THE JOHNS HOPKINS UNIVERSITY

ELECTRICAL ENGINEERING DEPARTMENT

Final Report

Intermittent/Transient Fault Phenomena
in Digital Systems

NASA Research Grant NSG 1265

by

Gerald M. Masson
Electrical Engineering Department
The Johns Hopkins University
Baltimore, Maryland 21218
(301) 338-7013

Submitted to

National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23655

Final Report

Intermittent/Transient Fault Phenomena in Digital Systems

NASA Research Grant NSG 1265

by

Gerald M. Masson
Electrical Engineering Department
The Johns Hopkins University
Baltimore, Maryland 21218
(301) 338-7013

Submitted to

National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23655

July, 1977

Project Participants

Name: Vinod K. Agarwal
Date & Place of Birth: 4/30/52 Mathura, India
Education: BE (Honors) 1973 Birla Institute of Tech. & Sci. Pilani, India
MS 1974 University of Pittsburgh, Pittsburgh, Pennsylvania
Ph.D Candidate
Interest: Fault interrelationships; resolvability of multiple fault situations; algorithms (network level) for resolution; testing theory.

Name: Robert E. Glaser
Date & Place of Birth: 3/12/54 Baltimore, Maryland
Education: BES (General and Departmental Honors) 1975 The Johns Hopkins University
Ph.D Candidate
Interest: Reliability of networks and systems; strategies and algorithms for recovery, roll-back, survivability and tolerance.

Name: Clifford L. Greenblatt
Date & Place of Birth: 6/16/52 Baltimore, Maryland
Education: BS (Honors) 1974 Case Western Reserve University
Ph.D Candidate
Interest: Causes and effects of intermittent/transient faults (component and network levels); models and parameters.

Name: Sivanarayana Mallela
Date & Place of Birth: 8/4/52 Madras, India
Education: B. Tech 1974 Indian Institute of Technology Bombay, India
MS 1976 State University of New York at Buffalo
Ph.D Candidate
Interest: Analysis (detection and location) of transient/ Intermittent faults (network level); reliability of testing procedures; goodness factors and bounds.

Name: Shinji Nakamura
Date & Place of Birth: 2/27/44 Tokyo, Japan
Education: BS 1966 Gakushuin University (Physics)
MS 1969 Gakushuin University (Physics)
Ph.D Candidate
Interest: Equivalence of fault situations (component and network level); functional approaches to coverage; transformations.

I. Preliminaries

1.1 Introduction

A study of intermittent/transient faults (I/T faults) in digital systems requires a confrontation with a multitude of issues. It soon becomes clear, moreover, that in the period of one year, not all the issues can be considered. We have, therefore, taken one approach - of many possible approaches - to this area, and have reached a point which we feel indicates reasonable progress. In the following sections we will detail the state of our investigation, and we will try to indicate the strong and weak points of our current position with this problem area. This section will serve as an overview of these results.

The ultimate goal of this study is to perform survivability evaluation of digital systems for I/T faults. The framework within which this evaluation is to be performed is generically described by the CARE II approach. However, survivability has heretofore been addressed primary from the point of view of long-term or uniform survivability. The explicit consideration of I/T faults requires instead the consideration of interval survivability. Interval survivability is a measure of the probability of the system surviving a fixed time interval of I/T fault activity in the sense that at the end of this interval the system can continue to operate (perhaps at the cost of some recovery operations) in an acceptable (but perhaps degraded) mode.

We have not, at this time, implemented a means of evaluating such a survivability number. It is clear, however, that the task of doing so is at least as complicated as that of writing the actual CARE II program. Moreover, to do so is beyond the scope of the work reported here, and would, in fact, overlap significantly with the development of CARE III. However, crucial to any such evaluation is the capability to detect and diagnose I/T faults. (This is sometimes referred to as the "D" and "I" of the DIR function). We have here, then, the motivation for much of the work to be reported in the following: the chief results to be reported will consist of the development of methodology for detecting and diagnosing I/T faults in digital systems. We will show that there are specific bounds and guidelines to this detection and diagnosis for I/T faults in general which must be taken into account in any interval survivability evaluation. These bounds and guidelines are detailed such that they can be incorporated into any testing and diagnosing methodology.

In addition we report on the status of an experimental attempt to determine the effect of various physical I/T faults (for example, those which might be determined to be important from the 1977 Learjet Experiments) on a type of module which could be a part of a general computer system characterized by the CARE II model. The goal here is to functionally determine the effect of various I/T faults so that the methodologies for testing and diagnosing I/T faults can be fully exploited by

taking into account the detailed test set requirements.

All of this, then leads to further refinements in interval survivability evaluation for digital systems in the presence of I/T faults.

2. Interval Survivability

Our objective is to enhance and evaluate the survivability of a computer system by use of fault tolerant techniques. Survivability is evaluated as the probability that the system will survive until a given time, t. The concept of survivability is extended by the inclusion of degraded modes of operation. For example, a system may survive to time τ in the undegraded mode but survive from time τ to time t in a degraded mode.

Various methods of increasing the fault tolerance of a computer system have been proposed in the literature. The method considered here is the use of standby sparing. The computer system is assumed to be partitioned into several stages. Within each stage are a number of identical units. A certain number of the units in a given stage are in active operation while several other units serve as standby units. In the event that an active unit fails, a spare unit is tested and then switched in to replace the bad active unit.

The principles of detection and location of faults and of recovery of the program in progress are essential to a gracefully degrading, standby sparing computing system. A system of both hardware and software detectors are provided to detect the presence of any faults and then to locate the fault to within a certain category. A strategy for recovering from faults in each category must be provided. When the presence of a fault is detected, further propagation of errors

is prevented by stopping the program in progress and holding information needed for recovery. After the fault is isolated to within a certain category and a spare unit switched in to replace the faulty unit (no spare unit is switched in for the cases of transient faults) a recovery strategy is then carried out to restore the program in progress. A failure to detect or isolate a fault or a failure to restore the program in progress will result in a system failure.

In order to design for fault tolerance, a great deal must be known about the nature of the faults that can occur. A transient fault is generated internal to the affected units; therefore one must be able to identify transient faults so that recovery may be done without replacing any units with a spare unit. An intermittent fault is a failure that is generated within a unit. If a fault is identified as intermittent, it is best to treat the fault as a permanent fault, i.e., replace the unit with a spare unit to avoid the consequences of reoccurance of the intermittency. In order to detect faults as quickly as possible, one must have a good model of the probability of occurance of each type of fault that may occur. Using an accurate fault model, one then designs hardware detectors and software procedures to detect the various faults in an efficient manner. For permanent faults, accurate failure rates are given by the manufacturer. However, additional study is required to model intermittent faults, which are generally

caused by loose wire bonds in integrated circuits, and transient faults, which are generally the result of electromagnetic disturbances generated in a thunderstorm by nearby lightning strikes.

The CARE II program is designed to evaluate the survivability of the gracefully degrading and standby sparing computing system described above. The program derives the probability that the system can survive until time t. Information required for CARE II includes failure rates of the units and rate of occurance of nonrecoverable, transient faults. The CARE II program includes a coverage model which demands a detailed knowledge of what faults may occur, the characteristics of the hardware and software detectors, and probability of recovery given the class of fault and the time elapsed from the occurance of the fault to its detection and to its isolation. Coverage is the part of the fault tolerance design that includes detection, isolation, and recovery from a fault. Once the fault is detected by a given detector, the isolation and recovery procedures follow according to a deterministic path. However, several detectors may be capable of detecting the same fault and which one actually succeeds will then determine the isolation and recovery procedures. Due to the nondeterministic nature of fault detection, CARE II requires categorization of faults into fault classes. Fault classes are chosen such that the

detection of a fault class by the set of detectors is statistically independent with respect to the detectors. To derive coverage coefficients, i.e., the probability of recovering from a fault given that a certain number of spare units must be tested before a good one is found, one must also know the probability of occurance of the various faults. Finally, one must also know the effectiveness of the recovery strategy for a given fault and given time delays included in detection and isolation. Specifically one must provide $r(\tau,\tau')$ where $r(\tau,\tau')$ is the probability for a given type of fault that the program will recover after a delay of τ seconds from fault occurance to fault detection and a delay of τ' seconds from fault detection to fault isolation. CARE II assumes that $r(\tau,\tau') = r'(\tau)+r''(\tau+\tau')$ where $r'(\tau)$ accounts for propagation of errors before the fault is detected and $r''(\tau+\tau')$ accounts simply for total time lost in the running of the program.

The present literature on gracefully degrading and standby sparing computing systems assume a uniform failure rate. The problem of lightning induced fualts brings up the question of interval survivability, that is design and evaluation of the computing system with respect to toleration of a severe transient fault inducing environment over a limited period of time. It will be necessary to modify the CARE II model to account for a change in detection and recovery

procedures for the severe transient interval. The emphasis of the detection procedure must be shifted to detection of the transient faults likely to occur in a thunderstorm. Such a shift may easily be initiated by the detection of electromagnetic disturbances by an external detector provided for such a purpose. The external detector may provide data on the nature of the transient environment so that the optimum fault tolerant strategy may be employed. For example, the fault tolerant strategy should be a function of the severity of each electromagnetic disturbance as it occurs and also the expected length of the severe transient fault inducing interval. If several operations are being carried out at once, those operations that are most affected by the transient environment should be postponed, if possible, or done in a way that is less sensitive to transient faults. To study the modification of gracefully degrading and standby sparing computer systems for interval survivability, one requires an extensive knowledge of the effects of a thunderstorm on the operation of the computing system and a knowledge of the effects of lightning induced transient faults on the various functions of the computing system.

3. Current Status of Module Self-Diagnosis Theory

3.1 Introduction

In this section we will give an overview of the theory of self-diagnosis of digital systems. It should be kept in mind that while our concern is with I/T faults, the majority of this reviewed work deals with permanent faults. However, this work must nevertheless be appreciated as it represents an initial Condition on the results of Section 4. Moreover, from Section 3 we have seen that for survivability evaluation, self-diagnosis is a crucial factor.

In particular, we will consider the design of such systems which will operate in a distributed processing/decentralized control mode. In order for such systems to operate in a fault tolerant enrivonment, it is imperative to incorporate into the design some degree of self-diagnosability.

In this section, a system is considered which can be partitioned into n functional units, or modules, each possessing some degree of intelligence. A major assumption to be made is that each module be completely capable of testing the correctness of other specified modules in the system. Such tests cannot be well defined in the general sense. (Indeed, their generation is the responsibility of the module designer). They may be considered to be strictly hardware, as in the case of dedicated hardware monitor. Similarly, a module may utilize various diagnostic software routines to generate and compare test patterns that will be applied to the module being tested. In the most general sense however, a test may be thought of as any combination of hardware and software which enables a unit to successfully base a conclusion as to the operational state of another module.

A unique test will probably need to be designed for each module based on the fact that a single universal test would not provide adequate fault coverage for each module, due to their functional differences.

It should be noted that the implementation of such tests is not a trivial matter and poses a major obstacle to the design of totally self-diagnosable systems.

The actual diagnosis problem is one of detection and location of all faulty modules that may be present in the system. Once all of the tests in the testing interconnection design have been completed, it is the function of the entire system to detect the presence of any single or multiple faults. A system diagnosis algorithm must be implemented to examine the set of test outcomes and determine if any faults have occurred. A factor which enhances the complexity of the problem is that of the possibility of incorrect test outcomes produced by modules which are themselves faulty.

A basic assumption affecting the testing interconnection of devices is one of generating an upper bound on the allowable number of modules which may become defective at any one time. Depending upon this figure, the testing scheme may be very simple or quite complex. It will be shown that the number of modules in the system and their testing interconnections have a direct dependence upon this upper bound.

The goals of attaining a totally self-dignosable system are two fold. The first is in enabling the system to operate in a fault tolerant environment. Upon the detection of any module failures, the system could conceivably isolate those devices, reconfigure and then recover to resume its processes, all without any external communications. Such performance would be essential

in situations where it would be disasterous for the entire system to crash. The second goal of attaining self-diagnosability would be to ease system maintenance. The detection of any faults in the system could signal an error condition. This in turn could affect a physical replacement of the faulty components, if possible, to update the system to resume correct operation. Thus, in any situation, self-dignosable systems are easily seen to be a major consideration in increasing system reliability.

3.2 SYSTEM MODELS AND THEIR ANALYSIS

Various system models have been proposed to aid in the analysis of diagnosable systems. The basic goal of each of the models is to demonstrate the testing interconnections employed and to deduce the performance characteristics of such schemes. These characteristics may include an upper bound on the number of faults that may occur in the system, while at the same time possess the ability to locate just a single fault or perhaps diagnosis the entire fault situation.

Probably the most well known had been proposed by Preparata [3.7] a decade ago. In this model the system is decomposed into n different subsystems, or modules, each with the capacity to test the correctness of the others. The model itself is a graph-theoretic one in which each of the n modes represents the n modules and a directed edge is included to denote a testing interconnection between two modules. Each of the testing links is represented by b_{ij} in which each unit U_i evaluates unit U_j. The weight associated with each b_{ij} is $a_{ij} = \{0, 1\}$. The test outcomes a_{ij} is a 0 if module U_i is fault free and tests U_j to be also fault free. However, if U_i is fault free and tests U_j to be faulty, then $a_{ij} = 1$. In the situation where the testing module U_i is faulty, its test output could possibly be a 0 or 1, regardless of the actual condition of the module U_j being tested. The testing connection of system can be represented by a connection matrix $C \equiv ||c_{ij}||$ where:

$$c_{ij} = \begin{cases} 1 \text{ if } b_{ij} \text{ exists.} \\ 0 \text{ if } b_{ij} \text{ does not exist.} \end{cases}$$

Once all of the tests in the model have been completed, each a_{ij}

has been assigned a corresponding binary weight. It is from this set of test outcomes, i.e. the system syndrome, that the system diagnostics will be performed.

The system diagnostics may be oriented towards one of two possible approaches. The first is often referred to as one-step t-fault diagnosibility. Is the goal of this method to identify (locate) all of the faulty units in the system, given its syndrome. A necessary constraint is that the number of faulty units does not exceed the upper bound t. Another approach is in viewing the system as being sequentially t-fault diagnosable. Here, it is guaranteed that at least one faulty unit can be detected directly from analysis of the system syndrome. Again, it is assumed that the maximum number of faulty modules does not exceed t. It is obvious that a one-step t-fault diagnosable system is also sequentially t-fault diagnosable. The motivation behind each of these situations should also be clear. In the one-step case, the syndrome is examined to identify each of the faulty units. These units may all be replaced at once, thus enabling the system to once again become fully operational. On the other hand, in the sequentially t-fault diagnosable situation, the syndrome is examined such that only one faulty unit is detected. This faulty unit is then replaced and a new syndrome is generated. This procedure would be reinterated up to t times, until all of the faulty units had been replaced. While one-step t-fault diagnosability is more efficient than sequential t-fault diagnosability, the complexity involved may not warrant the increased performance.

In his paper, Preparata made a few basic, but very important observations in relation to diagnosable systems. The first was in generating a lower bound on the number of units to ensure the system to be diagnosable. That is, given that a system is one-step t-fault diagnosable, then $n \geq 2t+1$. Conversely, if a system of n modules is said to be one-step t-fault diagnosable, then an upper bound on its degree of diagnosability is $t \leq \left\lceil \frac{n-1}{2} \right\rceil$. It should be emphasized that these bounds may not be reached if an inefficient connecting scheme is employed. Another important observation made was in bounding the smallest number of units needed to test another to ensure the system to be one-step t-fault diagnosable. It was found that it was necessary that each unit be tested by at least t other units. Thus, we see that for a system of n units, a minimum number of connections that enable one-step t-fault diagnosability is $N = nt$ links.

An interconnection design in which $n = 2t + 1$ and each unit is tested by exactly t other units in such a way as to be one-step t-fault diagnosable is said to be optimal. A well known class of optimal designs is the so-called $D_{\delta t}$ design. In this design, a testing link from U_i to U_j exists if and only if $j - i = \delta m$ (modulo n) and m assumes the values $1,2,\ldots,t$. Examples of D_{12} and D_{22} designs, with $n = 5$, are shown in Figure 3.1. It was shown that the $D_{\delta t}$ design is an optimal one whenever δ and t are relatively prime, and as such, would allow the system designer to employ a most efficient testing interconnection scheme into the overall design. Also, this type of

testing interconnection between devices provides the ability to synthesize an efficient diagnostic algorithm, such as the one proposed by Meyer and Masson [3.6]. (This algorithm is presented in detail in the Appendix.)

As had been already pointed out, the complexity of a one-step t-fault diagnosable interconnection scheme leads to a rather large number of testing links. It is for this reason that sequential t-fault diagnosable systems have been studied. Since we are utilizing the same model as before, the lower bound on the number of units, $n \geq 2t+1$, is still valid. However, Preparata showed that there exists a class of designs with the number of testing links, $N = n + 2t - 2$, such that the resulting system is sequentially t-fault diagnosable. Essentially, this design was that of a simple loop along with a subset of $2t - 2$ units all testing a common unit. Such a testing interconnection scheme is shown in Figure 3.2 with $n = 14$ and $t = 6$.

The simplest sequential t-fault analysis is through the use of a single loop system. With this interconnection, a lower bound on the number of units to guarantee sequential t-fault diagnosability is given by the following:

$$n \geq v = 1 + (m+1)^2 + \lambda(m+1)$$
$$\text{with } t = 2m + \lambda,\ m \text{ integral and } \lambda = 0,1$$

A table comparing one-step t-fault diagnosis to sequential t-fault diagnosis with respect to their lower bounds on the number of units and testing links is given in Figure 3.3. Upon examination of the table, it is obvious that as the allowable number of faults increases, sequential t-fault diagnosis becomes much more cost effective in

relation to the hardware needed to realize the design.

As one of the first to address the problem, Preparata has been shown to make a few important initial contributions to the study of diagnosable systems. Among them have been the concepts of one-step and sequential t-fault diagnosis, along with each of their respective lower bounds or the number of units and testing interconnections. Also, a class of optimal designs were proposed. What was noticeably missing however, was the means to determine the diagnosibility number, i.e. the maximum number of allowable faulty units, of a general interconnection scheme.

To this end, it was necessary to be able to fully characterize the connection assignment of diagnosable systems. Hakimi and Amin [3.4] essentially picked up the study where Preparata left off, in that they claimed to have shown both the necessary and sufficient conditions for a system to be t- diagnosable. In their terminology, a system that is t-diagnosable is directly analagous to Preparata's one-step t-fault diagnosis. The model used is exactly the same as the one that had previously been considered. That is, the system in question can be viewed as a directed graph explicitly showing the testing connections between modules.

One of the main results of Hakimi and Amin was the determination of necessary and sufficient conditions for a system to be t-diagnosable when the system has the property that no two units test each other. Very simply, they found that the system is t-diagnosable only if each unit is tested by at least t other units. As an example,

it is seen that the $D_{\delta t}$ design, which is known to be t-diagnosable, satisfies the above conditions.

What was obviously needed, however was a general characterization of any system, regardless of the interconnection scheme. The following approach was proposed. Consider a dipath from U_{i_1} to U_{i_k} in which a sequence of vertices and edges in G (the graph), U_{i_1}, (U_{i_1}, U_{i_2}), U_{i_2}, ..., $(U_{i_{k-1}}, U_{i_k})$, U_{i_k} exists, where (U_{i_1}, U_{i_2}) denotes a directed edge from U_{i_1} to U_{i_2}. When there is a dipath from U_i to U_j in G, then U_j is said to be reachable from U_i. G is defined as being strongly connected if any pair of vertices are mutally reachable. Hakimi and Amin claimed the following: If $n \geq 2t+1$ and $K(G) \geq t$, then the system is t-diagnosable, where the connectivity K(G) of a digraph G is the minimum number of vertices whose removal from G yields a graph that is not strongly connected. It is claimed that whenever these conditions are met by any system, then the system diagnosibility number can be verified.

An example which questions these results is shown in Figure 3.4. By inspection, it is obvious that the graph as shown is not strongly connected. By considering U_4 and U_3 it is seen that a dipath from U_4 to U_3 does not exist. Therefore, the connectivity of this system is $K(G) = 0$, which implies that the system is o-diagnosable. A descrepancy evolves here in that it is felt that the system is instead 1-diagnosable, or equivalently, sequentially 1-fault diagnosable. It will be shown in the next discussion of Russell and Kime's model, that the above system is indeed sequentially 1-fault diagnosable. Thus we see that given a general interconnection scheme, necessary

and sufficient conditions have been proposed by Hakimi and Amin to determine the degree of the systems diagnosibility, although at the present they do seem to be questionable.

Another diagnostic model has been proposed by Russell and Kime which tends to be somewhat more general that the ones previously studied. In this model a system can be represented by either the G array approach $S=(\mathcal{F},T,F,G)$ or by the diagnostic graph approach $S=(\mathcal{F},T,F,G,)$. Common to both is the set $\mathcal{F}$ which is the set of a possible faults that may occur in the system. Thus, $\mathcal{F}=\{f_1,f_2,\ldots,f_n\}$. Now, if we consider all of the possible 2^n possible subsets of $\mathcal{F}$, $F=\{F^1,F^2,\ldots,F^{2^n}\}$ where $F^k(k=1,2,\ldots,2^n)$ is one of the allowable fault patterns that may occur in the system. The entire class of fault patterns can be represented by a 2^n x n F array as follows:

$$F=\begin{array}{c} \\ F^1 \\ F^2 \\ \vdots \\ F^{2^n} \end{array}\begin{array}{c} \begin{array}{cccc} f_1 & f_2 & \cdots & f_n \end{array} \\ \left[\begin{array}{cccc} & & & \\ & F_i^K = 1 \iff f_i \in F^K & & \\ & & & \end{array}\right] \end{array}$$

The set $T=(t_1, t_2,\ldots,t_p)$ represents the p pass-fail tests that can be applied to S. A test t_j is a complete test for a fault f_i if and only if a) t_j always fails for f_i alone and b) t_j always passes for no faults. Thus, we see that the set of tests which are complete for a fault pattern F^k is given as $t(F^k) = t(f_i) \cup t(f_j) \cup \ldots \cup t(f_k)$, where $f_i, f_j \ldots f_K \in F^k$. An invalid test set $T(F^k)$ is defined as the set of tests that may not correctly specify the nature of the system in the presence of a fault pattern F^k. That is, the test outcomes may become unreliable in the case where a) t_j might pass if $t_j \in t(F^k)$ and

b) t_j might pass if $t_j \notin t(F^K)$. This leads us to the notion of a valid test set. In the presence of a fault pattern F^K, a valid test is one in which a) t_j always fails if $t_j \in t(F^K)$ and b) t_j always passes if $t_j \notin t(F^K)$. It can be shown that the valid test set for a fault pattern can be derived as follows: Valid tests = $\left\{ t(F^K) - T(F^K) \right\}$.

Up to this point we have completely described those parts of the model which are common to both the diagnostic graph and G array approaches. In the G array, or generalized fault table, the set of test outcomes, or syndromes, is represented by a 2^n x p matrix with the following structure:

$$G = \begin{array}{c} \\ F_1 \\ F_2 \\ \vdots \\ F^{2n} \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & t_2 & \cdot & \cdot & \cdot & t_p \end{array} \\ \left[\begin{array}{cccccc} & & & & & \end{array} \right. \end{array}$$

$$G_j^K = \begin{cases} 0, & t_j \notin t(F^K) \text{ and } t_j \notin T(F^K) \quad \text{(i.e. always passes)} \\ 1, & t_j \notin t(F^K) \text{ and } t_j \notin T(F^K) \quad \text{(i.e. always fails)} \\ X, & t_j \notin T(F^K) \quad \text{(i.e. don't know)} \end{cases}$$

To aid in visualizing the system, a diagnostic graph can be drawn. This graph is essentially a digraph in which each allowable single fault in the system is represented by a vertex of the graph. A directed edge from f_i to f_j represents the situation in which f_i being faulty invalidates test that is complete for f_j. Thus, we see that according to this model, the set of complete tests for a fault is just the set of all incoming edges to it. In the presence of a fault pattern F^K, the set of invalid tests is the set of all outgoing tests of $f_i \in F^K$. An example of a diagnostic graph of a given system is shown in Figure 3.5. Notice that this model differs significantly from the one previously discussed in that more than one unit may be needed to perform a test. Therefore, it is evident that

this type of model allows a finer partitioning of the system. For example, both Data Channel (C1) and Memory (M1) act in conjunction with each other in testing the ROM Control (R1).

This model as proposed is much more efficient than the one introduced by Preparata due to the fact that it considers much more general systems. In a normal design, a fault may invalidate a test that would be performed. In such cases, the system could be viewed to be morphic, that is, $T(F^i \cup F^j) = T(F^i) \cup T(F^j)$. The present model, however, also enables special systems to be represented. As an example, suppose there exists a design in which two or more simultaneous faults are necessary in order to invalidate a test. Such would be the situation in a triple modular redundant design. These type of systems would be semi-morphic systems in which $T(F^i \cup F^j) \supseteq T(F^i) \cup T(F^j)$. Semi-morphic systems could be modelled as above by making a morphic approximation. Thus, we see that the diagnostic model covers a rather large class of system designs.

In their first paper. Russell and Kime [3.8]define diagnosability with repair to be exactly the same as sequential t-fault diagnosis. In order to derive the conditions for diagnosability, the concept of a closed fault pattern was studied. A closed fault pattern F^k is one in which every test for each fault in F^k is invalidated in the presence of F^k. Note that this is analagous to the classical masking ideas of combinational logic circuits. The system closure index C(S) is the cardinality of the smallest closed fault pattern in the system. It is with this index that a necessary

condition is given for the system to be t-fault diagnosable with repair. It is stated as follows:

$$C(S) \geq \left\lfloor \left\{ \frac{(t+2)^2}{2} \right\} \right\rfloor + 1$$

It was also shown that for t = 1,2,3 the above is both necessary and sufficient. Now, recall that in Figure 3.4 the system was deemed to 0 - fault diagnosable according to Hakimi and Amin. Since 1 - fault diagnosis is equivalent to a system which is 1 - fault diagnosable with repair, we can also use Russell and Kime's results to determine the diagnosability of the hypothetical system. Upon examination of Figure 4, it is seen the smallest closed fault pattern is $\{U_1, U_2, U_3\}$ and therefore C(S) = 3. Since C(S) = $3 \geq 2t+1$, this implies that the system diagnosibility number t = 1. Therefore, it appears we have a contradiction between the two approaches, with Hakimi and Amin's results seeming to be in question.

In their second study, Russell and Kime [3.9]define diagnosability without repair to be equivalent to one-step t-fault diagnosis, or simply t-diagnosability. Fault masking was found to be essential in determining the conditions for t-diagnosability. A fault pattern F^j is said to be masked by F^K if every test for each fault in F^j is invalidated in the presence of F^K. This concept is similar to that of faults completely masking others in logic networks. The masking index is the cardinality of the smallest fault pattern F^K that masks F^j and is denoted by $M(F^j)$. Thus, we see that F^j is masked by F^K if and only if $t(F^j) \subseteq T(F^K)$. Also, if a fault in F^j is not masked by F^j, it is said to be exposed. The exposure index of a fault pattern

F^j, $e(F^j)$ is the number of faults exposed in it. The minimum of the exposure indices of the fault pattern containing k faults is termed the system exposure index of order k, $e_k(S)$. From these definitions, we can find the number of faulty elements in F^K to be $\left|M(F^K)\right| + \left|e(F^K)\right|$.

Necessary and sufficient condition for a system to be t-fault diagnosable without repair are given as the following:

a) $M(S) \geq t$
b) $C(S) \geq 2t + 1$
c) $e_k(S) \geq 2t + 1 - k$ for $k = t + 1, \ldots, \min(2t - 1, n)$

Thus, with these results one can determine to what degree a system is capable of being diagnosable without repair. A major obstacle in working with these ideas points directly to the amount of book keeping involved in determining the system masking, closure and exposure indices.

A final model to consider has been published recently by Barsi [3.1] The diagnostic model proposed is a slight modification of the one introduced by Preparata, with the notion of producing a more realistic representation of the system. The basic assumptions are the following:

1) each test is performed by a single unit;
2) each unit must be capable of testing any other unit;
3) no unit tests itself; and
4) for any pair (U_i, U_j), unit U_i performs at most one test of unit U_j.

The actual difference between this model and that of Preparata's is with regard to the set of possible tests outcomes. Assuming that a testing link exists from U_i to U_j, we have the following set of allowable test outcomes:

$$a_{ij} = \begin{cases} 0, \text{ if } U_i \text{ is fault-free and } U_j \text{ is fault-free} \\ 1, \text{ if } U_i \text{ is fault-free and } U_j \text{ is faulty} \\ 0 \text{ or } 1, \text{ if } U_i \text{ is faulty and } U_j \text{ is fault-free} \\ 1, \text{ if both } U_i \text{ and } U_j \text{ are faulty.} \end{cases}$$

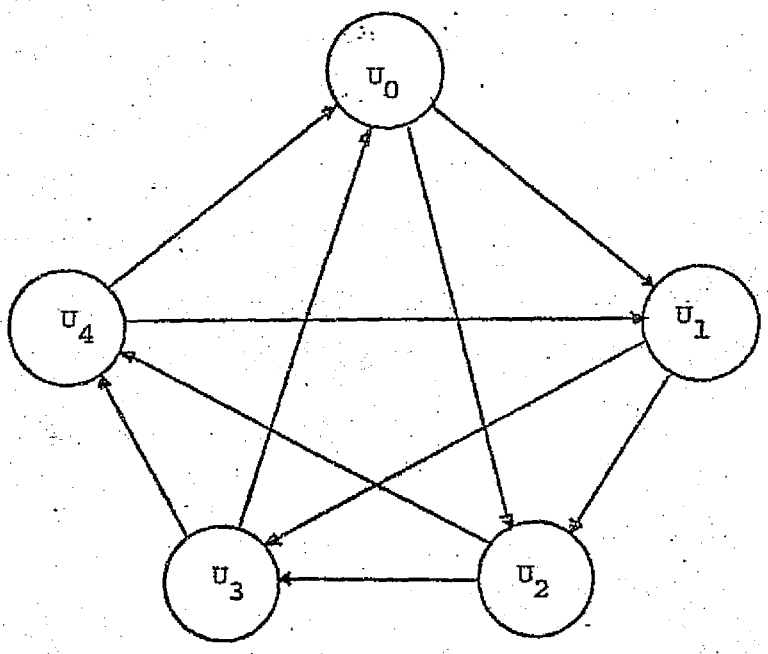
Notice that when both U_i and U_j are faulty, the test outcome is always a " 1." The reasoning behind this is that some type of self-checking design be incorporated into the critical parts of the testing devices. Thus, according to this approach, a " 1 " test outcome encountered specifies that the tested unit is guaranteed to be faulty.

Due to this slight difference in the model just discussed, the upper bound on the diagnosibility number t, is seen to increase. In fact, with a system of n units, the one-step diagnosibility of the system is $t \leq n-2$. Observe that this largely exceeds Preparata's bound of $t \leq \left\lfloor \frac{n-1}{2} \right\rfloor$. Necessary and sufficient conditions were also derived for the system to be one-step t-diagnosable. These are: a) $|B(x)| \geq t \; \forall x \in N$ and b) for each pair (x,y) with $x \in N$, $y \in N$, $|B(x)| = |B(y)| = t$ and $y \in B(x) \cap D(x)$ there exists at least one node u such that either $u \in B(x) - B(y) \cap B(x)$ and $B(u) \neq B(y)$, or $u \in B(y) - B(x) \cap B(y)$ and $B(u) \neq B(x)$, where B(x) is the predecessor set of x and D(x) is the sucessor set of x. Examination of the D_{1t} design reveals that only (a) of the above conditions reveals an optimal interconnection design. In fact, equality in (a) specifies that a design was indeed optimal. A technique for the synthesis of an optimal design was proposed, although it is essentially equivalent to the D_{1t} design previously discussed.

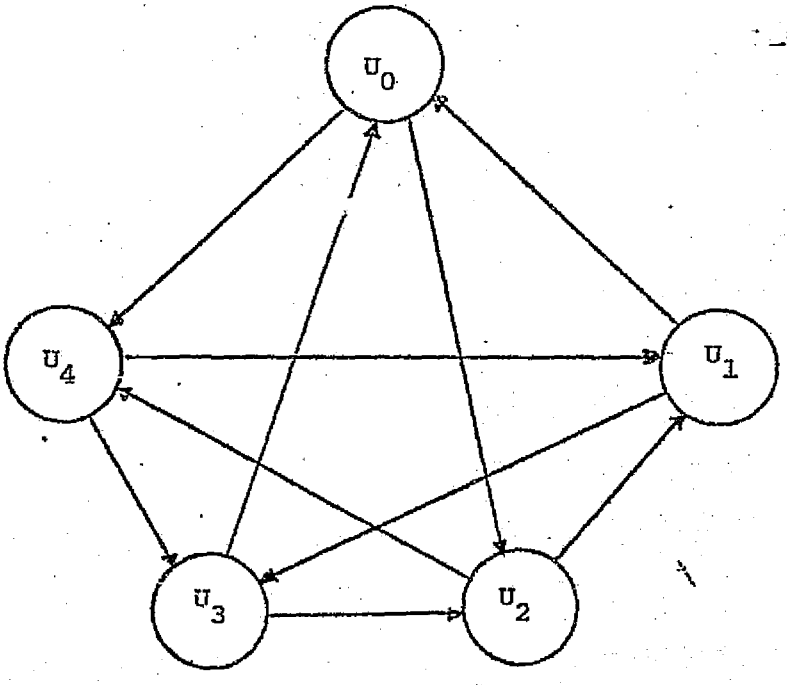
Barsi also addressed the problem of diagnosability without repair. In doing so, the index $f_1(x)$ is defined to be the minimum number of faulty units that can give rise to a syndrome of all 1's in the system, with the constraint that unit x is faulty. Similarly, the index $f_0(x)$ is defined to be the minimum number of faulty units that can give rise to a system syndrome of all 1's provided that unit x is fault-free. Using these, it is claimed that a system having a strongly connected graph is t-diagnosable with repair if and only if $t = \max_{i=1,\ldots,n} \left[f_0(U_i),\ f_1(U_i)\right] - 1$. Thus, we see that there exists but one more way of determining the diagnosability of a given system.

In conclusion, various models used in studying diagnosable systems have been presented. It is felt that they adequately represent the research that has been directed in the area, ranging from the basic concepts to the more advanced analysis which represents the current state of the art with respect to diagnosable systems.

With this in mind, we can now proceed to our extensions of this theory to I/T faults.



a) D_{12} Design



b) D_{22} Design

Figure 3.1. Example D_{St} Designs

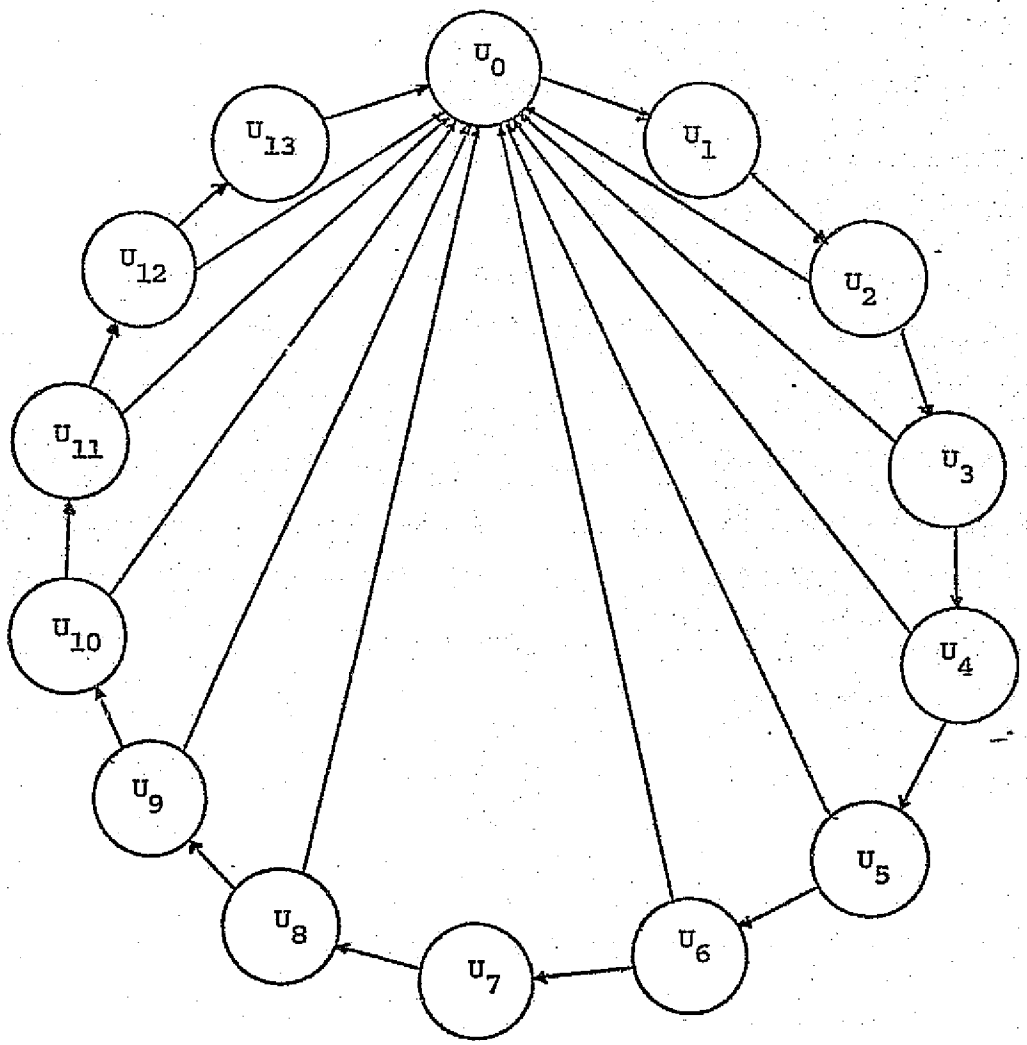


Figure 3.2. A sequential 6-fault diagnosis connection

Number of Allowable Faults	One-step Diagnosis		Sequential Diagnosi	
t	Units	Links	Units	Link
	$n \geq 2t + 1$	$N \geq nt$	$n \geq 1+(m+1)^2+\lambda(m+1)$	$N \geq n$
1	3	3	3	3
2	5	10	5	5
3	7	21	7	7
4	9	36	10	10
5	11	55	13	13
6	13	78	17	17
7	15	105	21	21

Figure 3.A comparison between one-step and sequential diagnosis systems.

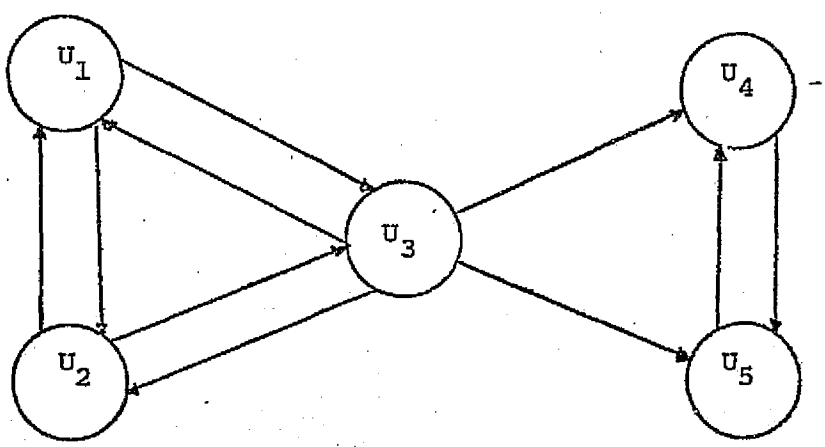


Figure 3.4. An interconnection scheme where $K(G) = 0$.

REFERENCES

3.1 F. Barsi, F. Grandoni and P. Moestrini, "A theory of diagnosability of digital systems," IEEE Transactions on Computers, Vol. C-25, NO. 6, June 1967, pp 585-593.

3.2 D.B. Burchby and L.W. Kern, "Specification of the fault-tolerant spaceborne computer (FTSC)," Proceedings of the Fault Tolerant Symposium, IEEE Cat. No. 76 CH 1094-2C, June 1967, pp 129-133.

3.3 A.M. Corluhan and S.L. Hakimi, "On an algorithm for identifying faults in a T-diagnosable system," Proceedings of the 1976 Conference on Information Sciences and Systems, The Johns Hopkins University, 1976, pp 370-375.

3.4 S.L. Hakimi and A.T. Amin, "Characterization of the connection assignment of diagnosable systems," IEEE Transactions on Computers, January 1974, pp 86-88.

3.5 T. Kameda, S. Toida and F.J. Allan, "A diagnosing algorithm for networks", Information and Control, Vol. 29, 1975, pp 141-148.

3.6 G.G.L. Meyer and G.M. Masson, "An efficient fault diagnosis algorithm for multiple processor architectures, to appear, IEEE Transactions on Computers.

3.7 F.P. Preparata, G. Metze and R.T. Chien, "On the connection assignment problem of diagnosable systems," IEEE Transactions on Computers, Vol. EC-16, No. 6, December 1967, pp 848-854.

3.8 J.D. Russell and C.R. Kime, "System fault diagnosis: Closure and diagnosability with repair," IEEE Transactions on Computers, Vol. C-24, No. 11, November 1975, pp 1078-1088.

3.9 J.D. Russell and C.R. Kime, "System fault diagnosis: Masking, exposure and diagnosability without repair," IEEE Transactions on Computers, Vol C-24, No. 12, December 1975, pp 1155-1161.

3.10 J.J. Stiffler, "Archtectural design for near - 100% fault coverage," Proceedings of the Fault Tolerant Symposium, IEEE Cat. No. 76 CH 1094-2C, June 1976, pp 134-137.

3.11 S. Toida, "System diagnosis and redundant tests," IEEE Transactions on Computers, November 1976, pp 1167-1170.

4. I/T Testing and Diagnosis

4.0 Introduction

Previous work on intermittent fault detection has been done by Breuer [3-1] and Kamal and Page [3-2]. Breuer assumed that the statistics of the intermittent fault can be modeled by a two state first-order Markov process. State FP corresponds to the fault being present at time t_g and state FN corresponds to the fault not being present at t_g. The transition probabilities for going from one state at t_g to either state FP or FN at t_{g+1} are assumed known. From this, the steady state probabilities associated with states FP and FN at any time t_K can be determined as a function of these probabilities at some initial time t_0. Let T be a collection of tests $X_1, X_2, \ldots X_K$ where each $X_k, k = 1,2,\ldots K$, is a single test pattern for an intermittent fault in a combinational circuit. T will detect the presence of the intermittent fault under test if the fault is present when at least one of the X_k is applied. The probability that T will detect the presence of the intermittent fault is a function of D_k, the time between the application of tests and K, the number of tests applied.

The model used by Kamal and Page is a special case of the above one. In this case, the transition probabilities between the two states are assumed to be equal. Thus, the first-order Markov process reduces to a zero-order process. It is assumed that $P(\omega_i)$, the prior probability that the circuit is in ω_i,

where ω_i denotes the condition of the circuit having the intermittent fault i, is known. It is also assumed that e_i, the probability that the effect of the intermittent fault ω_i is present knowing that the circuit already has the intermittent fault ω_i, is constant and known. After applying a test t_j to the circuit and observing the output, the posterior probabilities $P(\omega_i/\text{output when } t_j \text{ is applied})$ are calculated using Bayes' rule. These posterior probabilities are used as prior probabilities the next time a test is applied.

4.1 Combinational Circuits:-

A transient fault is intermittent if it occurs repeatedly. If a transient fault is not intermittent, it would be very difficult to test for it in a combinational network. This is because the combinational network would behave as if it is fault free after the transient has disappeared. If the transient does not occur repeatedly we might never catch it at all. Therefore, intermittent/transient faults will be considered here. It is necessary to characterize intermittent faults.

4.1.1 Model of Intermittent Faults:

Arrival:- We will assume that intermittent faults arrive in a random manner. The interarrival times of faults will be assumed to be independent and random with a known probability density function.

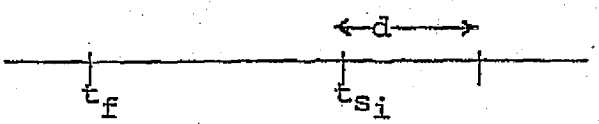
Duration:- After a fault arrives, it persists for certain time which is its duration. We will assume that the intermittent fault has a duration which is random with a known probability density function. We will also assume that the duration is independent of the arrival.

Depending on the nature of the fault, different density functions might be used to model the random nature of the interarrival times and the duration.

(i) If an assumption that short interarrival times (or durations) are more likely than longer ones is used, we arrive at an exponential or hyperexponential density as an approximation.

(ii) If an assumption that there is a definite mean inter-arrival time (or duration) with an associated spread is made, we would get gamma, normal, Rayleigh, Erlang or Weibull approximation. Naturally, the chosen density function should have a value of zero for negative values of the argument (time, in this case).

4.1.2 Fault Detection:-



t_f: time fault arrives

t_{s_i}: time test (set) is applied

d: duration of the test (set)

It will be assumed that a fault arrival can be detected by a test only if the effect of the fault arrival is present for th_ complete duration of the test, since otherwise, the output will change when observing for the presence of the fault, leading to uncertainty. When a test set is applied, a fault arrival can be detected only if it persists for the entire duration of the particular test(s) which tests for it. Since tests in a test set can be applied in any order, it is conservative to assume that a fault arrival is not detected by a test set if it does not persist for the entire duration of the test set. Therefore, a fault arriving at t_f can be

detected by a test set applied at t_{s_i} only if the duration of the fault t_d, is such that

$$t_d \geq t_{s_i} + d - t_f$$

Due to the non-permanent nature of the faults, it is necessary to apply the test set repeatedly. If the network gives an incorrect output under any test set application, the testing can be stopped because the presence of a fault is indicated. But it is necessary to have a decision rule which will permit us to stop further applications of the test set at some stage when the network responds correctly to all the past applications of the test set. The decision rule which will be used here is: The conditional probability of not detecting a fault given that the network has an intermittent fault is close to 0. That is,

$$P(\text{fault not detected/network is faulty}) \leq |\varepsilon|$$

Depending on the level of confidence required about the fault free condition of the network (if it responds correctly to all the test set applications), $|\varepsilon|$ can be chosen to be as close to 0 as needed.

4.1.3:- Sequential Analysis

Let $T_1, \ldots, T_k$ be k applications of the test set, applied at times $t_{s_1} < t_{s_2} \ldots < t_{s_k}$ respectively. All the

probabilities mentioned below are conditional probabilities given that the network has an intermittent fault.

$P(T_i)$ = Probability that T_i detects the presence of a fault

= Probability that there is a fault arrival at t_f before t_{s_i} such that its duration is $\geq t_{s_i}+d-t_f$

$P(\overline{T}_i)$ = Probability that T_i does not detect the presence of a fault

$= 1 - P(T_i)$

$P(\overline{T}_i \cap \overline{T}_{i-1} \cdots \cap \overline{T}_1)$ = Probability that none of the i applications of the test set detect a fault.

All the fault arrivals which can be detected by T_2 can be classified into two groups.

i) Arrivals which can be detected by both T_2 and T_1.

ii) Arrivals which can be detected by T_2 and not T_1.

Given that T_1 has not detected the presence of any fault, the probability that T_2 detects a fault is just the probability that there is an arrival belonging to group ii). Hence, $P(T_2|\overline{T}_1) = P(T_2) - P(T_2*T_1)$ where T_2*T_1 is the event that there is a fault arrival which can be detected by both T_2 and T_1.

$$P(\overline{T}_2|\overline{T}_1) = 1 - P(T_2|\overline{T}_1)$$

$$= 1 - P(T_2) + P(T_2*T_1)$$

$$\therefore P(\overline{T}_2 \cap \overline{T}_1) = P(\overline{T}_2|\overline{T}_1)\ P(\overline{T}_1)$$
$$= \left(1-P(T_2)+P(T_2{*}T_1)\right)\ \left(1-P(T_1)\right)$$

We need to compute $P(T_3|\overline{T}_2 \cap \overline{T}_1)$. Given that neither T_1 nor T_2 has detected the presence of a fault, the probability that T_3 detects a fault is given by

$$P(T_3\ \overline{T}_2 \cap \overline{T}_1) = P(T_3) - P\left((T_3{*}T_2) \cup (T_3{*}T_1)\right)$$

But the event $T_3{*}T_1$ is a subset of the event $T_3{*}T_2$. Hence,

$$P(T_3|\overline{T}_2 \cap \overline{T}_1) = P(T_3) - P(T_3{*}T_2)$$
$$P(\overline{T}_3|\overline{T}_2 \cap \overline{T}_1) = 1 - P(T_3)+P(T_3{*}T_2)$$

Similarly, for any T_i $i>1$,

$$P(\overline{T}_i|\overline{T}_{i-1} \cap \ldots \cap \overline{T}_1) = 1 - P(T_i) + P(T_i{*}T_{i-1})$$

The decision rule employed for the termination of the test set applications requires the computation of the probability $P(\overline{T}_k \cap \ldots \overline{T}_1)$ which can be done as follows,

$$P(\overline{T}_1) = 1 - P(T_1)$$

$$P(\overline{T}_j \cap \overline{T}_{j-1} \ldots \cap \overline{T}_1) = (1-P(T_j)+P(T_j{*}T_{j-1}))\ P(\overline{T}_{j-1}\ \ldots T_1) \qquad k \geq j > 1$$

As can be seen, this requires the computation of $P(T_j)$ for $1 \leq j \leq k$ and $P(T_j{*}T_{j-1})$ for $1 < j \leq k$.

4.1.4 Queueing Theory Applications:-

The intermittent fault model assumed can be noted for its similarity to the models used in Queueing theory. Hence, the probabilities which need to be computed can be done so using results from Queueing theory.

The fault system, where F_i is the intermittent fault (say on the i^{th} line in a network), has fault arrivals and duration. Naturally, when a fault arrives, till its duration ends, there cannot be another new arrival. Therefore, at any time, the arrival of a new fault depends on the previous arrival. But once a fault arrives, its duration is independent of previous arrivals and previous fault durations.

A queueing system is characterized by the following three factors:

1. The customer arrivals
2. The service time of customers
3. The service system

The customer arrivals and service times are expressed as statistical distributions. The service system can be described by the number of servers in the system and the queue discipline.

There is a one-to-one correspondence between the parameters of the fault model assumed and those associated with a queueing system. Thus, the arrival of faults corresponds to the customer arrivals, the duration of faults to the service

time of customers and the service system corresponding to the fault model is a single server system with no waiting permitted.

One implicit assumption made is that the arrival of faults begins before testing is started. However, the exact time of the beginning of the arrival process cannot be known. Hence, it will be assumed that the arrivals begin long before testing is started. This will permit us to use the steady state results of queueing theory (which are time independent and simple) rather than the transient ones (which are time dependent and hence complex).

The Erlang k-distribution will be used to approximate the statistical nature of the interarrival time and the fault duration. If k equals one, this reduces to the exponential distribution. When the Erlang k-distribution is used, the mean and standard deviation equal that of the practical problem and yet, some of the properties of the negative exponential distribution are maintained. The arrival (or duration) is divided into a fixed number of independent, identical (hypothetical) "phases", each phase having a negative exponential distribution. Each time a phase ends, an arrival takes place or the duration ends as the case may be. The Erlang k-distribution is,

$$f(t) = \frac{\lambda^k \, t^{k-1} \, e^{-\lambda t}}{(k-1)!}$$

With the above assumptions, some correspondence relations can be stated. Using these relations, probabilities required for fault analysis can be obtained from the equivalent Queueing model which is a single server system with no waiting allowed. F_i is the type of intermittent fault present, with arrivals and duration.

(i) P(effect of a fault arrival is present at time t) $\underline{\Delta}$ $P(f_i)$ $\longleftrightarrow$ P(A customer is in the service facility at time t)

(ii) P(effect of a fault arrival is present from time t to at least t+d) $\underline{\Delta}$ $P(f_i \cap rf_i \geq d)$ $\longleftrightarrow$ P(A customer is in the service facility at time t and will require at least d more units of service time).

(iii) P(effect of any fault arrival is not present at time t) $\underline{\Delta}$ $P(\overline{f}_i)$ $\longleftrightarrow$ P(there is no customer in the service facility at time t)

(iv) P(effect of any fault arrival is not present from time t to at least t+s) $\underline{\Delta}$ $P(\overline{f}_i \cap r\overline{f}_i \geq s)$ $\longleftrightarrow$ P(No customer is in the service facility at time t and no new customer will enter the service facility at least till time t+s).

4.1.5 Single Fault Detection Procedure:-

Let the set of faults in a circuit be $F = \{f_1, f_2, \ldots, f_n\}$, where each f_i is a single fault. A fault event E_f occurs when a single fault f_i from F occurs. When the assumption that only a single fault can occur is made, the n faults in F are not independent occurrences. Therefore, the arrivals and duration considered will be of E_f.

We will assume that each test set application T_j has a duration d. It will also be assumed that the elapsed time between successive test set applications T_{j-1} and T_j is a constant, L. A test set application T_i applied at t_{s_i} will detect the presence of a fault only if the effect of a fault arrival is present from t_{s_i} to at least $t_{s_i}+d$. Therefore,

$$P(T_j) = P(f_i \cap rf_i \geq d) \qquad (1)$$

A fault arrival which can be detected by T_{j-1} can also be detected by T_j only if the arrival occurs before $t_{s_{j-1}}$ and is such that the effect of the fault arrival is present from $t_{s_{j-1}}$ to at least $t_{s_{j-1}}+d+L$. Therefore,

$$P(T_j * T_{j-1}) = P(f_i \cap rf_i \geq L+d) \qquad (ii)$$

The above probabilities are independent of t_{s_i} because we use steady state results.

Hence, given k applications of the test set, the probability that none of them detects a fault is,

$$P(\overline{T}_k \cap \ldots \overline{T}_1) = [1-P(f_i \cap rf_i \geq d)] \prod_{j=2}^{k} (1-P(f_i \cap rf_i \geq d)+P(f_i \cap rf_i \geq L+d))$$

Note that above probability is the conditional probability given that the network has an intermittent fault. This is because the model we have assumed guarantees at least one fault arrival.

Now we consider 4 cases of arrival and duration.

i) Exponential interarrival time: $f(t_a)=\lambda e^{-\lambda t_a}$
Exponential duration: $f(t_d)=\mu e^{-\mu t_d}$

$$P(f) = \frac{\lambda}{\lambda+\mu}$$

$$P(\overline{f}) = \frac{\mu}{\lambda+\mu}$$

$$P(f \cap rf \geq d) = e^{-\mu d} \frac{\lambda}{\lambda+\mu}$$

$$P(\overline{f} \cap r\overline{f} \geq S) = e^{-\lambda S} \frac{\mu}{\lambda+\mu}$$

Hence,

$$P(\overline{T}_k \cap \overline{T}_{k-1} \ldots \cap \overline{T}_1) = \left(1-\frac{\lambda}{\lambda+\mu} e^{-\mu d}\right) \left(1-\frac{\lambda}{\lambda+\mu} e^{-\mu d}(1-e^{-\mu L})\right)^{k-1}$$

(ii) Exponential interarrival time:- $f(t_a) = \lambda e^{-\lambda t_a}$

k-Erlang duration (k=m):- $f(t_d) = \dfrac{\mu^m t_d^{m-1} e^{-\mu t_d}}{(m-1)!}$

$$P(f) = \sum_{n=1}^{m} P(\text{n stages of duration remaining}) = \frac{\lambda}{\lambda+\mu}$$

$$P(\bar{f}) = \frac{\mu}{\lambda+\mu}$$

$$P(f \cap rf \geq d) = \sum_{n=1}^{m} P(\text{n stages of duration remaining and remaining duration is } \geq d)$$

$$= \sum_{n=1}^{m} \frac{\lambda}{m(\lambda+\mu)} e^{-d\mu} \sum_{S=0}^{m-1} \frac{(\mu d)^S}{S!}$$

$$P(\bar{f} \cap r\bar{f} \geq S) = e^{-\lambda S} \frac{\mu}{\lambda+\mu}$$

Hence,

$$P(\bar{T}_k \cap \bar{T}_{k-1} \cdots \cap \bar{T}_1) =$$

$$\left[1 - \sum_{n=1}^{m} \frac{\lambda}{m(\lambda+\mu)} e^{-\mu d} \sum_{s=0}^{m-1} \frac{(\mu d)^S}{S!}\right] \times \left[1 - \sum_{n=1}^{m} \frac{\lambda}{m(\lambda+\mu)} e^{-\mu d} \sum_{S=0}^{m-1} \left\{ \frac{(\mu d)^S}{S!} - \frac{(\mu d+\mu L)^S}{S!} e^{-\mu L} \right\}\right]$$

(iii) k-Erlang interarrival time $(k=\ell)$, $f(t_a) = \dfrac{\lambda^\ell t_a^{\ell-1} e^{-\lambda t_a}}{(\ell-1)!}$

Exponential duration: $f(t_d) = \mu e^{-\mu t_d}$

$$P(f) = \sum_{n=1}^{\ell} P(\text{n stages of arrival remaining and fault in in duration}) = \frac{\lambda}{\mu}\left[1 - (1+\frac{\mu}{\lambda\ell})^{-\ell}\right]$$

$$P(\bar{f}) = 1-\frac{\lambda}{\mu} + \frac{\lambda}{\mu}\left(\frac{\mu}{\lambda\ell}\right)^{-\ell}$$

$$P(f \cap rf \geq d) = \bar{e}^{\mu d}\,\frac{\lambda}{\mu}\,(1-(1+\frac{\mu}{\lambda\ell})^{-\ell})$$

$$P(\bar{f} \cap r\bar{f} \geq S) = \sum_{n=1}^{\ell} P(\text{n stages of arrival remaining with no fault in duration and remaining stages need more than S units of time})$$

$$= \sum_{n=1}^{\ell} \frac{1}{\ell}\left(1+\frac{\mu}{\ell\lambda}\right)^{n-(\ell+1)} \sum_{q=0}^{\ell-1} \bar{e}^{\lambda S}\,\frac{(\lambda S)^q}{q!}$$

Hence,

$$P(\bar{T}_k \cap \ldots \bar{T}_1) =$$

$$\left(1-\bar{e}^{\mu d}\,\frac{\lambda}{\mu}\left[1-(1+\frac{\mu}{\lambda\ell})^{-\ell}\right]\right)\left(1-\bar{e}^{\mu d}\,\frac{\lambda}{\mu}\left(1-(1+\frac{\mu}{\lambda\ell})^{-\ell}\right)\left(1-\bar{e}^{\mu L}\right)\right)^{k-1}$$

(iv) k-Erlang interarrival time $(k=\ell)$, $f(t_a)=\frac{\lambda^{\ell} t_a^{\ell-1}\,\bar{e}^{\lambda t_a}}{(\ell-1)!}$

k-Erlang duration $(k=m)$, $f(t_d) = \frac{\mu^m t_d^{m-1}\,\bar{e}^{\lambda t_d}}{(m-1)!}$

First, it is necessary to solve for the following $m\ell+\ell$ simultaneous linear algebraic equations.

$$\lambda P(r,-;0) = \lambda P(r+1,-,0) + \mu P(r,1;1)$$

$$\lambda P(\ell,-;0) = \mu P(\ell,1;1) \qquad r=1,2,\ldots\ell-1$$

$$(\lambda+\mu)P(r,n;1) = \lambda P(r+1,n;1) + \mu P(r,n+1;1)$$

$$r=1,2,\ldots,\ell-1$$

$$n=1,2,\ldots,m-1$$

$$(\lambda+\mu)P(r,m;1) = \lambda P(r+1,m;1)$$

$$r=1,2,\ldots,\ell-1$$

$$(\lambda+\mu)P(\ell,n;1) = \mu P(\ell,n+1;1)$$

$$n=1,2,\ldots,m-1$$

$$(\lambda+\mu)P(\ell,m,1) = \lambda P(1,-;0)$$

$$\sum_{j=0}^{1} \sum_{n=1}^{m} \sum_{r=1}^{1} P(r,n,j) = 1 \qquad , \text{ where}$$

$P(i,j,1) \triangleq$ Probability that there are i stages of arrival remaining and j stages of duration remaining and a fault in duration.

$P(i,-,0) \triangleq$ Probability that there are i stages of arrival remaining and no fault in duration.

The required probabilities can be obtained as follows.

$$P(f) = \sum_{n=1}^{m} \sum_{q=1}^{\ell} P(q,n;1)$$

$$P(f \cap rf \geq d) =$$

$$\sum_{q=1}^{\ell} \sum_{n=1}^{m} P(q,n;1)\ e^{-\mu d} \sum_{S=0}^{m-1} \frac{(\mu d)^S}{S!}$$

$$P(\bar{f}) = \sum_{q=1}^{\ell} P(q,-;0) = 1-P(f)$$

$$P(\bar{f} \cap r\bar{f} \geq S) = \sum_{q=1}^{\ell} P(q,-;0)\ e^{-\lambda S} \sum_{c=0}^{\ell-1} \frac{(\lambda S)^c}{c!}$$

Hence,

$$P(\bar{T}_k \cap \ldots\ldots \cap \bar{T}_1) =$$

$$\left(1-\sum_{q=1}^{\ell}\sum_{n=1}^{m} P(q,n;1)\right) e^{-\mu d} \sum_{S=0}^{m-1} \frac{(\mu d)^S}{S!} \Bigg) [1 -$$

$$\sum_{q=1}^{\ell}\sum_{n=1}^{m} P(q,n;1)\ e^{-\mu d} \sum_{S=0}^{m-1} \left(\frac{(\mu d)^S}{S!} - \frac{(\mu d+\mu L)^S}{S!}\ e^{-\mu L} \right)^{k-1}]$$

The probability that a fault is detected is

$$P_k = 1 - P(\bar{T}_k \cap \ldots\ldots \cap \bar{T}_1)$$

In practice, a fault can be detected if it exists for the entire duration of the particular test(s) which test for it. Since we required the fault arrival to effect the entire duration of the test set, the actual probability of fault detection will be greater than the P_k obtained above. But the above procedure can be easily extended to give more accurate results.

Let the test set contain g tests. Each test has a duration of d/g, where d is the test set duration. Let the set of possible faults be $F = \{f_1, f_2, \ldots, f_n\}$. Let

P_{f_i} be the conditional probability that fault f_i has occurred given that E_f has occurred. Naturally,

$$\sum_{i=1}^{n} P_{f_i} = 1$$

It is necessary that P_{f_i} be known for each i. By considering those tests which test for f_i, the probability $P(i)$ that f_i is detected can be computed using the results developed above by substituting tests in place of test set everywhere. Then,

$$P_k = \sum_{i=1}^{n} P(i) P_{f_i}$$

This is the actual probability of detection of a fault.

4.1-6 Multiple Fault Detection Procedure

Let the number of lines which can be faulty be n. All these lines are assumed to have identical fault characteristics. It is also assumed that the probability that b lines are faulty is the same as the probability that a single line is faulty. Due to the large density of circuits in present day IC's, this seems a reasonable assumption. We also assume that an intermittent fault on a line is either of the s-a-1 or s-a-0 type but not both. The arrival and duration of the intermittent faults corresponding to each of the n lines are similar to the corresponding ones of E_f, as used in the single fault case.

Here, it will be assumed that a multiple fault can be detected only if it is present for the entire duration of the test set (MFDTS). This is a reasonable assumption because, if any component fault in the multiple fault is present only for part of the duration of the test set, neither of the two different fault situations would be detected if the particular tests which test for them are not applied during their presence. All the assumptions regarding the test set applications will be the same as before. The probability that a given line in a network is faulty = P_f.
The probability that the given network is faulty is,

$$P_F = \sum_{b=1}^{n} \binom{n}{b} P_f = (2^n - 1)P_f$$

The probability that exactly b of the n wires are faulty is,

$$P_b = \binom{n}{b} P_f$$

The conditional probability that exactly b wires are faulty given that the network is faulty is,

$$P_{cb} = \frac{\binom{n}{b} P_f}{2^n - 1\ P_f} = \frac{\binom{n}{b}}{2^n - 1}$$

clearly,

$$\sum_{b=1}^{n} P_{cb} = 1$$

Given that a b-wire multiple fault has occurred, each of the component wires of the multiple fault has fault arrivals with duration. Let the fault corresponding to the b wires be labeled $f_1,\ldots,f_b$. A multiple fault involving f_1 can be detected by a test set application if the effect of a fault arrival of f_1 exists for the complete duration of the test set, d and none of the other faults have an arrival whose effect exists for only part of the duration d.

P(multiple fault involving f_1 is detected)

$$= P(f \cap rf \geq d)\,(P(f \cap rf \geq d) + P(\bar{f} \cap r\bar{f} \geq d))^{b-1}$$

P(multiple fault involving f_2 but not f_1 is detected)

= P(f_1 has no fault arrival with an effect on test set and f_2 has an arrival with effect for the complete duration d and none of the other faults have an arrival with an effect for only part of the duration d)

$$= P(\bar{f} \cap r\bar{f} \geq d)\,P(f \cap rf \geq d)\,(P(f \cap rf \geq d) + P(\bar{f} \cap r\bar{f} \geq d))^{b-2}$$

The probability that the b-wire multiple fault is detected by a test is,

$$S_b(d) =$$

$$P(f \cap rf \geq d) \sum_{i=1}^{b} (P(\bar{f} \cap r\bar{f} \geq d)^{i-1} (P(f \cap rf \geq d) + P(\bar{f} \cap r\bar{f} \geq d))^{b-i}$$

Therefore, the conditional probability that a test set application will detect a multiple fault given that it has occurred is,

$$P(T_j) = \sum_{b=1}^{n} P_{cb} \, S_b(d)$$

$$= \sum_{b=1}^{n} \frac{\binom{n}{b}}{2^n - 1} \cdot P(f \cap rf \geq d) \times$$

$$\left(\sum_{i=1}^{b} (P(\bar{f} \cap r\bar{f} \geq d)^{i-1} (P(f \cap rf \geq d) + P(\bar{f} \cap r\bar{f} \geq d))^{b-i} \right.$$

A multiple fault arrival which can be detected by T_{j-1} can also be detected by T_j only if all the component fault arrivals occur before $t_{s_{j-1}}$ and are such that their effects are present from $t_{s_{j-1}}$ to at least $t_{s_{j-1}} + d + L$. This probability is $S_b(T+d)$, where we substitute T+d for d. Hence,

$$P(T_j * T_j\text{-}1) = \sum_{b=1}^{n} P_{cb} \, S_b(T+d)$$

$$= \sum_{b=1}^{n} \frac{\binom{n}{b}}{2^n - 1} \cdot P(f \cap rf \geq T+d) \times$$

$$\left(\sum_{i=1}^{b} (P(\bar{f} \cap r\bar{f} \geq T+d)^{i-1} \, (P(f \cap rf \geq T+d) + P(\bar{f} \cap r\bar{f} \geq T+d))^{b-i} \right.$$

Hence, given k applications of the test set, the probability that none of them detects a fault is,

$$P(\bar{T}_k \ldots \cap \bar{T}_1) =$$

$$(1 - \sum_{b=1}^{n} \frac{\binom{n}{b}}{2^n-1} \cdot P(f \cap rf \geq d) \times \Big(\sum_{i=1}^{b} (P(\bar{f} \cap r\bar{f} \geq d))^{i-1}$$

$$(P(f \cap rf \geq d) + P(\bar{f} \cap r\bar{f} \geq d))^{b-i}\Big)\Big)(1 - \sum_{b=1}^{n} \frac{\binom{n}{b}}{2^n-1} \cdot (P(f \cap rf \geq d) \times$$

$$\Big(\sum_{i=1}^{b} (P(\bar{f} \cap r\bar{f} \geq d))^{i-1} (P(f \cap rf \geq d) + P(\bar{f} \cap r\bar{f} \geq d))^{b-i}\Big) - P(f \cap rf \geq T+d) \times$$

$$\Big(\sum_{i=1}^{b} (P(\bar{f} \cap r\bar{f} \geq T+d))^{i-1} (P(f \cap rf \geq T+d) + P(\bar{f} \cap r\bar{f} \geq T+d))\Big)^{b-i}$$

Now we consider one of the 4 cases mentioned before

Exponential inter arrival time: $f(t_a) = \lambda \bar{e}^{\lambda t_a}$

Exponential duration : $f(t_d) = \mu \bar{e}^{\mu t_d}$

$$P(\bar{T}_k \cap \bar{T}_{k-1} \ldots \cap \bar{T}_1) =$$

$$(1 - \sum_{b=1}^{n} \frac{\binom{n}{b}}{2^n-1} \cdot \bar{e}^{\mu d} \frac{\lambda}{\lambda+\mu} \Big(\sum_{i=1}^{b} (\bar{e}^{\lambda d} \frac{\mu}{\lambda+\mu})^{i-1}$$

$$(\bar{e}^{\lambda d} \frac{\mu}{\lambda+\mu} + \bar{e}^{\mu d} \frac{\lambda}{\lambda+\mu})^{b-i}\Big)\Big) (1 - \sum_{b=1}^{n} \frac{\binom{n}{b}}{2^n-1} \frac{\lambda \bar{e}^{\mu d}}{\lambda+\mu}$$

$$\Big(\sum_{i=1}^{b} (\bar{e}^{\lambda d} \frac{\mu}{\lambda+\mu})^{i-1} (\bar{e}^{\lambda d} \frac{\mu}{\lambda+\mu} + \bar{e}^{\mu d} \frac{\lambda}{\lambda+\mu})^{b-i} - \bar{e}^{\mu T} \sum_{i=1}^{b} (\bar{e}^{\lambda (T+d)} \frac{\mu}{\lambda+\mu})^{i-1}$$

$$(\bar{e}^{\lambda (T+d)} \frac{\mu}{\lambda+\mu} + \bar{e}^{\mu (T+d)} \frac{\lambda}{\lambda+\mu})^{b-i}\Big)$$

4.1-7 Conclusion

As can be seen, the expression gets complex. However, if the actual values of the various parameters are substituted, the

computation can be performed in a systematic manner, very easily on a computer. The various parameters can be estimated using methods employed in Queuing systems.

We have obtained expressions to determine the number of times a combinational circuits has to be tested, when checking for the presence of single or multiple faults of the intermittent-transient type. Though dependent on the model used, since we have used quite a general model, these results should be useful.

4.2 Sequential Machines:-

A sequential machine can be tested by applying an appropriate sequence of input signals, termed a checking sequence, to the circuit and observing the output sequence that the circuit produces in response [3-3]. The checking sequence determines whether or not the sequential machine is operating in accordance with the given state table description rather than testing for specific hardware failures. Hence it is difficult to find the precise relationship between the presence of an I/T fault and its effect on the output sequence. Though the output of the sequential machine may be correct during the presence of the fault, it could be incorrect at a later stage. Therefore, it is convenient to model the faulty sequential machine as a probabilistic sequential machine.

4.2.1 The model:-

If the statistics of the I/T fault are known, at a given time, the probability that the effect of the I/T fault is present, can be calculated. A particular fault will affect the next state and output functions in a particular way. By knowing the exact way in which each fault will affect the next state and output functions along with the relative probabilities of occurrence of these faults and their statistics, the faulty sequential machine can be modeled as a probabilistic sequential machine. Instead of the exact model, it is possible

to set up an approximate model, with relative ease, by assuming that every possible combination of incorrect next state and/or output is equally likely when the effect of the I/T fault is present. In either case, we arrive at a probabilistic sequential machine model of the faulty machine.

4.2.2 Testing:-

The actual application of the checking sequence is preceded by the application of a homing sequence to bring the machine to a fixed starting state. Initially, if we assume that the machine is equally likely to be in any of its states, the probability of the machine being in any final state after the application of the homing sequence can be computed using the transition probabilities and the output response of the machine to the homing sequence.

If the initial state probabilities are known, the probability that the machines output response to the checking sequence is correct can be easily found [3-4]. This represents the probability that the test does not detect the fault, given that the machine is faulty. Therefore, the probability that n applications of the test fail to detect the fault given that the machine is faulty, can be calculated.

4.2.3 Conclusion:-

The testing of sequential machines for I/T faults is straightforward once an exact model of the faulty machine as a probabilistic sequential machine is obtained, due to the results already available in this area. Hence, in this section, we have just outlined the technique.

4.3 Self diagnosable systems:

Self diagnosing capability is becoming an important requirement of systems as their complexity increases and greater emphasis is being placed on their reliability. Design conditions for such systems where the units are capable of testing each other have been studied for permanent failures of units. One such system is the t-fault diagnosable system proposed by Preparata et al. We shall study the capability of the t-fault diagnosable systems to diagnose intermittent/transient (I/T) faults in units.

4.3.1 Preliminaries:

We shall assume that a fault free unit correctly evaluates the tested unit as being faulty or fault free while a faulty unit's evaluation of the tested unit could be incorrect. Under such circumstances, the diagnosis of the faulty units is achieved through the results of the tests performed by the fault free units.

When a unit has an I/T fault, it may have to be tested several times by a fault free unit before correct evaluation can be performed. Therefore, we will assume that after every test routine, an updated syndrome (set of test outcomes) is formed which describes the evaluation of all the units to date. Anytime the updated syndrome corresponds to a consistent set of faults, diagnosis can be performed. Because of the time delay from the initiation of the I/T fault in a unit to its detection, it is likely that additional units could have faults initiated in them in the mean time. Therefore, even if certain units are diagnosed as being faulty, one cannot be absolutely certain that no more units are faulty. Hence, incomplete diagnosis

is inevitable. We shall designate the I/T fault capability of a system, t', as the maximum number of units which could be faulty such that the diagnosis is at worse incomplete but never incorrect, i.e.; a fault free unit is never diagnosed as being faulty.

4.3.2 The Two Partitions

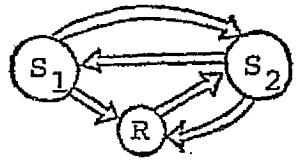


Fig. 4.3-1

S_1, S_2 are 2 sets of units, $S_1 \cap S_2 = \emptyset$ and $|S_1|, |S_2| \leq t$, R being the remaining units in the system. Because the system is t-fault diagnosable, it is not possible that neither S_1 nor S_2 receives any testing links from R. Therefore, there are only 2 possibilities:

i) Only one of S_1, S_2 receives links from R. In such a case, there is a non-zero probability of diagnosing $S_1(S_2)$ as being faulty when infact $S_2(S_1)$ is faulty, if $S_1(S_2)$ receives no testing links from R. In Fig. 4.3-1, if S_2 is the faulty set of units, it is possible to obtain an updated syndrome where, due to insuffient testing all links from R to S_2 are 0-links. If in addition, all links from S_2 to S_1 are 1-links and all links within S_2 are 0-links, regardless of the nature of the links from S_1 to S_2, this syndrome would be a valid syndrome and would correspond to a fault pattern where S_1 is the faulty set of units. So we could diagnose S_1 as being the faulty units when in fact S_2 is the set of faulty units.

ii) Both S_1 and S_2 receive links from R. In such a case, there is a zero probability of diagnosing $S_1(S_2)$ as being faulty when in fact $S_2(S_1)$ is faulty. This is

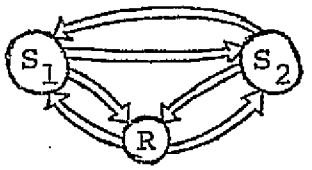
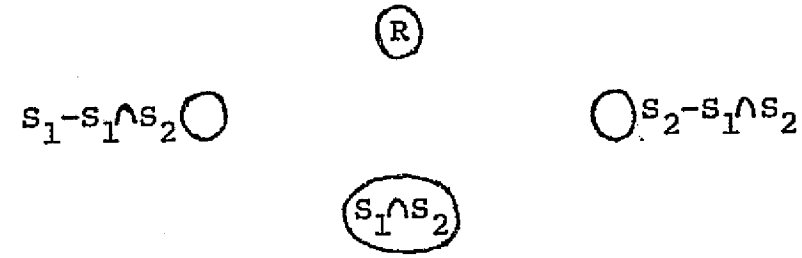


Fig. 4.3-2

because a valid syndrome corresponding to a fault pattern where $S_1(S_2)$ is the faulty set of units would require all the links from R to $S_1(S_2)$ to be 1-links and this could never happen if $S_2(S_1)$ is indeed the faulty set of units.

4.3.3 I/T Fault Diagnosability

We can describe the I/T fault diagnosing capability of a t-fault diagnosable system by an index t'. If the number of faulty units does not exceed t', there is a zero probability of diagnosing a fault free unit as faulty. This requires that given any 2 sets of units S_1 and S_2, $|S_1|, |S_2| \leq t'$, $S_1 \cap S_2 = \emptyset$, both S_1 and S_2 receive at least one link from R. This guarantees that even if S_1, S_2 were 2 sets of units such that $|S_1|, |S_2| \leq t'$ and $S_1 \cap S_2 \neq \emptyset$, $S_1 \cup S_2 \neq S_1$, $S_1 \cup S_2 \neq S_2$, there is a zero probability of diagnosing a fault free unit as faulty. The reasoning is as follows.



Because $(S_1 - S_1 \cap S_2)$ and S_2 are 2 disjoint sets of units with cardinality $\leq t'$, $S_1 - S_1 \cap S_2$ receives at least one link from R.

Therefore, if $S_1(S_2)$ is the faulty set of units, there is a zero probability of diagnosing $S_2(S_1)$ as being faulty because the links from R to $S_2-S_1 \cap S_2$ $(S_1-S_1 \cap S_2)$ would never be 1-links.

Therefore, when a set of units is diagnosed as faulty, there is a 100% probability that all those units are faulty if the number of faulty units is $\leq t'$. Hence the diagnosis is incomplete at worse but never incorrect as far as the faulty units are concerned. If a set of units S_1 is faulty, it is always possible to diagnose only a proper subset of S_1 as being faulty.

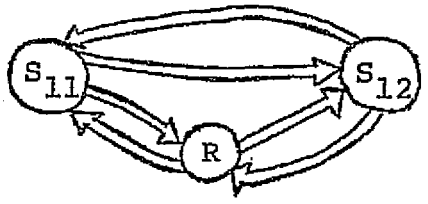


Fig. 4.3-3

In Fig. 4.3-3, S_{11} and S_{12} are proper subsets of S_1 such that $S_{11} \cup S_{12}=S_1$. When S_1 is faulty, due to insufficient testing, it is possible that all the links from R to S_{12} are 0-links, while all those from R to S_{11} are 1-links. If, in addition, all the links from S_{12} to S_{11} are 1-links and all links within S_{12} are 0-links, the diagnosis will designate S_{11} as the faulty units. As long as the number of allowable faulty units is greater than 1, this sort of incorrect diagnosis has a non-zero probability.

There are several partitions of a system into S_1, S_2 and R (as in Fig. 4.3-1) such that S_1 receives no testing links from R. t' has to be less than max $(|S_1|,|S_2|)$ of each such partition

$$\therefore \quad t' = \min_{\substack{\text{over all} \\ \text{partitions}}} (\max (|S_1|,|S_2|))-1$$

We shall now find bounds for t'. Let us define $\lfloor x \rfloor$ as the largest integer smaller than x.

4.2.4 Bounds for Asymmetric Testing

Lemma 1: In any t-fault diagnosable system where no two units test each other, the minimum value of t' is $\left\lfloor \frac{2t+1}{3} \right\rfloor$.

Proof: Let k be the number of units in S_1. The max number of links possible within $S_1 = k\,\frac{(k-1)}{2}$. This would require the smallest number of links incident on S_1, from outside S_1. Since each unit has to be tested by at least t others, the smallest number of links incident on S_1 from outside S_1 is $kt-\frac{(k-1)k}{2}$. If m is the cardinality of S_2, the smallest m will be needed when each unit in S_2 tests each unit in S_1. Therefore, the smallest size of S_2 is given by

$$Km = kt-\frac{(k-1)k}{2}$$

$$\therefore\ m = t-\frac{(k-1)}{2}$$

Since m has to be an integer, in any t-fault diagnosable system, if S_1 has size k, the minimum size of S_2 is $\left\lceil t-\frac{(k-1)}{2} \right\rceil$. It is possible to design a system

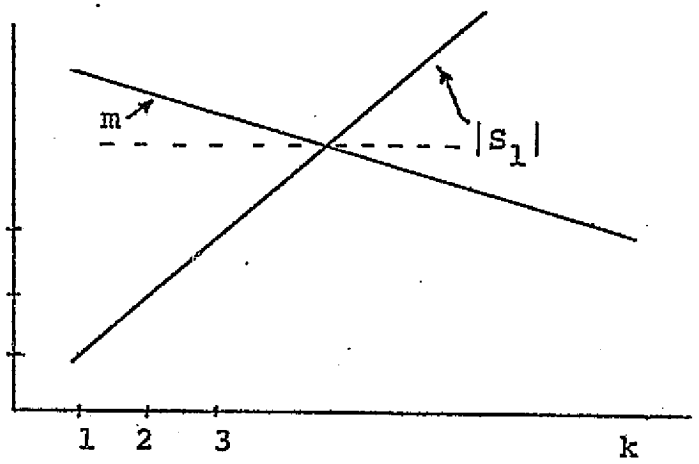


Fig. 4.3-4

which has a partition as in Figure 4.3-1, and these values of $|S_1|$, and $|S_2|$. As the value of k increases, the minimum size of S_2 decreases. The minimum value of max ($|S_1|, |S_2|$) occurs when m=k , as can be seen from Fig. 4.3-4.

$$m = k = t - \frac{(k-1)}{2}$$

$$\therefore \quad m = k = \frac{2t+1}{3}$$

If $\frac{2t+1}{3}$ is an integer , $\max(|S_1|, |S_2|) = \frac{2t+1}{3}$ since, $\frac{dm}{dk} = -1/2$, when $\frac{2t+1}{3}$ is not an integer, for $k = \left\lfloor \frac{2t+1}{3} \right\rfloor$, $m = \left\lceil \frac{2t+1}{3} \right\rceil$.

$\therefore$ Minimum value of $\max(|S_1|, |S_2|) = \left\lceil \frac{2t+1}{3} \right\rceil$.

Since $t' = \min_{\text{over all partitions}} (\max(|S_1|, |S_2|)) - 1$,

$$t'_{min} = \left\lceil \frac{2t+1}{3} \right\rceil - 1$$

$$= \left\langle \frac{2t+1}{3} \right\rangle$$

Q.E.D.

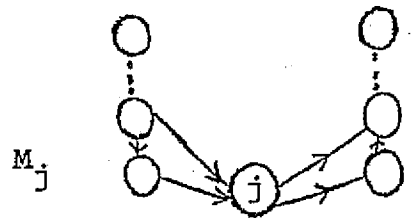
Lemma 2: The maximum value of t' is t-1.

Proof: Since the system we consider is t-fault diagnosable, there exists a fault pattern comprising (t+1) units which cannot be diagnosed. Therefore, t' cannot be greater than t.

Since the system is t-fault diagnosable, there is at least one unit i, which is tested by exactly t units. Hence, the system has a partition as in Fig. 4.3-1, with S_1 consisting of i and S_2 of the t units which test i. Therefore, in any system min(max($|S_1|, |S_2|$)) is at least as small as t. Therefore t' can never exceed (t-1).

We shall show that the maximum value of t' is (t-1) by citing a connection assignment where it is so. Let us consider the $D_{1,t}$

connection. Let us try to form a partition as in Fig. 4.3-1, by starting with a unit j in R.



Now S_1 can only be a subset of M_j, where M_j is the set of units not tested by j. Because of the $D_{1,t}$ connection, any S_1 will have one unit which is not tested by any of the remaining units in S_1. Therefore, S_2 has to have a size of at least t. Therefore t' has a value of t-1 in a system with a $D_{1,t}$ connection.

Q.E.D.

4.3.5 Non-Asymmetric Testing

Now, we shall consider a system which contains pairs of units that test each other. The necessary and sufficient conditions for such a systemto be t-fault diagnosable were formulated by Hakimi and Amin. We shall find bounds for t' for such systems.

Lemma 3: In any t-fault diagnosable system where some pairs of units test each other, t' cannot be less than $\left\lfloor \frac{2t+1}{3} \right\rfloor$ and can be at most, equal to t.

Proof: (I) One of the conditions necessary for a system to be t-fault diagnosable is that for each integer p with $0 \leq p < t$, given any set of units R with $|R|=n-2t+p$, the largest set of additional units S_2 with every unit in S_2 being tested by at least one unit in R, must be such that $|S_2|>p$.

For any given value of p and a partition as in Fig. 4.3-1, since $|S_1|$ decreases as $|S_2|$ increases, the smallest value of $\max(|S_1|,|S_2|)$ occurs when $|S_1| = |S_2|$. Let $|S_2| = p+k$ where $k > 0$. Of all possible values of k, for a given p, there exists only one, k^* when $|S_1| = |S_2|$.

$$|S_1| = n-|R| -|S_2| = n-(n-2t+p)-(p+k^*)=p+k^*$$

$$\therefore 3(p+k^*) = 2t+k^*$$

$$\therefore p+k^* = \frac{2t+k^*}{3} = |S_1|=|S_2|$$

The min mum value of k^* is 1 and there exists a p for which this equality holds.

Therefore, t' cannot be less than $\left\lfloor \frac{2t+1}{3} \right\rfloor$. (II) t' obviously cannot be greater than t. Consider a system with $2t+1 \leq n < 2t+3$. Construct a $D_{1,t}$ system with a bidirectional link between units i and j if $|i-j|=1 \bmod n$. In this case each unit is tested by (t+1) other units. If a unit k is in R, S_1 can only be a subset of M_k, the set of units not tested by k. Any such S_1 will have a unit m which is tested by atmost one other unit in S_1 and a unit n which is tested by at least one unit p not in S_1 such that p does not test m . Therefore, every S_2 has a size of at least t+1 and hence for such a system t'=t. Hence, the maximum value of t' is t.

Q.E.D.

We have established bounds for t'. For any given connection assignment, the exact value of t' can be determined by examining all partitions as in Fig. 4.3-1. We shall now give an algorithm to do this.

4.3.6 Procedure

Let us denote by t'_m , the smallest upper bound on the value of t', based on all the available information at any stage. We know the initial value of t'_m . Then, we examine all Type A partitions, partitions as in Fig. 4.3-1 with $|S_1| \leq t'_m$ & $|S_2| \leq t'_m$, which contain a unit i in R . If there exists such a partition, we update t'_m and repeat. If there exists no such partition we examine Type A partitions which contain a unit j but not i in R and so on till all possible partitions have been examined. After all the partitions have been examined, the current value of t'_m is the value of t'.

Let us define,

$\Gamma_i \triangleq$ set of units tested by unit i

$\Gamma_i^{-1} \triangleq$ set of units testing unit i

$P_{ij} \triangleq |(\Gamma_i \cup \Gamma_j \cup i \cup j)|$

$Q_{ij} \triangleq |(\Gamma_i^{-1} \cup \Gamma_j^{-1} \cup i \cup j)|$

Procedure 1: This will be used to find an upper bound, sufficiently lower than t'_m , on the maximum possible size of S_1 in a type A partition, when a unit i is in R.

Lemma 4: The minimum possible size of $(S_2 \cup R)$ is k only if there is a set C of at least $(k-t'_m)$ units such that for each pair of units $m,n \in C$, $P_{mn} \leq k$.

Proof: Deleted because it is obvious.

Every set C satisfying the condition in the Lemma is a possible candidate for R but $|S_2 \cup R|$ must be evaluated to make sure that it is so. The minimum possible size of $(S_2 \cup R)$ establishes a limit on $|S_1|_{max}$.

We start with a unit i in R and establish a lower limit on k by first finding the smallest k such that there are at least $(k-t'_m-1)$ units such that for every such unit j, $P_{ij} \leq k$. If the lower limit is to be increased, every possible C has to be formed and checked. If any C results in a type A partitions, the value of t'_m is updated, the new t'_m being $(\max(|S_1|,|S_2|)-1)$ for that partition and the procedure is repeated till the value of t'_m is unchanged during the iteration with that value of k. If there exists no type A partition with $|S_2 \cup R|=k$, we can increment k and start all over again. We can do this till we have $|S_1|_{max}$ to a number sufficiently smaller than t'_m.

The various C's can be evaluated by converting a Boolean expression in a product of sums form to a sum of products representation. e.g., if $P_{jn}, P_{jm} > k$, then j and n,m cannot be in the same C. We express it as $(X_j \bar{X}_n \bar{X}_m + \bar{X}_j)$. The C's are evaluated from

$$C_i = \prod_j (X_j \bar{N}_j + \bar{X}_j) X_i \quad \text{where} \quad N_j = \sum_{\substack{q, st \\ P_{jq} > k}} X_q$$

Only those X_j are considered which have $P_{ij} \leq k$ and at least $(k-t_m-1)$ such other terms not in N_j. No reductions are performed on the sum of products. Also note that if a product term contains more than $(k-t'_m)$ literals, all combinations of size $\geq k-t'_m$ are possible C's.

Procedure 2: When a unit i is in R, S_1 can be formed only from M_i, the set of units not tested by i. After establishing an upper bound on $|S_1|_{max}$ using procedure 1, procedure 2 can be used to look for type A partitions.

Lemma 5: A type A partition exists with $|S_1| = |S_1|_{max}$ only if there exists a set of units D in M_i such that $|D| = |S_1|_{max}$ and for every pair of units m,n in D. $Q_{mn} \leq |S_1|_{max} + t'_m$.

Proof: Deleted because it is obvious.

Every set D satisfying the condition in the Lemma is a possible candidate for S_1 in a type A partition. However, each D has to be examined individually to check if $|S_2| \leq t'_m$.

After starting with a unit i in R and arriving at a t'_m and $|S_1|_{max}$ by using procedure 1, we can find a lower value for $|S_1|_{max}$ by first finding the largest number w such that there are at least w units in M_i such that each of them satisfies the condition on Q with at least (w-1) other units. If $|S_1|_{max}$ is to be lowered, every possible D has to be formed and checked. If any D results in a type A partition, the value of t'_m is updated and the procedure is repeated for the same $|S_1|_{max}$. It is also possible to try to reduce $|S_1|_{max}$ by repeating procedure 1 using the new value of t'_m. If there exists no type A partition with $|S_1| = |S_1|_{max}$, we decrement $|S_1|_{max}$ and repeat. We do this till $|S_1|_{max}$ is reduced to 1.

The D's can be found in a manner analogous to that for finding the C's. The Xj's considered are those units in M_i which satisfy the condition on Q with at least $|S_1|_{max}$-1 other units in M_i, the X_q's representing units not satisfying the condition on Q with X_j. Also, if a product term in the sum of products representation has more than $|S_1|_{max}$ literals, only combinations of size $|S_1|_{max}$ are to be considered as candidates for D.

Procedure 2': After we have examined all partitions which contain a unit i in R and are examining partitions which contain a unit j in R, we can try to avoid examining some D's which have already been examined before. We can divide M_j into $M_j \cap M_i$ and $M_j \cap M_i^c$. Every D must have at least one unit from $M_j \cap M_i^c$ in order to be an unexamined one. Therefore, $|S_1|_{max}$ will be determined by the maximum size of D containing a unit from $M_j \cap M_i^c$.

After examining partitions containing units $i_1, i_2, \ldots i_p$ in R, the next unit we pick should be a unit j which has the smallest $P_{i_s j}$ for all j and all i_s in $i_1 \ldots i_p$. We will then partition M_j into $M_j \cap M_{i_s}$ and $M_j \cap M_{i_s}^c$ and use procedure 2 with the exception that D's not containing any units from $M_j \cap M_{i_s}$ are not examined at all.

4.3.7 An Example

We shall now use an example to clarify the algorithm. In Table 1 is given a connection assignment, with the P's and Q's given in Table 2.

Procedure 1: We shall start with unit 3 in R. Initially $t'_m = t-1 = 5$. We need at least (k-6) units with $P_{3i} \leq k$. Therefore, the smallest value of k is 9. The possible candidates for R with $|R \cup S_2| = 9$, are units 3,2,6 and 10. We now form C_3.

$$C_3 = x_3(x_2\bar{x}_{10}+\bar{x}_2)(x_6\bar{x}_{10}+\bar{x}_6)(x_{10}\bar{x}_2\bar{x}_6+\bar{x}_{10})$$

At any stage, we multiply 2 sum of products terms only if they have at least one common x_j. The C's can be easily formed from these disjoint sum of products terms.

$$C_3 = x_3(x_2x_6\bar{x}_{10}+x_{10}\bar{x}_2\bar{x}_6)$$

The maximum possible size of C is 3 and Lemma 4 is not satisfied. Therefore, we check for k=10. Now, the C's can be selected from units 3,2,5,6,10,13,15. We again form C_3.

$$C_3 = x_3(x_2\bar{x}_{10}\bar{x}_{13}+\bar{x}_2)(x_5\bar{x}_6\bar{x}_{10}\bar{x}_{13}+\bar{x}_5)(x_6\bar{x}_5\bar{x}_{11}+\bar{x}_6)(\bar{x}_{10}+ x_{10}\bar{x}_2\bar{x}_5\bar{x}_{15})(x_{15}\bar{x}_{10}\bar{x}_{13}+\bar{x}_{15})(x_{13}\bar{x}_2\bar{x}_5\bar{x}_{15}+\bar{x}_{13})$$

x_{10},x_5 and x_{13} cannot be in any C because every C must have a size of at least 5 and they satisfy the condition on P with only 3 other x_j's. Since there are only 4 possible cardidates remaining, there exists no C satisfying Lemma 4 for k=10. Therefore, the smallest $|S_2 \cup R|$ is greater than 10. Now we can switch to procedure 2.

Procedure 2: Since the minimum size of k is >10, $|S_1|_{max}$ =4. The units not tested by 3 are 2,4,6,8,9,10,12,14 and 15. It can be seen from the table that none of these units has more than 1 unit satisfying the condition on Q. Therefore, there is no type A partition containing unit 3 in R. Now we look for type A partitions not containing unit 3 in i.

4.3.8 Conclusion

We have established bounds on the I/T fault diagnosing capability of t-fault diagnosable systems and given an algorithm to determine this value for any connection assignment. Since this does not take into account the statistics of the I/T fault, this can be looked on as the minimum capability of the system.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	1	0	1	0	0	1	0	0	1	1	1	0	0
2	0	0	0	1	1	0	1	0	0	0	1	0	1	1	0
3	0	1	0	0	0	1	0	1	1	0	0	1	0	0	1
4	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1
5	0	0	1	0	0	0	1	1	0	1	0	0	1	1	0
6	1	0	0	1	0	0	0	1	0	1	0	1	1	0	0
7	1	0	1	1	0	1	0	0	1	1	0	0	1	0	0
8	0	1	0	1	0	0	1	0	0	0	1	1	0	0	1
9	1	0	0	1	1	1	0	1	0	0	1	0	0	0	0
10	1	0	0	1	0	0	0	1	1	0	0	1	0	1	0
11	0	0	1	0	1	0	1	0	0	1	0	1	0	0	1
12	0	1	0	0	1	0	1	0	1	0	0	0	0	1	1
13	0	0	1	0	0	0	0	1	0	1	1	1	0	1	0
14	1	0	0	0	0	1	1	1	1	0	1	0	0	0	0
15	1	0	0	0	1	0	1	0	1	1	0	0	1	0	0

(m,n) = 1 iff m is tested by n

Number of units = 15 t = 6

TABLE 1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	x	12	12	10	12	9	13	12	10	11	12	13	11	12	12
2	11	x	9	9	9	8	9	11	9	11	10	10	11	9	7
3	11	13	x	12	10	9	11	11	11	9	11	11	10	11	10
4	10	10	12	x	12	9	12	12	11	11	11	12	11	11	11
5	10	10	12	12	x	11	10	13	11	11	11	13	11	11	10
6	10	12	11	10	11	x	11	10	8	10	11	12	11	11	10
7	12	12	12	12	11	10	x	14	11	11	12	14	12	11	10
8	11	10	10	10	12	11	13	x	12	12	12	12	13	12	13
9	10	11	11	11	12	10	11	11	x	11	13	13	13	11	10
10	11	12	11	10	11	9	11	11	10	x	13	12	10	11	11
11	10	11	11	11	10	12	12	10	12	12	x	12	12	11	11
12	12	10	10	11	11	13	13	10	12	11	10	x	13	12	10
13	9	11	11	10	9	10	12	11	12	10	10	12	x	11	12
14	11	11	11	11	11	11	11	11	9	10	12	11	11	x	11
15	11	11	12	11	10	11	10	12	11	11	10	10	12	11	x

(m,n) , $m>n = P_{mn}$

(m,n) , $m<n = Q_{mn}$

TABLE 2

3-1 M. A. Breuer, "Testing for Intermittent faults in digital circuits", IEEE Trans. Comput., vol. C-22, No. 3, pp. 241-246, March 1973.

3-2 S. Kamal and C. V. Page, "Intermittent faults: a model and a detection procedure", IEEE Trans. Comput. (Special Issue on Fault-Tolerant Computing), vol. C-23, pp. 713-719, July 1974.

3-3 F. C. Hennie, "Fault detection experiments for sequential circuits".

3-4 T. L. Booth, ' Sequential Machines and Automata Theory', John Wiley,1967.

3-5 F. P. Preparata, G. Metze and R. T. Chien, "On the connection assignment problem of diagnosable systems", IEEE Trans. Comput., vol. EC-16, No. 6, pp. 848-854, December 1967.

3-6 S. L. Hakimi and A. T. Amin, "Characterization of the connection assignment of diagnosable systems", IEEE Trans. Comput., pp. 86-88, January 1974.

3-7 T. L. Saaty, 'Elements of Queueing Theory', McGRAW-HILL, 1961.

5. Tolerance Strategies

5.1 Introductory Statements:

Given that a machine is imperfect, information may or may not be known about the types of phenomena which will produce I/T faults in it. In either case, the problem is the same: how can the circuitry be arranged to eliminate the I/T faults or the effects of the I/T faults? Is enough information available to do this? If not, what can be done to make possible this goal? This report is intended to address these problems. Material pertinent to this subject is listed in references 5-1 through 5-10.

5.2 I/T Fault Intolerance:

The first method is to attempt to eliminate the I/T faults. This approach is called fault intolerance. The most reliable components are used in constructing equipment to lower the probability of overall failure for a given mission time. No redundancy is employed, and in general, all components must function properly for the system to operate. The overall reliability is the product of the component reliabilities, and since these have been arranged to be as high as possible, the system will have a high reliability.

There is a limit to how much the reliability can be increased, both physically and economically. The I/T fault intolerance approach adds nothing new to the structure of a system, and better reliabilities are required, so the next approach is considered.

5.3.2 Sequential Circuits:

When dealing with sequential circuits, I/T faults may affect them after the fault disappears. Wakerly points this out in [5-2]. If triple modular redundancy is used on a level low enough so that the replicated modules are combinational, the overall circuit may be sequential in nature but the redundancy scheme will filter out all single I/T faults per module. If it is desired to triplicate modules which are themselves sequential, then a build-up of faults can occur. An I/T fault in a sequential machine can change the machine's state. This can clearly lead to improper execution. Therefore, applying triple modular redundancy to sequential modules is more complicated than to combinational modules. However, it has become increasingly more important to apply redundancy to sequential modules. Breaking a circuit down into such small portions so that they are all combinational results in modules which are comparable in complexity to the voters themselves (not very complex). It is doubtful that fault tolerant machinery constructed in this manner would be much more reliable than the original non-redundant machine.

The level of complexity of commercially available integrated circuits is constantly rising, and due to this constraint many times it is impossible to break a circuit down into combinational modules. Wakerly shows that the modules

5.3 I/T Fault Tolerance:

The idea behind fault tolerance is to utilize redundant circuitry to eliminate the effects at the output to internal faults. The fact that faults will occur is accepted; only through redundancy can the effects be eliminated.

5.3.1 Faults in Combinational Circuits:

In a combinational circuit, the worst that an I/T fault can do is to cause the circuit to produce erroneous output while the fault is present. If measures are taken to insure that these erroneous outputs are masked, the task is accomplished. The job is simplified because there can be no after-effects of the I/T faults.

Triple modular redundancy has been used at this level, as shown in Figure 5-1. If no more than one module in a triple experiences a fault at the same time, the voter output will be correct. Depending upon the voter reliability, single or triple voter schemes are used. With combinational circuits, triple modular redundancy may be used at any level. Circuits may consist of a single overall triplication, or of triplication of modules which are triplications of modules, etc. Note that while transient faults have been implied in this discussion, single permanent faults per module will also be masked. Naturally, another I/T fault in another portion of that same group can cause system failure.

must be restorable, and that restoring inputs must be applied in normal operation. Some circuit structures directly lend themselves to this automatically in normal operation, while for others special resynchronizing inputs must be devised and applied. In the latter case, the problem is that only single faults per module can be guaranteed not to affect the output, between applications of the resynchronizing inputs.

If enough is statistically known about the I/T faults, then the overall reliability can be computed for various resynchronizing input frequencies.

5.3.2.1 Larger Scale Modularization:

In [5-3] Wakerly describes a method of utilizing triple modular redundancy with microprocessors with associated memories. Considerable discussion is given to the voter placement problem. The resultant system constantly runs a program of resynchronizing routines which restore the registers on the central processor chip as well as the external memory. To prevent a single microprocessor which has gone awry from deteriorating the system periodically the processors are restarted.

Wakerly's scheme will certainly work, but it has limitations. Due to the choice of placing the voters after the memory instead of between the processor and the memory, if a failure occurs within a processor, its faulty data can be placed anywhere within its external memory. To keep the memory clean, the entire memory must periodically be completely rewritten.

It is not sufficient to wait until an error is detected and only rewrite the present address location. The problem is that for any reasonably large memory it will take a considerable amount of time to totally rewrite. Furthermore, the cleansing must be done fairly frequently to capture as many I/T faults as possible. This leaves very little time for the processors to perform the original task assigned them. For these reasons it believed that the triple modular redundancy scheme for microprocessors described in [5-3] needs improvement.

5.3.2.2 Special Problems with Microprocessor Implementations

To a limited degree, fault tolerance implementations for low level circuits exist. Designing a random logic-circuit to be fault tolerant is possible primarily for two reasons: analysis and test set generation. Due to the low level nature of the circuit the effects of any particular fault can be analyzed. It is only because I/T faults injected into a circuit can be analyzed that it then becomes possible to generate test sets. The problem of test set generation is complex even in low level circuit descriptions, however there are techniques available which give best solutions, at least in theory if not in practice.

There are also practices devised suitable for connecting large computers together to provide some measure of fault tolerance. Even in this case, it is not clear what a good measure

of fault tolerance is. It is also necessary to specify what the faults are. I/T and permanent faults may be considered. The problem under investigation falls between the above two categories. How can microcomputers be best configured to provide fault tolerance? Microprocessors have peculiarities which must be considered in planning tolerance strategies. The relative price of each component changes what should be duplicated in the overall system. The capabilities of microcomputers are not as great as large computers, and strategies developed for large computers are often not at all suitable for implementation with microprocessors.

5.3.2.2.1 Microprocessor Redundancy Schemes

Redundancy schemes applied to microprocessors fall into two categories: those which are specifically designed for the microprocessor, and those which are general in nature and are originally intended as reliability schemes for larger processor systems, but are adapted to microprocessor based processors. Ideas based on larger systems which are later applied to microprocessor systems sometimes make little sense. There are some features of microprocessors which must be taken into account when devising tolerance schemes. The first is complexity. If the redundancy scheme uses so much extra hardware so as to overshadow the amount of hardware that the microprocessor itself has, then the reliability of the hardware added for the extra reliability will probably be such that when compared to the reliability of the original nonredundant system, little will be

added, if indeed the so called reliable system is not less reliable than the simple system. Another feature which makes the microprocessor very different from larger processors is speed. The microprocessor is quite slow when compared with other large computers and minicomputers. Elaborate reconfiguration schemes implemented in software may take a long time to execute, and depending on the application, may or may not be suitable.

The report from Ultrasystems [5-7] on reconfigurable computer systems is quite complete. Many of the ideas presented there can be adapted to microprocessor designs. They categorize their approaches as mostly software, hardware aided software, and mostly hardware. The mostly hardware proposals involve a large amount of hardware, and would not be desirable to implement on a processor system using microprocessors as the processor elements, as the complexity of the extra hardware is large compared to the relative small amount of hardware which the individual microprocessors require. Reliability is closely connected with the amount of interconnections, and hardware designs involving large quantities of integrated circuits demanding many interconnections tend to become unreliable. Therefore, any hardware added to microprocessors for I/T fault tolerance should be small to moderate when compared with the complexity of the microprocessor itself.

The major thrust of microprocessor based controllers is to replace hardware with software. Continuing in this manner, it would seem that any fault tolerant microprocessor system should use as little added hardware as is possible for the

added fault tolerance. The techniques mentioned in [5-7] under mostly software can be adapted reasonably well to microprocessor systems. This ranges from minimal additions to vast reconfiguration software monitors. The large monitors should be avoided with microprocessor implementations since microprocessors do not usually have a large amount of memory to hold elaborate programs, and very elaborate monitor programs would tend to take a long time to execute on microprocessor systems. Nevertheless, there are some very good techniques discussed in [5-7], and those which can be used on microprocessor systems will be briefly outlined here.

The applications program is broken into program segments. The choice of program segments can greatly affect the reliability of the end product. No more than one output statement should be in any one program segment, and large calculations should be broken down into several segments. A set of variables called the state vector is associated with each program segment. The state vector is such that in order to leave a particular segment with the correct data, all that should be needed is the state vector input to that segment. Naturally, the larger the program segment, the larger will be the state vector. If the program is operating properly, and if the state vector is correct, then the output of that program segment should be correct. That data can then be used as the state vector for the next program segment. Comparison of state vectors is the major reliability addition made in multiple processor imple-

mentations in this approach. Multiple processors all execute the same program, and produce state vectors. Before a program segment is initiated, the state vectors of each processor are compared. If all agree, the processing continues in the normal manner. If a disagreement is found, one of several things will occur. If there are more than two processors, then program rollahead can be used. A vote is taken on the state vectors, and any processor which disagrees with the outcome of the vote has its state vector forcibly changed to what the others have. Program execution continues as normal. If there are only two processors or if there is a tie vote with an even number of processors, rollback must be used. It is known that a mistake has occurred, but it is not known where. The previous state vector is reloaded, and program execution of the prior program segment is repeated. Rollback, of course, takes longer than rollahead, and the larger the program segment, the longer the recovery time when rollback is used.

Many other considerations come into play with multiple processor systems, such as keeping track of the frequency of errors in each module, knowing to remove a processor from the system, and trying to restart faulty processors at a slow rate. Implementations can be made on microprocessors with these techniques.

Reliability schemes have been specifically developed for microprocessor systems. Wakerly [5-3] describes a triple modular redundancy system for microprocessors. He replicates

processor/memory pairs and adds voters. He discusses where the optimum place is to put the voters, and decides that the voters on the output of the memory is best. This system is simple, and the hardware automatically assures that the processors receive the proper data from memory. The biggest problem with this implementation is that it is possible for memory locations to be changed to bad data, so that periodically it is necessary to read and rewrite the entire memory contents. This cleans up any errors, however, for larger memories it could take a considerably long period of time to execute. Even this minimal arrangement requires a lot of circuitry: 24 voters for an 8 bit machine. Reliability curves can be provided for the various systems.

5.3.2.2.2 Possible Use of Bit Slice Microprocessors for Tolerance

Many microprocessor fault tolerance approaches utilize a modular structure. Processors are replicated and comparisons are made between them. There is a lot of overhead in these designs for the limited amount of reliability gained, and an alternative approach is desired. An interesting possibility is the use of the bit slice microprocessor designs for this purpose. What would be significantly useful would be a tolerance structure which built a sixteen bit microprocessor out of five four bit slice microprocessors, leaving one extra for redundancy. This would be an overhead of only

25% as versus 200% for a triple modular redundant system. However, the only part of the bit slice microprocessor design which is actually modular in the slice sense is the register, arithmetic, and logic unit. The contemporary bit slice processor chips are powerful and include registers and shifters. These are useful for multiplication and similar powerful instructions. These are sequential in nature, and this in itself is a problem. To make a transparent redundant system from these devices with voters on the outputs of the RALUs would require that the sequential portions of them not be used. This destroys most of their power, and is unreasonable. Even if this were not a problem, the RALU is a minor portion of the overall circuitry of which the bit slice microprocessor is composed, and it is not reasonable to make the RALU tolerant while not doing anything to the rest of the circuitry to improve the fault tolerance. If the bit slice microprocessors included most of the slice properties throughout most of the circuitry, then perhaps good advantage could be made of them for fault tolerance implementations. No way is seen to do this with present bit slice microprocessors which is any better than non bit slice microprocessors, and no way is evident to design a new type of bit slice microprocessor which would allow one to take advantage of the slice properties for fault tolerance implementations.

5.3.2.2.3 The Problem of Determining the Effectiveness of Designs

There are many schemes proposed to achieve fault tolerant microcomputer systems, most incorporating multiple processors for the redundancy needed for the fault tolerance. The present situation is such that short of construction and operation in hostile environments, there is no good way to determine the relative effectiveness of the various approaches - indeed, even to verify that a particular implementation will perform as claimed. The reason that there is so much difficulty in determining these parameters is that the fault class being considered is phenomenally large - namely, all intermittent/transient faults. Parameters to be determined are such things as the sensitivity of the strategy to burst type faults, dependent faults, how long the recovery times are for different faults, the longest expected and the mean recovery times, and catastrophic faults. Very little is understood about the various categories of faults which can be utilized in analyzing fault tolerant approaches to microprocessor designs. What is needed then is a general model for I/T faults. It is not clear, however, that a general model for such faults exists. Realizing this problem, the research emphasis has been shifted from fault tolerant strategies for microprocessors to that of measuring and modeling the intermittent/transient faults which can influence microprocessor based systems.

5.3.2.2.4 Plan for Resolving the Problem

The work thus far has been directed towrads obtaining the means to further examine proposed multiple processor schemes. This is being accomplished somewhat experimentally. A microprocessor system has been constructed for this use. The first step was to subject the processor to various induced faults. These induced faults are faults that are supposed to copy the real world intermittent/transient faults to which such a microprocessor system might reasonably be expected to be exposed. Data which is to be collected from the Lear jet experiments to be conducted in Florida will be a more realistic guide in choosing realistic faults to induce. In fact, there is little distinction between actual fault situations and induced faults. The faults which a circuit is exposed to in normal operation are the faults of interest, but by the very nature of the fact that a study of those faults is being made, the circuit under test is not in normal operation. This is particularly true in view of the fact that one cannot wait around for the natural faults to manifest themselves, but must force the circuit into a faulty situation. Any faults to which a network is purposefully exposed are called induced faults. The induced faults are as close an approximation to real faults that the circuit would normally be exposed to as is possible. Naturally, the fault rate will be higher in the induced faults than in a mildly hostile

environment, but this is necessary in order to accomplish our goals.

The first step in the experimental procedure was to subject the test microprocessor to different induced fault situations. Such things as high or low power supplies, noisy power supplies, heat, and electromagnetic interference are possible induced faults which we have attempted to consider. Our choices will be soon coupled with the Lear jet experiments which will ultimately be the guide in choosing and implementing the induced faults. The experiments we are performing can be described as follows: It is not first of all known how these induced faults will affect the processor - these are decentralized faults, and the individual lines actually driven to faulty values are not known. A diagnostic program which will give data on the faults as they occur is running on the processor during the time that faults are being induced. The purpose is to collect enough data on each induced fault so that the data will be a signature of each fault, and give an indication of the severity of that fault. When enough data is collected on each induced fault, the fault emulation stage begins. This work is still in its initial stages, but, briefly, it characterizes fault emulation.

Induced faults will be approximated by emulated faults. Emulated faults are faults for which it is known how, and

more importantly, where they affect the circuit. For example, suppose a noisy power supply causes intermittent fault situations to occur in the processor. The cause of the fault is known, the power supply. However, it is not known where that fault is affecting the circuit to cause the failures. It could be internal to any of the integrated circuits, and may not even be directly observable on the pins of the packages. A logic level somewhere in the processor must be changed to cause the failure, but it is not known which level has been changed to the other, nor which line on which the level has been changed. Emulated faults will be postulated for each induced fault, and the same tests will be run with each emulated fault as was done with the induced faults. Comparison of the emulated fault data and the induced fault data will serve as a feedback loop to improve the accuracy of the postulated emulated faults. In this manner, a set of emulated faults will be constructed which in a sense are a model of the induced faults which are a close representation of the actual intermittent/transient faults encountered in a real situation. The emulated faults for a specific induced fault may be used as a model of that induced fault because the direct effect of the emulated faults are known in the processor. This allows the entire system to be simulated on a large computer, and evaluations can be made of the effectiveness of the fault tolerance strategy. The

emulated faults do not have to be known deterministically, but may only be known statistically. The proposed way to generate these emulated faults is to have some intelligent device (minicomputer or specialized hardware) drive various fault injection networks which are imbedded throughout the microprocessor. This will give a large degree of freedom in arriving at emulated faults in the hope that a match can be obtained.

The plan is to construct a microprocessor, expose it to various hostile environments, measure the effects, postulate an equivalent emulated fault, expose it to that proposed emulated fault, measure the effects, and arrive at a reasonably approximate class of emulated faults which can be used to model a large class of real hostile fault environments. The concept of an emulated fault includes any faults which can be injected into the microprocessor in a manner such that its direct effects are known, either deterministically or statistically. These effects must be first order effects, meaning that it is clear that the particular fault emulation is directly causing some effect, and is not indirectly caused by that injection.

On the other side are induced faults. These are an attempt to expose the processor to a real hostile environment without waiting for the processor to experience intermittent/transient faults on its own. In order to test the validity of the postulated emulated faults, faults must be induced into

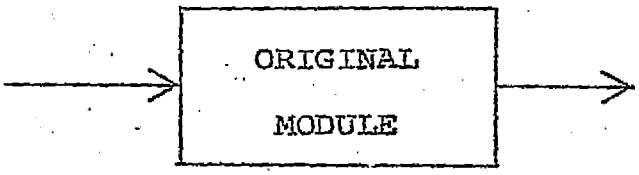
the processor which can be expected to closely parallel a real hostile environment. How these induced faults cause failures in the processor need not be known - indeed, this is the entire point of the current research; in general this is not known. In short, the induced faults are being substituted for real world faults, and the emulated faults are effectively modeling the induced faults, though perhaps not in a conventional manner. The whole purpose of this procedure is to produce a methodology to test the effectiveness of various fault tolerance microprocessor strategies.

A suitable microprocessor for these proposed experiments has been constructed. Figure 5-2 shows the configuration. The processor has been constructed on plug boards, so that it may be easily modified. This allows any of the lines to be broken for the insertion of the fault injection networks. The processor is done, and the testing program is to be developed and checked. Figure 5-3 shows the diagnostic program. When running, it periodically prints a message to indicate that it is still working. Implementation on the 8080 microprocessor has the advantage that if the stack gets changed, and a non-memory location is used for the stack, the processor will jump to a nonexistant location, and receive the data hexadecimal FF, which corresponds to the interrupt instruction on the 8080. Advantage is taken of this in the diagnostic program, and in normal operation the interrupt instruction is never reached.

A count is kept of the times an interrupt is received as an indication of the processor running awry. The checking program checks all of the microprocessor instructions, and prints a message if execution is improper, perhaps also with a time tag. It includes memory checks.

The exact nature of the induced faults must be chosen, and suitable circuitry designed and built to create the faults. All of the data collecting must be done, and then the fault injection networks must be built for the emulations. Strategies for the emulations must be developed, and again data must be collected until the end goal is reached. Figure 5-4 shows the proposed method for emulating faults.

This plan will yield the needed information on intermittent/transient faults as microprocessors are affected, and permit further investigation of microprocessor tolerance structures.



VOTER LOGIC

INPUTS	OUTPUT
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	1

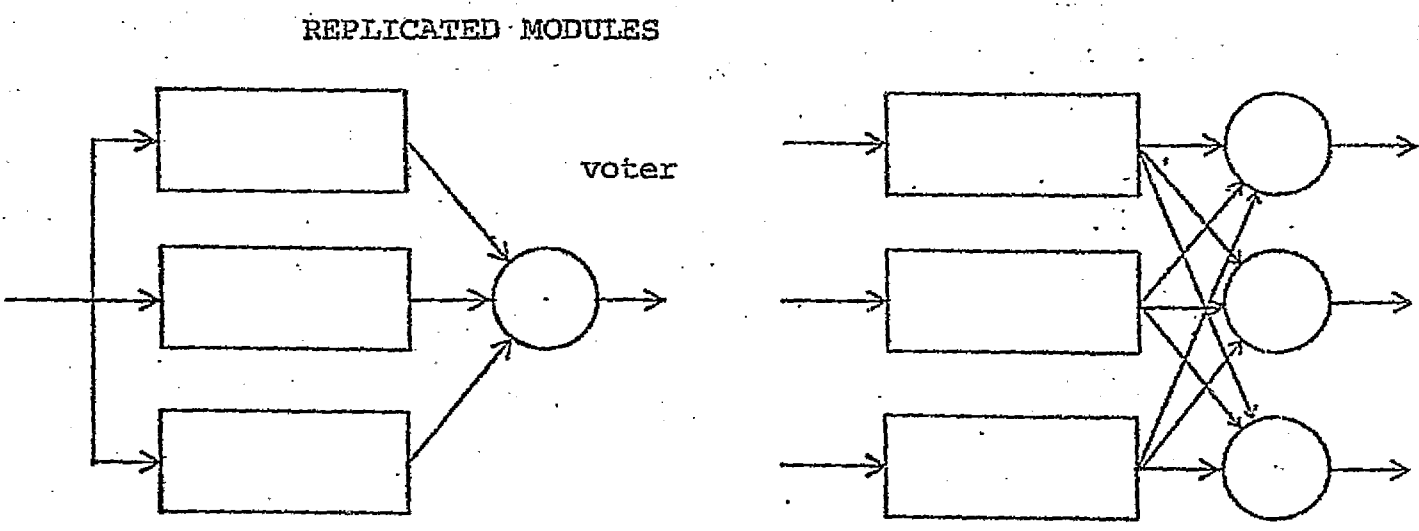


Figure 5-1: Triple Modular Redundancy

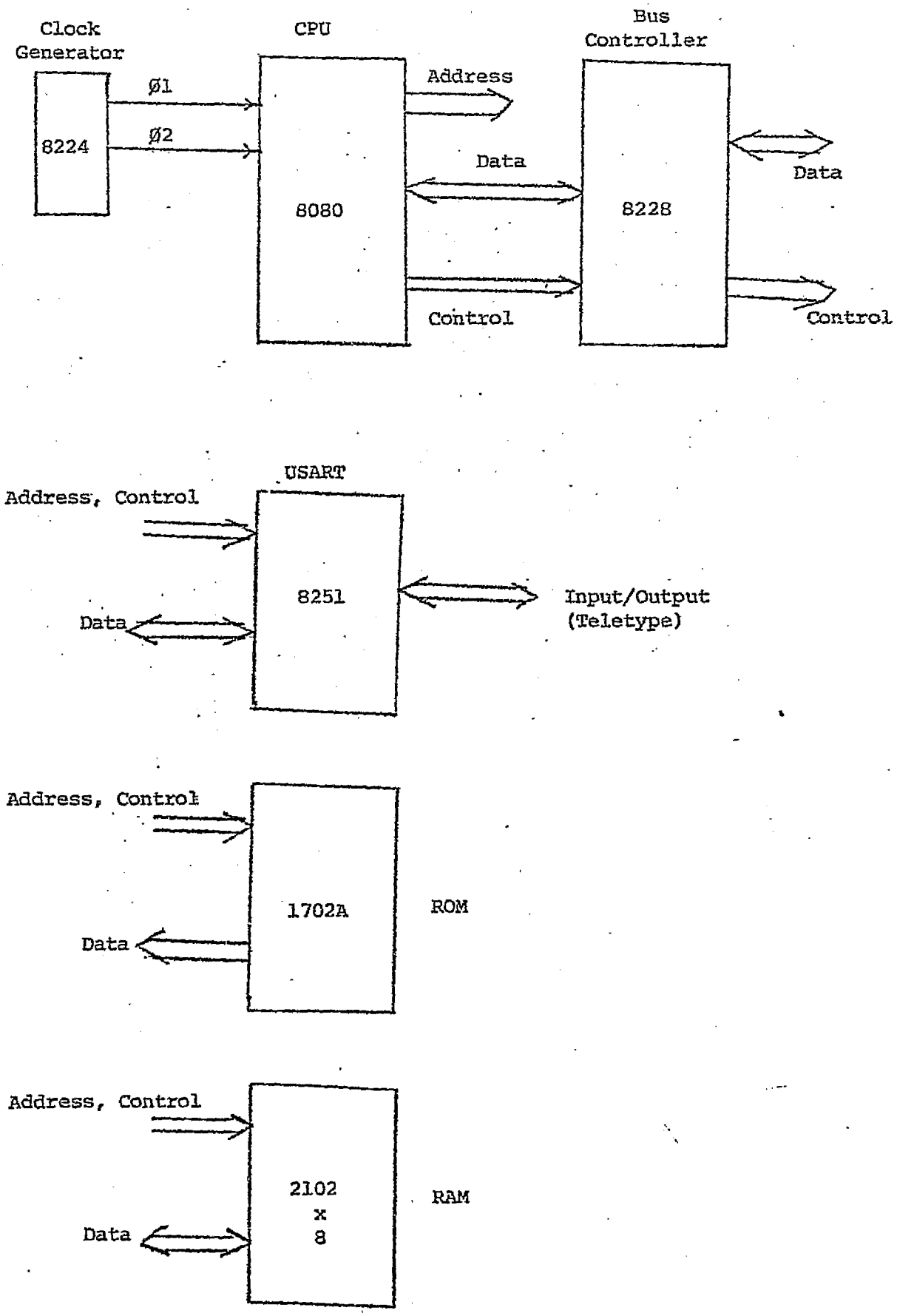


Figure 5-2: Processor

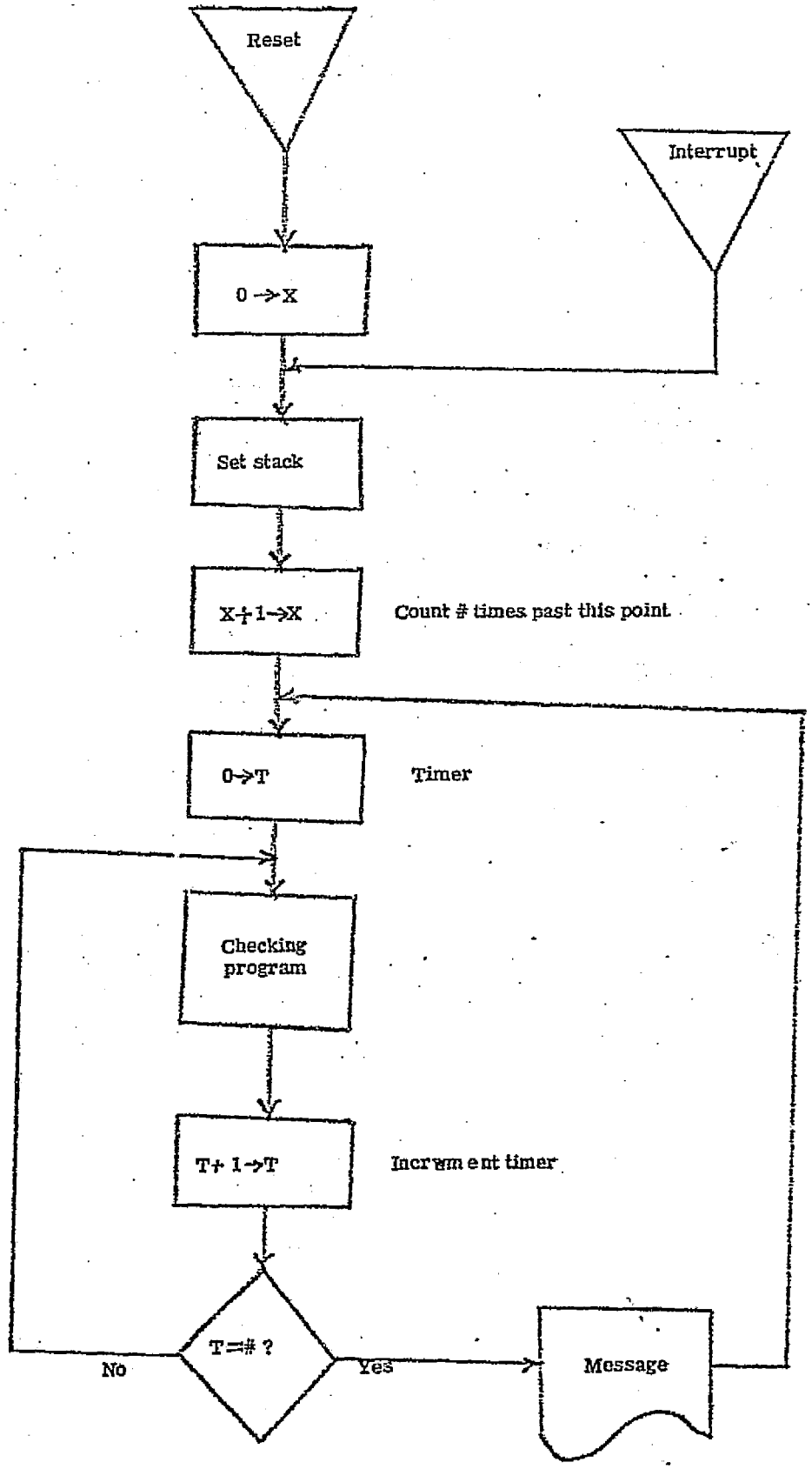


Figure 5-3: Diagnostic Program for Fault Detection

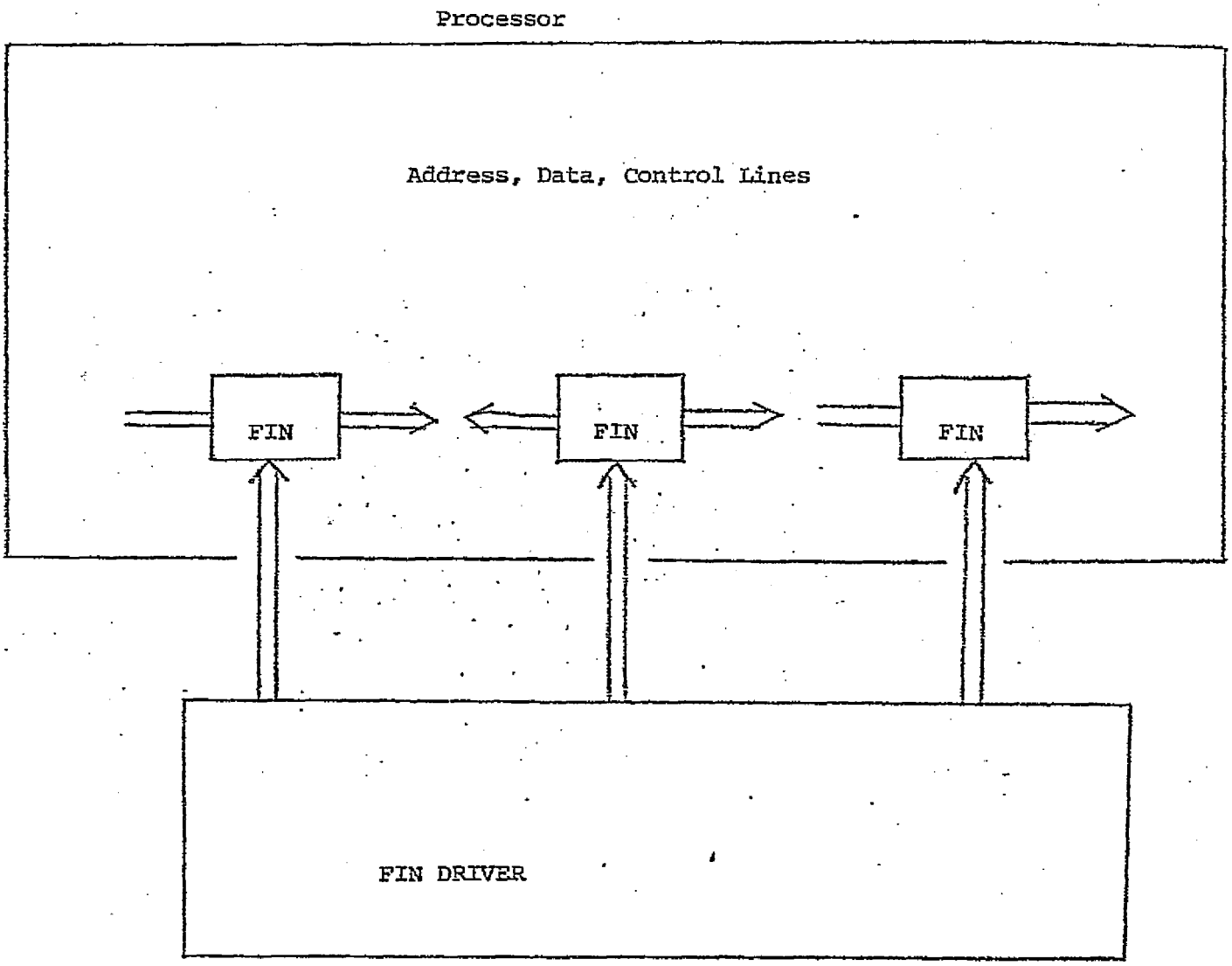


Figure 5-4: Fault Emulation Experiment

5-1 P. M. Merryman, A. A. Avizienis, "Modeling Transient Faults in TMR Computer System", Proceedings 1975 Annual Reliability and Maintainability Symposium.

5-2 John F. Wakerly, "Transient Failures in Triple Modular Redundancy Systems with Sequential Modules", IEEE Trans. on Computers, May 1975.

5-3 John F. Wakerly, "Reliability of Microcomputer Systems Using Triple Modular Redundancy", IEEE Trans. Comp., April, 1975.

5-4 J. A. Abraham and D. P. Siewiorek, "An algorithm for the accurate reliability evaluation of Triple Modular Redundancy networks", IEEE Trans. Comp., vol. C-23, No. 7, pp. 682-692, 1974.

5-5 John F. Wakerly and E. J. McCluskey, "Design of Low-cost general-purpose self-diagnosing computers", Technical Note No. 38, Digital Systems Laboratory, Stanford University, Stanford, Ca.

5-6 John F. Wakerly, "Checked binary addition using check symbol prediction and checksum codes", Technical Note 39, Digital Systems Laboratory, Stanford University, Stanford, Ca.

5-7 "Definition and trade-off study of reconfigurable airborne digital computer system organizations", Final report, November 1974.

5-8 A. Avizienis, "The Methodology of Fault-Tolerant Computing", Proc. First USA - Japan Computer Conference, 1972.

5-9 W. G. Bouricius, et al., "Reliability Modeling Techniques for Self-Repairing Computer Systems", Proc. ACM 1969 Am. Conf.

5-10 W. G. Bouricius, et al., "Reliability Modeling for Fault-Tolerant Computers", IEEE Trans. Comp., vol. C-20, No. 11, November 1971.

Appendix

AN EFFICIENT FAULT DIAGNOSIS ALGORITHM FOR SYMMETRIC MULTIPLE PROCESSOR ARCHITECTURES

INTRODUCTION

Consider a general model of a multiple processor architecture consisting of n digital modules denoted U_0, U_1, $\cdots$ U_{n-1} and some associated interconnection design, denoted $D_{\delta,t}$. These modules, for example, could be n processors implementing a segmented algorithm [6]. Regardless of the use of the multiple processor architecture, we will assume that each U_i is capable of testing the other U_j's to which it is directly connected for some specified cl of faults. If a module contains any such fault we will refer to it as faulty. The problem we will study in this paper is the diagnosis of an existing fault situation among the modules given their respective testing results. This problem is not new and has been examined elsewhere in the literature [1,3,4,5,7,8]. The results to be presented here represent a new approach to such diagnosis. In particular, the diagnosis procedure described will be seen to be sufficiently straightforward to be easily implementable on a simple processor, e.g., a microprocessor, and for a proper interconnection design among the processors and upper bound on the number of simultaneous faults which can occur, will always yield the correct diagnosis of the existing fault situation.

PRELIMINARIES

Given n modules $U_0, U_1, \cdots, U_{n-1}$, we will denote the modules which U_i tests by $U_{f(r,i)}$, $r = 1,2,\cdots,t$, where $f(r,i) \in [0,1,\cdots,n-1]$, $i = 0,1,\cdots,n-1$. For convenience, we will always assume that U_i tests itself and, regardless of its state, concludes that it is fault free. The outcome of the test of module $U_{f(r,i)}$ by module U_i will be denoted $a(i,f(r,i))$ where

$$a(i,f(r,i)) = \begin{cases} 0, & \text{if } U_i \text{ concludes that } U_{f(r,i)} \text{ is fault free} \\ 1 & \text{otherwise.} \end{cases}$$

It should be noted that the conclusion of, say, U_i regarding the state (faulty or fault free) of the modules to which it is connected is only reliable if indeed U_i is fault free. If with each module U_i, we associate a test table B_i, $i = 0,1,\cdots, n-1$, where B_i represents the conclusion of U_i regarding the states of all the modules, we have the problem of determining the existing fault situation based on the available test results. Whether or not this is feasible clearly depends on the number of faults and the interconnection design. We will assume in the following that at most t modules can be simultaneously faulty and that every module is tested by at least t other modules. Under some assumptions on the interconnection design, Preparata, Metze and Chien [7] have shown that it is feasible to diagnose any valid fault situation. However, the diagnosis algorithms which have been proposed to do so are quite complex [1,4,5]. We propose here a new diagnosis algorithm for this problem. For the purpose of explanation we will assume in the following that the interconnection design between the modules is the so-called D_{1t} design of [7],

wherein there is a testing interconnection from U_i to U_j if and only if $j - i = m$ (modulo n) and m assumes the values from 1 to t. The results presented here have been extended to more general interconnection designs, but since they are descriptively cumbersome, these extension will not be detailed.

DIAGNOSIS ALGORITHM

Each test table B_i has components $B_{i,0}, B_{i,1}, \cdots, B_{i,n-1}$ where $B_{i,j}$ represents the conclusion of module U_i regarding the state of module U_j. If module U_i "believes" that module U_j is fault free, then $B_{i,j}$ is set to the value 0, otherwise $B_{i,j}$ is set to the value 1. Suppose that $B_0, B_1, \cdots, B_{n-1}$ are complete in the sense that every module has a conclusion regarding the state of each of the modules U_i, $i = 0,1,\cdots,n-1$. We will assume here that if a module is fault-free, its corresponding table is correct.

Lemma 1 : There exists at least n-t of the B_i tables which are identical.

Proof: Since at most t modules are faulty, and since a table corresponding to a fault-free module correctly describes the fault situation, the theorem follows.

Lemma 2 : If there exists only one set of identical tables $B_{i(1)}, B_{i(2)}, \cdots, B_{i(s)}$, such that $s \geq n-t$, then each of these tables in this set correctly describes the existing fault situation.

Proof: We already know that there exists at least n-t correct and therefore identical tables. Therefore, if only one set of identical tables has a cardinality larger or equal to n-t, this set must consist of the correct tables.

It should be clear that no conclusion can be made regarding the fault situation if there exists more than one set of identical tables with cardinality larger than or equal to n-t.

Theorem 1 : Suppose that $n \geq 2t + 1$; then there exists one and only one set of identical tables with cardinality larger than or equal to n-t.

Proof: Suppose that $n \geq 2t+1$ and assume that there exist two sets of identical tables of cardinality n_1 and n_2 respectively.

Assume $n_1 \geq n-t$

and $n_2 \geq n-t$.

Then, $n_1 + n_2 \geq 2n-2t$.

We know that $n \geq n_1 + n_2$ and therefore

$n \geq 2n -2t$.

This inequality, used in conjunction with $n \geq 2t + 1$ yields

$$2n \geq 2n + 1$$

and we conclude that we cannot have two sets of identical tables of cardinality larger than or equal to n-t when $n \geq 2t + 1$.

At this point we need an efficient procedure to build the complete n tables $B_0, B_1, \cdots, B_{n-1}$ such that if module U_i is fault free, then the table B_i reflects accurately the fault situation of the multiple processor architecture. Such an algorithm is presented in the following to compute the tables $B_0, B_1, \cdots, B_{n-1}$.

Algorithm 1: Let i in [0,1,...,n-1] and t in [1,2,...,n-1] be given.

Step 0: Set $B_{i,m}=0$ for $m=0,1,\cdots,n-1$, set j=i, set k=i+1 and set $N_F=0$.

Step 1: If $N_F \geq t$, stop; else, go to Step 2.

Step 2: If k=i, stop; else, go to Step 3.

Step 3: If a(j,k) = 1, set $B_{i,k}=1$, set $N_F=N_F+1$ and go to Step 4; else, set j=k and go to Step 4.

Step 4: Set k=k+1 and go to Step 1.

Notes:

(i) All additions are performed modulo n;

(ii) We assume a $D_{1,t}$ interconnection design, i.e., $f(r,j)=j+r$ (modulo n); and therefore, we use the notation $a(j,j+r)$ instead of the more general notation $a(j,f(r,j))$.

Theorem 2: If a $D_{1,t}$ interconnection design is used, if the maximum number of faults which may occur is t and if module U_i is fault-free, then the table B_i constructed by the algorithm accurately reflects the existing fault situation.

Proof: We need to show that the algorithm is well defined and that it produces tables B_i which are correct whenever U_i is not faulty. The technique we use to prove the theorem is based on the use of invariant assertions as described in [2] (see Fig. 1).

We assume that a $D_{1,t}$ interconnection design is used, i.e., module U_i tests the modules $U_{i+1}, U_{i+2}, \cdots, U_{i+t}$. The algorithm uses the quantity $a(j,k)$ which contains the result of the test of module k by module j. It follows that the algorithm is well defined if and only if j and k are related by

$$k = j+r$$

where r is some integer in $[1,2,\cdots,t]$.

Assume that before executing Step 3, the following assertion holds:

(A1) $$j+1 \leq k \leq j+1+N_F .$$

Then it can be shown that (A1) still holds after the execution of Step 4. Clearly (A1) is satisfied by the initial values given to

j, k and N_F and therefore we conclude that (A1) is always satisfied before the execution of Step 3.

It is only possible to reach Step 3 if

$$N_F < t$$

It follows that just before the execution of Step 3, the quantities j and k are related by the assertion

(A2) $$j+1 \leq k \leq j+t$$

which shows that the algorithm is well defined.

The first part of the proof showed that the algorithm is well defined. We now prove that if U_i is fault free, then the table B_i reflects the actual fault situation. Following again the approach described in [2], we show that the following assertions are always satisfied before the execution of Step 3:

(A3) The module U_j is not faulty

(A4) B_i accurately reflects the existing fault situation up to k-1, i.e., for all m in [i,i+1,i+2,···,k-1]:

$B_{i,m} = 0$ if and only if module U_m is not faulty.

and $B_{i,m} = 1$ if and only if module U_m is faulty.

(A5) N_F contains the number of faulty modules up to k-1, i.e.,

$$N_F = \sum_{m=i}^{k-1} B_{i,m}$$

It can be shown that if (A3), (A4) and (A5) are true before the execution of Step 3, then they are still true after the execution of Step 4. Clearly (A3), (A4) and (A5) hold after the execution of Step 0 and therefore we conclude that (A3), (A4) and A(5) are always true before the execution of Step 3.

Now, suppose that the algorithm stops in Step 1. We know that B_i is correct up to k and that N_F=t. In other words, $B_{i,m}$ correctly reflects the fault situation for m=i, i+1,···,k and t faults have been detected. But we have assumed that at most t faults may occur and therefore this implies that the remaining modules are not faulty. The $B_{i,m}$ for m = k+1, k+2,...,i-1 are equa to 0 and therefore the complete table B_i is correct.

Suppose that the algorithm instead stops in Step 2; then B_i is correct up to k=i and therefore B_i is correct.

Although we have shown that when the algorithm stops, it produces the correct table. It remains to be shown that it indeed stops after a finite number of iterations. We note that k takes the values i, i+1, i+2,··· and therefore if the algorithm does not stop in Step 1, it must necessarily stop in Step 2. This concludes the proof of the theorem.

ACCELERATED ALGORITHM

The diagnosis of the set of faulty modules based on the results of Lemma 1, Lemma 2, and Theorem 1 requires that the table B_i , i = 0, 1,...,n-1 be compared. This process is time consuming and may be avoided. For each j = 0,1,..., n-1, let γ_j be the number of indices i for which $B_{i,j}$ = 1, i.e.,

$$\gamma_j = \text{cardinality of } \{ i \in [0,1,\ldots,n-1] \mid B_{i,j} = 1 \},$$

then, these quantities may be used in a diagnostic algorithm as follows:

Algorithm 2: Let t in [1,n-1] be given.

Step 0: Compute the tables $B_0, B_1, \ldots, B_{n-1}$ by using Algorithm 1.

Step 1: Compute the quantities $\gamma_0, \gamma_1, \ldots, \gamma_{n-1}$.

Step 2: Let $V = \{ j \in [0,1,\ldots,n-1] \mid \gamma_j \geq t+1 \}$.

Theorem 3: If a $D_{1,t}$ interconnection design is used, if the maximum number of faults which may occur is t and if $n \geq 2t + 1$, then U_j is faulty if and only if j is in V.

Proof: The result is a direct consequence of Lemmas 1 and 2 and Theorems 1 and 2.

Algorithm 2 is well suited for implementation on a microprocessor. For example, on an Intel 8080 microprocessor, the total amount of memory necessary to store the data and the program in the case n = 8 and t = 2 is 176 words of 8 bits, i.e., 1408 bits.

We note that Algorithm 2 may be implemented in parallel on a network of N microprocessors, with $N \leq n$. In particular, if N microprocessors are used, then it is possible to compute in parallel all the tables B_i and all the quantities γ_j. The computational time necessary to diagnose the network of n modules, using N microprocessors for implementing Algorithm 2 is essentially $T\lceil n/N \rceil / n$, where T is the computational time necessary to execute the instructions of Algorithm 2 when a single microprocessor is used and $\lceil n/N \rceil$ is the smallest integer larger than n/N.

EXAMPLE

In order to demonstrate the simplicity of the algorithms, we apply them to the network given in Figure 2. The network contains n=9 modules, t=3, i.e, at most three modules may be faulty, and a $D_{1,3}$ interconnection design is used, i.e., module U_0 tests U_1, U_2 and U_3, module U_1 tests U_2, U_3 and U_4, etc. Assume that the modules U_1, U_3 and U_6 are faulty. Figure 3 contains a possible set of test outcomes. The application of the algorithm to these test outcomes yields the tables B_i, $i=0,1,2,\cdots,8$ given in Figure 4. We find that the tables B_0, B_2, B_4, B_5, B_7 and B_8 are identical. We have 6 identical tables and using Lemma 2, we conclude that these tables reflect the correct fault situation of the network, i.e., we conclude that the modules U_1, U_3 and U_6 are faulty. Alternatively, we may compute the quantities γ_j, i.e., $\gamma_0 = 0$, $\gamma_1 = 8$, $\gamma_2 = 0$, $\gamma_3 = 8$, $\gamma_4 = 0$, $\gamma_5 = 0$, $\gamma_6 = 8$, $\gamma_7 = 1$, and $\gamma_8 = 1$, and then compute the set $V = \{ j \mid \gamma_j \geq 4 \} = \{1,3,6\}$. Using Theorem 3, we conclude once again that U_1, U_3, and U_6 are faulty.

CONCLUSION

An approach to the problem of fault diagnosis of symmetric multiple processor architectures has been proposed. It consists of constructing tables B_i, assuming that the corresponding modules U_i are not faulty, followed by a voting procedure. The construction of the tables B_i is decoupled in the sense that each table may be constructed independently of the others. It is possible to decrease the amount of computation necessary to obtain all the tables B_i, $i = 0, 1, \ldots, n-1$ by increasing the dependency between the construction of the various tables. It is not difficult to find schemes in which the construction of the table U_j depends on the tables U_0, U_1, ..., U_{j-1}. Such schemes are more complicated to code than the one we propose, require more memory to store the program and do not lend themselves to parallel implementation. Therefore, we feel that our scheme, i.e., Algorithm 2 which has a time complexity of $0(n^2)$ if sequentially implemented and $0(n)$ if implemented on a network of n microprocessor, is ideally suited for the fault diagnostic of $D_{1,t}$ networks.

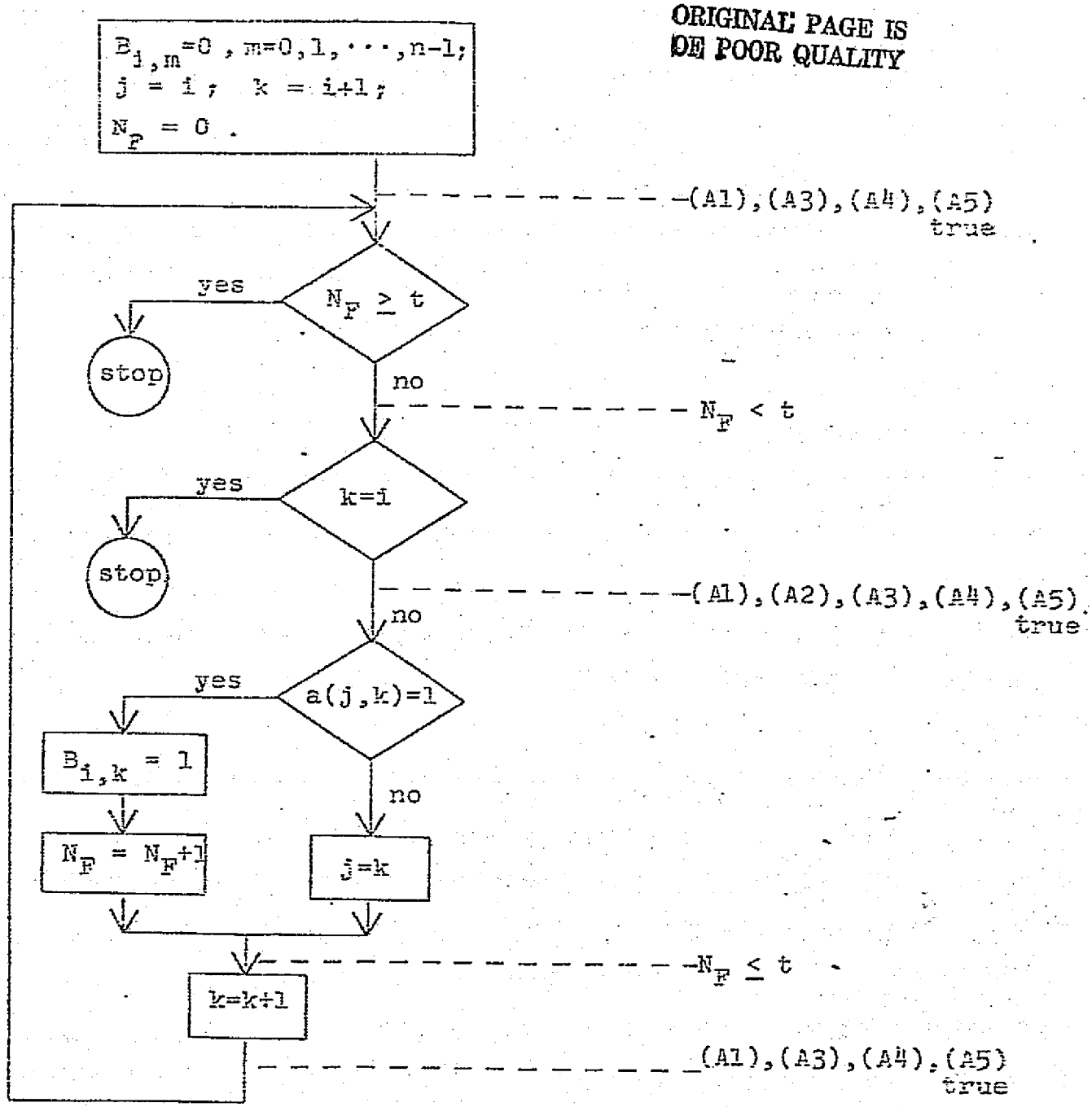


Figure 1: Flowchart of Algorithm 1:
Interpretation when module U_i is not faulty.

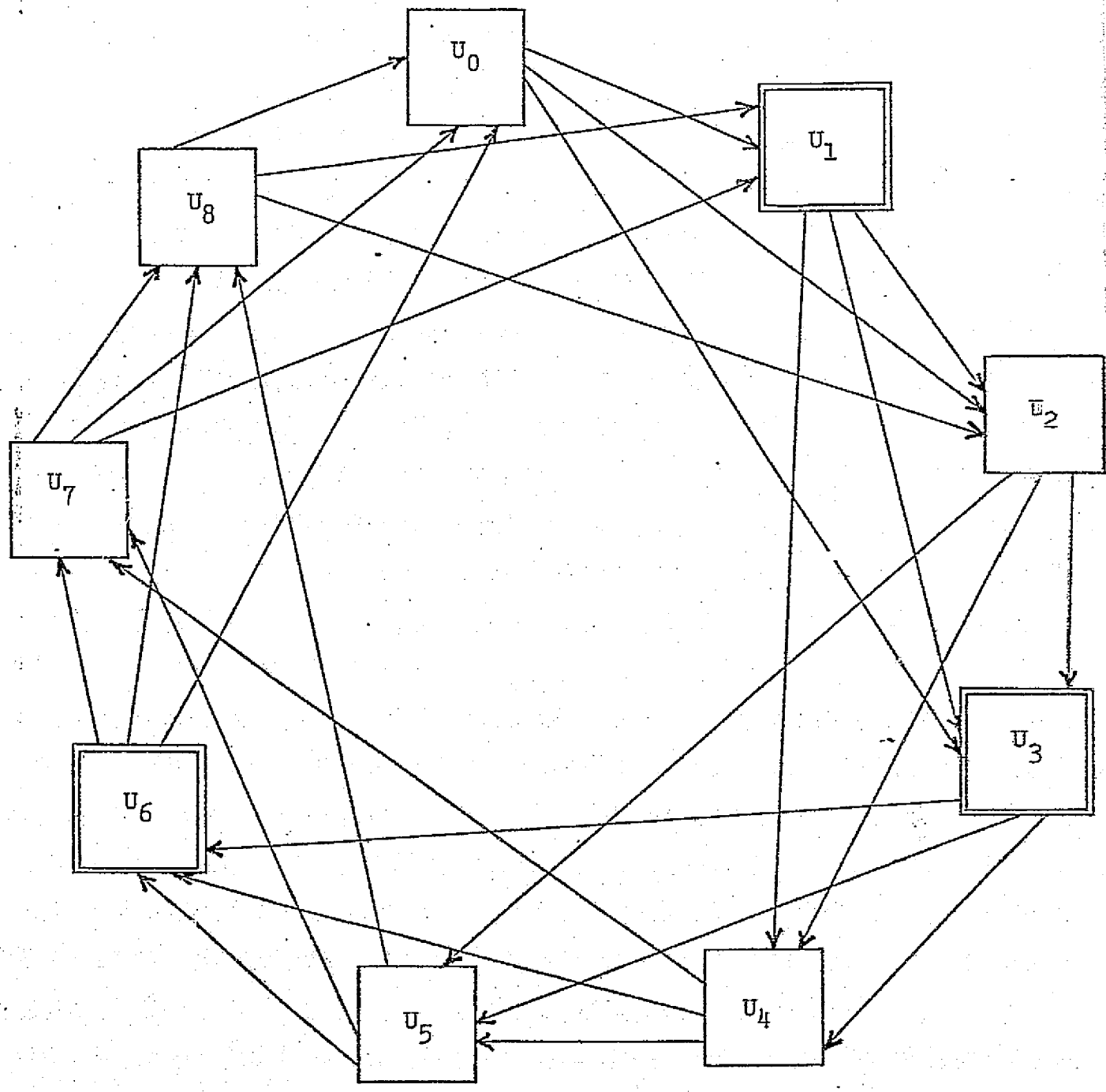


Figure 2. $D_{1,3}$ interconnection of 9 modules; modules U_1, U_3 and U_4 faulty.

j \ k	0	1	2	3	4	5	6	7	8
0		1	0	1					
1			0	1	1				
2				1	0	0			
3					0	0	1		
4						0	1	0	
5							1	0	0
6	0							1	1
7	0	1							0
8	0	1	0						

Figure 3. Tests outcomes a(j,k), Modules U_1, U_3 and U_6 faulty.

i \ j	0	1	2	3	4	5	6	7	8
0	0	1	0	1	0	0	1	0	0
1	0	0	0	1	0	0	1	0	0
2	0	1	0	1	0	0	1	0	0
3	0	1	0	0	0	0	1	0	0
4	0	1	0	1	0	0	1	0	0
5	0	1	0	1	0	0	1	0	0
6	0	1	0	1	0	0	0	1	1
7	0	1	0	1	0	0	1	0	0
8	0	1	0	1	0	0	1	0	0

Figure 4. Quantity B_{ij} obtained by using the algorithm to process the test results given in Fig. 3.

REFERENCES

[1] A.M. Corluhan and S.L. Hakimi, "On an algorithm for identifying faults in a T-diagnosable system", Proceedings of the 1976 Conference on Information Sciences and Systems, The Johns Hopkins University, 1976, pp. 370-375.

[2] R.W. Floyd, "Assigning meanings to programs, mathematical aspects of computer science", Proceedings of Symposia in Applied Mathematics, American Mathematical Society, Providence, Rhode Island, 1967, pp. 19-32.

[3] S.L. Hakimi and A.T. Amin, "Characterization of the connection assignment of diagnosable systems", IEEE Transactions on Computers, Vol. 23, January 1974, pp. 86-88.

[4] T. Kameda, S. Toida and F. Allan, " A Diagnosing Algorithm for Networks", Information and Control, Vol. 29 (1975), pp. 141-148.

[5] S.N. Maheshwari and S.L. Hakimi, "On models of diagnosable systems and probabilistic fault diagnosis", IEEE Trans. on Computers, March 1976, pp. 228-236.

[6] G.G.L. Meyer, "A segmented algorithm for solving a class of constrained discrete optimal control problems", IEEE Trans. on Automatic Control, Vol. AC-19, No. 2, April 1974, pp. 134-136.

[7] F.P. Preparata, G. Metze and R.T. Chien, "On the connection assignment problem of diagnosable systems", IEEE Trans. on Computers, Vol. EC-16, No. 6, December 1967, pp. 848-854.

[8] J.D. Russell and C.R. Kime, "On the diagnosability of digital systems", 1973 Int'l. Symposium on Fault Tolerant Computing, IEEE Computer Society Publications, June 1973, pp. 139-144.