and the distribution of commands to a set of effector servos that control the pattern of flight. A longer interval between receipt of effector commands is possible while still being able to recover stable flight operation but an upward bound of 200 ms is predicted for the maximum time that the F-8 can sustain a fault under critical flight conditions [Ref. 21].

The solution must meet the reliability requirements of both NASA and the FAA. The 10^{-9} per flight hour failure rate over a 10 hour mission time is quite stringent. This stringent reliability requirement makes the problem significantly more difficult and means the reliability cannot be met by an ad hoc patch work but requires a systems approach to reliability.

The avionics control cycle consists of gathering data (via sensors), performing calculations (via a computer), and exercising control operations (via effectors). A portion of the cycle that is not explicitly stated in this cycle is the communication of data to and from the computer (combining the sensors and effectors into one category). The system consists of three major components (as was shown in Figure 1-2).

The ultra reliability of each of the major components of the system is achieved through the use of redundancy. A basic description of hybrid redundancy was given in Chapter II. An extension of this process can be applied to the sensor/effector component resulting in three sources of equivalent data that are made available to the computational element. A voting scheme is used at the effectors to determine the correct signal in a manner similar to the approach proposed by Siewiorek [Refs. 1,13]. A midvalue vote process is not needed because further calculations are not necessary at the effector. The simpler, and therefore more reliable, hybrid redundancy hardware is sufficient.

Not only is the communication path a possible source of data corruption, for sensory data as well as effector commands, but the link may also be physically damaged. A simple approach for achieving the communications element of the overall control system is to provide a direct connection between the computational element and the sensor/effector nodes. This is not desirable because one component failure (the bus) could destroy the whole control system.

A bus scheme such as the one shown in Figure 3-1 uses redundancy to increase the reliability of the communications paths. While this approach is the commonly accepted method, it has some drawbacks. The concept is based on redundant components and not adaptability to possible system states. A single failure on any of the terminals on a bus can cause the entire bus to fail.

An example is the case of a babbling node. In this case the whole bus effectively fails because bus control is destroyed. It is even possible for a failure of a single remote terminal to render the entire redundant bus system useless. [Ref. 1]

The dispersed sensor processor mesh (DSPM) is a system of communications links that is designed to overcome the drawbacks of redundant bus schemes while avoiding the hardware overkill that is implied in a fully connected communication system. The system is discussed in length by Dr Abbott in Ref. 1, so a detailed description of the DSPM system will not be given in this paper. An overview of the system is necessary however, because it impacts the reliability model that will be generated for the SIR computer system.

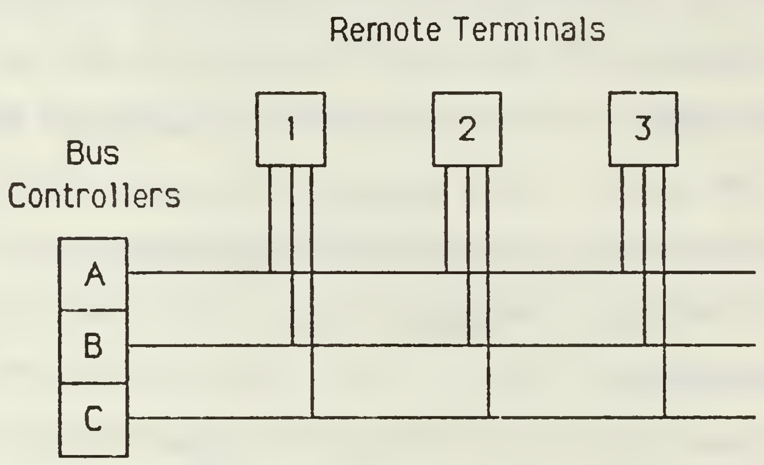


Figure 3-1 Conventional Approach to RedundantCommunications

A typical DSPM network is shown in Figure 3-2. The essence of the reliability enhancement achieved by this system lies in two major system characteristics. First, not all links in the system are in use. Active links, displayed in the figure as solid lines, carry actual information. The links displayed with dotted lines are inactive and carry no information. The active links in the DSPM network form a set of tree structures that originate at the bus controller and grow out to the furtherest link in the system of nodes. Each of the nodes in the system control one or more sensors or effectors and are distributed throughout the airframe.

The second characteristic of the DSPM network is that network control is centrally located in the bus controller. The bus controller manages the network of links by three main algorithms; the growth, repair, and modify algorithms.

The growth algorithm is a network initiation algorithm that determines which links are used to form the trees shown in Figure 3-2. The growth algorithm is a breadth first growth process. Note that each node of the system has a number of links (full duplex). The nodes are directed by the bus controller to identify one of its ports as an inbound port through which the node will receive bus controller (BC) commands and through which the node will relay the BC commands through the other ports to the rest of the tree. Each tree in the network has a different bus controller port as its root. The direction of data flow and the state of each link in the network is determined by the growth algorithm resident in the bus controller. A necessary system property is that there are no closed loops in the network. This is important to the algorithms that control the DSPM network. (See Ref. 1 for details of why this is so.)

When a fault is detected by the system, the repair algorithm circumvents the failed link or node by activating an inactive one to reconfigure around the fault. There can be a very large number of system configurations that generate an acceptable network structure given the occurrence of a fault in the system. If the repair algorithm encounters a second fault during the repair process, the links in the nodes are reset and the growth algorithm is used again. This retreat to the growth algorithm greatly decreases the complexity and processing requirements of the repair algorithm and is acceptable so long as the growth algorithm is of sufficient speed that the control cycle can be completed within the bounds of safe operation. (The growth algorithm must also have the means to accommodate failed link/node states in the growth process, which it does.)

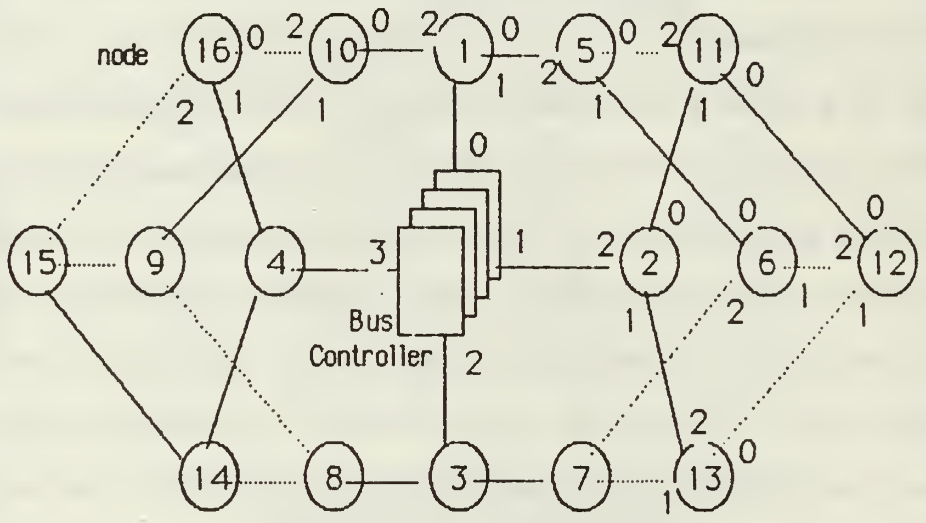


Figure 3-2 Dispersed Sensor Processing Mesh Approach

The DSPM modify algorithm is a method for discovering failures in inactive links before they can become a critical factor in a repair or growth process. Faults that occur in inactive links are not observable. The modify algorithm makes these latent faults observable by periodically exchanging the inactive links with their active counterparts while retaining the requisite connections between the affected node sensor/effector components. The algorithm is designed to be distributed over many control cycles. The process is continued at the end of a number of control cycles until all of the inactive links have been activated for some portion of the current modify cycle. At the end of the modify cycle the process is begun again. Detection of a "discovered" link fault will of course generate a repair task.

The DSPM relies on the bus controller to make all network configuration decisions. Network configuration management requires a significant computational element and as such may be combined with the computational element required for the overall control system. Of course this requires that the SIR hardware and operating system software support the additonal task of controlling a complex communication network. The entire task cycle in the SIR computer must run within a 20 ms control cycle under fault free conditions. The SIR system must also be able to correctly respond to both internal as well as external faults within the upper bound of the control cycle.

B. SYSTEM TASK STRUCTURE

The operating system that is to manage the SIR task cycle can be implemented as an event driven real time operating system. The set of tasks that must be scheduled is shown in Figure 3-3. These tasks are scheduled as events by the operating system using a system of priorities to determine the next event that is to be executed. There are other tasks that the operating system must also schedule such as memory management and I/O port control functions. These are not included in the set of tasks shown in Figure 3-3 because they are standard tasks in general purpose operating systems and have no impact on the reliability model that will be generated for the SIR computer system.

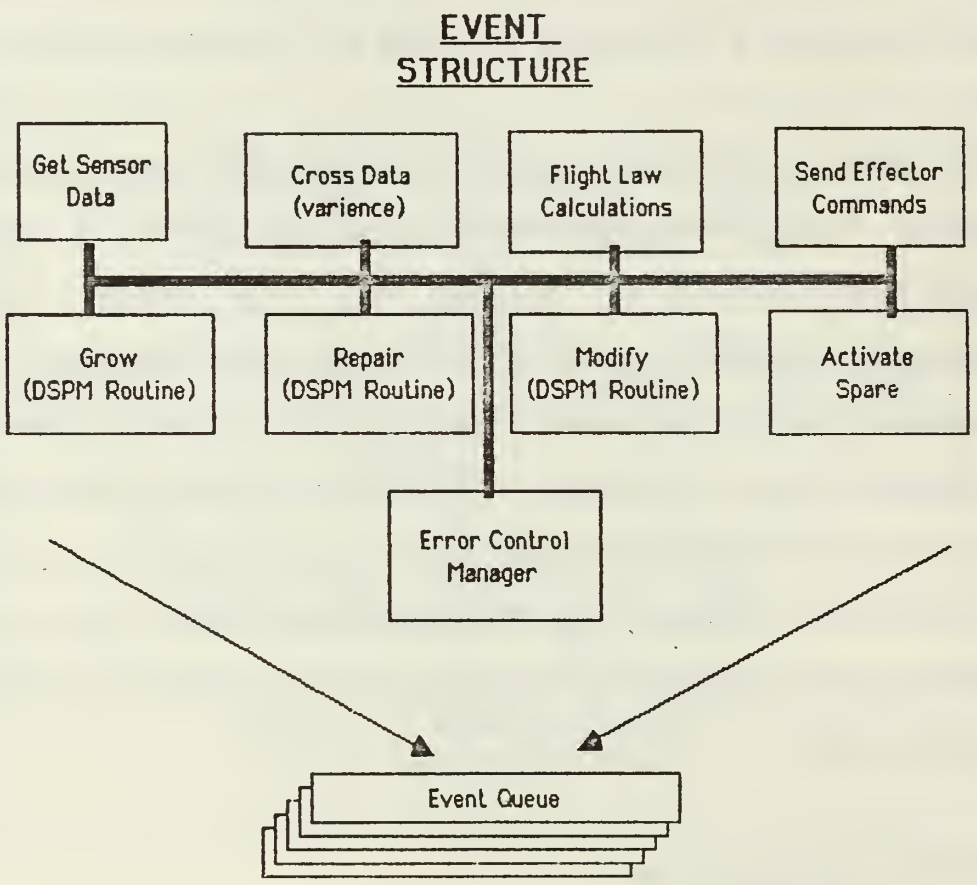


Figure 3-3 Operating System Task Structure

A base line scheduling structure is used to manage the tasks when the system is in an error free state. Because the SIR system is designed with a high reliability as a goal, it is expected that the majority of the control cycles will fall into this task execution pattern. Figure 3-4 graphically displays the error free task execution flow as a continuous bar of tasks that conforms to the control problem to be executed by the SIR computer system.

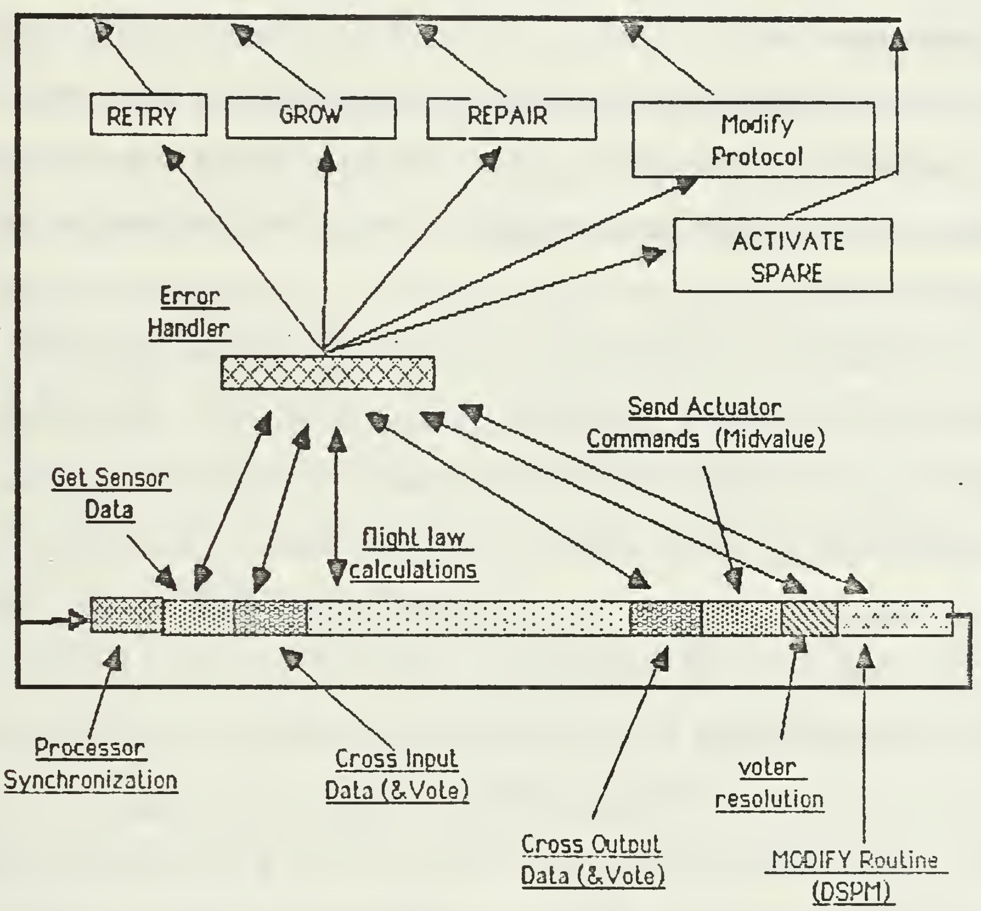


Figure 3-4 System Task Flow

There are of course errors that can be encountered due to hardware and software faults as discussed in Chapter II. An error handling module is shown that controls the response to encountered errors. (The algorithms for error detection and error state maintenence will be discussed in Chapter V.)

Upon encountering an error in the execution of any of the main loop tasks, an error handling task is generated. The error handler is of a higher priority than the main execution loop so control is passed to this module after saving the necessary register set and temporary variables in the currently executing module.

The error handler locates the error and generates a task or set of tasks to respond to the error appropriately. The priorities of the error correction tasks are adjusted for correct execution order. The error handler sets these priorities based on the severity of the error. The execution control may in fact be returned to the main loop routine that detected the error, with the error correcting tasks scheduled for execution after completion of the control cycle. In this case, the Modify task would not be implemented in the current control cycle, but delayed until completion of the next control cycle.

As stated in Chapter II, the operating systems of the active SIR computer nodes are completely independant. All of the software written for the SIR computer must also be independant, or protocols must be designed to distribute system information correctly among the processors. Each of the SIR processor nodes contains one of the BC ports into the DSPM network (as described in Chapter II). Because the DSPM management algorithms depend on the absense of any closed loops in the subtree structures (within a tree or among the bus controller and any combination of trees), there is definite dependance among the software that resides in each of the SIR procesor nodes.

The system information needed to manage the DSPM network consists of a set of tables that describe the DSPM network state, which links are active, inactive, or failed, and information on the tree structure that forms the communications paths from the bus controller to the sensor/effector nodes of the DSPM network. Each of the active SIR processors has independant control over one of the trees, but state information must be global among the processors if the DSPM algorithms are to function properly.

It is fairly obvious that there must be a high degree of confidence in the values of the state information that is passed between the SIR processor nodes. The method in which this information is passed must provide a degree of verification or any reliability models that describe the SIR system will be incomplete and inaccurate.

The programs that will implement the DSPM algorithms will be coded as if the SIR system is a single processor computer. A mechanism that can be used to control the actual three processor environment is a set of traps embedded in the operating system. These traps treat the section of memory that contains the DSPM states tables as a special memory category. When an update operation is performed on data within this section of memory, the trap routine communicates the update to the other active processors in the SIR processor network. System state table congruency between the processors is assured by this trap system.

IV. SYSTEM RELIABILITY

The model for the complete sensor/DSPM/SIR system is quite complex and is best approached as a series of models, one for each major subsystem. The reliability of the DSPM system has been estimated as 10^{-12} system failures per flight hour during a 10 hour flight [Ref. 1]. Achieving a similiar reliability figure is a goal of the SIR architecture.

There are several levels that can be viewed in developing a reliability model for the SIR system. The major levels are the component reliability model and a systems reliability model for calculating the effect of the components taken as a system. The reliability model for the components in the system has been developed in Chapter II and a set of component reliabilities have been generated from this component level model.

The SIR processor network is based on hybrid redundancy, which itself is based on a TMR operating environment. The purpose of the TMR model and its supporting architecture is to tolerate a number of faults while still providing results that are judged to be correct within a defined range of confidence. The TMR model specifically judges that a minimum acceptable confidence level can only be obtained when at least two communicating components (processors) agree on the result of a test applied to system data values. The confidence level of correct system operation increases as more components can be used in the verification process. Ideally, the configuration control software will modify the system configuration to one that provides the optimum confidence in system output given the occurrence of one or more specific faults. Figure 4-1 graphically shows the basic TMR system configuration from a logical component level. The components of the system that are subject to faults (and fault management) are the individual processors and the system links connecting the processors.

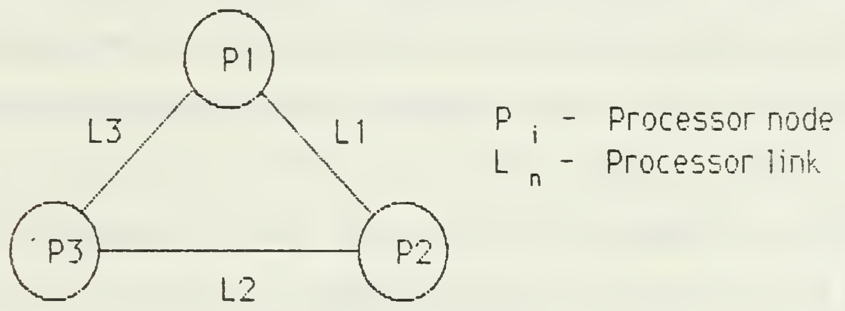


Figure 4-1 Basic TMR System

Modeling the reliability of the basic TMR system requires the definition of all possible system states where the minimum configuration exists. The model also requires a specification of the set of operating procedures that are necessary to manage all of the possible "good" states. These operating procedures must include a test set for each good system state such that additional faults can be detected and the model can progress to another good state. Each good system state can require a unique operational procedure and test set since the combination of good components in the system will vary with fault occurrences.

Effort has been expended to determine how the reliability of the system increases as spares are added to the basic TMR system [Refs. 5,22,23]. In order to achieve greater system reliability the spares are assumed to be in an unpowered state. A model for predicting the reliability of unpowered spares is very difficult to develop. The difficulty arises due to being unable to test the component in an unpowered state. The process of powering up the component most likely introduces more stress on the component than the entire time the component is in the unpowered state. As a result, periodically activating a major component like a computer node for fault testing is probably not reasonable. This thesis will not concern itself with the subtleties of modeling the unpowered state. An assumption will be made that the component reliabilities of the unpowered components is a single order of magnitude less than that of the components in the powered state.

The goal of the system as defined above is two fold. First, and most important, the goal is the operation of the system in the presence of a number of faults. A second goal is operation in the configuration giving the most confidence in the results being generated by the system.

What are the implications when spares are introduced to the basic TMR system? The standard system operation remains the same: three active processors compare outputs to determine correct operation. The difference in reliability is that the number of system links grows in a nonlinear fashion as spares are added to the system ($n(n-1)/2$ where n is the number of nodes in the system).

Nonlinear growth in the number of system links implies that the growth of system states is also non linear. Each of the system components can be modeled as being in one of two states - functional or nonfunctional. This is a simplification since there is also the possibility of improper component functioning, but that will be ignored for the present. The number of states possible given the starting configurations, is then 2^n, where n is the number of system components. An indication of the rate of growth in system states is evident in Table 4-1. A systematic method is needed to identify the set of states necessary for a valid reliability model and determination of the necessary operating protocols and test sets.

TABLE 4-1
STATE PROGRESSION
AS SPARES ARE ADDED TO BASIC TMR

Configuration	Processors	Links	States
Basic TMR	3	3	2^6 = 64
TMR + 1 Spare	4	6	2^{10} = 1024
TMR + 2 Spares	5	10	2^{15} = 32,768
TMR + 3 Spares	6	15	2^{21} = 2,097,152

System operation always occurs with an active set of processors of two or three. This property of sytem operation allows the development of an incremental system reliability model for the SIR processor network. The basic model covers the three processor (basic TMR) case. Each of the incremented models, corresponding to the addition of spares to the system, will degenerate to the basic TMR case after some combination of component faults in the overall system.

Since there is no single control point in the system being modeled, the total number of possible system states is not needed to either calculate the number of distinct operating protocols required or to calculate the reliability of the system. Clearly, some aggregation of states into like configurations is possible (differing only by label changes of the nodes and appropiate connecting links).

A. THE SEMI-MARKOV MODEL

The Markov process model is a powerful tool for analyzing complex probabilistic systems. The central concepts of such models are states and state transitions. The states of a system have already been defined, but it should be pointed out that each state of the system represents all that must be known to describe the system at any instant. A second key concept in this model is the state transition. As time passes and faults are introduced, the system passes from state to state. These changes of state are called state transitions. Discrete-time models require all of the transitions to occur at fixed intervals and assign probabilities to each possible transition. For reliability models, the transitions represent failure occurrences and configuration functions (or repair functions for other than real time applications). [Ref. 16]

The basic assumption of Markov models is that the probability of a given state transition depends only on the current state. The length of time spent in a state does not influence the probability distribution of the next state or the distribution of time remaining in the present state. This assumption is rather strong but it fits naturally with the assumption that failure rates are constant. The constant failure rate assumption applies to the operational phase of component operation and results in an exponential distribution of arrival times of failures. The Weibull distribution, based on non-constant failure rates, apply to the burn in and wear out phases of component operation. The model developed in this paper will apply to the operational phase.

The reliability model that will be developed for the SIR computer system does not require the entire set of system states. There are several cases where the system can catastrophically failed before the system states are exhausted by the arrival of faults. These failure conditions arise due to the confidence requirements of TMR systems. For example, failure of two of the active processor nodes without a correcting reconfiguration is a catastrophic condition, even if there are several spares in the system that are in a non-failed state. System failure states are referred to as death states; no transitions from death states are possible.

The probability of entering a death state is precisely what is needed to determine the reliability of the system. The calculation of the probability of entering the death state of a semi-Markov model requires the solution of a set of coupled differential equations. The large disparity between rates of fault arrivals and the rate of recovery (based on reconfiguration) usually leads to numerically stiff differential equations. This problem along with the high computational cost of solving large state space problems has led to the use of tools such as CARE III and HARP, and ARIES [Refs. 10,16]

A tool that was recently developed at NASA, Langley is the Semi-Markov Unreliability Range Evaluator (SURE) [Ref. 10]. The program is based on a mathematical theorem developed by White [Ref. 11] that enables efficient computation of the death state probabilities. The technique provides a means of bounding the probability of entering a death state of a semi-Markov model using simple model parameters such as the means and variances of the state transitions. The advantage of the SURE technique is that the bounds are algebraic in form and thus computationally efficient. Because of this computational simplicity, very large models can be analyzed by the program.

The modeling of highly reliable fault tolerant systems generally exhibits both slow and fast processes with respect to mission time. When these systems are modeled stochastically, some state transitions are many orders of magnitude faster than others. The slower transitions correspond to the arrival of faults in the system while the faster transitions represent the system response to the fault. Fault arrivals are modeled as exponentially distributed and are therefore time invarient with respect to the length of time that the system resides in a specific state. System recovery transitions are generally not exponentially distributed and therefore the rates are time dependent. In order to preserve the semi-Markov nature of the system, the time since entering the current state is used to calculate the system recovery time probability. Because the TMR system uses three way voting to mask a fault, there is a race between system recovery and the occurrence of another fault.

The state and state transition basis of the semi-Markov model is represented very nicely by a directed graph. An example of a graph representation of a Markov process is shown in Figure 4-2. The states are represented by the labeled nodes in the figure while the state transitions are represented by the directed arcs.

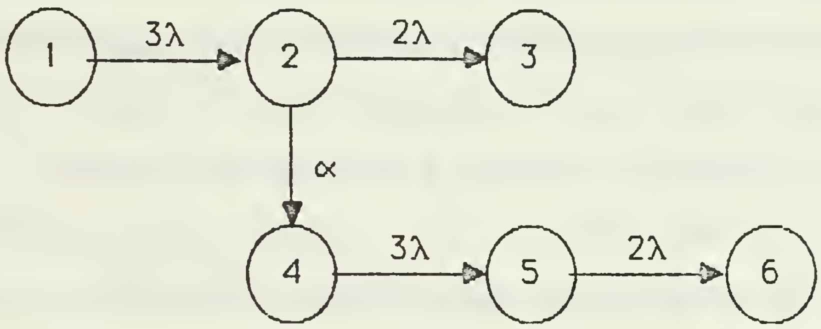


Figure 4-2 Graphical Representation of a Semi-Markov Model

The horizontal arcs correspond to the slow transitions of fault arrivals. These occur with exponential rate λ, with the coefficients of λ representing the number of components that can fail. Vertical arcs represent fast transitions that correspond to system recovery.

White's theorem is based on a graphical analysis of a semi-Markov model. The theorem calculates the bounds on the probability of traversing a specific path within a specific time. Applying the theorem to all of the possible paths of the model results in determination of the probability of the system reaching any death states bounded by a narrow interval. [Ref. 24]

The SURE program will be used as a tool for determining the reliability of the SIR system model. The model developed in this thesis will conform to the graphical representation conventions that are described for Figure 4-2. See References 10,11, and 24 for a more detailed discussion of White's theorem.

B. THE SINGLE FAULT ARRIVAL ASSUMPTION

The SIR system of processors performs a self diagnosis to detect and locate faults within the active processor set. The reliability model of such a system can assume a variety of fault arrival conditions. The level of confidence in fault detection and correction in a discrete system is dependent on the number of faults that can arrive during one interval. The confidence level will decrease as the number of simultaneous component faults increases.

The reasoning behind this property is straight forward. The detection of a fault in the system becomes increasingly more difficult and the algorithms to perform this detection becomes more complex. An example illustrates the point nicely. Suppose that a host processor (Processor A) in the basic TMR system receives a vote after a data exchange which indicates an unacceptable inequality condition on one of the external active SIR processor nodes (Processor B). An acceptable conclusion could be that the link connecting nodes A and B was corrupted by noise, or that processor node B is faulty. But suppose that there is also the possibility that the Processor A's voter could have malfunctioned. This poses a problem to the operating protocol in Processor A. If the processor assumes that the fault was due to the voter, it will take itself out of service. If there is the possibility that the voter is malfunctioning, then any tests received by processor A by way of its voter is also suspect. If the fault actually occurred in Processor B then two of the three processors are effectively in a faulty state, and the system collapses.

The best case for system confidence is obviously for there to be only one fault occurrence during any one test cycle. Reliability based on the arrival of faults for single components is modeled as a exponentially distributed probability function and has the form shown in equation 4-1.

$$R = e^{-\lambda t} \quad , \qquad (4\text{-}1)$$

where

R is the reliability
λ is the component failure rate
t is the time interval since the last known good state was observed.

The reliability of the TMR system is a multiplicative combination of the reliabilities of the TMR components. An equation for reliability for a fully operational TMR system is given in equation 4-2,

$$R_S = (R_P)^3 \times (R_L)^3, \qquad (4\text{-}2)$$

where

R_S is the system reliability,
R_P is the reliability of a single processor node
R_L is the reliability of a single processor link

Because the processors and links used in the TMR system are identical, there is no reason for a unique labeling system, hense the terms in equation 4-2. The equation is a statement of the probability of there being exactly zero component faults in the TMR system during a specified interval. This is one of a set of probabilities that cover all of the possible component fault states that the TMR system can be in during that interval. The occurrence of multiple faults within a single component is not germaine to the system reliability equation because a single fault is assumed to cause the incorrect operation of the component.

Suppose that a hypothesis is made that states that there can only be a single component fault during the test interval. The reliability figure desired for the system is 10^{-12} system failures per hour during a 10 hour flight. The probability of more than one component fault occurring during a test cycle should be at least an order of magnitude greater than the desired system reliability over the mission time. If it can be proven that this is the case, then the assumption that only one component fault can occur within one test cycle is valid. The operating protocols, given a single fault arrival assumption, will then be a great deal less complex as will the reliability model describing the SIR system.

The probability that there will be exactly one faulty component within the test cycle is the summation of the probabilities that a unique component will fail and the remaining components will not fail. In a system consisting of six components this requires six probability terms. An equation for the case of exactly one faulty component during a test interval is shown in equation 4-3.

$$P_S(1) = 3(1 - R_P)R_P^2R_L^3 + 3(1 - R_L)R_P^3R_L^2 , \qquad (4\text{-}3)$$

where

$P_S(i)$ is the probability that i components will fail in time t

$(1 - R_K)$ is the probability that component k has 1 or more faults

The probability that two components will fail in the test interval is calculated in a similiar manner. The number of terms in the equation will be equal to the number of unique cases of components taken two at a time from the set of system components. Equation 4-4 shows this relation.

$$P_S(2) = 3(1 - R_P)^2 R_P R_L{}^3 + 3(1 - R_L)^2 R_P{}^3 R_L + 9(1 - R_P)(1 - R_L) R_P{}^2 R_L{}^2 \qquad (4\text{-}4)$$

Substituting the values for component reliabilities developed in Chapter II into the equations for probability of faulty components during a test interval leads to the probabilities shown in Table 4-2. A test for faults is assumed to take place during every control cycle as depicted in Figure 3-4. The test interval used in the fault probability equations is the maximum time interval allowable for a control cycle for the F-8 digital fly-by-wire system.

TABLE 4-2
PROBABILITY FOR MULTIPLE SIR COMPONENT FAILURES WITHIN MAXIMUM BOUNDS OF A CONTROL CYCLE

Component failures within t	probability
>0	8.97000×10^{-8}
1	8.96999×10^{-8}
2	2.69985×10^{-15}

It is obvious that the probability of more than one component becoming faulty during a single test interval is insignificant when compared to the overall reliability requirement for the system. The protocols and system reliability models that are developed in this paper assume that the fault arrivals are singular during a test interval.

V. THREE PROCESSOR CASE

This chapter will discuss the three processor case of the SIR architecture. Expansions to the three processor case are possible using additional computer nodes as spares, however all of these cases will effectively degenerate to some form of the three processor case by the introduction of enough system component failures.

The operation of the SIR system is based on the TMR principle. System spares are in a cold, nonpowered state and can only be activated by actions taken by the other two active SIR processors in concert. These two system characteristics cause the three processor case to be important to larger systems with spares, even before the introduction of component failures. The SIR architecture uses an active processor set consisting of three processors; all the tests for fault detection and location and the decisions based on the outcomes of these tests are made by the active three processor set. The internode communication links do not require a rotary multiplexer for the three processor case. The multiplexer hardware is, however, needed for all cases that contain spares, so the links (using the standard link hardware discussed in Chapter II) will be included. The results will apply to all cases regardless of the number of spares.

There are several issues that apply to the three processor case of the SIR architecture. Each of these issues will be discussed in a separate section. Section A will discuss the operation of data congruency in the three (good) processor case. Section B will develop a set of system states that meets the TMR requirement for two connected, good processors. This set of system states will consist of acceptable "views" of a partially failed system that can still be managed. Section C will develop a high level communications protocol that is necessary for the data congruency operations discussed in section A. The protocol will also address the fault location process. Finally, section D will develop a semi-Markov model of the three processor case based on aggregations of acceptable system states discussed in section B.

A. DATA CONGRUENCY AND FAULT DETECTION

The SIR architecture was developed to meet a real time control problem that has a specific cycle of events to process. The event cycle includes a series of data words that must be exchanged reliably between the active processors. These data words can be exchanged in two basic modes. The first mode requires an exact match of the value that each of the three active processors generate. This type of word will consists of system state information as discussed in Chapter III (states of the DSPM system). A seperate case where bit invarient data exchange is necessary, is in the command words of the inter-SIR communications protocol discussed in section D of this chapter. The test for equality of the triad of data words is made at each of the active processors by their respective voter elements. The equality condition is observable to the host processor by viewing the contents of the slave status register.

A second category of data word communication is also required. The sources of these words are the sensor inputs to the SIR system and the output values generated by the flight law calculations in the separate active processors (that are subsequently communicated by the DSPM to the effector servos in the aircraft). There is a large probability that data of this type will slightly vary in the low order bits as discussed in Chapter II. There are bounds on the amount of variation acceptable in the data however. Observability of the size of the variation between the minimum and maximum values of the data triad is achieved by calculation in the node's host processor. The preset bounds, with respect to the particular word being voted, is applied to this calculated difference to determine whether any of the values deviate unacceptably.

Both of these data transfer types will be used in each of the control cycles that were discussed in Chapter III. The cross communication and vote of these data words provide a comprehensive test of the active components in the SIR system. The process of communication itself is a test of the links that connect the active processors; each active processor's internal components are thoroughly exercised during the execution of the flight law modules.

Although exercising the functionality of a component is not in itself a test for a fault, the verification process inherent in the voting process is indeed a test for a component fault. So long as two of the three processors agree on the outcome of a particular vote process, then the agreed upon value can be taken as valid with the confidence level of the overall system remaining at an acceptable level. (Of course a three way agreement generates a higher level of confidence.)

When an unacceptable comparison of the active processor generated data values occurs, of either the varient or invarient type, an indication of a component fault condition is established. The fault can be caused by quite a number of faults internal to the offending node, for example the node's voter, floating point processor, or memory. The fault could also be caused by the link connecting a pair of the active processors.

The fault that has been detected by the unacceptable data comparison can be caused by either a permanent fault or a transient fault. Transient faults have many causes such as excessive electrical noise in the aircraft environment and marginal operation of some set of component circuits. The effects of transient faults have a low probability of occurring over an extended time so repeated the tests over a number of control cycles can eliminate the possibility (with high probability) that the error is a transient one. [Refs. 1,13]

There is an external system characteristic that can cause a fault detectable by data disagreement. In the ideal case the data supplied by the external sensors and transmitted over the DSPM system will arrive as a data triad with one data word being supplied to each node. An unacceptable variation in the values of these data words could be caused by faults in the sensor set or the DSPM system. In either case, the SIR system would generate a data disagreement detected fault. A preset test for correct operation of the SIR system is required to isolate the fault within the SIR system or within the external system when sensor data is being cross linked and voted. If the fault is isolated to the external system (consisting of the sensor set or the DSPM system) then this information is passed to

DSPM redundancy management routines. (These routines will not be considered in this paper.)

1. Simplex Data Transfer

Simplex data is propagated from one SIR node to the remaining two active nodes by an algorithm designed to insure congruency in the data between processors. Of course the transfer of simplex data must be made in an invarient manner because only one copy of the data is made available to the SIR system. Figure 5-1 displays a graphical representation of the simplex data distribution algorithm. Data supplied to node A (either by external input or a DSPM state table update as a result of a DSPM algorithm) is loaded into each of its interstage registers. In the second phase of the transfer, the interstage registers (B and C shown in Figure 2-3) transfer their contents to the associated interstages of the remaining two active nodes. The value in the home register (A in Figure 2-3) remains the same.

In the third stage each of the interstages loads the received data into its register set complement. At this point all the interstage registers contain the original data value. In the forth stage, the contents of the interstages are again cross linked. At this point, each of the interstages have a copy of the data received by each of the remaining interstages. The interstages vote the data at this point. If the votes indicate a three way equality (observable by there being no min or max indication in the slave status register) then the data transfer is correct. A maximum or minimum indication by the slave status register indicates an error condition in the system that must be located.

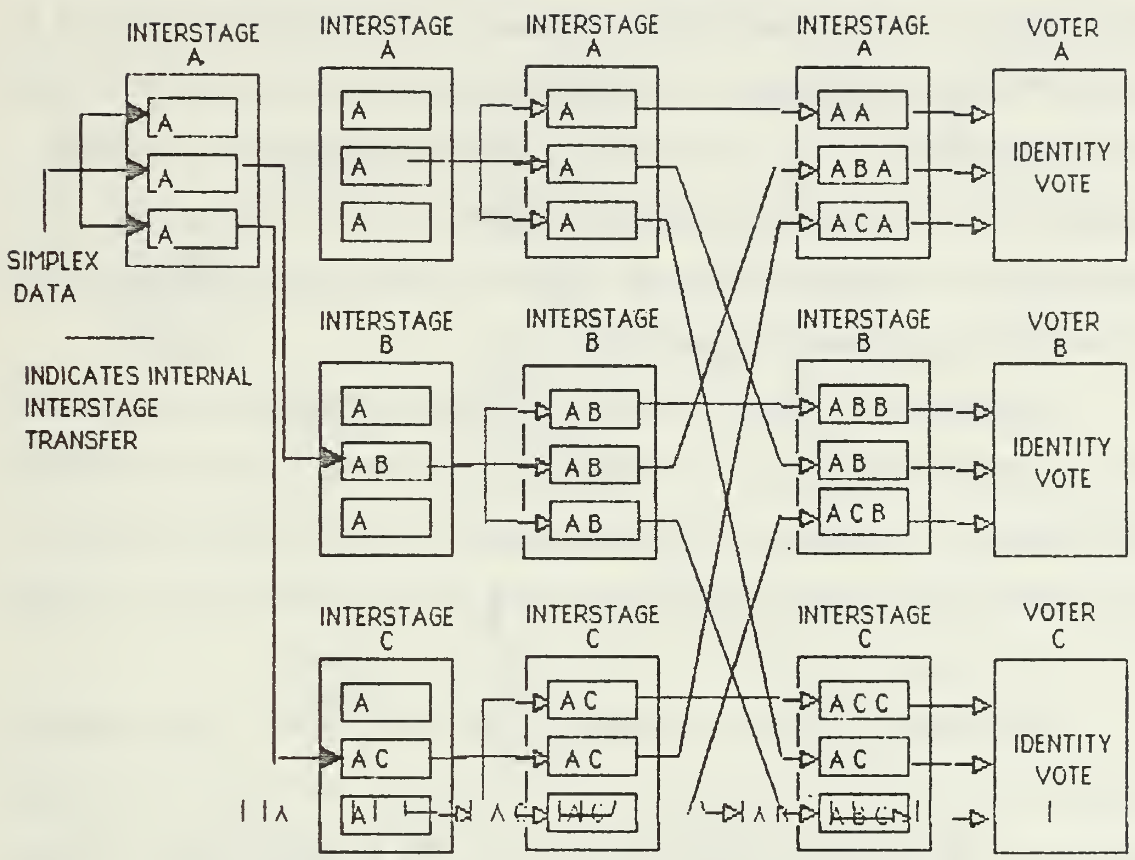


Figure 5-1 Interprocessor Communication and Voting Through a SIR Interstage (Simplex)

2. <u>Three Value Data Transfer</u>

The procedure for exchanging and comparing the data generated by the external sensors (assuming that one sensor value of a data triad is supplied to each of the active node's external ports) follows a similiar approach. As shown in Figure 5-2, the exchange can be made in a more concurrent manner in this case. The nodes are assumed to be in loose synchronization which is justified by the relatively short duration of the control cycle being executed in the SIR system and the synchronization task that is the precursor to each control cycle execution (as shown in Figure 3-4).

The data to be voted is supplied to each of the active node's interstages by either SIR external sensors or generated internally by the flight law calculation task. An additional source of data to be voted is the command words of the inter-SIR communication protocol discussed in section C. The data can be considered to be a triad of simplex data at the beginning of the exchange process. Each of the data values that are to be exchanged are independently generated.

The process of exchange is begins by each of the nodes transfering their simplex data values to registers A, B, and C of their respective interstages. The process of loading a starting value in the interstages is concurrently performed as contrasted with the procedure shown in Figure 5-1.

After one exchange cycle among the interstages, each interstage contains the complete triad of simplex data values. A vote is then made on the data triad by each of the interstages. The procedure after the vote depends on the type of data transfer that is being made. If the data transfer is of the invarient type, ie the values must exactly match, then the slave status register contains all of the information necessary to test for three way equality or, conversely, the indication of an error condition within the active SIR component set. The value in register A after the vote can then be transfered to the node's host processor and execution can continue. (Register A will contain the mid value of the data triad as determined by the voter.)

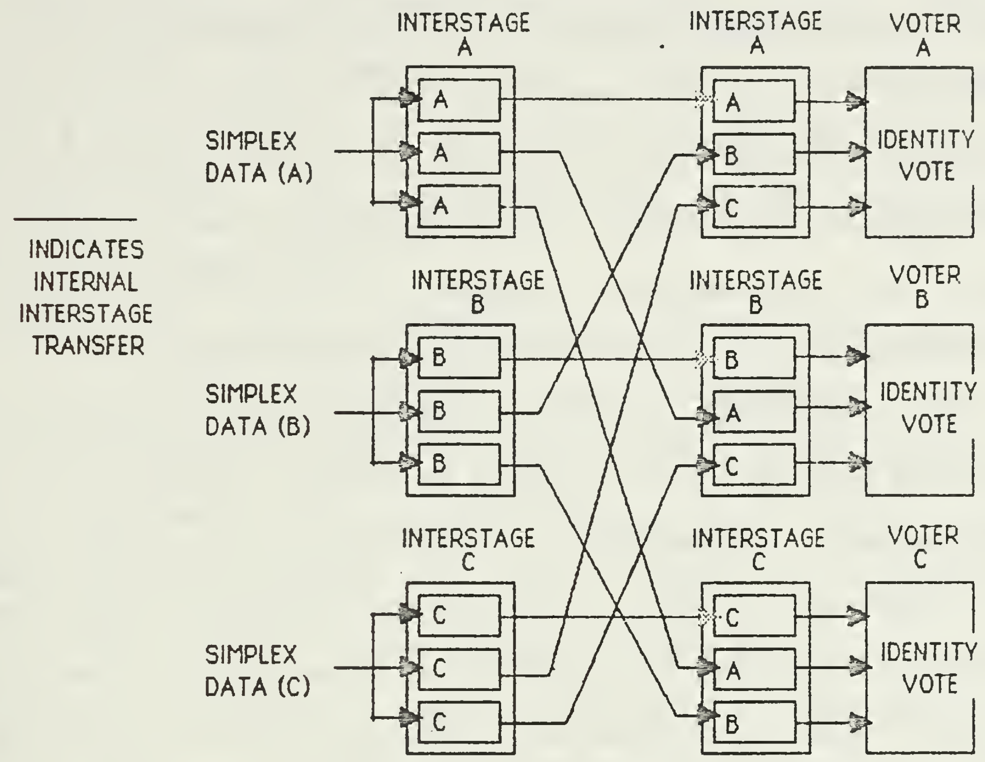


Figure 5-2 Interprocessor Communication and Voting Through a SIR Interstage (Three Valued)

If the data to be exchanged and voted upon is of the varient type then the contents of the A, B, and C registers must be transfered to the node's host processor. At the completion of the vote process register A contains the mid value while registers B and C contain the maximum and minimum values respectively. The host calculates the difference between the minimum and maximum values and compares the difference with the maximum bound for the data word that is being cross linked and voted. If the difference exceeds the maximum bounds for the data word, then the error condition indicates either a failed component within the active set of SIR components or a failed component in the sensor/DSPM system. A preset test must be performed if the data triad originated externally to the SIR system. An out of bounds condition on internally generated data indicates a SIR component failure. The cause of an internally generated failure can be attributed to either hardware or to one of the N-version programs that carry out the flight law calculations. Fault location algorithms must then be used to locate the faulty component and remove it from the set of components that interface with the systems external to the SIR architecture.

The midvalue that is contained in register A at the completion of the vote will be taken as the correct value for further program execution. The differences between the midvalue and the minimum and maximum values is used to locate which of the two values caused the variation bound violation. The value that is closest to the midvalue is the value that is assumed to be fault free. The value associated with the larger difference is assumed to be the value caused by a component (or software) fault. Again, the fault could have been caused by the node, link, software, or an externally applied SIR input.

3. Two Value Data Transfer

There is the possibility that only two values are made available to the SIR system. The data must be cross linked and voted in this case also. The condition can be caused in two separate ways. The first cause is a characteristic of the external system that is used to supply the SIR architecture with input data. A sensor failure in the external sensor set that supplies the SIR input data would cause only two values to be presented to the active processor set. A DSPM system link failure could also isolate a sensor and prevent communication of its data to the SIR system.

Internal SIR fault conditions cause the second category where two data values must be cross linked and voted. The removal of a SIR processor by either a processor failure, or a combination of SIR link failures that isolate a good processor, sets up a condition where the active processor set consists of only two processors.

In the first case, an assumption is made that there is a full complement of active SIR processors and the appropiate links. The discovery of the external fault which causes the two valued data must be globally distributed to the active SIR processor set. With this global knowledge, the SIR processors are aware of which two processors will receive the data. The data is cross linked in the same way as in the simplex data case for each of the data values that is recieved by the SIR system. At this point, each of the three active processors contains both of the externally supplied values.

With only two values available, the concept of a midvalue vote is not appropiate. The values are instead transfered to the host computers in each of the nodes. The difference is taken between the two values and the variation bound test is applied to this difference. If the difference between the two values is within the variation bounds, then an average is taken in each processor. These averages are then cross linked among the processors by the three value communication algorithm. The averages that are cross linked should be an exact match, because the node components are identical from both a hardware and software view. (The software for system control is not performed by use of N-version programming techniques.) If the two data values generate an out of bounds variation when they are compared, then reasonability tests are applied to the two values and the more reasonable of the two is selected. The confidence is less than is achieved with values that remain within the limits on deviation, but continued system operation is maintained. The reasonability tests are based on past values and the rates of change for the sensor type as well as implications that can be drawn from combinations of data supplied from other types of sensors. Some upper limit is necessary on the number of different sensors that can be in this reduced sensor set condition, but the upper limit and the level of confidence required will not be addressed in this paper.

If the two valued case arises due to a fault that is internal to the SIR architecture, then a slight variation of the approach described above can be used. In this case there are only two communicating processors; the faulty processor has been powered down. The interstage registers that would have been used to connect the faulty processor are in a cleared state so the vote process still correctly functions in the remaining processors. (the vote instruction from the host processor to the interstage is the only means of generating an output from the slave status register to the host. This restriction was designed into the interstage controller to simplify the controller design.)

The two received values, one at each of the active processors, are processed as discussed for the case of two data values and three good processors. The only difference is that the minimum value is excluded from consideration in the final cross link and vote of the averages.

B. SYSTEM STATES

There are 2^n system states in systems that contain n components and where each component can be in either a correctly operational state or a faulty state. For the three processor case the number of components is 6 when the links are included. This means that there are 64 system states for the three processor SIR architecture. This set of system states includes all permutations of 6 components where each component can be failed or good.

The SIR architecture is operationally dependant on a TMR system. This condition results in the complete set of 64 system states not being useful. Acceptable confidence in the correct system output in TMR systems can only be achieved when at least two of the three processors agree on the value to be output from the system. System states where there are more than one faulty processor can no longer provide even minimum confidence in the values being generated for output to the external system. Combinations of processor faults and link faults that isolate two fault free processors are also not useful; if the two good processors cannot communicate, then there is no possibility of an explicit agreement on data values even if both processors are producing identical values.

The SIR system has another pertinent characteristic that effects the number of system states that are useful in redundancy management. There is no overall system controlling processor or component. Decisions on the correctness of the values that are supplied by the remaining active processors are made independently in each of the active processors.

There will several system states that are identical in the number and type of components that are faulty. The difference in these system states is apparent only in the labeling of the faulty components. If there is no overall system that controls the redundancy management, then the label changes are meaningless to the reliability calculations. This is true because the duplicated components used in the SIR architecture have an equal impact on the system if they fail and the probability of components of a like class failing is identical. A failure of processor A has no less and no more effect than the failure of processor B, if there are no other component failures in the system.

The calculation of the number of "good" system states in a TMR system, where good implies that the minimum TMR confidence level is met, is straightforward. The binomial coefficient notation is a compact way of describing the number of ways a subset of objects can be selected from a larger set. The notation for the binomial coefficient is shown in Figure 5-3. The binomial coefficient calculates the number of ways that a subset of objects can be selected from a larger global set of objects where the order of selection is unimportant.

$$\binom{X}{Y} = \frac{X!}{Y!(X-Y)!} \quad , X > Y$$

Figure 5-3 Binomial Coefficient Notation

For the purposes of modeling TMR system states, the X term in the binomial coefficient will represent the number of components in the system. The Y term in the binomial coefficient will represent the number of faulty components in the set of X components.

A further breakdown is necessary to correctly represent the system state aggregations that will be developed. Two fault free communicating processors is the condition required for continued operation of a TMR system. There are two component types in the system (the nodes and the communications links), so a complete representation of possible fault combinations must take the two component nature of the system into account. This is easily accommodated by the binomial coefficient notation by simply treating each component class separately and multiplying the two resulting values. The notation convention used will place the processor term first in the processor/link multiplicative pair.

Figure 5-4 shows the resulting equation for calculating the number of system states where continued TMR operation is possible, where the first factor of each term is the number of ways to choose fault-free processors and the second factor is the number of ways to choose fault-free links. These states are referred to as good states, although the confidence level varies among the states groups this set categorizes.

Number of "good" system states

$$\binom{3}{0}\binom{3}{0}+\binom{3}{0}\binom{3}{1}+\binom{3}{0}\binom{3}{2}+\binom{3}{1}\binom{3}{0}+\binom{3}{1}\left[\binom{3}{1}-1\right]+\binom{3}{1}\left[\binom{3}{2}-2\right]$$

$$= (1)(1)+(1)(3)+(1)(3)+(3)(1)+(3)(3-1)+(3)(3-2)$$

$$= 19$$

Figure 5-4 Calculation of the Number of Good TMR System States

A graphical representation of the fault classes that are sustainable by the TMR system is shown in Figure 5-5. The figure shows components that are faulty by using highlighted lines. Each fault class shows the corresponding term from Figure 5-4 to indicate the number of system states that can exist for that particular fault class (identical but for label changes).

A larger aggregation of fault classes can be made after studying the fault classes shown in Figure 5-5. The classes shown in the Figure 5-5c through 5-5f have a. important characteristic in common; one of the processors in the active set of processors is either faulty, or isolated from the remaining active processors by faulty communications links or a combination of both. Any of these four fault classes will of necessity be treated in the same fashion; the two remaining, communicating processors will assume that the third processor is faulty and proceed accordingly.

There will be a number of death states in the TMR system. The death states are composed of fault classes that do not allow the minimum TMR confidence level to be met. The number of death states is calculated in the same manner as for the number of good system states. A naive assumption can be made that the number of death states is simply the number of states remaining after subtracting the good states from the total number of states. This is not the case. The definition of a death state is one in which there are no further transitions possible. Implied in this statement is that there is some starting state for the system, ideally a fault free state. The assumption for only single fault arrivals in the SIR system has already been justified in Chapter IV. The combination of these two characteristics results in a reduced set of death states.

Figures 5-6 and 5-7 show the calculation of the number of death states possible in a TMR system and a graphical representation of the fault classes for system death states respectively. Each of the fault classes that are depicted in Figures 5-6 and 5-7 correspond to a fault class that is entered by a single fault arrival.

The system states remaining after the death states and the good states are subtracted from the total number of system states are classified as impossible system states for a TMR system. The reasoning for this classification is a result of the TMR characteristics together with the single fault arrival characteristic of the SIR system. All of the impossible state fault classes can only be achieved by way of a death state. By definition, there are no transitions allowed leaving a death state. The impossible states simply cannot be reached. (This assumes that the system shuts itself down when a death state occurs.)

The enumeration of the impossible states for the SIR (TMR) system is shown in Figures 5-8 and 5-9. These states can not be reached in the SIR system, so they will not be included in any configuration management algorithms or in the semi-Markov reliability model that will be generated in section D.

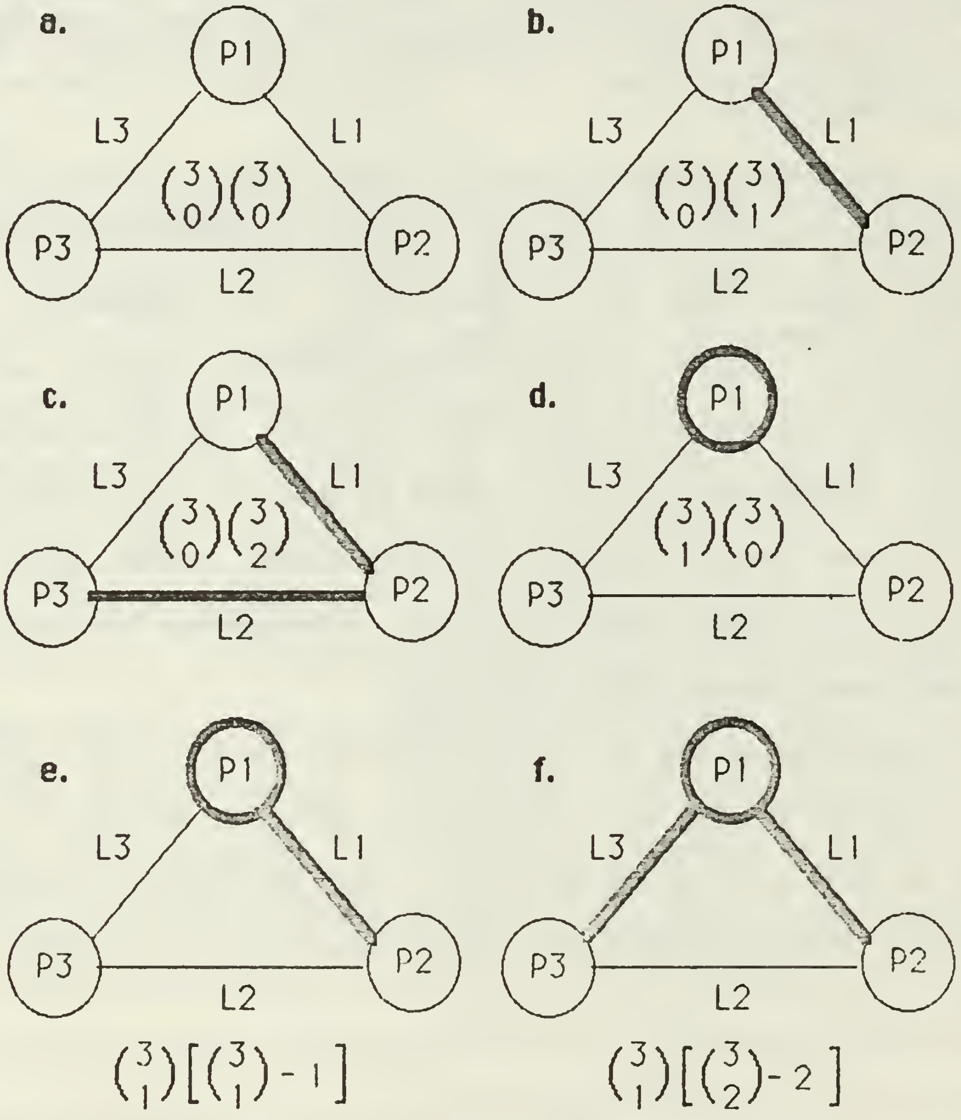

a.
P1
P2
P3
L1
L2
L3
$\binom{3}{0}\binom{3}{0}$
b.
P1
P2
P3
L1
L2
L3
$\binom{3}{0}\binom{3}{1}$
c.
P1
P2
P3
L1
L2
L3
$\binom{3}{0}\binom{3}{2}$
d.
P1
P2
P3
L1
L2
L3
$\binom{3}{1}\binom{3}{0}$
e.
P1
P2
P3
L1
L2
L3
$\binom{3}{1}\left[\binom{3}{1}-1\right]$
f.
P1
P2
P3
L1
L2
L3
$\binom{3}{1}\left[\binom{3}{2}-2\right]$

FIGURE 5-5 Views of Good TMR System States

Number of system "death" states

$$\binom{3}{0}\binom{3}{3} + \binom{3}{1}\left[\binom{3}{2} - 1\right] + \binom{3}{1}\binom{3}{3} + \binom{3}{2}\binom{3}{0} + \binom{3}{1}\left[\binom{3}{1} - 2\right]$$
$$+ \binom{3}{2}\binom{3}{1} + \binom{3}{2}\left[\binom{3}{2} - 1\right]$$

$$= (1)(1) + (3)(2) + (3)(1) + (3)(1) + (3)(1) + (3)(3) + (3)(2)$$

$$= 31$$

Figure 5-6 Calculation of the Number of TMR System Death States

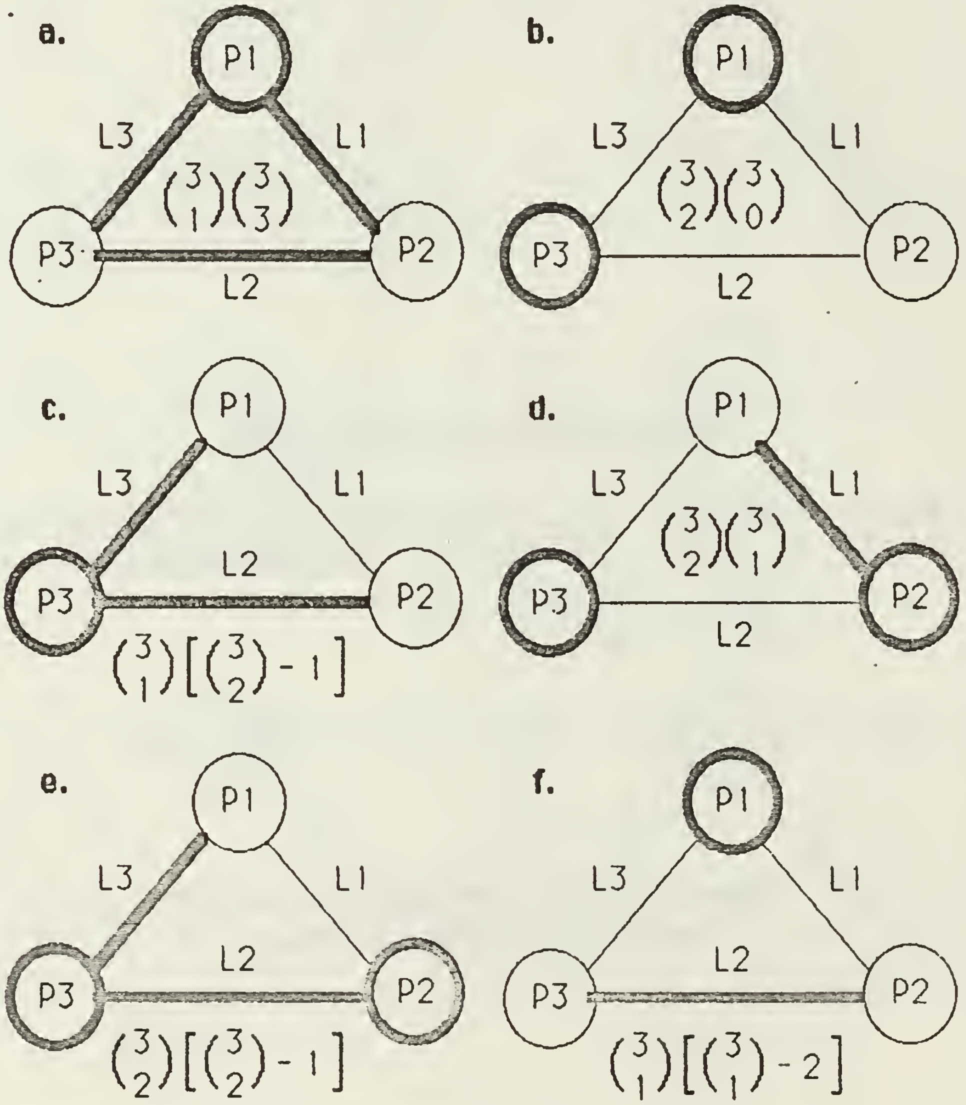

a.
P1
P2
P3
L1
L2
L3
$\binom{3}{1}\binom{3}{3}$
b.
P1
P2
P3
L1
L2
L3
$\binom{3}{2}\binom{3}{0}$
c.
P1
P2
P3
L1
L2
L3
$\binom{3}{1}\left[\binom{3}{2}-1\right]$
d.
P1
P2
P3
L1
L2
L3
$\binom{3}{2}\binom{3}{1}$
e.
P1
P2
P3
L1
L2
L3
$\binom{3}{2}\left[\binom{3}{2}-1\right]$
f.
P1
P2
P3
L1
L2
L3
$\binom{3}{1}\left[\binom{3}{1}-2\right]$

FIGURE 5-7 Views of TMR System Death States

Number of system "impossible" states

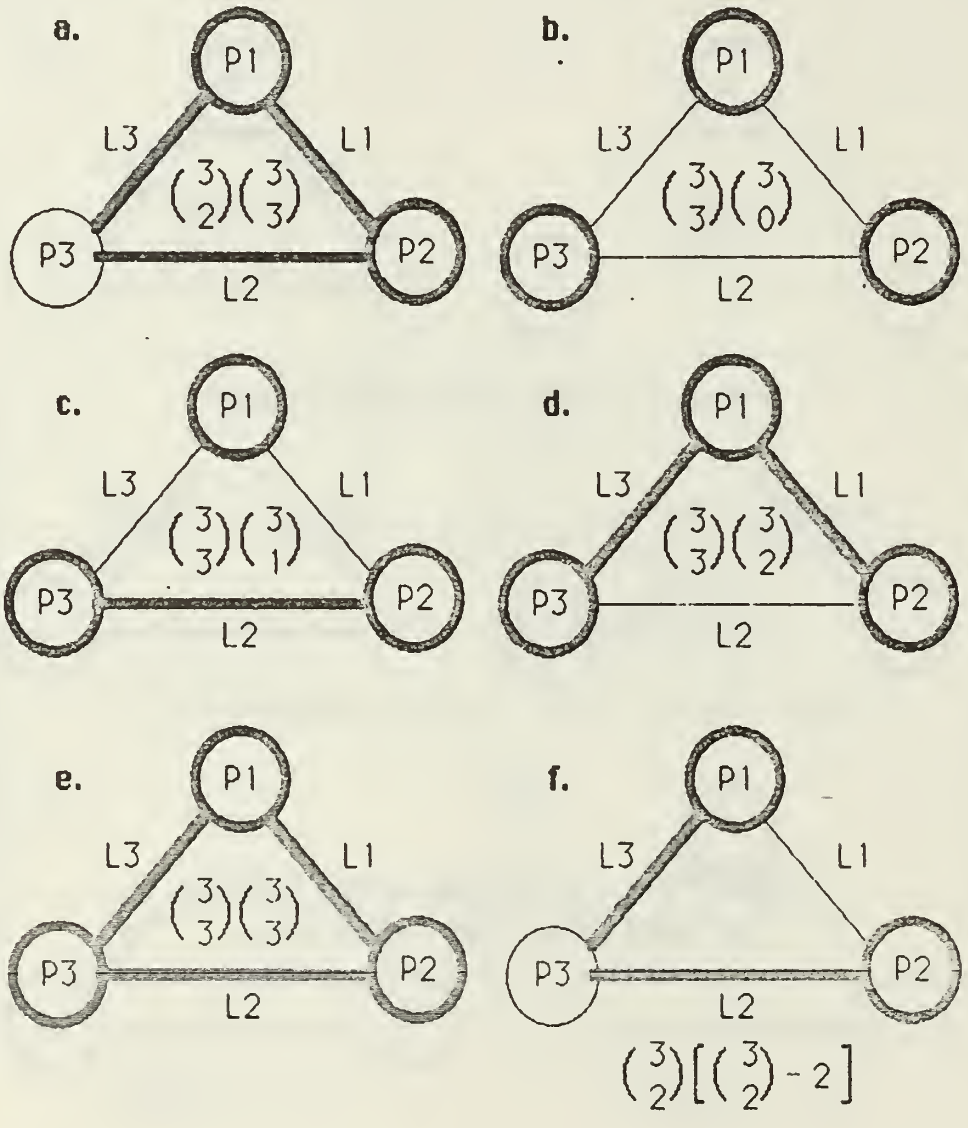
$$\binom{3}{2}\binom{3}{3}+\binom{3}{3}\binom{3}{0}+\binom{3}{3}\binom{3}{1}+\binom{3}{3}\binom{3}{2}+\binom{3}{3}\binom{3}{3}$$

$$+\binom{3}{2}\left[\binom{3}{2}-2\right]$$

$$= (3)(1)+(1)(1)+(1)(3)+(1)(3)+(1)(1)+(3)(1)$$

$$= 14$$

Figure 5-8 Calculation of the Number of TMR System Impossible States



a.
P1
L3
L1
$\binom{3}{2}\binom{3}{3}$
P3
P2
L2
b.
P1
L3
L1
$\binom{3}{3}\binom{3}{0}$
P3
P2
L2
c.
P1
L3
L1
$\binom{3}{3}\binom{3}{1}$
P3
P2
L2
d.
P1
L3
L1
$\binom{3}{3}\binom{3}{2}$
P3
P2
L2
e.
P1
L3
L1
$\binom{3}{3}\binom{3}{3}$
P3
P2
L2
f.
P1
L3
L1
P3
P2
L2
$\binom{3}{2}\left[\binom{3}{2}-2\right]$

FIGURE 5-9 Views of Impossible TMR System States

C. INTER-SIR COMMUNCATIONS AND FAULT LOCATION PROTOCOLS

The cross communication of data values between the active set of SIR nodes is necessarily dependant on a rigidly defined order of events and reactions to the outcomes of those events. This rigid operating procedure, called a communications protocol, is a straight forward method of exchanging information reliably between the node processors. Section A of this chapter described the method in which data is exchanged and voted in order to assure a reliable, congruent data exchange.

The TMR operating environment must also be able to react correctly to a fault arrival. Correct reaction to a fault means that the system must reconfigure the system components such that the set of active core components comprises, as close as possible, a complete set of good components. Correct reaction to a fault arrival requires that the fault not only be detected, but also be located. The algorithms for cross linking data values between the active processor set is adequate to detect a faulty component, however using these algorithms alone, the fault can only be located to a possible set of components consisting of the node from which the offending value arrived, the link over which it arrived, and the external input system (for the case of system input data).

The knowledge of a fault occurrence must also be tranmitted to all of the active nodes. It is quite possible that the fault can be detected at only one of the nodes in the active processor set. For example, this could happen by a failure of the control signal that enables the shift out operation (shift right) on the B or C registers in one of the system's interstages (re, Figure 2-3). If the failure occurs in node A and the failure effects the path to node

B, then node B is the only node in the system that can detect the error. As an example, consider the case where processor A operates correctly, other than the inability to enable the shift right instruction of the register connected to node B. Node C will not be able to detect the fault because the path to node A is functioning correctly. Processor A will not be able to detect the fault because the shift right operation of the offending shift register does not effect either the node's voter or the ability to recieve a data word from node B correctly.

In a TMR system, at least two communicating processors must agree on the result before confidence is placed in the result. This requirement also applies to fault detection. For the example above, if node B does not obtain corroboration of the detection of the fault in processor A (or the link between nodes A and B), then the consensus of the active processor set will be that node B is in error, even though node A is actually the faulty node.

A system for the exchange of fault detection must therefore be included in the communications protocol. Because the occurrence of a fault in the system is not the expected result (the system uses relatively reliable components), there must be a preset, recognizable method that explicitly states the intent to communicate a fault arrival rather than the next data word to be communicated. The preset signals that are used to indicate conditions within the SIR active processor set will be referred to as inter-SIR protocol command words, or just command words.

There are two methods that are generally used to differentiate the command words and the data words. The first is to send the word in a message format where there is a preset sequence of words expected; for example, a command word followed by a data word. If enough information needs to be transmitted with the data, then more than one command word can be sent in the message, again with the words in a preset sequence. The second method that can be used is to modify a frame synch bit, as in the case of the MIL-STD-1553 protocol. When a synch bit is available for use in this manner, there is more latitude available in the order of the words sent. A reduction in the total transmitted volume of information can be achieved in this manner. Each data word does not necessarily need a command word preceding it; expected data traffic can procede in an ordered manner, and if errors are detected, then a command word can be inserted into the data stream and be recognized as such by the difference in the synch bit.

The SIR architecture does not use synch bits in the transfer of data between it's internal nodes. The former method must therefore be used so that there is no possibility that data can be misinterpreted as command words and vice versa.

The structure of the command words must be designed so that the originator of a command word is included in the word. This knowledge is needed in the activation of an offline spare to insure that the active processors are all aware of the actual configuration of active nodes. The set of command words must also be large enough to indicate all of the possible faults that can be generated by a cross link process. The indication of a fault can be communicated among processors by sending the status of a vote. In this case the command word structure must be robust enough to include all possibilities of results that can be generated. The indication of an out of bounds condition is not sufficient; the processor that deviated from the allowable bound must be indicated in the command word.

The detailed design of the command word structure will not be addressed in this paper. The high level protocols developed will assume an appropiate command word structure for the task. The discussion above is included for completeness of the discussion on the communications protocols.

The node processor acts as the controlling element of a fully controllable n x 2 full duplex switch, where the switch is actually the node's rotary multiplexer (n is the number of remaining processors in the given SIR architecture). Control of the paths selected to connect the rotary multiplexer input ports with the multiplexer output ports is established by the control word supplied to the rotary multiplexer by the node's host processor. The process of changing the routing of the paths through the multiplexer also changes the composition of link subcomponents that comprises a particular system link. The single fault arrival assumption assures that if the component that has actually failed is the link, then only one terminal end of the link has failed. Although the subcomponent compositions of the links are now different, each new path is an independent circuit once the control word has been loaded into the set of rotary multiplexer flip flops (as discussed in Chapter II B).

The notation shown in Figure 5-10 will make the following discussion easier to follow. Each of the nodes contains 4 parts of a communications path: two are incoming simplex paths and two are outgoing simplex paths. These paths are indicated as 1 and 2 in the Figure 5-10, with the direction indicated by the subscript **i** for input and **o** for output. The paths will always be manipulated as pairs. For example, a node's number 2 path will be directed to the same external node for both the input and output subscripts.

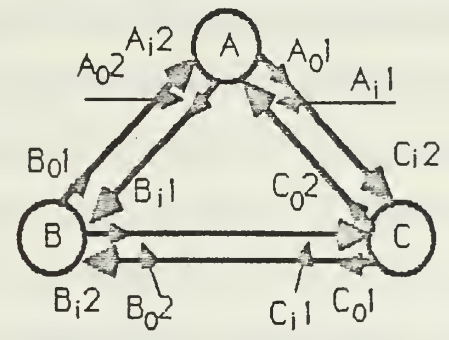


Figure 5-10 Basic SIR System

Using the notation introduced by Figure 5-10, a link is composed of 2 pairs of components. For example, the link between nodes A and B in the figure are represented by the component pair B1A2 and the link between the nodes A and C is represented by the component pair A1C2.

The communication requirement between processor pairs consists of a communications path in both directions. Each path has hardware components located at the terminal ends of the path, however all of the link components must be operational for the link between the connected nodes to be classified as good. No inconsistency with the concept of a link component being the combination of the hardware at both ends of the link is introduced by the additional information shown in Figure 5-10.

Because the rotary multiplexer is fully controllable, each of the selected paths through the rotary multiplexer can be associated with either of the interstage communications registers. The B interstage register pair (consisting of registers B and B' in Figure 2-3) of node A can be routed to either node B or node C.

The ability to switch the interstage register pair associated with a link path allows isolation of the fault in the communications path (consisting of the interstage and the link) to either the interstage or the link itself, thereby providing fault location. If the fault is caused by the inability of node A's interstage register (B) to shift right, as in the example, then changing the external node that is connected with the B register to the other active node will provide that second node with an indication of a communications fault. Now two of the three nodes have received an indication of a communiations fault associated with node A, and the fault is isolated to the interstage of node A.

The approach of switching the external nodes connected to a particular interstage register pair is a sufficient location test for node faults that effect only the interface with the rotary multiplexer. The same method also correctly identifies a node that fails to correctly input communicated data into it's primed interstage registers.

An example is the best way to show that the path switching method can uniquely locate a fault under the single fault arrival assumption. The results of a cross link of a data word can manifest itself in three basic ways. The first case is for two nodes to indicate the same fault. Under a single fault arrival assumption, this leads to the unique conclusion that the indicated node is bad. A second case is for all three nodes to agree on the vote outcome, in which case the voting process is assumed correct. If the vote outcome is that all data values are correct, then all system components are considered fault free and the system proceeds. If the vote outcome indicates a particular word is bad (not equal or out of range) then two cases can apply. If the word is internally generated (a command word or a data word produced by the flight law calculations), then the node that produced the erring word must be bad. If the word is produced by a source external to the SIR architecture (DSPM) then a preset test word must be used in a cross link test. If each voter status agrees the outcome of this test (using invarient data transfer) then the fault is external to the SIR and the DSPM redundancy management routines are notified of the fault.

The third case applies if the results of the vote test do not agree (for the preset word test or for the original vote outcome). A series of additional tests must be performed to isolate the fault. If two nodes indicate the same node as faulty, then by the single fault arrival assumption, the indicated node must be faulty. If only one node has indicated a fault then more tests must be implemented. (If the node indicates that both the other nodes are faulty, then the single fault arrival assumption implies that the node which indicates the faults is bad.)

Figure 5-11 depicts the possible test outcomes for a set of three tests (a, b, and c). Test outcome (a) in Figure 5-11 shows a possible senario where only one node indicates a fault. (This test outcome is generated by the data cross link that supplied the initial fault detection.) The rows in the figure indicate the test outcomes of each of the nodes with the columns indicating the nodes on which the test was performed. A good test outcome is represented by a 0 and a faulty test outcome is represented by a X. The possible faults that could cause these outcomes are also indicated in the figure. There are several possible faults that could generate the test outcome shown in Figure 5-11a. A second test is needed to isolate the actual fault condition that caused the test outcome shown in Figure 5-11a. A logical second test switches the paths through node C's rotary multiplexer and repeats the data cross link which initially generated the test outcome shown in Figure 5-11a. The possible outcomes of the second test are shown as test outcomes b.1 through b.3 in Figure 5-11. (The single fault arrival assumption limits the possible outcomes.) Unique fault locations are provided for each of the test outcomes except for test outcome b.2 in Figure 5-11 In this case the fault can't be isolated with just this test. A third test is needed. The logical approach is to exchange the paths through the rotary multiplexer of node A and cross link and vote again. The possible outcomes of this new test is indicated in test outcomes c.1 through c.2 in Figure 5-11. The outcomes of this second test are unambiguous under a single fault arrival assumption so no further test are necessary. The fault has been isolated. The set of faults listed as the possible causes of the test outcome shown in Figure 5-11c.1 is a simplification. The test outcome isolates the fault to the outgoing end of a link that resides at processor A. The failure could also be in the interstage of processor A, however. If that is in fact the case, then node A is actually faulty and not the link. The test isolates the fault to that single outbound path from node A however, and verifies that the remaining path out of node A is functioning correctly. For the three processor case, the fault has been isolated to a sufficient degree and no other test is necessary. For the case of the SIR architecture with spares, an indication should be registered that either the outbound path through the interstage in node A is faulty or that the outbound link from node A is faulty. In either case the node should be replaced with a completely fault free spare (with requisite links). The node is not however eliminated from consideration for possible use in a future active node configuration.

TEST OUTCOMES		A	B	C	POSSIBLE INDICATED FAULTS
	A	-	0	0	LINKS C_i2, $A_0$1
a	B	0	-	0	NODES A, C
	C	X	0	-	
	A	-	0	0	
b.1	B	0	-	0	LINK C_i2
	C	0	X	-	
	A	-	0	0	LINK $A_0$1
b.2	B	0	-	0	NODES A, C
	C	X	0	-	
	A	-	0	0	
b.3	B	0	-	0	NODE C
	C	X	X	-	
	A	-	0	0	
c.1	B	X	-	0	LINK $A_0$1
	C	0	0	-	
	A	-	0	0	
c.2	B	0	-	0	NODE C
	C	X	0	-	

Figure 5-11 Possible Test Outcomes For SIR Fault Location Algorithm

The two cases of link failure (the input half of a link or the output half of a link) can actually be thought of a single case because the link must be full duplex for correct operation. Failure of both simplex paths associated with a single duplex link is not consistant with the single fault arrival assumption that was validated in Chapter IV, unless the fault lies in a node's hardware or software and not the node's link hardware. This will happen in two separate cases: the node processor supplies an incorrectly ordered set of bits to the set of flip flops in the rotary multiplexer; or the failure of the AND gate that controls the loading of new values into the rotary multiplexer flip flops. In either case, the nodes that are connected by the rotary multiplexer have no opportunity to agree on data values. The ideal TMR operation is no longer possible and an alternative is required.

While ideal TMR operation is no longer possible in the nodes that are connected by the faulty link, the remaining active processor can still perform an ideal TMR operation. If all three processors remain in service, then the node not directly affected by the faulty link has become a single point of failure for the system. It must relay data words between the two processors that have the faulty link in common. Neither of the remaining links is a single point of error for the system, nor are the nodes that have the failed link in common. A single failure of either of the remaining good links results in the isolation of a processor, but there are still two communicating processors in the system and, as discussed in section A.2, operation can continue.

Figure 5-12 shows a flowchart of the communications and fault location protocol. The operational modes for cross linking data under varying fault classes, discussed in section A of this chapter, are referred to in the

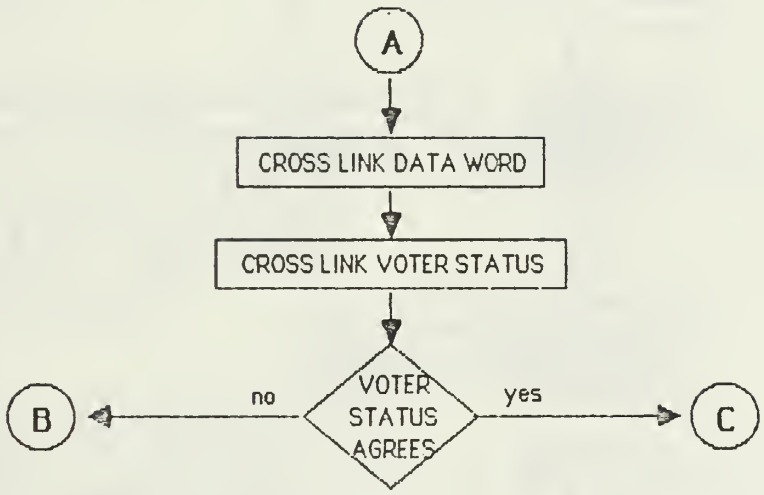


Figure 5-12 Three Processor mode Inter-SIR Communications and Fault Location Protocol

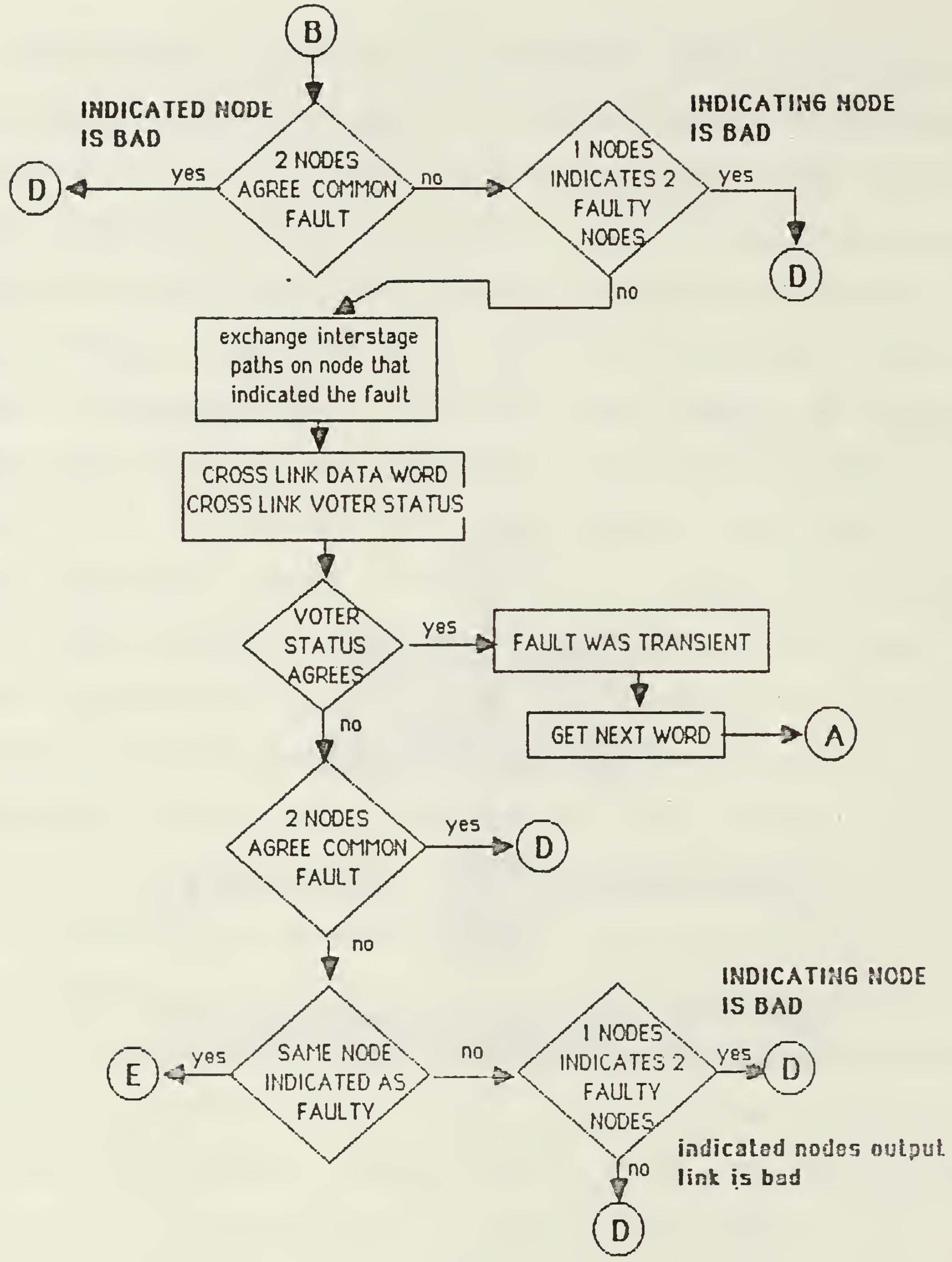

B
INDICATED NODE IS BAD
2 NODES AGREE COMMON FAULT
yes
D
no
1 NODES INDICATES 2 FAULTY NODES
INDICATING NODE IS BAD
yes
D
no
exchange interstage paths on node that indicated the fault
CROSS LINK DATA WORD
CROSS LINK VOTER STATUS
VOTER STATUS AGREES
yes
FAULT WAS TRANSIENT
GET NEXT WORD
A
no
2 NODES AGREE COMMON FAULT
yes
D
no
SAME NODE INDICATED AS FAULTY
yes
E
no
1 NODES INDICATES 2 FAULTY NODES
INDICATING NODE IS BAD
yes
D
no
indicated nodes output link is bad
D

Figure 5-12 (continued) Three Processor mode
Inter-SIR Communications and Fault Location Protocol

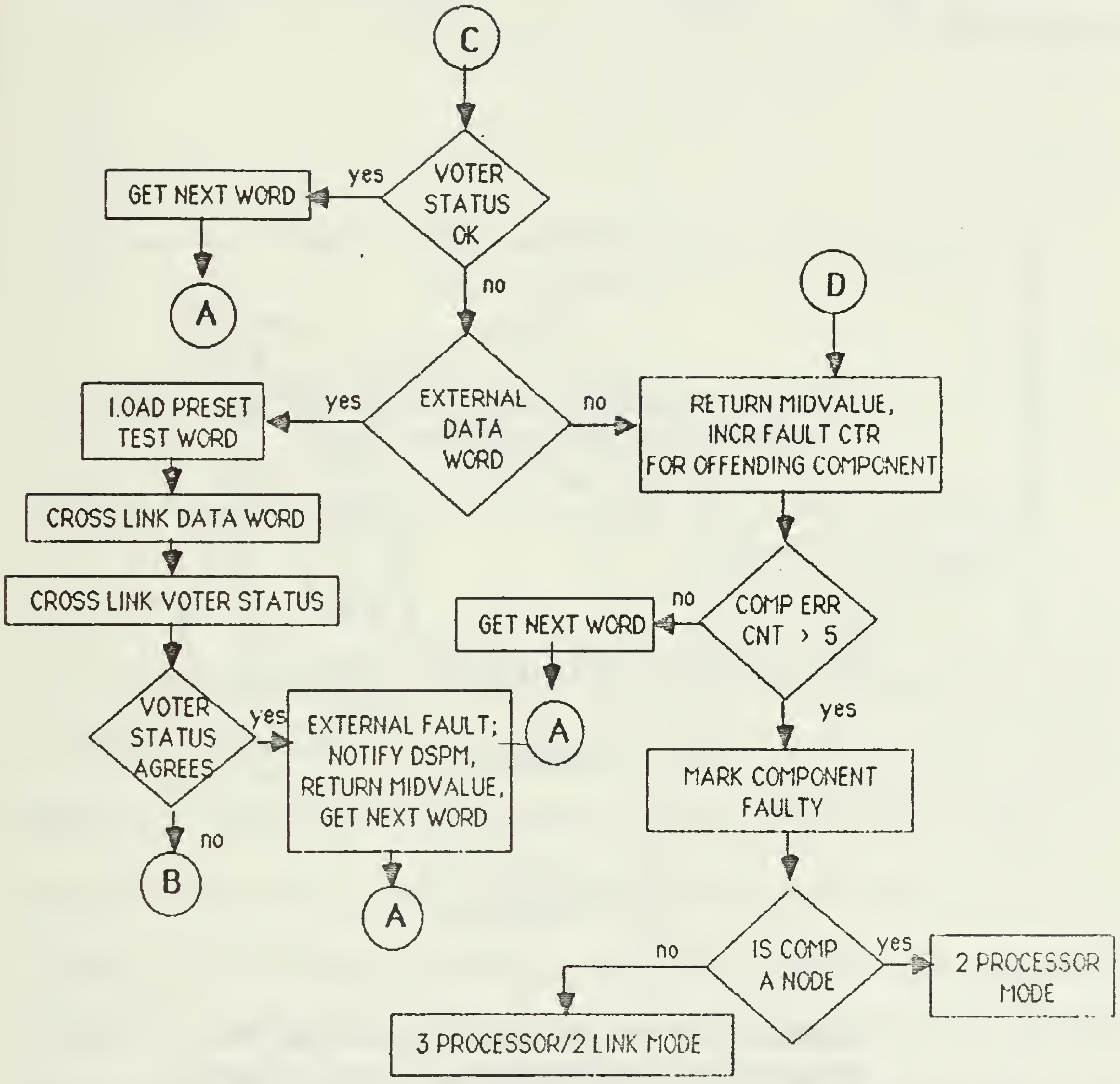
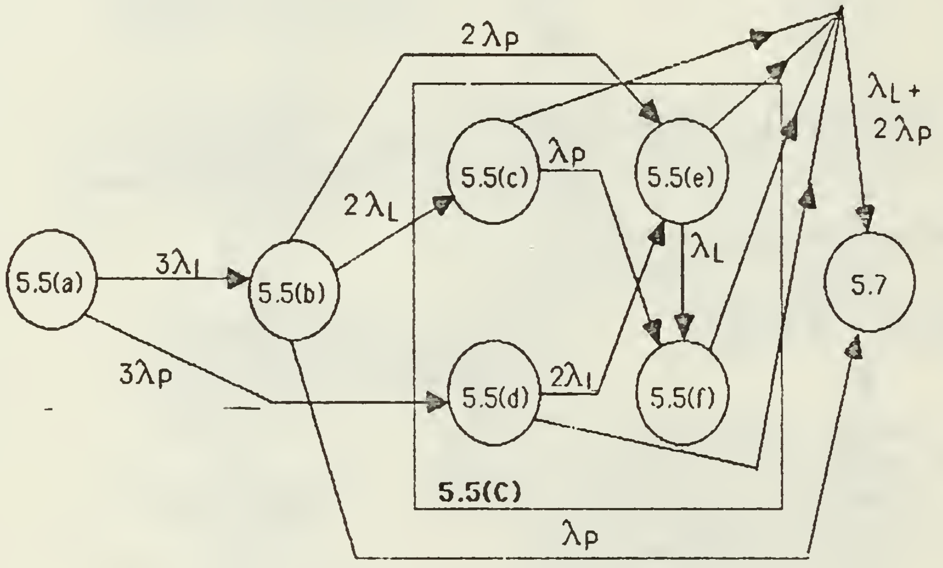


Figure 5-12 (continued) Three Processor mode Inter-SIR Communications and Fault Location Protocol

D. RELIABILITY MODEL

All of the necessary information has now been developed for applying the semi-Markov model to the three processor case of the SIR architecture. The model will be shown graphically as discussed in Chapter IV, and is shown in Figure 5-13.



$\lambda_P = 5.9423 \times 10^{-4}$
$\lambda_L = 3.4519 \times 10^{-6}$

Figure 5-13 (a) Reliability Model for the Three Processor SIR Architecture (complete)

$$\begin{tikzcd}
5.5(a) \arrow[r, "3\lambda_L"] \arrow[rr, bend left, "3\lambda_P"] & 5.5(b) \arrow[r, "2\lambda_L + 2\lambda_P"] \arrow[rr, bend right, "\lambda_P"'] & 5.5(C) \arrow[r, "\lambda_L + 2\lambda_P"] & 5.7
\end{tikzcd}$$

$\lambda_P = 5.9423 \times 10^{-4}$

$\lambda_L = 3.4519 \times 10^{-6}$

Figure 5-13 (b) Reliability Model for the Three Processor SIR Architecture

Note that there are two seperate λs that are used to indicate the arrival of a fault. (Recall that the arrival of a fault corresponds to a state transition in the semi-Markov model.) λ_P will denote the arrival of a processor fault, while λ_L will denote the arrival of a link fault.

There are no system recovery transitions possible for the three processor case of the SIR architecture. This is because there are no spares associated with the system. The system is tolerant of faults, with the number of faults that can be tolerated being a function of the order in which the faults arrive. Therefore, use of the notation descibed in Chapter IV corresponds to a linear state transition graph.

The notation that is used in the reliability model refers to the fault classes that were described in section B of this chapter. For example, the fault free category is notationally listed as state 5.5(a). The notation refers to Figure 5.5 (a) to describe the fault classification that corresponds to the state.

A good deal of state aggregation is used in the reliability model shown in Figure 5-13. The death state categories that are shown in Figure 5-7 are unique classifications of faults, however for the purposes of the model, the classification of subcategories of a death state is not important. Therefore, the figure shows a single death state. The path that is followed in the figure to arrive at the death state identifies the fault class of the death state.

Figure 5-13a shows the complete set of good states for the three processor case and the possible transitions due to fault arrivals. The states that are shown enclosed within the dotted box are effectively identical with respect to system states that follow and with respect to the communications and operational algorithms that are used in each of the states. A super state aggregation can therefore be made and no reliability information will be lost.

The transitions that are made inside the dotted box are included only to show a complete exhaustion of system components. The algorithm that detects and locates a fault would not actually allow the transitions internal to the dotted box to take place. This is because when a processor is determined to be faulty the fault is considered to be a permanent fault and the processor is powered down. Because the fault decision is made by the two nodes that aren't faulty (or isolated) the actual condition of the node that is voted as faulty is not relevant. The single fault arrival assumption assures that the correct node is powered down because both of the nodes that decide the fault cannot be faulty. The states shown as (c), (d), (e), and (f) in Figure 5-5 are therefore actually one distinct state as viewed by the redundancy management algorithm. This is shown in Figure 5-13b where the state shown as 5.5(C) represents all of the states within the dotted box in Figure 5-13a.

Rather than show a seperate transition for the link and node fault arrivals, a notation is used that is more compact notationally, while losing none of the actual reliability information. If two transitions leaving a particular state both arrive at the same next state, then the probability of transitioning between the two states is just the summation of the two transitional probabilities. The collapsed notation helps keep the model diagram from becoming cluttered unnecessarily.

VI. THE SIR ARCHITECTURE WITH SPARES

The SIR architecture is obviously designed with the operation of spares in mind. The addition of the rotary multiplexer is made for just this purpose. The development of the three processor case was necessary because in all of the cases of the SIR architecture that utilize spares, the active set of processors remains a three processor core. The method in which the system detects faults is identical whether the spares are included or not. The introduction of faults to the system over time eventually leads to some mode of the three processor case. This will happen when processors fail and are replaced in the active core by spares. A failed processor is now placed in a spare position, however there is no possibility of restoring the failed processor to active status. The set of available spares is therefore reduced by one processor. Continuation of this process leads to the depletion of the spares and thus effectively results in a mode of the three processor case. In a like manner, the failure of links can isolate processors. The results are the same; effective reduction to some mode of the three processor case.

The task of the operating protocol is a little different in the SIR system with spares. The location process is even more meaningful in this case, because a history must be kept of all the faults that have occurred in the system to date. The loss of a single link may cause the swapout of a processor. However the processor that is placed in spare status is still functional, and may be used in a different configuration of the active core so the core is always composed of fully functional components. Of course this depends on the number of spares in the system (with the requisite number of links for full interconnection) and the fault history. With this possibility in mind, the algorithm that determines how the spares are to be managed is key to the reliability model of the system as a whole.

The amount of system state information and the complexity of the system grows at an exponential rate as can be seen by the 2^n figure for the number of system states. The number of processors grows in a linear fashion with the addition of spares to the system. The number of links grows at a greater rate however. Because the system is composed of fully connected processors the addition of a single processor causes the addition of a number of links equal to the number of processors in the system before the addition of the new processor. Addition of the first spare causes the addition of three new links to the system. The second added spare causes the addition of four new links. All the while, the complexity of the system in terms of system states is growing at 2^n.

There are aggregations of system states that can be made in the four processor case similar to the way in which state aggregations were made for the three processor case. For the same reasons as given in Chapter V, there will be a number of impossible system states that can be immediately eliminated from consideration in the reliability model as well as the operating protocol.

The death states in the SIR system with spares will be a larger set of fault classes for several reasons. The obvious reason is that there are more components in the system. Secondly, the system will always be in a death state when two of the active core of processors fail, even when there is a full complement of good spares. This condition occurs whenever the fault arrivals in the active core of components happens faster than the recovery process. This possibility of system failure prior to the exhaustion of the spare set emphasizes the need to make the recovery process as quick as possible. When a link or a processor fails, the processors remaining in the active core must replace a processor by an appropriate spare in order to return the active core to a fully operational component set.

The operation includes selection of the appropriate spare and testing of the spare and its link with the two processors selected to remain active (If the fault was a link failure, then there are two choices for which active processor will be placed in a spare status.) A process of updating the new addition to the set of active processors must be performed when an acceptable node is found. The DSPM state table must be transferred to the new node as well as the fault history in the SIR system. State information for the application software must also be transferred, particularily past data input history that is used in calculating the reasonableness of a suspect new data word. All of this information is necessary and it's communication will consume time. Of course the data transfer must be of the invarient type for transfer of system state information.

The TMR operation of the SIR system and the single fault arrival assumption assure that that there will not be more than one fault in the active set of processors unless there is no possibility to transition to a configuration (through the use of components in the set of spares) that contains less faults than in the present active core of components. When a fault arrives in the active core a search is undertaken to locate a fault free spare node such that its links with respect to the two communicating nodes from the active set are also fault free. Which of the spares selected (if any) is dependant on the location of the detected fault in the active set of components as well as the state of both the spare nodes and the spare links (with respect to the active set of nodes).

A question arises as to the number of states that are needed to completely specify all possible system states that can exist under the SIR operating environment. A search for a replacement node for the active set is not undertaken until a fault that arrives in the active set of components, is detected, and located. Recall that the system nodes are fully interconnected but that the search for a configuration that will improve that of the active core (with respect to the number of faulty components) is performed only by the nodes currently in the active set. The links that are terminated only on spare nodes can be discounted because the state of these links are not observable by the nodes in the active set which are making the decisions. This means that there are a set of three links which connect each spare node with the active set of nodes (see Figure 6-1). The number of possible states that can exist among the spare links (discounting links connecting spares) is thus 2^{3n}, where n is the number of spares in the system. Because the single fault arrival assumption is not valid for the spare conponents (the test interval for the spare components could be much longer than the test interval for active components) any of these states are possible when a fault arrives within active core. The fault that arrives in the active core will be detected and located, however because the probability of a fault occurring is equal among like components (nodes or links), all possible positions of that fault in the active core must be accounted for in order to assure that all possible paths through the Semi-Markov model are considered. The number of states is then three times the number of possible states within the set of spares, or 3×2^{3n}, which reduces to $3(8)^n$.

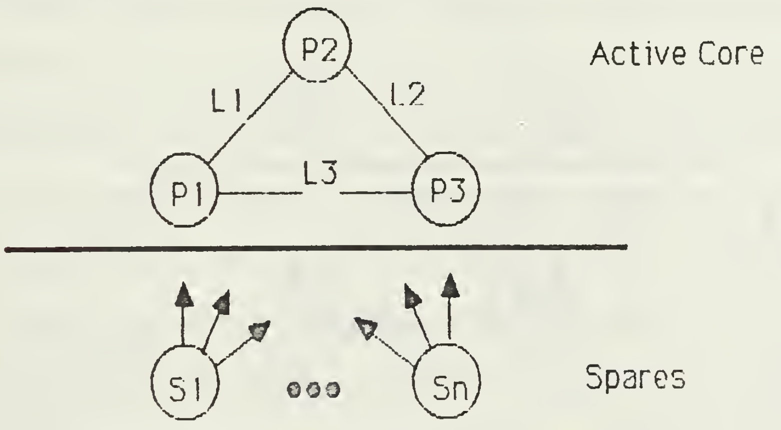


Figure 6-1 SIR Architecture with Spares

Suppose however that only the number of link faults for each of the spares is known (with respect to the nodes in the active core). Note that three link faults between a particular spare and the nodes in the active core will isolate the spare and is thus equivalent to a failed spare (with respect to the current configuration of the active core). Four states are therefore required to indicate the information concerning each set (which consists of a spare and the links that terminate on both the spare and one of the nodes in the active core). Because there are n such sets, the total number of states required is $(4)^n$. The total number of states required to describe the system is then $(4)^n$ times the three possible positions of the fault that has just arrived in the active core, or $3(4)^n$.

Suppose that the following conjecture is true. All nonredundant state information is retained in the latter system state description. An example for which this is true is shown in Figure 6-2.

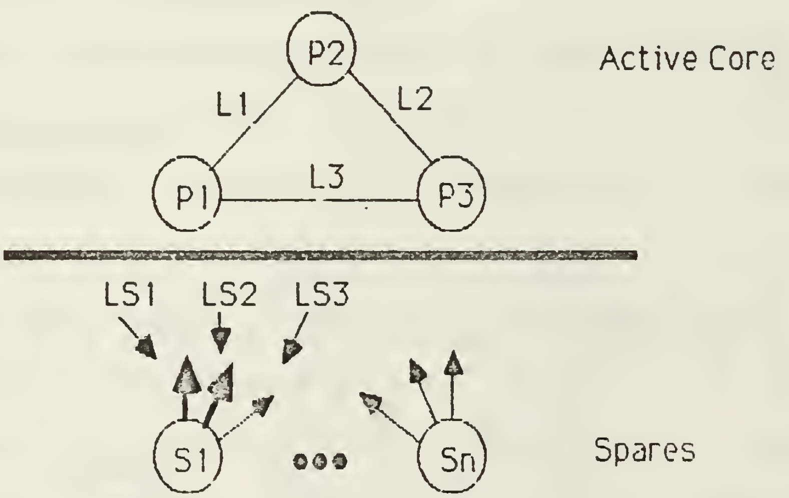


Figure 6-2 Sample Fault Senario for SIR Architecture with Spares

Assume a link failure has occurred within the active core and that there are 2 faulty links between spare S1 and the nodes in the active core. There are 3 possible ways that the 2 failed spare links could be configured with respect to the failed link in the active core. Two of the ways are topologically equivalent. This case arises when L2 or L3 is the failed link in the active core and LS1 and LS2 are the failed links associated with S1. These are equivalent topologies because in either case two of the active nodes each have 1 failed link with respect to the remaining active core and the spare S1 while the remaining node in the active core has 2 failed links associated with it. In the remaining case (with L1, LS1, and LS2 faulty) 2 active nodes have 2 associated faulty links. The same number of possible system topologies would have resulted by fixing the active link failure and varying the 2 failed spare links in all possible combinations.

If the state is represented by the unique number of failed links associated with the spares and the occurrence of a link failure in the active core, then all of the state information is still contained in the model only if three transitions are shown for the state recovery. Effectively, some of the state information is represented in the transitions leaving a state. The positional dependance of the failure in the active core actually represents three states. The specification of which state is actually present in the three state aggregation is contained in the recovery transition that is selected in leaving the three state aggregation. These transitions must correspond to fixing the failed spare links and varying the position of the active link failure. In the case shown above, two of these transitions are equivalent so the probability of transitioning to the recovery state indicated by these equivalent topologies is 2 times as great as the probability for the transition indicated by the remaining topology (the active link failure is L2).

An assumption will be made in this paper that the conjecture discussed above is, in general, valid. The number of states needed in the models are thereby reduced from $3(8)^n$ to $3(4)^n$ which is significant for cases of n as small as 2 (the five processor case).

A. THE FOUR PROCESSOR CASE

The four processor case of the SIR architecture consists of the three processor active core and a spare processor (and the links that connect the processors). Following the procedure that was developed in Chapter V, the set of system states can be broken down into a set of good system states, a set of death states, and a set of impossible states.

Figure 6-3 shows the set of possible states in which TMR operation is possible within the active processor core. The notation used in the figure is slightly modified, so that "S" represents a spare processor.

The labels on the links were not added to this figure. No real information is provided by the labeling of the links because each of the fault classes represents all combinations possible by label changing of the particular faults composing the fault class.

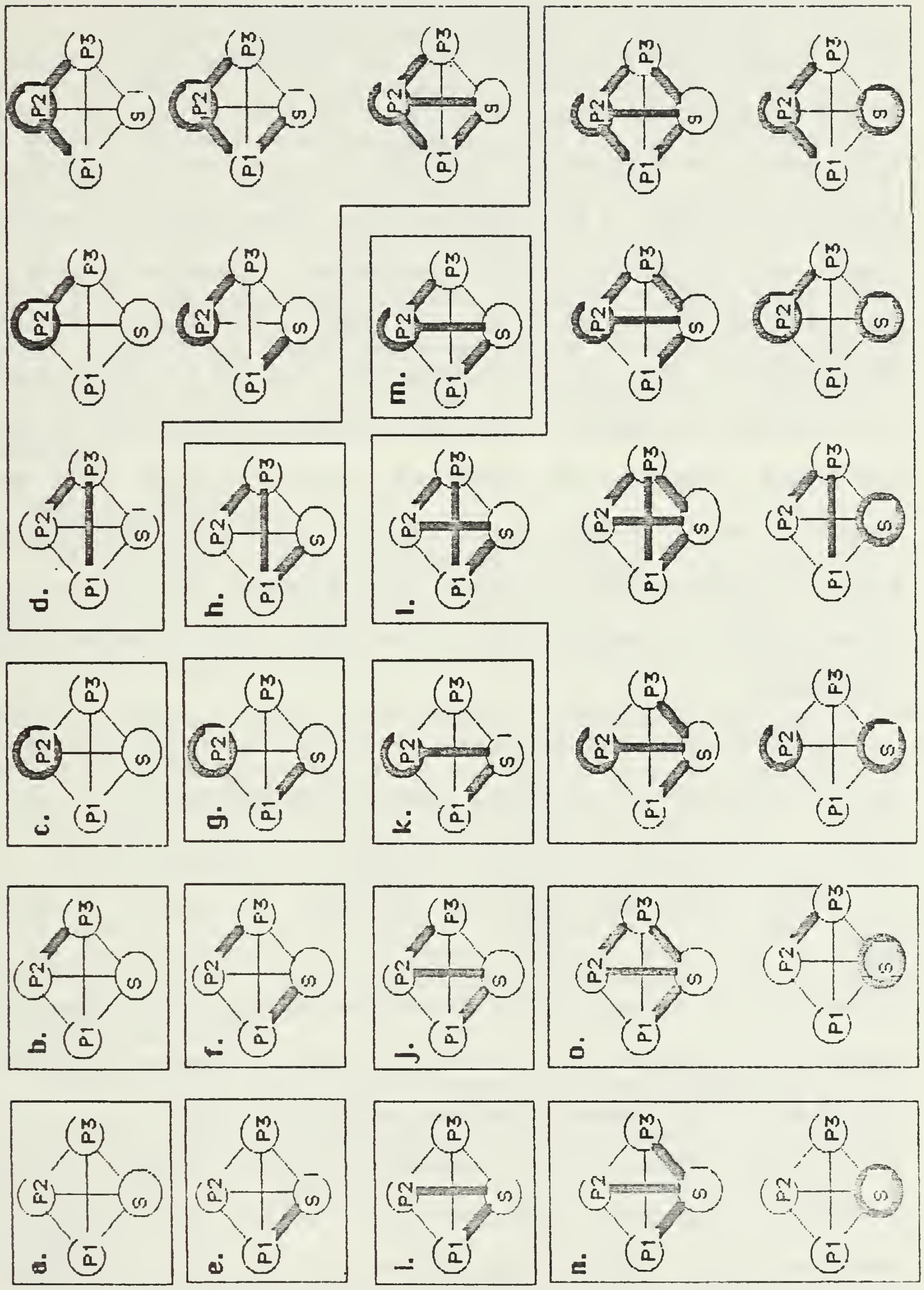


Figure 6-3 Good system states for
Four Processor Case

Several aggregations of the fault classes that are labeled good are made in the figure. Figure 6-3 (d) represents a class of "good" system states that will be assumed to be impossible states in the model for the four processor SIR system. The impetus for this decision is based on the assumption of single fault arrivals in the components that comprise the active core. A system reconfiguration is always implemented, if possible, when a single fault is detected in the active core. The length of time that will be allowed to perform the reconfiguration will be limited to the maximum bounds on the control cycle. Chapter IV validated a single fault arrival assumption for the active component set, therefore under the single fault arrival assumption the set of fault classes shown in Figure 6-3 (d) are impossible. Each of the fault classes have more that one fault in the active core and a reconfiguration is possible to a core with a larger number of good components.

The fault classes that are shown in Figure 6-3 (l.), (n.), and (o.) are each aggregate classes that effectively equate to the fault classes for the three processor case shown in Figure 5-5. Each of the aggregates shown in (l.), (n.), and (o.) are actually larger than indicated in the figure. For reasons of space in the figure, the three cases of a failed spare and the links connecting it to the active core were not shown (one, two, or three failed links). The aggregations do not change for these cases because a failed node causes an effective failure in the links associated with it.

The death states and the remaining impossible states for the four processor system do not need to be shown for cases other than the three processor case. The TMR requirement for two active processors is applicable only to the active core; any faults that take the active core to less that two active processors that are able to communicate with each other is a death state. All combinations of spare processors and links in their good or failed states can be aggregated into this overall death state.

The number of states that must be managed for the four processor case has been reduced from 2^{10} to just 15 (including the death state). The task remaining is to develop an algorithm that effectively manages these states and sets up appropriate transitions between them. Given the algorithm and the state aggregations shown in Figure 6-3, a semi-Markov model can be constructed.

1. Recovery Algorithm

The algorithms that were discussed in Chapter V for detection of faults in the active core apply, with minor changes, to the case of the SIR architecture with spares. The major difference is that once a fault has been detected and located, a search for a spare is undertaken. The goal of the search is to find a spare that, when activated, brings the complement of good components in the core back to 6 (3 processors and 3 links). Short of this, a reconfiguration is desired that will connect three good processors with 2 good links. The final course of action is to use just 2 good processors and a single link connecting them, and power down the remaining processors in the system..

For the case of one spare, there are not many possibilities to be checked in the search. If the detection algorithm indicates that a processor has failed, then the spare is activated, and the failed processor is deactivated.

There are several cases that can occur when the spare is activated. The spare components are in a deactivated state and it is not possible to test these components using the algorithms described in Chapter V. The set of spare components is therefore subject to fault arrival rates greater than for the case of the active core. The spare can be in a failed state, or either of the links, or both, connecting the spare with the remaining good processors can be failed. If both links are failed, the spare will behave as if it is failed. If the spare is failed, or both links associated with the spare are failed, then the two processors in the active core will deactivate the spare and operate in a 2 processor mode.

If one of the links between the spare and the active processors is failed, then a 3 processor/2 link mode of operation is set up. Note that the initial load of system state values to the spare must be validated as correct. To validate the state values, the spare relays the data back to the active processor with which the spare has a good link. At the same time the node that has 2 good links (the "center node") sends the state information to the remaining active processor. In this way, the center node performs a check on the state data contained in all three processors. Note that the center processor has become a single point of failure for the system.

A similar process is performed for the case of a link failure detection in the active core. In this case, a decision must be made as to which of the two nodes coincident with the failed link is to be deactivated. If there is a link failure in one of the links that connects to the spare, then the choice is critical. The wrong choice leads to selection of a configuration that is not optimum for the set of nonfailed components. A way out of this delimma is to not deactivate a node. Instead, one of the nodes that is coincident with the failed link can be placed in a wait state by setting that node's watch dog timer appropriately. The two remaining nodes attempt to establish communication with the spare in the manner indicated above. If a complete set of components for the core is established, then the active node in the wait state is deactivated and held in reserve (only one link is faulty, there is still a chance that the node may have a good link with the newly activated spare). If a complete set of core components is not achieved, then the process is performed again with the roles of the nodes that are incident with the failed active link reversed. Of course, this algorithm requires establishing communication and performing a synchronization with the node that is in the wait state.

The history of fault occurrences and where they are located is maintained in each processor. This enables past knowledge to assist in choosing the best strategy on the occurrence of a new fault in the active core. The algorithm described allows recognition of all of the states that are shown in Figure 6-3 (except for (d) which is an impossible state).

2. <u>Reliability Model</u>

A graphical display of the reliability model for the four processor case of the SIR architecture is shown in Figure 6-4. The model shows all of the possible transitions between the state aggregations that are shown in Figure 6-3. The notation used in Figure 6-4 is modified slightly from that used in Chapter V. Because of the complexity introduced in the system by the addition of the spare node and the associated links, it was not convenient to follow the horizontal and vertical paths to represent fault arrivals and recoveries respectively. Each of the transitions in Figure 6-4 are instead labeled with λ's and α's. The λ's represent fault arrivals and are further classified to indicate which type of fault has occurred. The subscripts shown indicate a fault as being an active processor (P), an active link (L), an inactive link to the spare (SL), or the spare processor (S). The recovery times are assumed to be identical for each of the fault classes. The labeling of the states in Figure 6-4 refer to the fault classes shown in Figure 6-3.

The (f) fault class shown in Figure 6-3 was seperated into two distinct fault classes in Figure 6-4. If the link fault in the active core is the link between P1 and P2, a recovery can be made to state (i). This recovery restores a full complement of good components to the active core. The same recovery is possible if the faulty link in the active core is between P1 and P3. If the faulty link is the one between P2 and P3, however, the only configurations possible in this case are composed of an active core with three good processors and two good links and no improvement can be achieved. This case is labeled (f2) in Figure 6-4 and the former two cases are labeled (f1). All recovery transitions from a single state are assumed to have equal probability. The probability of a particular recovery transition is indicated notationally as α_n, where n equals the inverse of the probability of that transition (and the number of transition that leave that particular state). For example, in state (k) of Figure 6-4, there are three recovery transitions possible depending on the position of the failed processor in the active core. Two of these three transitions go to the same state and are represented as $2\alpha_3$. The remaining trransition has a probability of 1/3 and is represented as α_3.

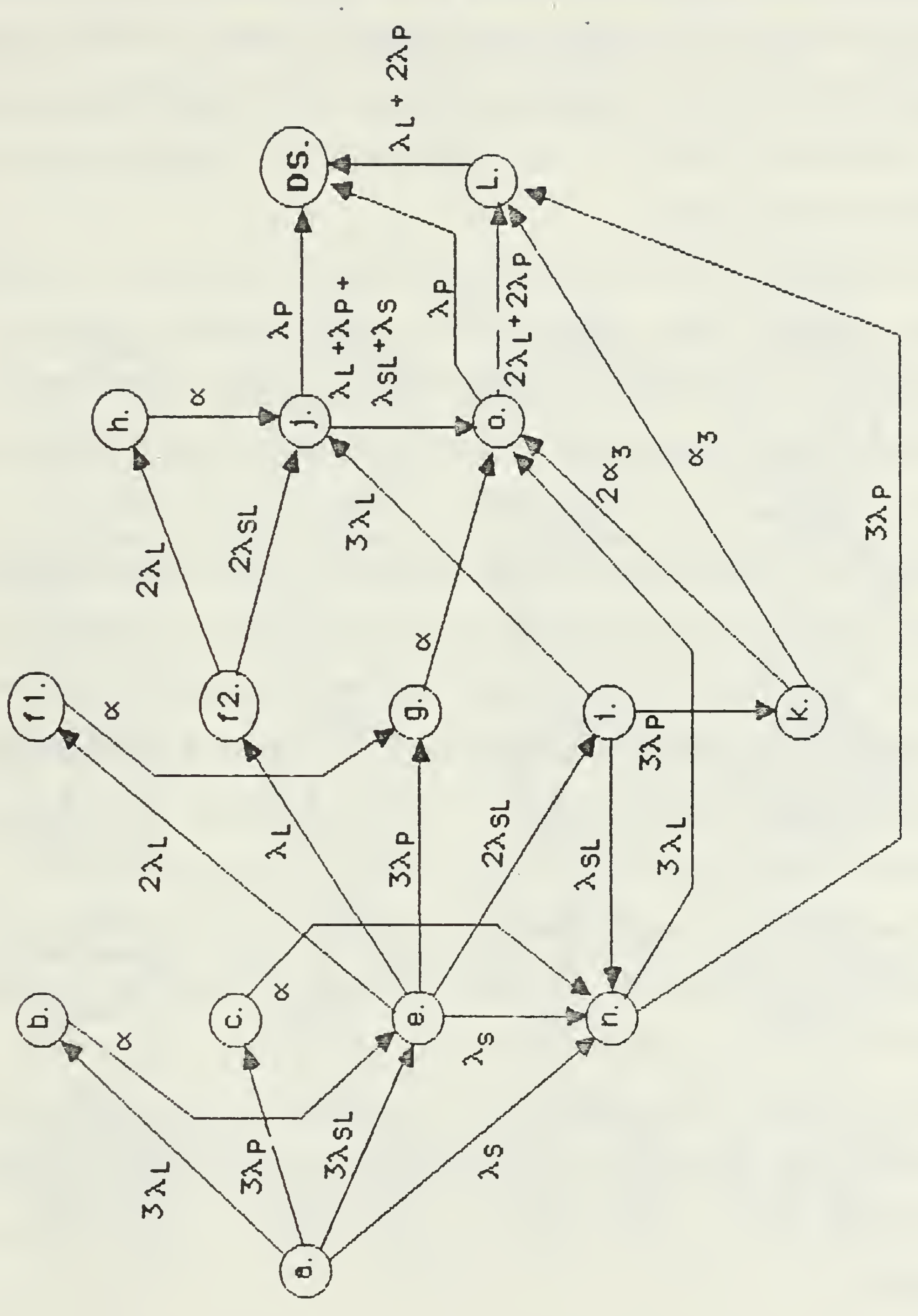


Figure 6-4 Reliability Model for the Four Processor SIR Architecture

B. THE FIVE PROCESSOR CASE

The process described in section A of this chapter can be extended to architectures that start with a larger number of spares. The algorithm that is used for selection of a spare to replace a failed component in the active set can be used without change. Only the five processor case will be considered in this paper.

The addition of an extra spare in the architecture will increase the number of fault classes in the system state aggregation categories. The reasons for the increases have already been discussed.

Figure 6-5 shows the set of impossible states that would be good states if the single fault arrival assumption were not applied to the active core components. This set of impossible fault classes cannot occur because the system will reconfigure to an improved fault condition for the active core by use of the recovery algorithm described in section A of this chapter.

The set of possible good states is shown in Figure 6-6. Numbers are used to label the state aggregations for the five processor case because the increase in system complexity makes the use of letters inconvenient.

A semi-Markov model can be constructed using the set of state aggregations in Figure 6-6. This model is shown in Figure 6-7. Because the model for the five processor case is more complex than for the three or four processor cases, the representation of the model requires more room to show all of the aggregation states and transitions. Therefore, Figure 6-7 is spread over several pages. The states in the model refer to those shown in Figure 6-6.

In several cases, the position of the fault in a single fault class changes the state to which recovery is possible, as has been previously discussed.

The five processor case contains quite a few more states than for the four processor case. In one case the state aggregation was split into two seperate states (50 and 51) in order to more clearly show the differences in recovery possibilities.

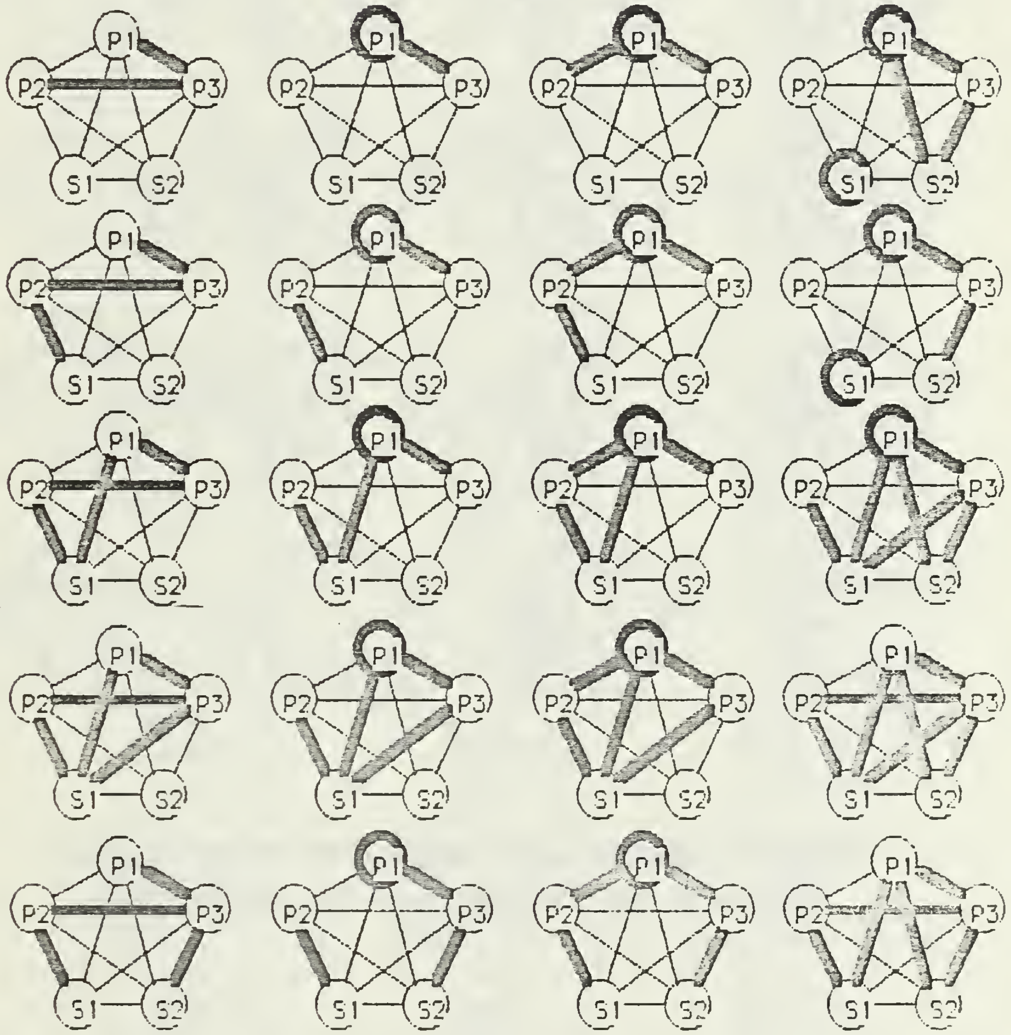


Figure 6-5 Impossible (Good) System States For The Five Processor SIR Architecture

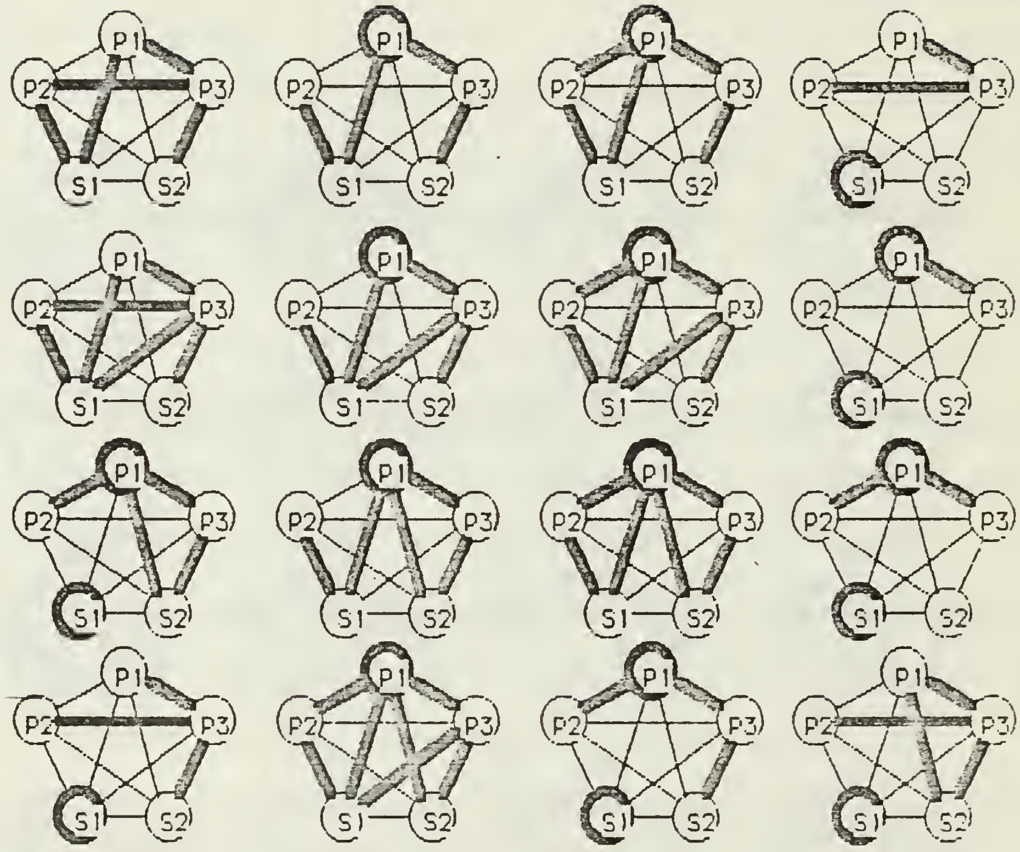


Figure 6-5 (Continued) Impossible (Good) System States For The Five Processor SIR Architecture

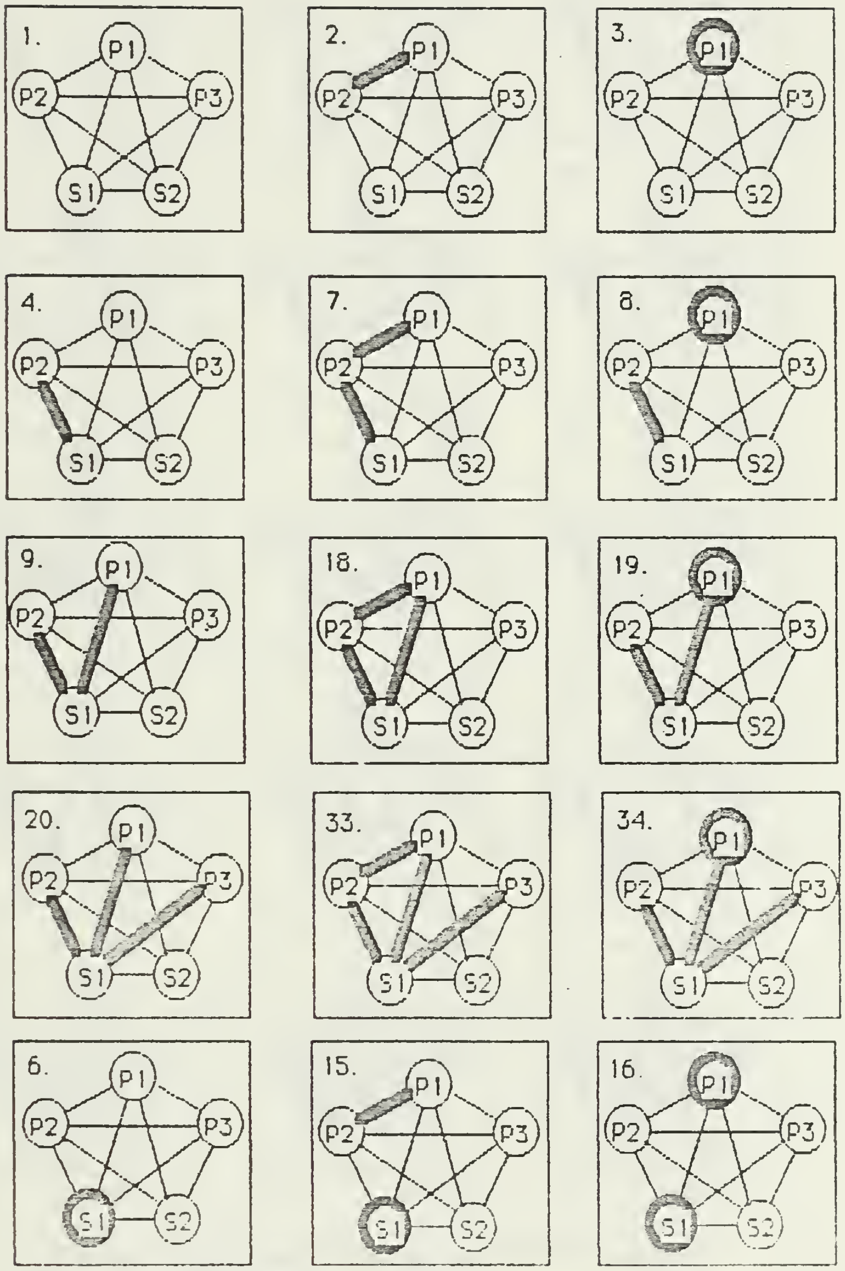


Figure 6-6 Good System States for the Five Processor SIR Architecture

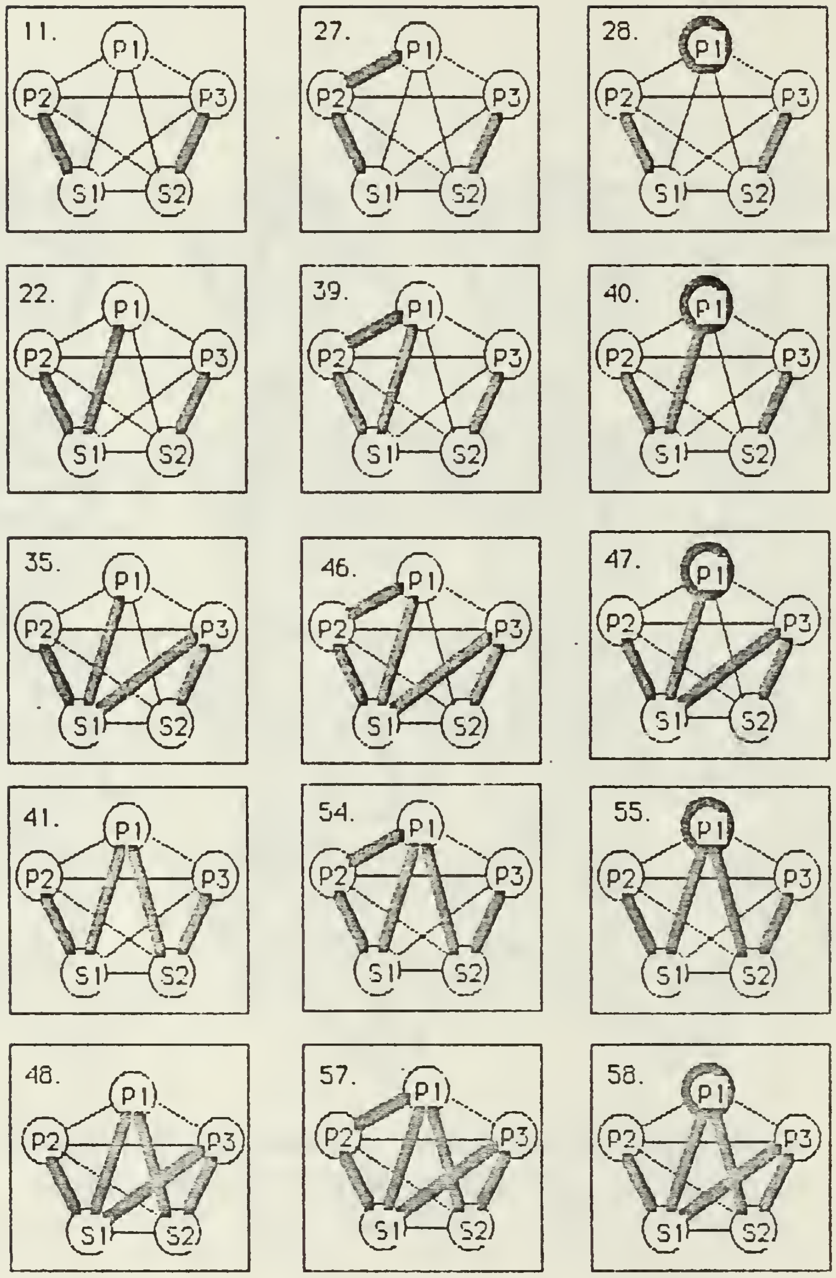


Figure 6-6 (Continued) Good System States for the Five Processor SIR Architecture

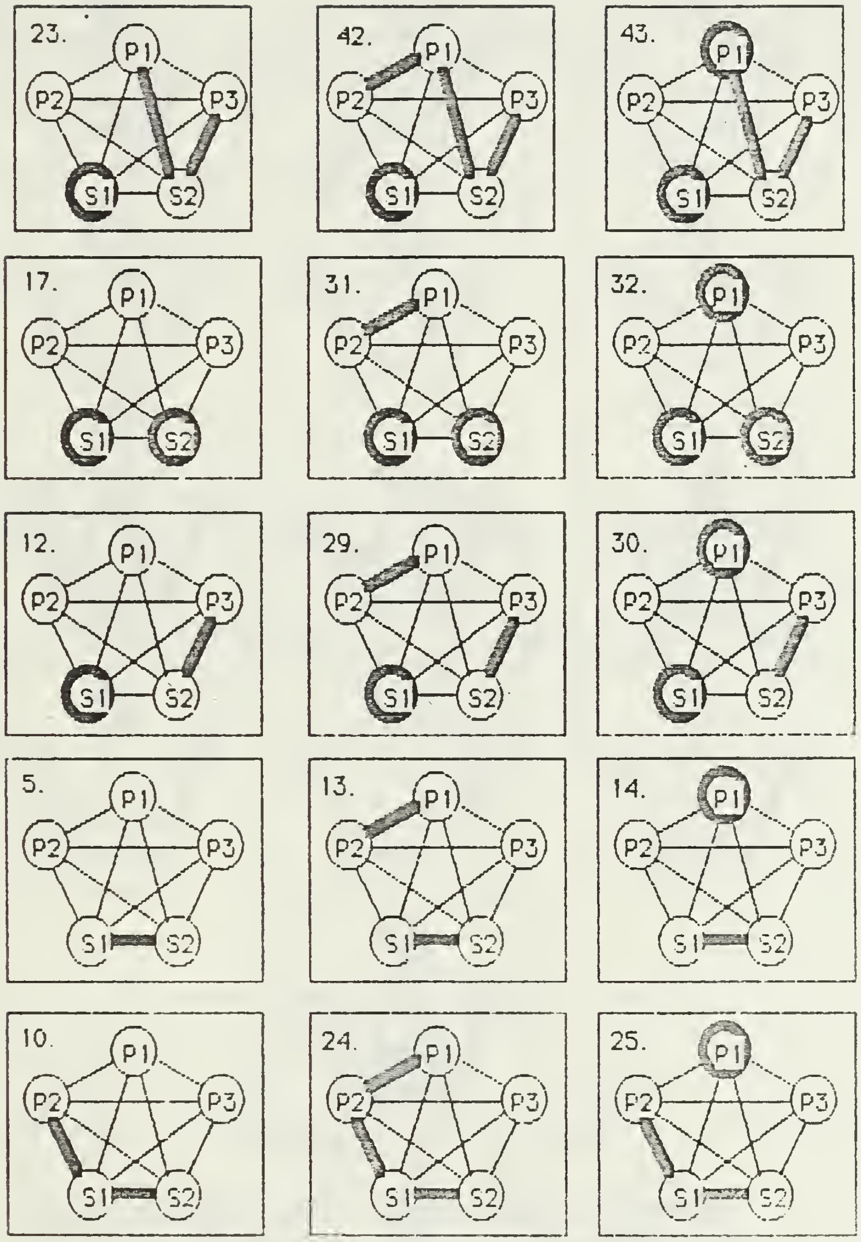


Figure 6-6 (Continued) Good System States for the Five Processor SIR Architecture

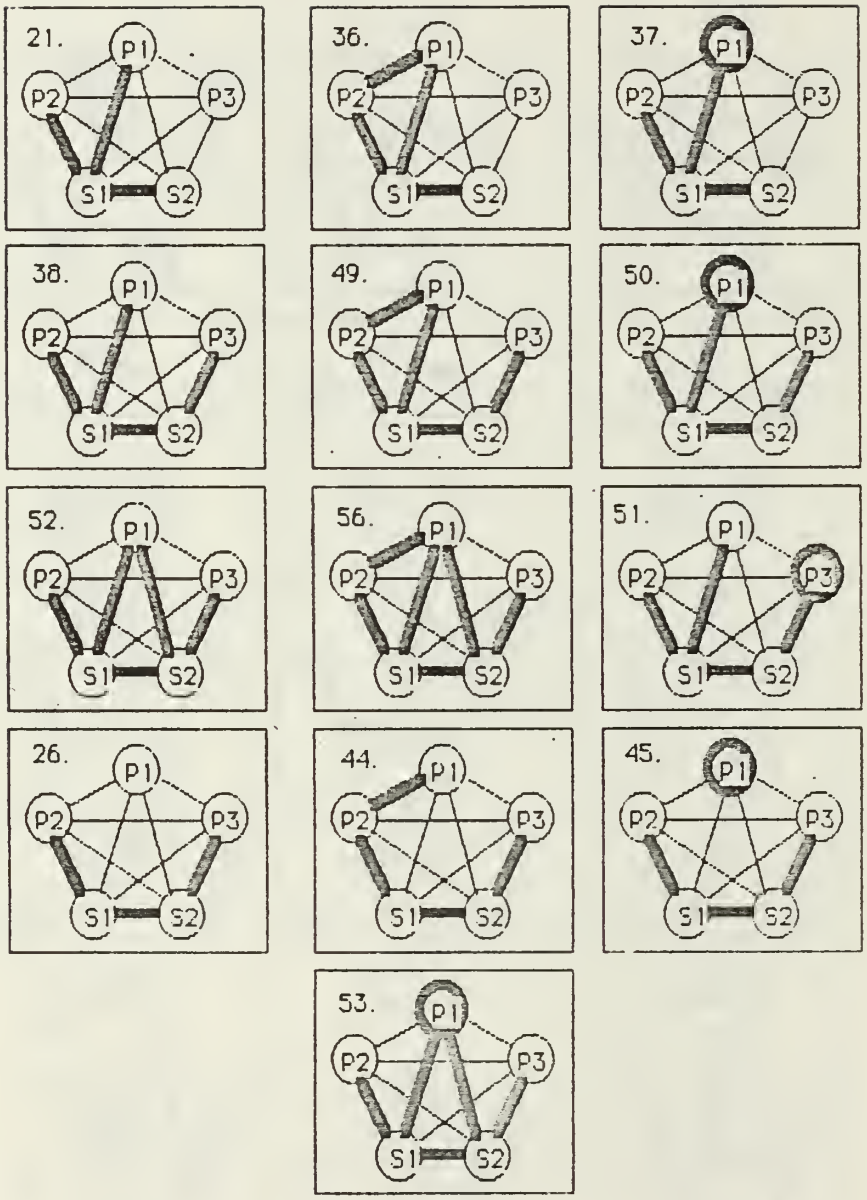


Figure 6-6 (Continued) Good System States for the Five Processor SIR Architecture

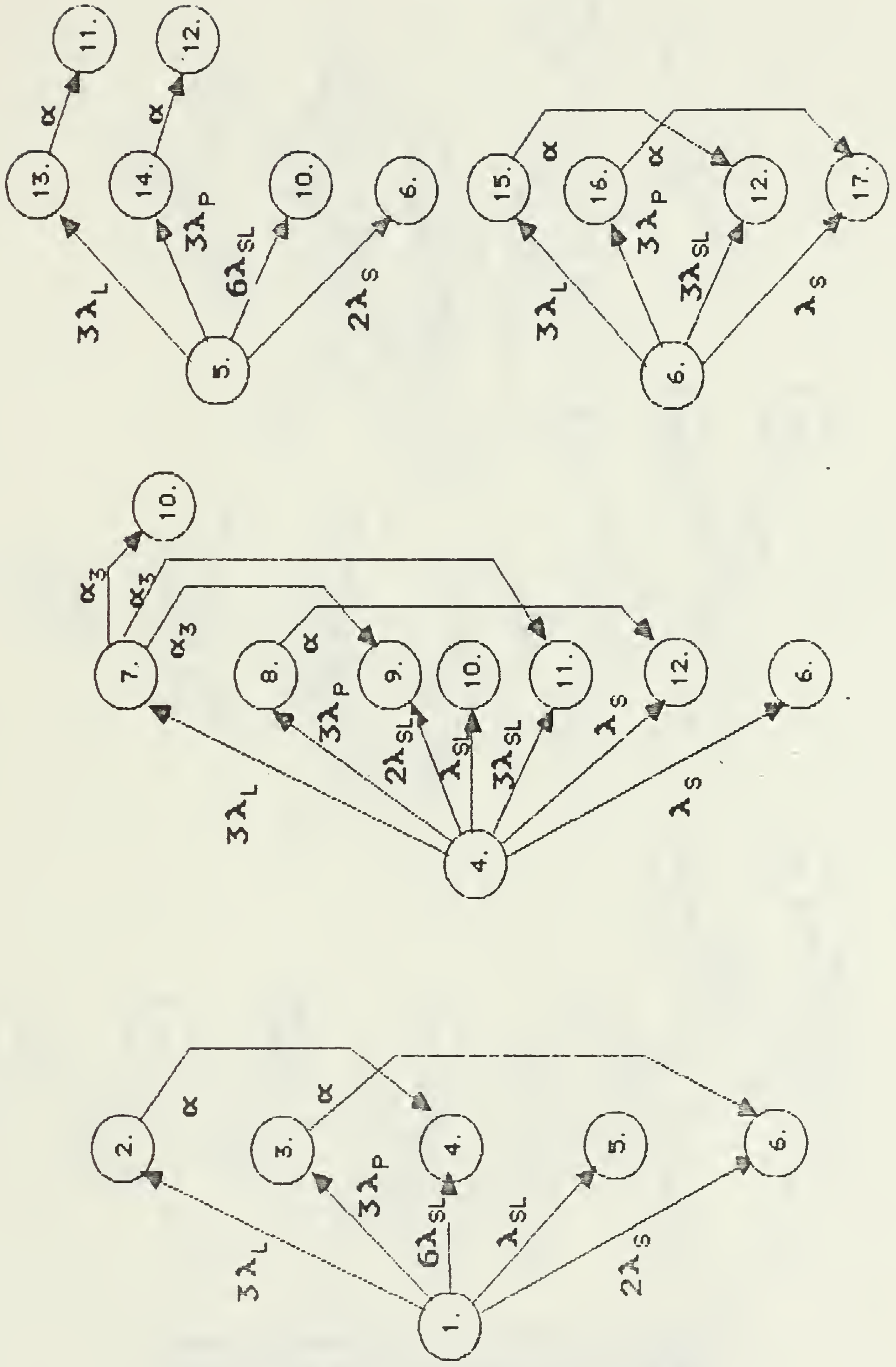


Figure 6-7 Reliability Model for
the Five Processor SIR Archltecure

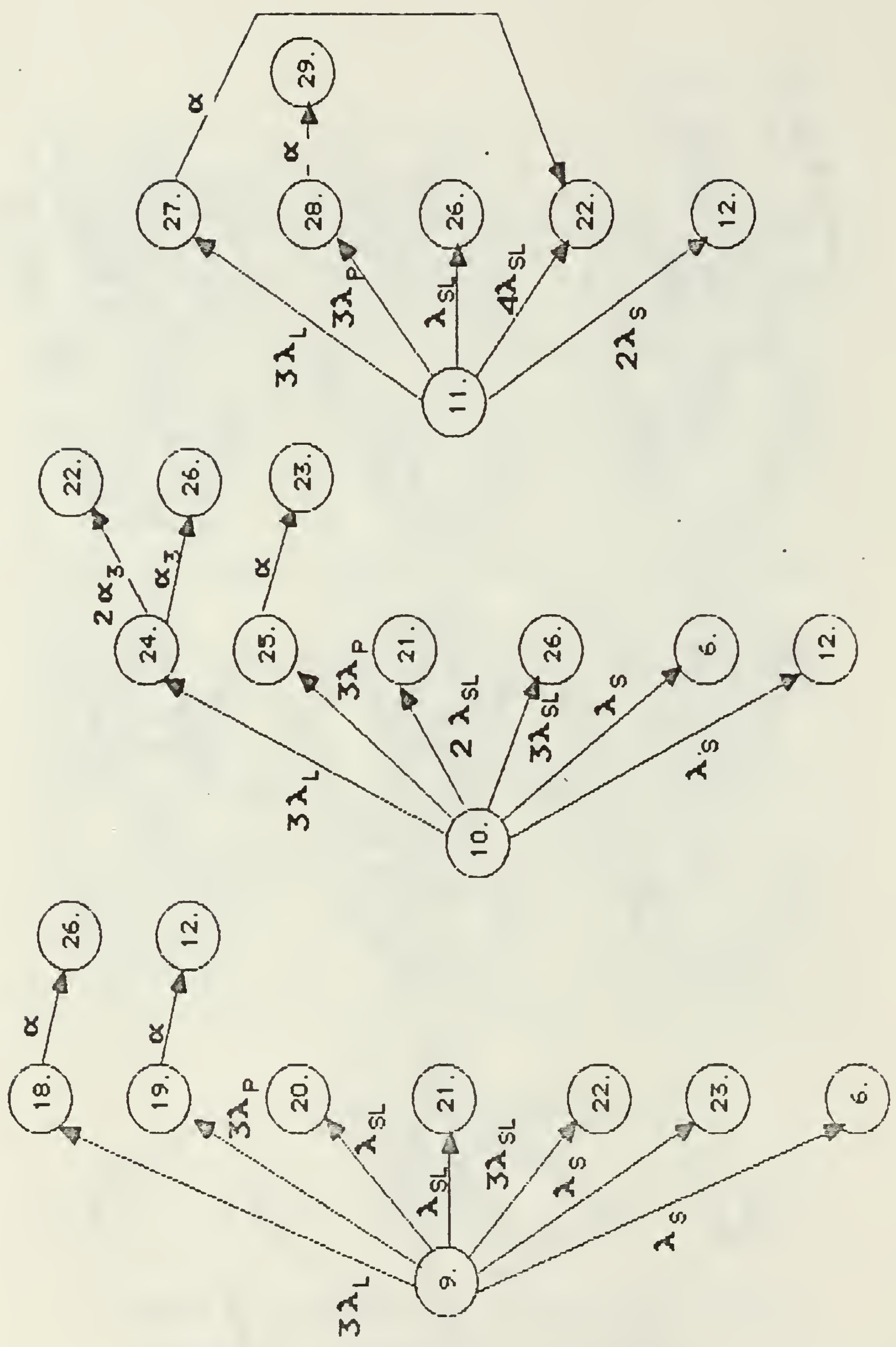


Figure 6-7 (Cont'd) Reliability Model for the Five Processor SIR Architecure

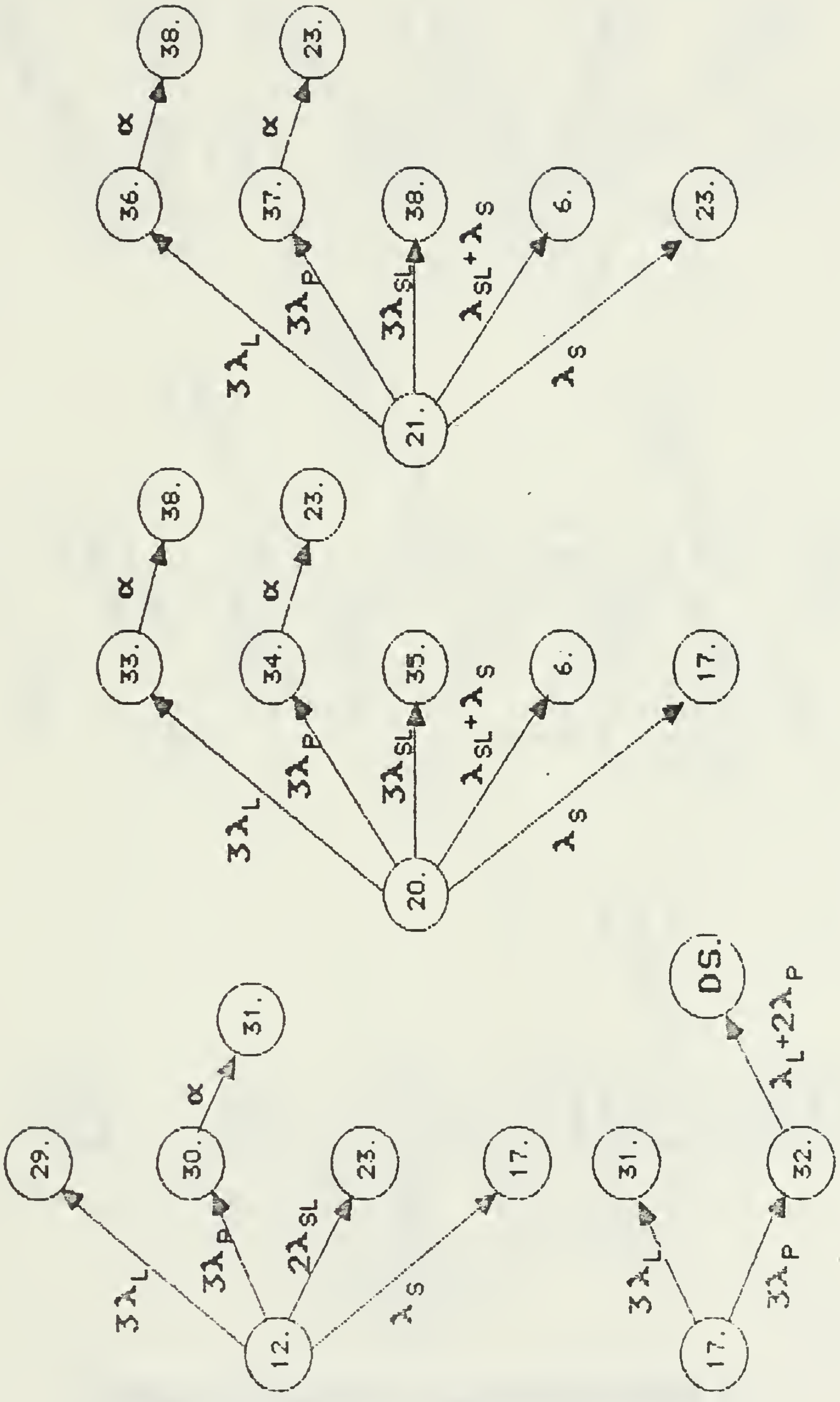


Figure 6-7 (cont'd) Reliability Model for the Five Processor SIR Architecture

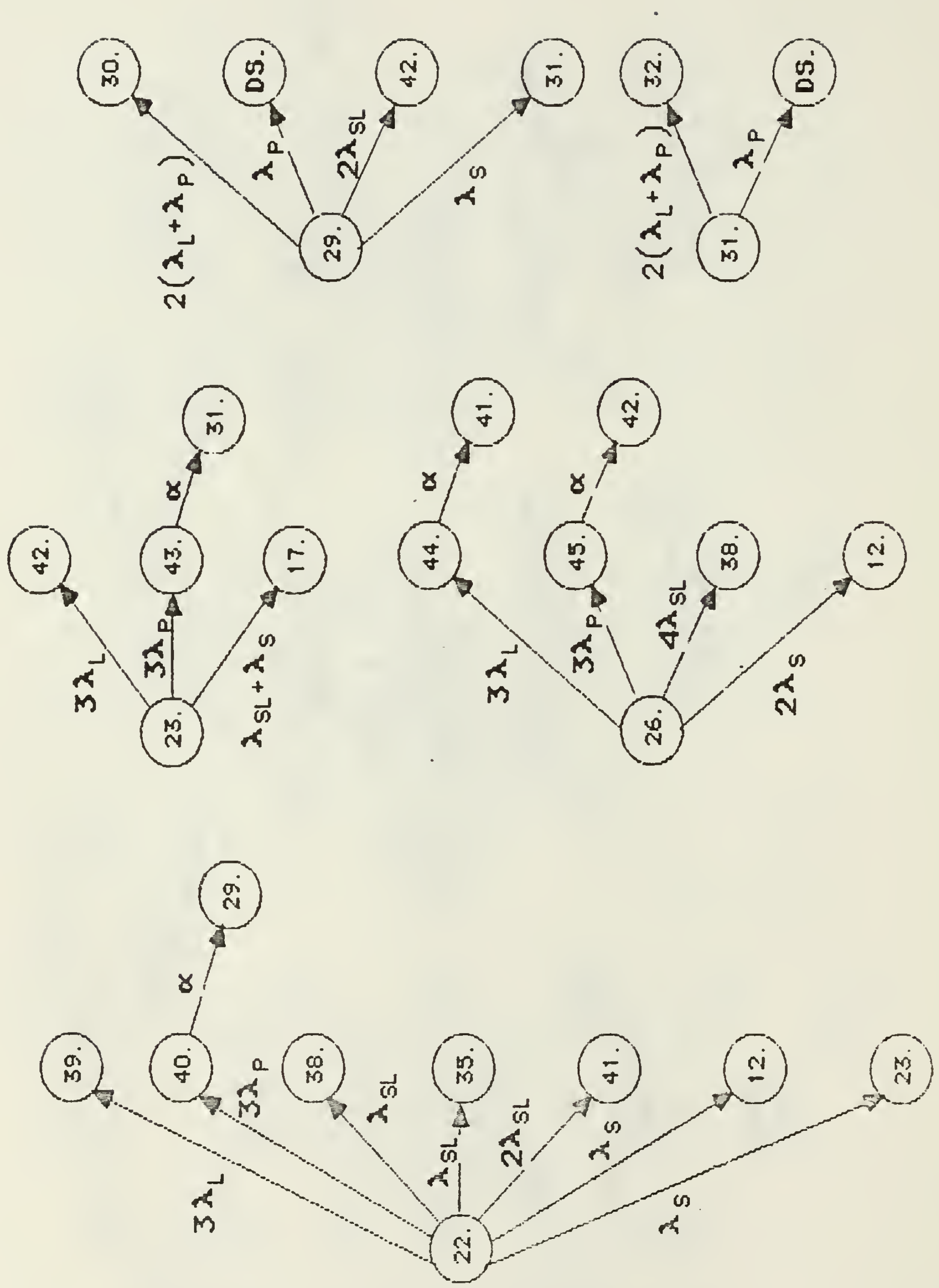


Figure 6-7 (cont'd) Reliability Model for the Five Processor SIR Architecture

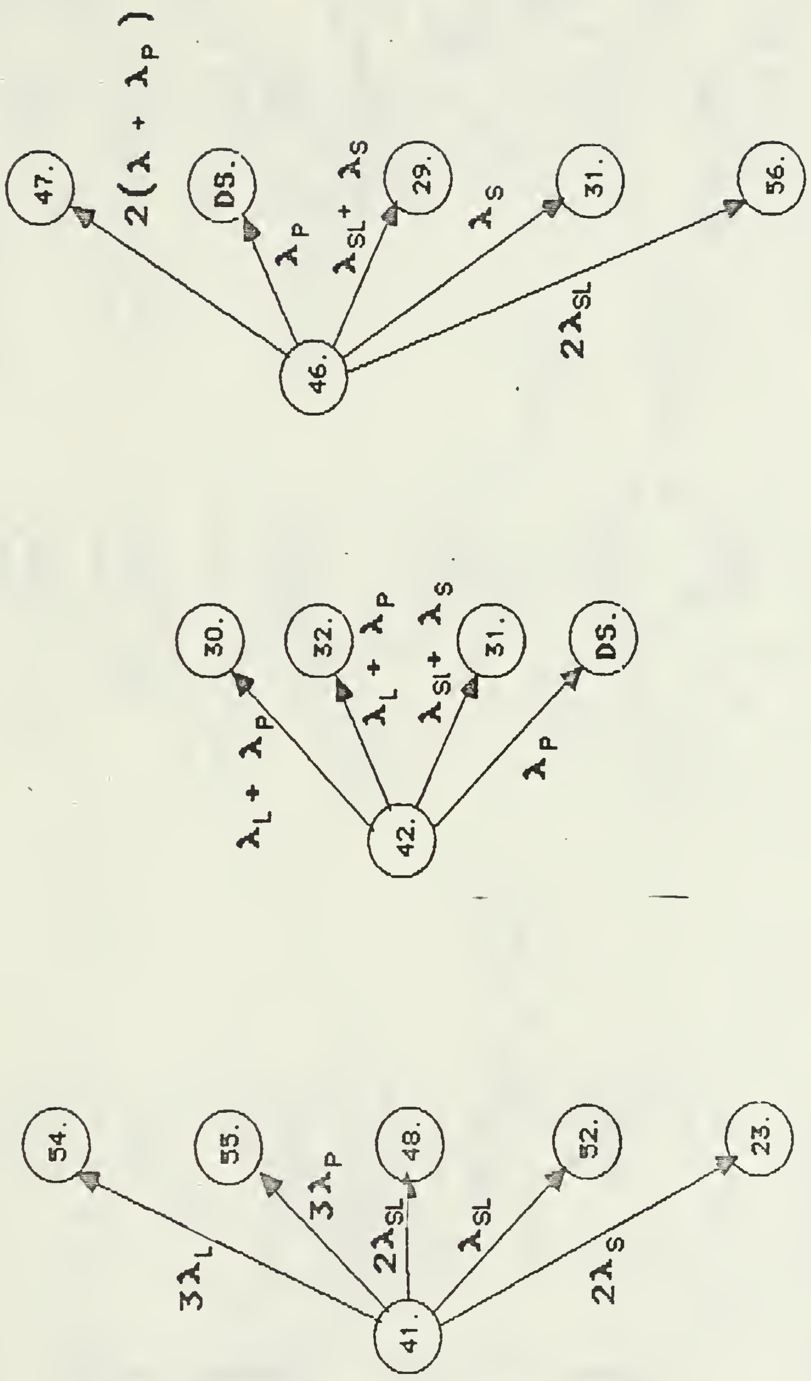


Figure 6-7 (Cont'd) Reliability Model for the Five Processor SIR Architecure

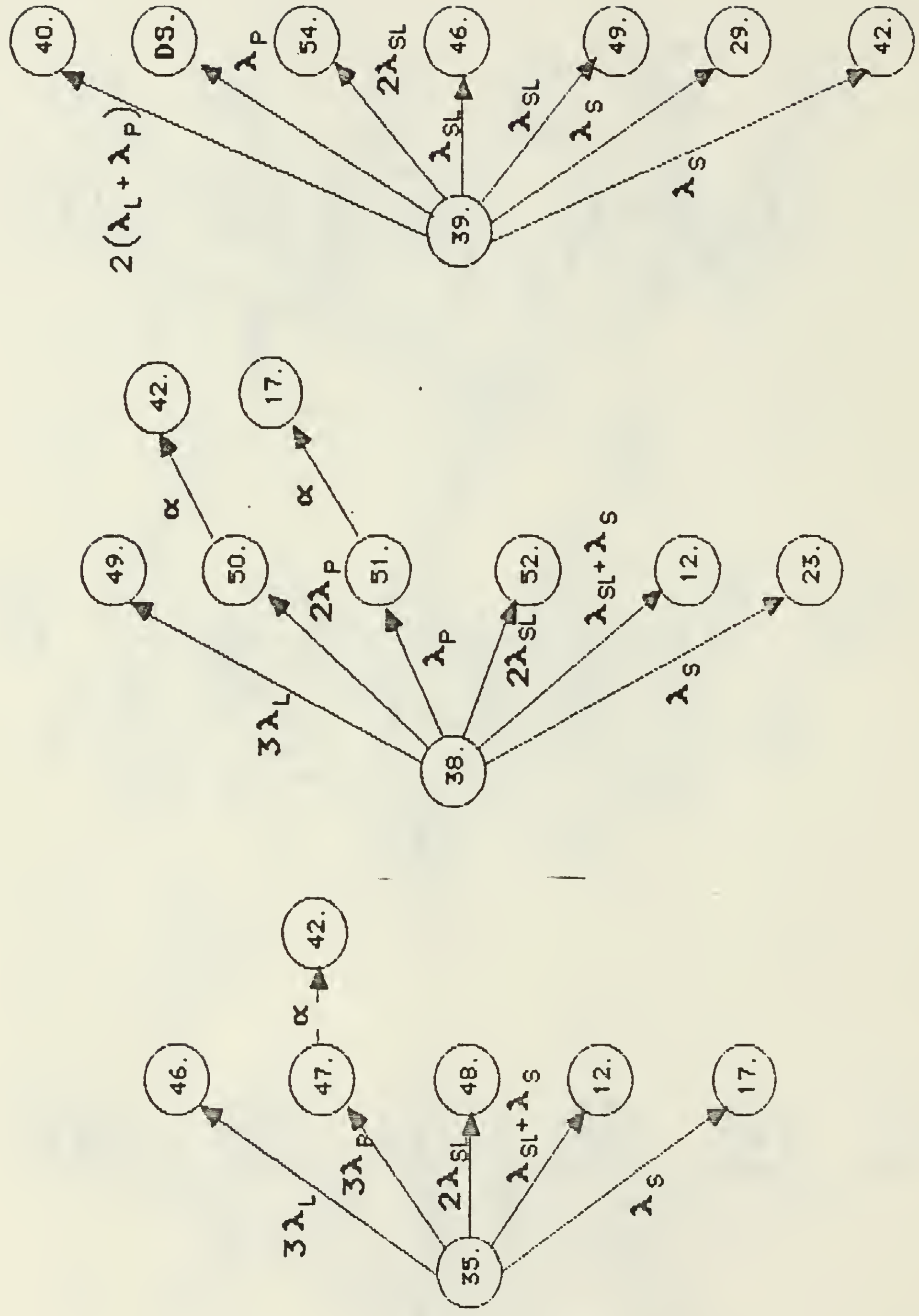


Figure 6-7 (Cont'd) Reliability Model for the Five Processor SIR Architecure

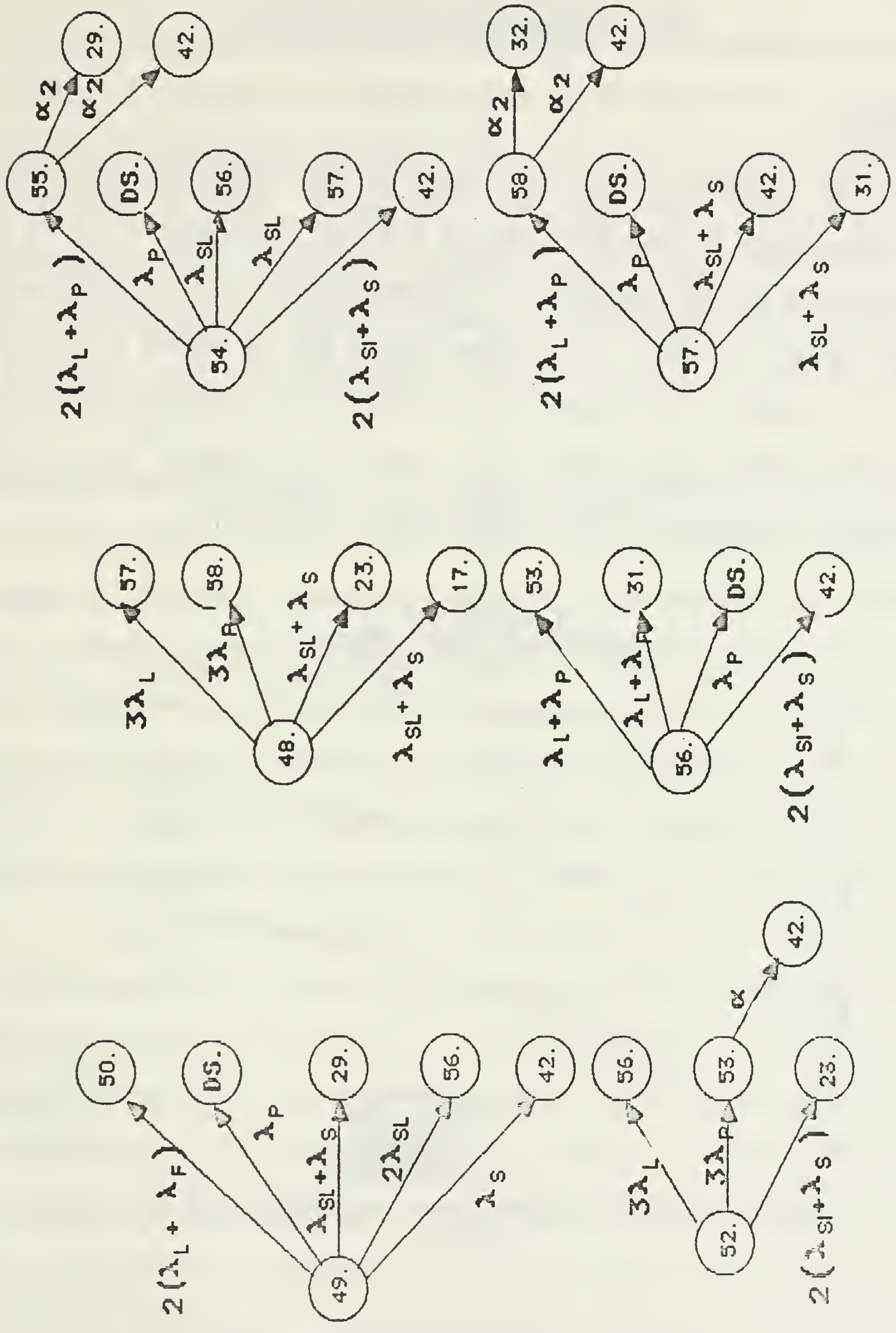


Figure 6-7 (Cont'd) Reliability Model for the Five Processor SIR Architecure

VII. RESULTS AND CONCLUSIONS

A. RESULTS

The results of applying the SURE Semi-Markov analysis program to the reliability models developed in Chapters V and VI are shown in Figure 7-1. The scale used for the probability of system failure axis is logarithmic so that the results of each of the three models can be shown in the same figure. The graph was constructed by plotting 15 equidistant points for each model, varying the mission time from 1 to 15 hours. Cubic spline interpolation was used to estimate the shape of the curve.

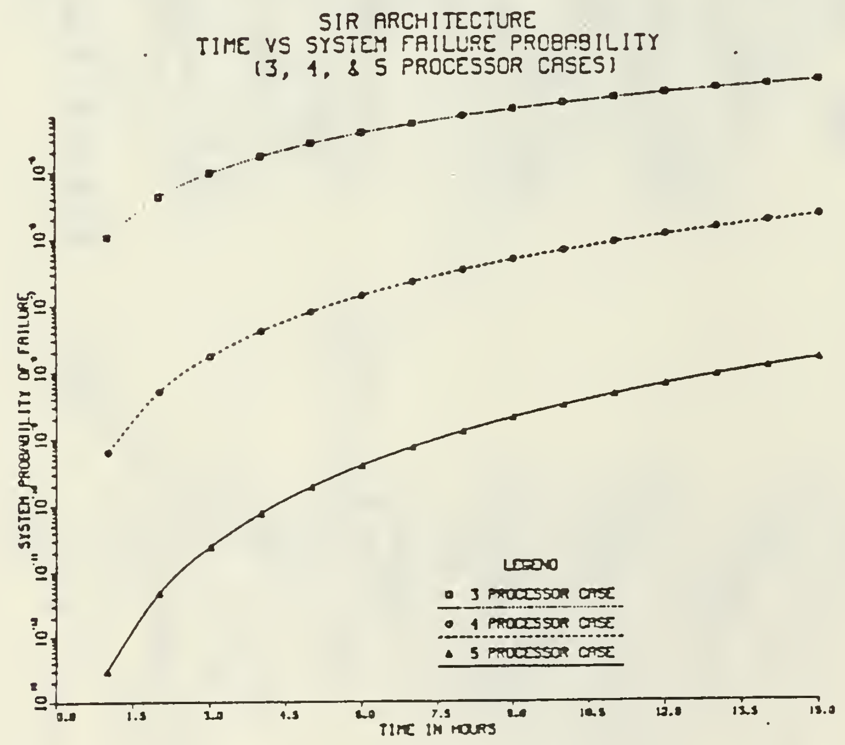


Figure 7-1 Reliability Model Results

The graphs in the figure show the improvement in reliability achieved by the addition of spares to the basic three processor case of the SIR architecture. The mission time that is required by the FAA for reliability calculations is 10 hours. Only the 5 processor case meets the FAA requirement for a 10^{-9} probability of system catastrophic failure for the 10 hour mission time. (The exact numbers produced by the SURE program place the bounds on the failure rate to within 3.07550×10^{-9} and 3.13041×10^{-9}.)

The model that is developed in this paper can be extended to cases of the SIR architecture that use more than 2 spares. The algorithm that searchs for an appropriate spare (discussed in Chapter V) still applies for the case of the six processor architecture. As the number of spares increase however, there are two factors that combine to limit the increase in reliability that can be achieved by the addition of more spares. The first factor is the increased time required to exhaustively search the system for an optimum configuration for a given set of faults. The reason that the search time becomes critical stims from the topological richness of the node interconnections for the SIR architecture, and the fact that only two processors are active in the search for a third good processor (and the links connecting it with the two active processors). The two active processors that comprise the active core for the search can be circulated through the set of nodes until a good spare is found that has links to both the active nodes. The circulating process is performed by deactivating one of the nodes in the original set of two, substituting the newly activated spare in its place, and proceding in the search with the new set of two search nodes. This requires a complete communication of the system state to the newly activated node prior to deactivation the node selected for elimination from the active core. The newly activated node must also be tested to establish that it is fault free prior to deactivating the node in the original set of two. At some point, the time limit of the control problem will require that the search be stopped. Although the search process could be continued at the conclusion of the next control cycle, the introduction of the extra time invalidates the single fault arrival and reconfiguration assumption that has been made for the models developed in this paper. The model would then require a larger number of states, in fact the number of additional states required would be the states listed as good but impossible by the single fault arrival and reconfiguration assumption.

A second reason that increasing the number of spares fails to improve the reliability is that the complexity of the link components increases as nodes are added to the system. The models developed in this paper all used a standard link circuit that supports a 6 node system. There should actually be some improvement in the numbers that are used in the 3 and four processor cases (the 5 procesor case remains unchanged because the number of flip flops required does not change). For systems larger than 6 nodes, a extra flip flops and AND gates are required at each end of the link. This will increase the failure rate of the link component although the rate will still be less than that of the node. (The node failure rate is changed by the addition of 2 AND gates for each additional node to the system.)

It should be noted that the reliability models that are developed in this paper are based on a visual analysis of a semi-Markov state representation of the arrival of faults and recovery transitions in the SIR system. The model becomes very complex as spares are added to the system. A conjecture was made in Chapter VII that signifacantly reduces the number of states that are necessary for the semi-Markov model to capture all nonredundant state information. There was not enough time remaining to completely prove the conjecture, but an assumption of its correctness allowed a visual analysis of the five processor case of the SIR architecture (2 spares). Without use of this conjecture, the number of states necessary in order to completely specify all state information would make a visual analysis unworkable.

The conversion of the SURE program code to operate on the IBM PC-AT was not completed in time for use in this paper. The calculations graphically displayed in Figure 7-1 were performed with a version of SURE that runs on a VAX 11/780 minicomputer under the VMS operating system. The graphics package of the system is not operable at NPS so the numbers that were produced by SURE for the three SIR reliability models were plotted using DISSPLA, a collection of Fortran plotting routines installed on the IBM 3033 mainframe computer resident at NPS.

B. RECOMMENDATIONS

The conjecture presented in Chapter VII should be proven. The formal statement of the conjecture made in Chapter VII should be of assistance in the proof. The state aggregations that are possible by use of the conjecture still results in a reliability model that grows exponentially in complexity as spares are added to the SIR system. Visual analysis would be of limited benefit for systems containing more than 2 spares. A computer algorithm should be constructed to automate the analysis procedures that are graphically displayed for the five processor case of the SIR architecture.

Such an automated system for applying the results of the conjecture would, in conjunction with the SURE semi-Markov analysis program, provide a very powerfull tool for the analysis of very complex systems.

There are several changes that can be made to the architecture of the SIR system that could improve the system failure rates predicted by the model presented in this paper. Contributions to component reliabilities made by component subsystems indicate that a prime target for improvement is the memory that is used in the node's computer. The failure rate of this subsystem is an order of magnitude higher than for the remaining node components. The use of such techniques as Hamming codes in the memory element should be investigated. This technique of course requires that more bits be included in the memory word which leads to greater complexity for the memory. The Hamming codes allow a particular memory word to operated acceptably after the arrival of a fault to the circuitry that contains that word. This means that the memory is fault tolerant for a whole class of faults (1 fault per word for a Hamming code that detects 2 faults and corrects 1 fault). There are obviously trades offs that have to be analyzed.

Another place where the node could be changed is in the design of the interface between the SIR node's microprocessor and the node's interstage. The custom slave processor mode of operation was selected to control the interstage and the communications protocol operating between the microprocessor and the interstage. This required a relatively complex controller element for decoding the commands sent by the microprocessor. The ROM used in the designed interstage controller contributed significantly to the failure rate of the interstage subcomponent of the SIR node. A peripheral interface between the microprocessor and the interstage should be explored as a possible reduction of required complexity in the interstage controller.

Finally, gains could also be made in the manner in which the internode communications are performed in the SIR network. The design using seperate clocks and a 5 register interstage could be reduced by a more elaborate protocol between the nodes. A system of flags could be managed in software resident in the host microprocessor and the SIR nodes interprocessor communication could share the microprocessor ports that are reserved exclusively for external communication in the present design. These approaches would result in a slower exchange of information between the SIR nodes, but the decrease in hardware complexity may result in increased reliability for the system. (Changes in the SIR node intercommunication hardware could require a modification to the model developed in this paper.)

LIST OF REFERENCES

1. Abbott, Larry W., An Ultra Reliable Computer Communication Network - The Dispersed Sensor Processing Mesh, D.Engr. Dissertation, University of Kansas, 1984.

2. Szelai, Kenneth J., and others, Digital Fly By Wire Flight Control Validation Experience, NASA Technical Manual 72860, December 1978.

3. Rediess, Herman A., and Szelai, Kenneth J., Status and Trends on Active Control Technology, NASA paper presented at NASA/University Conference on Aeronautics, Lawrence, Kansas, October 23-24, 1974.

4. Abbott, Larry W., Operational Characteristics of the Dispersed Sensor Processor Mesh, IEEE/AIAA 5th Digital Avionics Systems Conference, Seattle, Washington, October 31- November 3, 1983.

5. Abbott, Larry W., Test Experience on an Ultrareliable Computer Communication Network, IEEE/AIAA 6th Digital Avionics Systems Conference, Baltimore, Maryland, December 3-6, 1984.

6. Abbott, Larry W., A Synergistic Fault Tolerant Computer Design for an N-Version Programming Element, Naval Postgraduate School, Monterey, California, November 1984.

7. Spurlock, Virgil K., Design and Simulation of an Ultra Reliable Fault Tolerant Computing System Voter and Interstage, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1986.

8. Smith, T. Basil, Fault Tolerant Processor Concepts and Operation, CSDL-P-1727, Charles Stark Draper Laboratory, Cambridge, Massachusetts, May 1, 1983.

9. Lala, Parag K., Fault Tolerant & Fault Testable Hardware Design, pp. 69 - 134, Prentice/Hall International, 1985.

10. Butler, Ricky W., The SURE Reliability Analysis Program (Draft 8), NASA Technical Memorandum, Langley Research Center, Hampton, Virginia, September 1985.

11. White, Allan L., Upper and Lower Bounds for Semi-Markov Reliability Models of Reconfigurable Systems, NASA CR-172340, 1984.

12. Lee, Larry D., Reliability Bounds for Fault Tolerant Systems with Competing Responses to Component Failures, NASA TP-2409, 1985.

13. Siewiorek, Daniel P. and McCluskey, Edward J., "An Iterative Cell Switch Design for Hybrid Redundancy," IEEE Transactions on Computers, Vol C-22, No. 3, March 1973.

14. Chen,L. and Avizienis, A., "N-Version Programming: a Fault-Tolerant Approach to Reliability of Software Operation," Proc. Int. Symp. Fault-Tolerant Computing, pp. 3-9, 1978.

15. U.S. Department of Defense, MIL-HDBK-217B, Military Standardization Handbook: Reliability Prediction of Electronic Equipment, Washington, DC, 1976.

16. Siewiorek, Daniel P. and Swarz, Robert S., The Theory and Practice of Reliable System Design, pp. 201-302 and Appendix D, Digital Press, 1982.

17. Fairchild Camera & Instrument Company, Fairchild TTL Databook, Mountainview, California, 1978.

18. National Semiconductor Corporation, Reliability Data - 32000 Family (I8504002 and I8504003), Santa Clara, California.

19. Intel Corporation, Reliability Report (RR-18 Update): HMOS Reliability, pp. 7-8, August 1979.

20. Wensley, J. H., and others, "SIFT: Design and analysis of a Fault-Tolerant Computer for Aircraft Control," Proc. IEEE, pp. 1240 - 1255, October 1978.

21. Brock, L.D. and Goodman, H.A., Reliability Analysis of the F-8 Digital Fly-by-Wire System, NASA Contractor Report 163110, p. 116, October 1981.

22. Ingle, Ashok D. and Siewiorek, Daniel P., "A Reliability Model fro Various Switch Designs in Hybrid Redundancy," IEEE Trans on Computers, Vol C-25, No. 2, February 1976.

23. Siewiorek, Daniel P. and McCluskey, Edward J., "Switch Complexity in Systems with Hybrid Redundancy, IEEE Trans on Computers," Vol C-22, No. 3, March 1973.

24. White, Allan D., Synthetic Bounds for Semi-Markov Reliability Models, NASA CR-XXXXX, 1985.

BIBLIOGRAPHY

Proceedings IEEE/AIAA 5th Digital Systems Conference, Seattle, Washington, October 31 - November 3, 1983.

Shooman, Martin L., Probabilistic Reliability: An Engineering Approach, McGraw-Hill Inc, New York, 1968.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
2.	Chairman, Department of Electrical and Computer Engineering, Code 62 Naval Postgraduate School Monterey, California 93943-5000	2
3.	Professor Larry Abbott, Code 62At Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	5
4.	Professor Jon T. Butler, Code 62Uj Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	2
5.	Captain Ronald J. Nelson, USA 40 Apple Tree Lane Hillsdale, New Jersey 07642	5
6.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2

Thesis
N36665
c.1

220335

Nelson
The Synergistically Integrated Reliability architecture: reliability analysis of an ultra-reliable fault tolerant computer design.