STAN-CS-75-527 SU-SEL-75-044

CENTER FOR RELIABLE COMPUTING

CURRENT RESEARCH

by

Edward J. McCluskey
John F. Wakerly
Roy C. Ogus

October 1975

Technical Report No. 100

DIGITAL SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES

STANFORD UNIVERSITY · STANFORD, CALIFORNIA

STAN-CS-75-527 SU-SEL-75-044

CENTER FOR RELIABLE COMPUTING

CURRENT RESEARCH

by

Edward J. McCluskey

John F. Wakerly

Roy C. Ogus

October 1975

Technical Report no. 100

DIGITAL SYSTEMS LABORATORY

Department of Electrical Engineering Department of Computer Science

Stanford University

Stanford, California

C E N T E R F O R R E L I A B L E C O M P U T I N G

CURRENT RESEARCH

by

Edward J. McCluskey

John F. Wakerly

Roy C. Ogus

Technical Report no. 100

October 1975

Digital Systems Laboratory
Department of Electrical Engineering   Department of Computer Science
Stanford University
Stanford, California

ABSTRACT

This report summarizes the research work which has been performed, and is currently active in the Center for Reliable Computing in the Digital Systems Laboratory, Stanford University.

TABLE OF CONTENTS

1. INTRODUCTION. 1

2. THEORY OF FAULTS IN LOGIC SYSTEMS 3

2.1 Reliability Modeling. 3

2.1.1 Static Redundancy 5
2.1.2 Dynamic Redundancy. 10
2.1.3 Hybrid Redundancy 14
2.1.4 Comparisons 18

2.2 Signal Reliability. 19

2.3 Self-diagnosing Computer Design 21

2.3.1 Error-detecting Codes 23
2.3.2 Self-checking Circuits. 23
2.3.3 Fail-safe Circuits. 24

2.4 Multiple Fault Studies. 24

2.4.1 Fault Masking 25
2.4.2 Iterative Networks. 25

2.5 References. 26

3. STUDY OF MAINTAINABLE COMPUTERS . . . . , ... 37

3.1 Applying Triple Modular Redundancy to Small Computers 37

3.1.1 Effects of Transient Failures in Sequential Modules. 37
3.1.2 Reliability of TMR Memory Systems 39
3.1.3 A TMR Microcomputer System. 41

3.2 Self-Checking-Circuits. 53

3.2.1 Checked Binary Addition Using Checksum Codes and Check Symbol Prediction 53
3.2.2 A Self-Checking Checker for Periodic Signals. 54
3.2.3 Fail-Safe Sequential Machines 57

3.3 Self-Diagnosing Computer Design 59

3.3.1 Data Path Checking 60
3.3.2 Control Unit Checking. 65
3.3.3 Diagnosis. 68
3.3.4 Peripheral and Input/Output Checking 69

3.4 References Prepared under Previous NSF Grants. 71

3.5 Additional References. 73

4. INVESTIGATION AND EVALUATION OF DUAL COMPUTER CONFIGURATIONS

4.1 Introduction 75

4.2 Summary of Technical Areas in the Project. 76

4.2.1 The SIRU Computer System 76
4.2.2 Error Recovery Techniques in Computer Systems. 78
4.2.3 System Reliability Modeling. 79
4.2.4 Signal Reliability of a Circuit. 80
4.2.5 Software Reliability 81

4.3 References 82

5. BIOGRAPHICAL SKETCHES OF PRINICPAL RESEARCHERS IN CENTER FOR RELIABLE COMPUTING 83

6. LISTING OF PERSONNEL IN CENTER FOR RELIABLE COMPUTING. . . . 85

7. PREVIOUS RESEARCHERS IN CENTER FOR RELIABLE COMPUTING. . . . 87

8. CENTER FOR RELIABLE COMPUTING BIBLIOGRAPHY 89

8.1 Journal Papers 89

8.2 Conference Papers. 91

8.3 Technical Papers 94

0.4 Technical Notes. 97

1. INTRODUCTION

This report summarizes the research work which has been performed and is currently active in the Center for Reliable Computing (CRC) in the Digital Systems Laboratory, Stanford University. The report covers three projects in the CRC, and presents recent results obtained by researchers in the group, as well as an indication of the current status of each project.

The first project concerns the theory of faults in logic systems. Theoretical studies of the effects of faults on logic networks have been conducted and techniques for the modelling and evaluation of the reliability of redundant systems have been developed. Other areas involve the design of ultra-reliable systems, self-diagnosing computer systems, and the testing of digital circuits. Studies of fail-safe and self-checking circuits have also been carried out. The summary of the research in this project is presented in Section 2.

The second project involves the study of maintainable computers. Techniques for implementing fault-detection and diagnosis of small scale processors at a low cost are being investigated, as well as the study of improving the reliability of I/O controllers and peripheral devices. The design of redundant microprocessor systems and implementation of a low-cost maintainable minicomputer are current studies being performed. The description of the research in this project is found in Section 3.

The third project deals with the study of dual computer configurations. A dual processor system used in a guidance and navigation application was implemented for NASA-Ames Research Center and the CRC has been evaluating the performance of the system.

Section 5 presents a short biography of the principal researchers on the three projects, and Section 6 describes some previous researchers in the CRC.

2. THEORY OF FAULTS IN LOGIC SYSTEMS

Summary of Previous Work and Work in Progress

2.1 Reliability Modeling

A large variety of redundancy techniques -- TMR, Standby Sparing, Coding, Multiprocessing, etc. -- exists for improving the reliability of a digital system. When a system is being designed for a specific mission with given reliability requirements, it is necessary to:

(1) decide whether use of a redundancy technique is required,

(2) choose the optimum redundancy technique (if any is required) for the given application, and

(3) evaluate the resulting system reliability to determine whether the design requirements have been met.

The objective of research into reliability modeling is to develop tools to permit the previous three steps to be carried out as accurately as possible. Inaccuracy can result in systems which either (1) are more expensive than necessary, or (2) (unknowingly) fail to meet the reliability requirements.

Because of its serious practical importance, reliability modeling has received a great deal of attention [Borgerson, 1975; Bouricius, 1971; Rennels, 1973; Mathur, 1971].*

However, this earlier work results in only partially satisfactory results due to the inherent complexity of the problem. Straightforward combinatorial techniques usually have a computational

*The references for each section are compiled at the end of the particular section.

complexity too great to be useful so that more sophisticated approaches are necessary.

Redundancy techniques are usually classified as being either static, dynamic or hybrid. In _static_ redundancy, also called _masking_ or _massive_ redundancy, the effect of a fault is masked instantaneously by the non-faulty components of the system. No alteration in the system interconnection pattern is required for this error correction to take place. The principal static redundancy technique is _triple modular redundancy_ (_TMR_) which was used in the Saturn V launch vehicle computer [Dickenson, 1964] and has been proposed for use in other flight control systems [Masreliez, 1975]. Another important form of static redundancy is _interwoven redundant logic_ [Pierce, 1965]. Finally, the possibility of achieving static redundancy by replication of circuit components [Creveling, 1956; Lewis, 1963] should be mentioned. Interwoven logic is limited in its application since it must be applied at the gate level which can be an important restriction when the trend is towards higher levels of integration with consequent implications for non-independence of gate failures. TMR is more generally applicable since it can be applied at the gate, subsystem, or entire system level.

Dynamic redundancy, also called _selective_, _stand-by_ or _sparing_ redundancy, is characterized by the detection of a fault and the implementation of a corrective action. Fault detection is either concurrent or periodic. In concurrent detection, use is made of a coding technique relying on redundant signals [Kautz, 1962]. An important special case is the duplication of the system to generate two copies of the outputs with fault detection caused by a mismatch between duplicate signals [Wachter, 1975]. Periodic detection involves stopping the normal operation of the system at intervals to allow diagnostic tests to be carried out. Correction can be implemented by error-correcting circuitry if an error-correcting code is being used. Otherwise it is necessary to resort to some diagnostic routine to determine the proper signal values and fault-free system components and often to initiate a reconfiguration which eliminates the faulty subsystem.

In hybrid redundancy [Mathur, 1970; Siewiorek, 1973; Ogus, 1974], use is made of a static redundancy technique -- probably TMR -- to provide both error masking and error detection. Hybrid redundancy differs from static redundancy in that a reconfiguration action is taken when an error is detected. The major types of hybrid redundancy are TMR + spares [Ogus, 1974] and self-purging redundancy [Pierce, 1962; Chandy, 1972; Losq, 1975A].

2.1.1 Static Redundancy. A TMR system which consists solely of three copies of the desired circuit such as that shown in Figure 2.1, followed by a rank of voters from which the system output is derived is called a one-level TMR system. A system in which an input signal must pass through more than one vote before reaching the output, as in Figure 2.2, is called a multi-level TMR system.

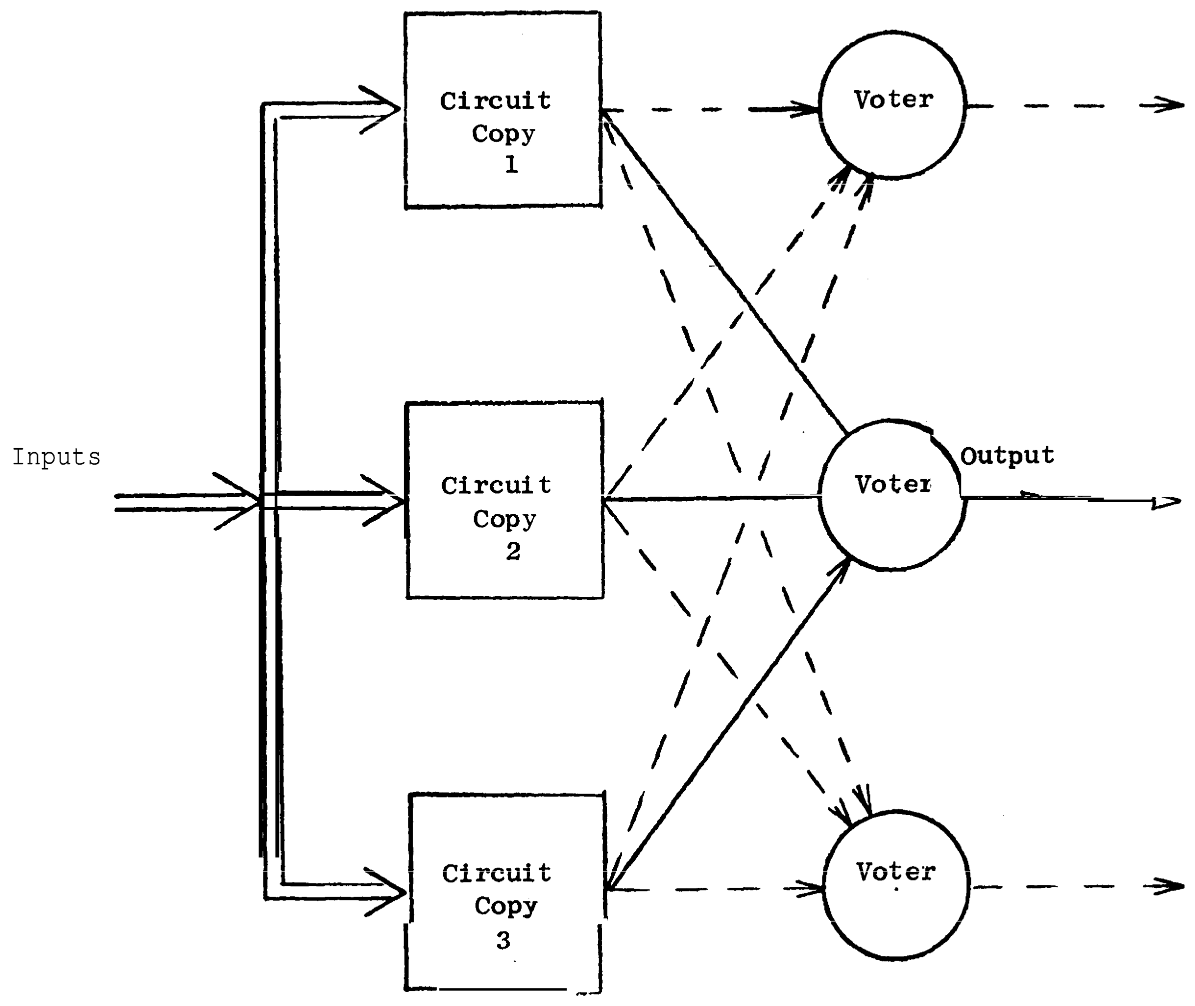


Figure 2.1   One-level TMR System

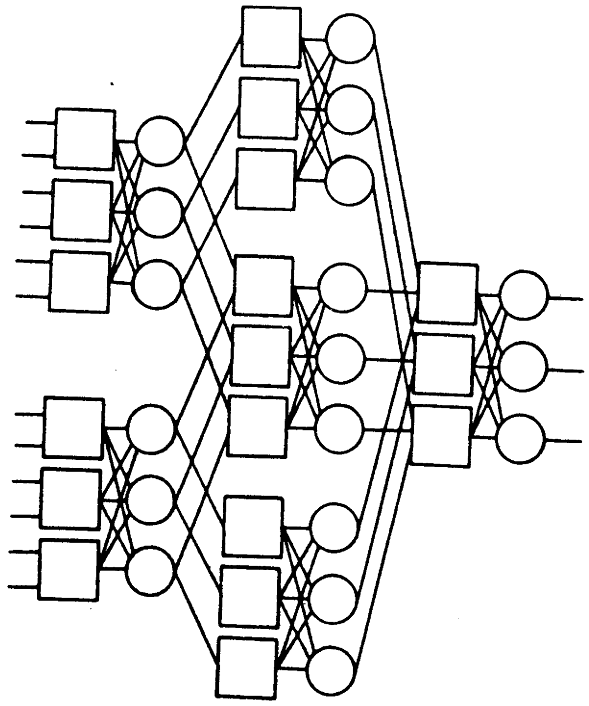
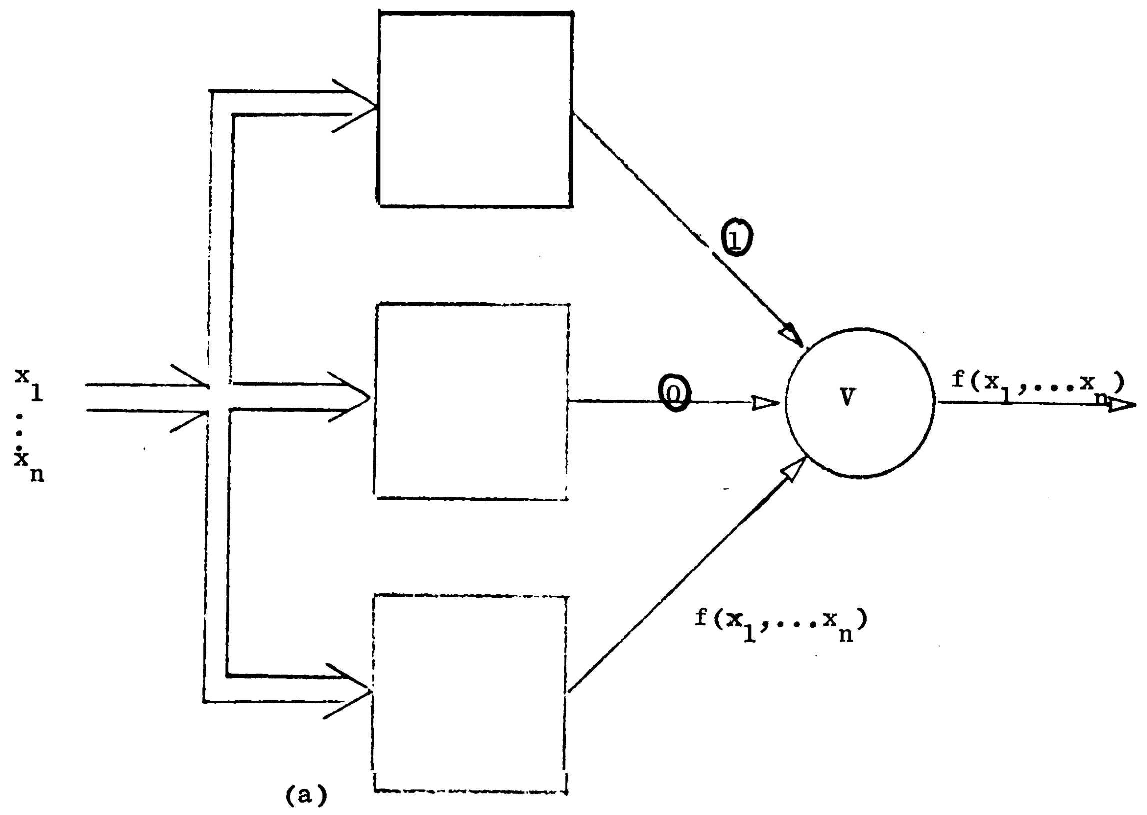


Figure 2.2 - Multi-level TMR Systems

One-level TMR Modeling. It is customary to model the reliability of a one-level TMR system by assuming that the system will perform correctly as long as no more than one of the three copies of the circuit has failed. In other words, it is assumed that for any failures affecting two or more copies, the system will fail. This assumption is much too pessimistic since there are many two-circuit failure situations which do not result in system failure. For example, consider the situation where one of the circuits has its output stuck-at-1, shown in Figure 2.3a, a second circuit has its output stuck-at-0, and the third circuit is fault-free. The system output from the voter will be correct. Another drastically over-simplified situation is shown in Figure 2.3b. Here the circuit being protected by TMR is a two-input AND gate. The figure shows the situation which occurs if an input (x) to one of the copies of the



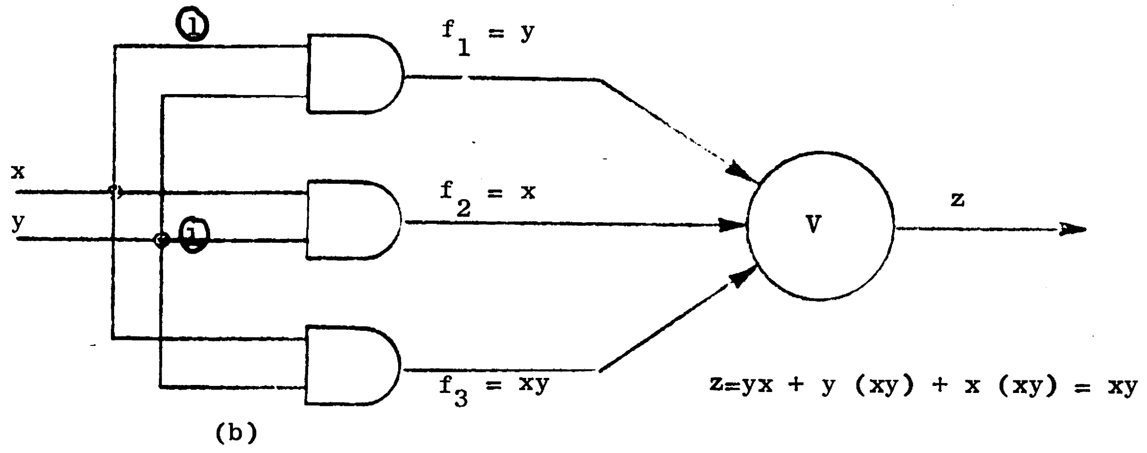


Figure 2.3- One-level TMR Circuits with Multiple Faults

gate is stuck-at-1 and the other input (y) to a second copy is also stuck-at-1. The system output is z=xy as required for correct operation.

A technique for calculating one-level TMR reliability which accurately models the effects of multiple failures which do not cause system failure was developed under a previous NSF grant* [Siewiorek, 1975]. It was shown that neglect of the multiple fault phenomenon just described can result in mission time predictions which are as much as 30% lower than the more accurate value obtained using Siewiorek's method.

Multiple-level TMR Modeling. A great deal of attention has been devoted to the problem of calculating the reliability of a multiple-level TMR network [Brown, 1961; Teoste, 1962; Rhodes, 1964; Longden, 1966; Rubin, 1967; Lyons, 1962; Gurzi, 1965; Jensen, 1964]. None of these approaches produced a technique for determing the exact reliability of a multiple-level TMR network; only bounds were obtained. The difficulty with obtaining an exact solution stems from the necessity for taking into account the complex interactions among the various stages of the network. Thus any straightforward combinatorial approach leads to a technique of computational complexity. In [Abraham, 1974], an algorithm for calculating the exact reliability of a multiple-level TMR network is described. The algorithm was developed by Abraham and Siewiorek

- - - - - -
* NSF grant GJ 27527

under a previous NSF grant*. The success of' the Abraham-Siewiorek algorithm depends on their technique of associating voters with circuit inputs and then partitioning the network into subnetworks which have the property that the overall network reliability can be calculated as the product of the subnetwork reliabilities.

Interwoven redundant logic modelling. In an interwoven redundant logic network the tasks of error correction and calculation of the output function are not separated as they are in TMR networks so that one-stage interwoven systems are of no importance. The problem of modelling interwoven networks was studied in [Jensen, 1963] and [Teoste, 1961] but only approximate techniques were discovered. The exact analysis was considered too difficult and costly [Teoste, 1964; Goldberg, 1966]. Our earlier success in modelling multiple level TMR networks led to the hope that a similar approach would be useful for interwoven networks. This turned out to be true, and a technique for exact modelling of interwoven redundant logic was developed under our present NSF grants# and is reported upon in [Abraham, 1975].

2.1.2 Dynamic Redundancy. The accurate modelling of dynamic redundant systems is difficult because the system reliability is very sensitive to the reliability of the mechanisms for fault detection and reconfiguration, and it is usually quite difficult to

- - - - - -

* NSF grant GJ 27527

# NSF grant GJ 40286 and DCR 7307973

model the reliability of these mechanisms accurately. In [Bouricius, 1969] a technique for estimating dynamic system redundancy is presented in which the problem of deriving detection and recovery mechanism reliabilities is avoided by introducing the concept of a coverage factor. The coverage factor is defined as the probability of system recovery given that a failure has occurred. The concept of coverage is very important for dynamic redundant systems. It has been used to demonstrate the extreme sensitivity of these systems to the recovery mechanisms [Arnold, 1973] and thus to focus attention on these mechanisms as being the critical aspect of such systems. The major problem with making use of coverage in modelling is the practical difficulty of determining the coverage factor for a specific system and evaluating the effect of the number of spares on the value of the coverage factor. As the number of spare modules is increased, so also is increased the number of module failures which the system can withstand and still continue to function. However, an increase in the number of spares also causes an increase in the complexity of the detection and reconfiguration mechanisms and thus a decrease in their reliability. Of major importance in modelling dynamic redundant systems is the ability to determine the optimum number of spares. This requires that the interrelationship between more spares giving better protection against module failures but worse detection and reconfiguration reliability be explicitly accounted for. Bouricius et al. attempt to take this interrelationship into account by letting the module failure rate (λ) depend on the number of spares. While this allows them to show that there exists an optimum number of spares for any coverage factor less than one, it is a completely artificial strategem which is hard to relate to the details of a specific system.

In dynamic redundant systems there are two techniques possible for replacing a failed unit by one of the standby spares: logic switching or power switching. In logic switching power is applied to all the units and reconfiguration consists of substituting the outputs from a standby unit for those of a failed on-line unit. In power switching spares are unpowered until the time at which they are switched on-line. Power switching has two advantages: (1) the savings in power consumption by providing power only to on-line units and (2) the possibility of a reduced failure rate for the non-powered spares [Nerber, 1965]. The ratio of the powered device failure rate to the unpowered device failure rate is called the dormancy factor (usually assumed ≥ 1).

Under the present grant, a very detailed analysis of stand-by systems [Losq, 1975 B; Losq, 1975 C] has been carried out. Rather than relying only on the coverage factor parameter, the analysis is carried out in terms of the parameters:

V_o, the rate of fail-safe failures, those failures which result in discarding a fault-free module but successfully replacing it with a spare module;

V_1, the rate of unsafe failures, those failures which result from a failed module not being successfully replaced (either because the failure is not detected or because a detected failure does not result in a successful reconfiguration).

This study has produced the following results:

(i) A technique for determining the optimum number of spares for a given system design and reliability specification.

(ii) A proof that for extremely short mission times systems with one spare are optimum.

(iii) A proof that for mission times which are not extremely short, but which are less than one-tenth of the mean lifetime of a single unit, the optimum number of spares is still small -- five or fewer for most systems.

(iv) The demonstration that it is possible to calculate a parameter, $\underline{\tau}$, which specifies the <u>useful life</u> of a stand-by system. The system reliability is very high for mission times less than τ and drops sharply towards zero for mission times greater than τ .

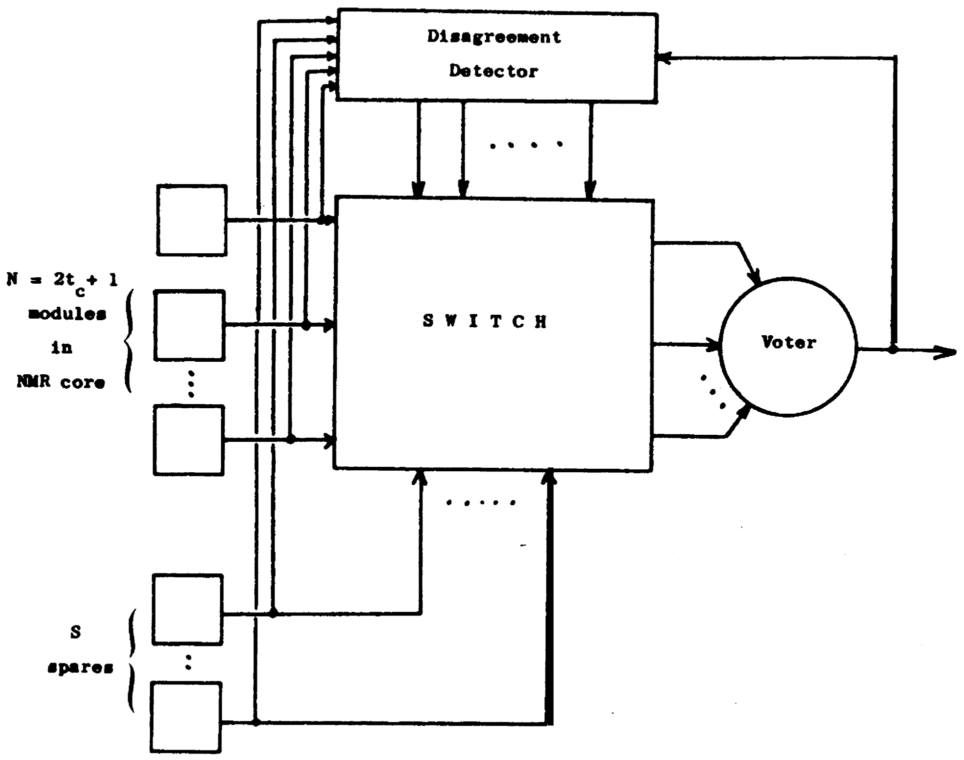
(v) The effects of imperfect fault detection mechanisms -- mechanisms which are designed to catch only a fraction of all possible module errors -- have been studied. A simple technique for determining the optimum number of spares for such systems and calculating their reliability is given.

(vi) If fail-safe techniques [Usas, 1975B; Mine, 1967] are used to design the fault-detection and recovery mechanisms, it is possible to design a stand-by system whose reliability is <u>always</u> greater than a system having no spares and whose reliability increases monotonically with the number of spares in the system.

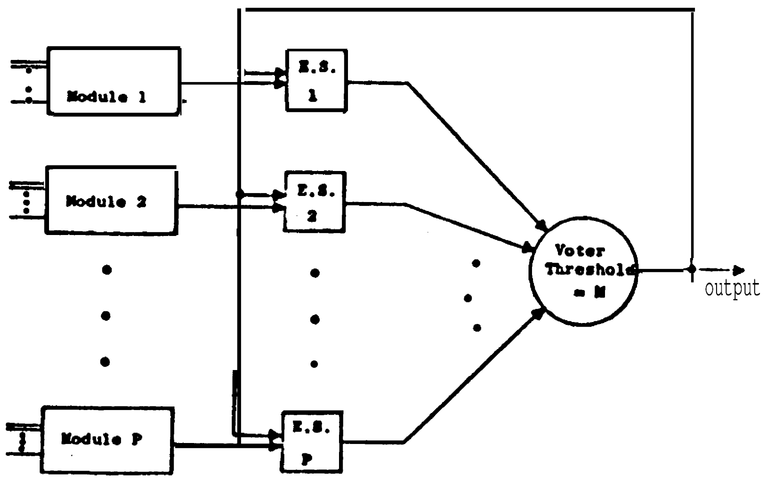
2.1.3 Hybrid Redundancy. Two general types of hybrid redundancy have been proposed: standby hybrid redundancy and self-purging redundancy, see Fig.2.4. In standby hybrid redundancy [Mathur, 1970; Siewiorek, 1973A; Ogus, 1974A] the system is initially placed into operation with three modules (for TMR, N for NMR) active and connected to the Voter from which the system output is derived. The remaining modules are designated as spares and are actively connected to the voter only when one of the on-line modules has failed and been disconnected from the voter. A major advantage of this form of redundancy is the possibility of keeping the spare modules unpowered until they are placed on line. It is thus possible to take advantage of the lower failure rate of unpowered modules [Nerber, 1965]. Self-purging redundancy [Pierce, 1962; Chandy, 1972; Losq, 1975A] has all of the modules initially connected to the voter. Only when a failure has been discovered is a module disconnected from the voter. This form of redundancy suffers from the necessity of keeping power on all non-failed modules. It has the advantage of not requiring a complex interconnection network and associated control as for standby hybrid redundancy. The mechanism for disconnecting a failed module from the self-purging system voter is a very simple device which is local to the module.

Standby Hybrid Redundancy. Work was carried out under a previous NSF grant* to arrive at an efficient design for implementing

*GJ-27527



(a) Standby hybrid redundancy



(b) Self-purging redundancy

Figure 2.4

standby hybrid redundancy. The resulting design, the iterative cell hybrid redundant system, was presented at the FTC/2 conference* [Siewiorek, 1972] and published in [Siewiorek, 1973A]. This design was compared with the only other "published" implementation of standby hybrid redundancy, the status register hybrid redundant system, [Roth, 1967B; Goldberg, 1966] and it was shown that the iterative cell design requires substantially less equipment than the status register design, thus making the iterative cell design both less costly and more reliable. A study was also made of various strategies for choosing which spare to use to replace a failed on-line module [Siewiorek, 1973B]. It was shown that a strategy in which each spare is used for only a subset of all voter inputs has as good reliability as a scheme in which any spare can be used to replace any failed on-line module.

Since the reliability of any hybrid redundant system depends critically on the mechanisms for failure detection and reconfiguration, an investigation was carried out to modify the iterative cell design so as to incorporate redundancy into these critical portions of the design. Two designs were arrived at -- one of which uses TMR and the other which uses fail-safe techniques [Mine, 1967]. These designs were both shown to provide about the same substantial improvement in reliability over the unprotected iterative cell design and to require approximately the same amount of additional hardware. This work was started under a previous

*Intl. Symp. on Fault-Tolerant Computing, Newton, Massachusetts, 1972.

NSF grant* and finished under the present grant#. It was presented at FTC/3† [Ogus, 1973] and published in [Ogus, 1974A].

Self-purging redundancy. Although self-purging systems have very good reliability properties, they have previously received very little attention. Work begun under a previous NSF grant* and continued under a present grant# has resulted in a detailed design of the switching and retry mechanisms for self-purging systems as well as the development of techniques for determining both the exact reliability or very tight bounds on the reliability [Losq, 1975A].

In analysis of redundant systems it is standard practice to make the assumption that only single-module failures occur. However there are two situations in which this assumption is invalid:

(i) Any application in which power is not always applied to the system as, for example, in a long space mission in which there are periods of time during which power is conserved by turning off the computers. Since there is a probability of failure associated with unpowered equipment, more than one module can fail during the power-off time without the failure being detected.

(ii) Even when the system is powered, it is possible for a failure to occur but for the failure not to cause an error until a later time. This phenomenon is called error latency and is discussed in [Shedletsky, 1975A] which reports on work carried out under a present NSF grant#. In this paper it is shown that there are situations for which the error latency (time between occurrence

----

*GJ 27527 #GJ 40286

†Intl. Symp. on Fault-Tolerant Computing, Palo Alto, California, 1973.

of a fault and its detection) is comparable to the mean time between failures. Thus it is possible for a second fault to occur before the first fault has been discovered. This has the same effect as the occurrence of a multiple fault.

In work carried out under a present NSF grant* two designs of self-purging systems which withstand multiple failures have been developed [Losq, 1974]. These designs resulted from a detailed theoretical study of the requirements for multiple-fault tolerance in redundant systems. The reliability analysis of the designs shows that their reliability is equal to that of a standby hybrid system with powered spares for single-module failures. While the standby hybrid systems can easily have a system failure as a result of a multiple fault, the designs of [Losq, 1974] are shown to have a very high probability of recovering from a multiple failure.

2.1.4 Comparisons. One general objective of reliability modeling is the development of techniques for choosing a particular redundancy scheme and configuring the system to satisfy a particular design objective. As a result of the research just summarized, it is possible to provide the following guidelines for choosing a redundancy technique:

(i) For extremely short mission times masking redundancy is optimal.

(ii) For mission times whose duration is of the same order of magnitude as the simplex (non-redundant) system mean life self-purging redundancy provides the best performance.

---

*NSF GJ 40286

(iii) For very long mission times standby redundancy is best, unless unpowered modules have the same failure rate as powered modules -- in which case self-purging redundancy should be used.

2.2 Signal Reliability.

Almost all of the literature on reliability modelling of digital systems has been concerned with functional reliability, the probability that the system realizes the desired design function. We have found that many useful results can be obtained by using a different reliability measure, the signal reliability, which is defined as the probability that the given signal is correct. The signal reliability differs from the functional reliability in that it allows for situations in which a signal will take on the correct value for a given input even in the presence of a fault. Signal reliability was discussed in [Amarel, 1962] in which it was called input-output reliability but has been neglected since.

Fundamental to the study of signal reliabilities is the ability to calculate the probability that a circuit output will take on a given value (usually taken to be 1) when the probabilities that the inputs take on given values are known. Research carried out under a present grant* has resulted in techniques for calculating these probabilities for combinational circuits [Parker, 1975A; Parker 1975B; McCluskey, 1974] and for sequential circuits [Parker, 1975C]. These techniques were then used to develop a method for calculating the signal reliability of a combinational circuit [Ogus, 1975].

- - - - - -

* NSF grant GJ 40286

An important application of the signal reliability parameter is in calculating the error latency of faults in digital circuits. The error latency of a fault is defined as the amount of time which elapses between an occurrence of the fault in a circuit and the first appearance of an error. Techniques have been developed for calculating error latencies and are reported in [Shedletsky, 1974A; Shedletsky, 1975A] for combinational circuits and in [Shedletsky, 1974B] for sequential circuits.

The concepts of signal reliability and error latency are applicable to situations in which the signals being applied to the circuit inputs can be modelled as random variables. There are two general types of situation for which such a model of the inputs is valid. One occurs when the circuit inputs are being generated by a random (or pseudorandom) source as in random testing or random test set generation. The other situation arises when the inputs are, in fact, deterministically generated but the generation mechanism is sufficiently complex so that it is not possible to characterize it more simply than as a random variable. An example of the second situation might be an adder circuit contained within a computer's arithmetic unit; the inputs to the adder are deterministic but the generation mechanisms are too complex to model directly.

In random test set generation a test set for a given circuit is constructed by simulating the fault-free circuit in parallel with circuits containing all faults to be tested for [Agrawal, 1972]. Inputs for the simulated circuits are determined by some random mechanism. The input sequence is continued until all of the faulty circuits have produced at least one output which differs from the output of the fault-free circuit or until a sufficiently high percentage of all the faulty circuits have met this condition. It is usually not possible to achieve detection of all the faults; in a practical circuit some of the faults may be untestable because they involve (inadvertently) redundant equipment or it may be uneconomic to insist on 100% testing.

In random testing randomly-generated inputs are applied directly to the physical circuit to be tested and also to a reference circuit. If the two circuits' outputs do not differ during the application of the input sequence, the circuit under test is accepted as good.

Both in random test set generation and in random testing, questions arise concerning the length of the required input sequence to guarantee that a fault is detected with a given probability, the best statistics to use for the input source, etc. Such questions were addressed in [Agrawal, 1972; Parker, 1973; Rault, 1971; Schnurmann, 1975; David, 1975]. The techniques developed for studying error latency have been found to be directly applicable to these problems of random testing and random test set generation. The error latency approach has produced much more precise results concerning these questions [Shedletsky, 1974A; Shedletsky, 1974B; Shedletsky, 19758; Shedletsky, 1975B].

2.3 Self-diagnosing Computer Design.

Decreasing hardware costs and increasing reliability requirements have led us to investigate the possibility of designing a low-cost self-diagnosing computer. Techniques have been demonstrated for detecting failures in all of the subsystems of a typical computer processor [Wakerly, 1973A] as well as in a large variety of peripheral and I/O subsystems [Usas, 1975C]. A simplified design example of a 16-bit self-diagnosing computer was given in [Wakerly, 1973] and a proposal to carry out the detailed design and construction of a 32-bit self-diagnosing computer has been submitted to NSF by John Wakerly. These computer designs incorporate the results of theoretical studies in the areas of error-detecting codes, self-checking circuits and fail-safe circuits.

In designing a self-diagnosing computer it is necessary to choose the error-detecting codes, to design the checking circuits which provide error indications, and to design the translation circuits necessary to convert from one code to another if more than one type of code is used. The translation and checking circuits should be self-checking so that failures in these circuits do not override the fault-detection capability of the computer.

The use of error-detecting codes in computers has received a great deal of theoretical attention and is actually fairly widespread in actual systems. However, most of the previous use of error-detecting codes has been confined to one subsystem of the computer. In the self-diagnosing computer with which we have been concerned it is necessary to use error-detecting codes throughout the entire computer system. Thus a study was conducted of error-detecting codes from the point of view of their applicability for use for error-detection in all of the parts of the computer system. This study has led to some new results on properties of error-correcting codes.

2.3.1 Error-detecting Codes. One of the most attractive error-detecting codes for use in arithmetic units is the low-cost residue code [Avizienis, 1971]. Study of this code led to the demonstration that it is also quite effective in detecting unidirectional multiple errors in mass storage devices and repeated-use multiple faults in byte-serial data transmision [Wakerly, 1975]. Rules for using residue codes for checking various arithmetic and non-arithmetic shift operations were also developed [Wakerly, 1973].

Checksum codes were studied and techniques for check symbol prediction were developed, thus making the checksum codes competitive with residue codes for checking arithmetic operations and data transmission and storage in some situations [Wakerly, 1974A].

2.3.2 Self-checking Circuits. In a computer arithmetic unit it is necessary to check both arithmetic and logical operations. The logical operations are difficult to check with the same codes which are effective for arithmetic operations. In studying this problem a new class of circuit called "partially self-checking circuits" was conceived which alleviates this problem [Wakerly, 1974B].

A number of new designs for self-checking circuits and checkers were developed. They are presented in [Wakerly, 1973] along with efficient practical MSI and LSI implementations.

It may be desirable to use different error-detecting codes for main memory and for the processor. For use in such a situation totally self-checking interfaces between codes were developed [Wakerly, 1973].

A persistent problem in designing error-detection circuitry for digital systems has been the design of a circuit to detect failures in the timing signal. A circuit to provide such a capability was discovered in connection with the studies of failure detection techniques for peripherals [Usas, 1975A].

2.3.3 Fail-safe Circuits. Our studies of failure-detection techniques for input-output systems has led to the conclusion that fail-safe circuits (circuits which produce unidirectional errors) are useful for controlling errors in such systems [Usas, 1975B]. Previous realization techniques for fail-safe circuits have assumed delay line memory elements [Diaz, 1973; Sawin, 1974] or have assumed that JK flip-flops which always have complementary outputs (even in the presence of failures) are used [Tohma, 1974]. Our studies of fail-safe circuits have led to a realization technique which uses D flip-flops and does not require the assumption that failures do not destroy the complementarity of their outputs [Usas, 1975C].

2.4 Multiple Fault Studies.

The bulk of the theoretical work on reliable computation has been concerned with situations in which it is assumed that only single faults occur. This is justified on the basis of independence among faults. Two phenomena act to invalidate this single-fault assumption: the trend towards higher levels of integration and the error latency property discussed in Section 2.2.

2.4.1 Fault Masking. A test set for single faults may fail to detect all multiple faults because of the property of fault masking: the presence of one fault may mask the effect of a test in detecting another fault. This property of fault masking has been studied and a technique for determining the existence of fault masking with respect to a given test has been developed [Diaz, 1975A). This study also results in a method for extending a test set which detects all single faults but not all multiple faults into a test set for all multiple faults.

2.4.2 Iterative Networks. An important special class of combinational circuit is the (unilateral) iterative network or iterative logic array. Because of the importance of this class of circuit, special testing techniques have been developed [Kautz, 1967; Menon, 1971; Landgraff, 1971; Friedman, 1973]. These methods all make the assumption that at most one cell of the network is faulty. We have been able to develop a testing method which does not require this assumption but instead permits any number of cells to be faulty [Diaz, 1975B]. A test procedure is given whose length is independent of the number of cells in the network. This procedure is applicable for any iterative network whose basic cell is reduced and has strongly connected components. Many practical networks satisfy these conditions. For those that do not, a simple modification is developed to make them testable with a constant number of tests.

2.5 REFERENCES

[Abraham, 1974]	Abraham, J. A. and Siewiorek D. P., "An Algorithm for the Accurate Reliability Evaluation of Triple Modular Redundancy Networks," IEEE Transactions on Computers, Vol. C-23, No. 7, July 1974, pp. 632-693.
[Abraham, 1975]	Abraham, J. A., "A Combinatorial Solution to the Reliability of Interwoven Redundant Logic Networks," IEEE Transactions on Computers, Vol. C-24, No. 5, May 1975, pp. 578-584.
[Agrawal, 1972]	Agrawal, V. D. and P. Agrawal, "An Automatic Test Generation System for ILLIAC IV Logic Boards,' IEEE Transactions on Computers, September 1972, Vol. C-21, No. 9, pp. 1015-1016.
[Agrawal, 1975]	Agrawal, P. and V. D. Agrawal, "Probabilistic Analysis of Random Test Generation Methods for Irredundant Combinational Logic Networks," IEEE Transactions on Computers, Vol. C-24, No. 7, July 1975, pp. 691-695.
[Amarcl, 1962)	Amarel, S. and J. A. Brzozowski, 'Theoretical Considerations on Reliability Properties of Recursive Triangular Switch Networks," Redundancy Techniques for Computing Systems, Wilcox and Mann, Editors, Spartan Books, Washington, D. C., 1962.
[Anderson, 1962]	Anderson, J. P., Hoffman, S. A., Shifman, J. and Williams, R. J., "A Multiple-Computer System for Command and Control," Chap. 36, Computer Structures, C. G. Bell and A. Newell, McGraw-Hill, Inc. New York, N. Y., 1971, pp. 447-455.
[Armstrong, 1966]	Armstrong, D. B., "On Finding a Nearly Minimal Set of Fault Detection Test for Combinational Logic Nets," IEEE Transactions on Electronic Computers, Vol. EC-15, No. 1, pp. 66-73, February 1966.
[Arnold, 1973]	Arnold, T. F., "The Concept of Coverage and its Effect on the Reliability Model of a Repairable System," IEEE Transactions on Computers, Vol. C-22, No. 2, March 1973, pp. 251-254.
[Avizienis, 1971]	Avizienis, A., "Arithmetic Codes: Cost and Effectiveness Studies for Applications in Digital Systems Design," IEEE Transactions on Computers, Vol. C-20, pp. 1322-1331, November 1971.
[Ball, 1969]	Ball, M. and F. Hardie, "Effects and Detection of Intermittent Failures in Digital Systems," FJCC, 1969, AFIPS, pp. 329--335.

[Baum, 1975] Baum, A. and D. Senzig, "Hardware Considerations in a Microcomputer Multiprocessing Systems," Digest of Papers, CompCon - Spring 75, Tenth IEEE Computer-Society International Conference, San Francisco, February 26, 27, 1975, pp. 27-30.

[Borgerson, 1975] Borgerson, B. R. and Freitas, R. F., "A Reliability Model for Gracefully Degrading and Standby-Sparing Systems," IEEE Transactions on Computers, Vol. C-24, No. 5, pp. 517-525, May 1975.

[Bouricius, 1969] Bouricius, W. G., Carter, W. C. and Schneider, P. R., 'Reliability Modeling Techniques for Self-Repairing Computer Systems" in Proc. ACM 1969 Ann. Conf., pp. 295-309, also IBM Report RC-2378, Watson Research Center, Yorktown Heights, New York.

[Bouricius, 1971] Bouricius, W. G., Carter, W. G., Jessup, D. C., Schnider, P. R., Wadia, A. B., "Reliability Modeling for Fault Tolerant Computers," IEEE Transactions on Computers, Vol. C-20, No. 11, pp. 1306-1311, November 1971.

[Breuer, 1972] Breuer, M. A., "Generation of Fault Detection Tests for Intermittent Faults in Sequential Circuits,' in Digest, 1972 International Symposium on Fault-Tolerant Computing, pp. 53-57.

[Brown, 1961] Brown, W. G., J. Tierney, and R. Wasserman, "Improvement of Electronic Computer Reliability Through the Use of Redundancy," IRE Transactions on Electronic Computers, Vol. EC-10, pp. 407-416, October 1961.

[Carter, 1971] Carter, W. C. et al., "Logic Design for Dynamic and Interactive Recovery," IEEE Transactions on Computers, Vol. C-20, No. 11, pp. 1300-1305, November 1971.

[Chandy, 1972] Chandy, K. N., Ramamoorthy, C. V. and Cowan, A., "A Framework for Hardware-Software Tradeoffs in the Design of Fault-Tolerant Computers," FJCC, 1972, AFIPS, pp. 55-63.

[Clegg, 1972] Clegg, F. W., 'Use of Spoofs for Faulty Logic Network Analysis," Digest of 1972 International Symposium on Fault-Tolerant Computing, Newton, Massachusetts, June 19-21, 1972, pp. 143-148.

[Clegg, 1973] Clegg, F. W., "Use of Spoofs for Faulty Logic Network Analysis," IEEE Transactions on Computers, Vol. C-22, No. 3, pp. 229-234, March 1973.

[Creveling, 1956]	Creveling, C. J., "Increasing the Reliability of Electronic Equipment by the Use of Redundant Circuits," Proc. IRE, Vol. 44, No. 4, pp. 409-415, April 1956.
[David, 1975]	David, R., G. Blanchet, "Sur la Detection des Pannes dans les Circuits Combinatoires par des Sequences d'Entrees Aleatoires," Digest of 1975 International Symposium on Fault-Tolerant Computing, Paris, France, June 18-20, 1975, pp. 210-214.
[Dias, 1975a]	Dias, F. J. O., "Fault Masking in Combinational Logic Circuits," IEEE Transactions on Computers, Vol. C-24, No. 5, May 1975, pp. 476-482.
[Diaz, 1973]	Diaz, M., J. C. Geffory and M. Courvoisier, "On-Set Realization of Fail-Safe Sequential Machines," Digest of 1973 International Symposium on Fault-Tolerant Computing, pp. 145-149.
[Dickinson, 1964]	Dickinson, M. M., Jackson, J. B. and Randa, G. C., "Saturn V Launch Vehicle Digital Computer and Data Adapter," AFIPS Conf. Procs., Vol. 26, (1964 FJCC, Washington, D. C., Spartan, 1964) pp. 501-516.
[Downing, 1964]	Downing, R. W., J. S. Nowack and L. S. Tvomenoksa, "No. 1 ESS Maintenance Plan," BSTJ, Vol. 43, No. 5, Part 1, September 1964, pp. 1961-2019.
[Fabry, 1973]	Fabry, R. S., "Dynamic Verification of Operating System Decisions," Communications ACM, Vol. 16, No. 11, November 1973, pp. 659-668.
[Fregni, 1974A]	Fregni, E. and R. C. Ogus, "Error Recovery Techniques in Computer Systems: A Survey," Technical Note no. 42, Digital Systems Laboratory, Stanford University, Stanford, California, June 1974.
[Fregni, 1974B]	Fregni, E., Beaudry, M. D. and Ogus, R. C., "A Markov Model of a Reconfigurable System," Technical Note no. 43, Digital Systems Laboratory, Stanford University, Stanford, California, August 1974.
[Friedman, 1973]	Friedman, A. D., "Easily Testable Iterative Systems," IEEE Transactions on Computers, Vol. C-22, No. 12, pp. 1061-1064, December 1973.
[Goldberg, 1966]	Goldberg,J.,Levitt, K. N. and Short, R. A., "Techniques for the Realization of Ultra-Reliable Space-Borne Computers," Final Report - Phase 1, Contract NAS12-33 Stanford Research Institute, Menlo Park, California, September 1966.

[Gurzi, 1965] Gurzi, K. J., "Estimates for the Best Placement of Voters in a Triplicated Logic Network," IEEE Transactions on Electronic Computers, Vol. EC-14, pp. 711-717, October 1965.

[Hayes, 1975] Hayes, J. P., "Testing Logic Circuits by Transition Counting," Digest 1975 Symposium on Fault-Tolerant Computing, Paris, France, June 1975.

[Hodges, 1972] Hodges, K. J. H., "Fault Resistance and Recovery within System 250," Proc. International Conference on Computer Communications, Washington, D. C., October 1972, pp. 290-296.

[Jensen, 1963) Jensen, P. A., "Quadded NOR Logic," IEEE Transactions on Reliability, Vol. R-12, pp. 22-31, September 1963.

[Jensen, 1964] Jensen, P. A., "The Reliability of Redundant Multiple-Line Networks," IEEE Transactions on Reliability, Vol. 13, pp. 23-33, 1964.

[Kamel, 1974] Kamel, S. and C. V. Page, "Intermittent Faults: A Model and a Detection Procedure," IEEE Transactions on Computers (Special Issue on Fault-Tolerant Computing), Vol. C-23, pp. 713-719, July 1974.

[Kautz, 1962) Kautz, W. H., "Codes and Coding Circuitry for Automatic Error Correction within Digital Systems," Redundancy Techniques for Computing Systems, Spartan Press, Inc., Washington, D. C., 1962, pp. 152-195.

[Kautz, 1967] Kautz, W., "Testing for Faults in Cellular Logic Arrays," IEEE Symposium on Automata Theory and Logic Design, pp. 161-174, 1967.

[Landgraff, 1971] Landgraff, R. W. and S. S. Yau, "Design of Diagnosable Iterative Arrays," IEEE Transactions on Computers, Vol. C-20, No. 8, pp. 867-877, August 1971.

[Lewis, 1963] Lewis, T. B., "Primary Processor and Data Storage Equipment for the Orbiting Astronomical Observatory," IEEE Transactions on Computers, Vol. EC-12, No. 5, pp. 677-686, December 1963.

[Longden, 1966] Longden, M., Page, L. J. and Scantlebury, R. A., "An Assessment of the Value of Triplicated Redundancy in Digital Systems," Microelectronics and Reliability, Vol. 5, Pergamon Press, Elmsford, N. Y., 1966.

[Losq, 1974]	Losq, J., "Redundancy Scheme for Optimum Multiple Fault Tolerance," Technical Note no. 33, Digital Systems Laboratory, Stanford University, Stanford, California, January 1974.
[Losq, 1975A]	Losq, J., "A Highly Efficient Redundancy Scheme: Self-Purging Redundancy," Technical Report no. 62, Digital Systems Laboratory, Stanford, California, July 1975.
[Losq, 1975B]	Losq, J., "Influence of Fault-Detection and Switching Mechanisms on the Reliability of Stand-By Systems," Digest of 1975 International Symposium on Fault-Tolerant Computing, Paris, France, June 18-20, 1975, pp. 81-86.
[Losq, 1975C]	Losq, J., "Influence of Fault-Detection and Switching Mechanisms on the Reliability of Stand-By Systems," Technical Report no. 75, Digital Systems Laboratory, Stanford University, Stanford, California, July 1975.
[Lyons, 1962)	Lyons, R. E., Vanderkuck, W., "The Use of Triple-Modular Redundancy to Improve Computer Reliability," IBM J. Res. Develop., Vol. 6, pp. 200-209, April, 1962.
[McCluskey, 1971]	McCluskey, E. J. and Clegg, F. W., "Fault Equivalence in Combinational Logic Networks," IEEE Transactions on Computers, Vol. C-20, No. 11, November 1971, pp. 1286-1293.
[McCluskey, 1974]	McCluskey, E. J., "Probability Models for Logic Networks," Proc. of the Fourth Manitoba Conference on Numerical Mathematics, Winnipeg, Canada, October 2-5, 1974, pp. 21-28.
[McCluskey, 1975A]	McCluskey, E. J., "Micros, Minis and Networks," Proceedings of the Meeting on Twenty Years of Computer Science, Pisa, Italy, June 16-19, 1975, pp. 23-33.
[Masreliez, 1975]	Masreliez, C, J., "Reliability Enhancement Through -Monitored Redundancy;" Digest of Papers, 1975 International Symposium on Fault-Tolerant Computing, Paris, France, June 18-20, 1975, pp. 227-231.
[Mathur, 1970]	Mathur, F. P. and Avizienis, A., "Reliability Analysis of a Hybrid Redundant Digital System: Generalized Triple Modular Redundancy with Self-Repair," Proc. SJCC, Vol. 36, pp. 375-383, 1970.
[Mathur, 1971]	Mathur, F. P., "On Reliability Modeling and Analysis of Ultrareliable Fault-Tolerant Digital Systems," IEEE Transactions on Computers, Vol. C-20, No. 11, pp. 1376-1381, November 1971.

[Mei, 1970] Mei, K. C. Y., "Fault Dominance in Combinational Circuits," Technical Note no. 2, Digital Systems Laboratory, Stanford University, Stanford, California, August 1970.

[Mei, 1974] Mei, K. C. Y., "Bridging and Stuck-At Faults," IEEE Transactions on Computers, Vol. C-23, No. 7, July i974, pp. 720-727.

[Menon, 1971) Menon, P. R. and A. D. Friedman, "Fault Detection in Iterative Logic Arrays," IEEE Transactions on Computers, Vol. C-20, No. 5, pp. 524-535, May 1971.

[Mine, 1967] Mine, H. and Y. Koga, "Basic Properties and a Construction Method for Fail-Safe Logical Systems," IEEE Transactions on Electronic Computers, Vol. EC-16, No. 3, June 1967, pp. 282-289.

[Nerber, 1965] Nerber, P. O., "Power Off Time Impact on Reliability Estimates," IEEE Int. Convention Rec., Part 10, pp. 1-5, March 22-26, 1965, New York.

[ogus, 19733 Ogus, R. C., "Fault-Tolerance of the Iterative Cell Array Switch for Hybrid Redundancy," Digest of 1973 International Symposium on Fault-Tolerant Computing, Palo Alto, California, June 20-22, 1973, pp. 107-113.

[Ogus, 1974A] Ogus, R. C., "Fault-Tolerance of the Iterative Cell Array Switch for Hybrid Redundancy," IEEE Transactions on Computers, Vol. C-23, No. 7, July 1974, pp. 667-682.

[ogus, 19751 Ogus, R. C., "The Probability of a Correct Output from a Combinational Circuit," IEEE Transactions on Computers, Vol. C-24, No. 5, May 1975, pp. 534-544.

[Ornstein, 19753 Ornstein, S. M., Crowther, W. R. Kraley, M. F., Bressler, R. D., Michel, A. and Heart, F. E., "Pluribus-- A Reliable Multiprocessor," Proc. AFIPS, 1975 National Computer Conference, pp. 551-559.

[Parker, 1973] Parker, K. P., "Probabilistic Test Generation," Technical Note no. 18, Digital Systems Laboratory, Stanford University, Stanford, California, June 1973.

[Parker, 1975A] Parker, K. P. and McCluskey, E. J., "Analysis of Logic Circuits with Faults Using Input Signal Probabilities," IEEE Transactions on Computers, Vol. C-24, No. 5, May 1975, pp. 573-578.

[Parker, 1975B] Parker, K. P. and E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks.' IEEE Transactions on Computers, Vol. C-24, No. 6, pp. 668-670, June 1975.

[Parker, 1975C] Parker, K. P. and E. J. McCluskey, "Sequential Circuit Output Probabilities from Regular Expressions," Technical Report no. 93, Digital Systems Laboratory, Stanford University, Stanford, California, June 1975.

[Pierce, 1962] Pierce, W. H., "Adaptive Vote-Takers Improve the Use of Redundancy," in Redundancy Techniques for Computing Systems, R. H. Wilcox and W. C. Mann, eds., Spartan Books, Washington, D. C., 1962, pp. 229-250.

[Pierce, 1965] Pierce, W. H., Failure-Tolerant Computer Design, Academic Press, New York, 1965.

[Rault, 1971] Rault, J. C., "A Graph Theoretical and Probabilistic Approach to Fault Detection of Digital Circuits,' Digest 1971 Symposium on Fault Tolerant Computing, June 1971.

[Reese, 1973] Reese, R. D. and McCluskey, E. J., "A Gate Equivalent Model for Combinational Logic Network Analysis," Digest of 1973 International Symposium on Fault-Tolerant Computing, Palo Alto, California, June 20-22, 1973, pp. 79-85.

[Rennels, 1973] Rennels, D. A., Avizienis, A., "RMS: A Reliability Modeling System for Self-Repairing Computers," Proc. of the Third International Symposium on Fault-Tolerant Computing, June 21-22, 1973, Palo Alto, California, pp. 131-135.

[Ressler, 1973] Ressler, B. E., 'Design of a Dual Computer Configuration for Redundant Computation," M.S. Thesis, Dept. of Electrical Engineering, M.I.T., Cambridge, Massachusetts, June 1973. .

[Rhodes, 1964] Rhodes, L. J., "Effects of Failure Modes on Redundancy," in Proc. 10th National Symposium on Reliability and Quality Control, Washington, D. C., pp. 360-364.

[Roth, 1967A] Roth, J. P., W. G. Bouricius, and P. R. Schneider, "Programmed Algorithms to Compute Tests to Detect and Distinguish between Failures in Logic Circuits," IEEE Transactions on Electronic Computers, Vol. EC-16, pp. 567-579, May 1967.

[Roth, 1967B] Roth, J. P., W. C. Carter and R. P. Schneider, "Phase II of an Architectural Study for a Self-Repairing Computer," SAMSO TR67-106, November 1967.

[Rubin, 1967] Rubin, D. K., "The Approximate Reliability of Triply Redundant Majority-Voted Systems," 1st Annual IEEE Computer Conference Digest, Chicago, Ill., IEEE Publ. 16051, September 1967, pp. 46-49.

[Sawin, 1974] Sawin, D. H., "Fail-Safe Synchronous Sequential Machines Using Modified On-Set Realizations," Digest of 1974 International Symposium on Fault-Tolerant Computing, pp. 7-12.

[Schnurmann, 1975] Schnurmann, H. D., E. Lindbloom and R. G. Carpenter, "The Weighted Random Test Pattern Generator," IEEE Transactions on Computers, July 1975, Vol. C-24, No. 7, pp. 695-700.

[Shedletsky, 1974A] Shedletsky, J. J. and E. J. McCluskey, "The Error Latency of a Fault in a Combinational Digital Circuit," Technical Note no. 55, Digital Systems Laboratory, Stanford University, Stanford, California, November 1974.

[Shedletsky, 1974B] Shedletsky, J. J. and E. J. McCluskey, "The Error Latency of a Fault in a Sequential Digital Circuit," Technical Note no. 56, Digital Systems Laboratory, Stanford University, Stanford, California, December 1974.

[Shedletsky, 1975A] Shedletsky, J. J., and E. J. McCluskey, "The Error Latency of a Fault in a Combinational Digital Circuit," Digest of 1975 International Symposium on Fault-Tolerant Computing, Paris, France, June 18-20, 1975, pp. 210-214.

[Shedletsky, 1975B] Shedletsky, J. J., "A Rationale For Random Testing Combinational Digital Circuits," Digest, Eleventh Annual IEEE Computer Society Conference (COMPCON), Washington , D. C., September 9-11, 1975.

[Siewiorek, 1972) Siewiorek, D. P. and McCluskey, E. J., "An Iterative Cell Switch Design for Hybrid Redundancy,' Digest of 1972 International Symposium on Fault-Tolerant Computing, Newton, Massachusetts, June 19-21, 1972, pp. 182-189.

[Siewiorek, 1973A] Siewiorek, D. P. and McCluskey, E. J., "An Iterative Cell Switch Design for Hybrid Redundancy," IEEE Transactions on Computers, Vol. C-22, No. 3, pp. 290-297, March 1973.

[Siewiorek, 1973B] Siewiorek, D. P. and E. J. McCluskey, "Switch Complexity in Systems with Hybrid Redundancy," IEEE Transactions on Computers_, Vol. C-22, No. 3, pp. 276-282, March 1973.

[Siewiorek, 1975] Siewiorek, D. P., "Reliability Modeling of Compensating Nodule Failures in Majority Voted Redundancy," IEEE Transactions on Computers, Vol. C-24, No. 5, May 1975, pp. 525-533.

[Teoste, 1961] Teoste, R., "Reliability of Redundant Computers," Lincoln Laboratory, M.I.T., Cambridge, Massachusetts, Report 21G-0029, ASTIA, Doc. 260494, 1961.

[Teoste, 1962] Teoste, R., "Design of a Repairable Redundant Computer," IRE Transactions on Electronic Computers, Vol. EC-11, No. 5, pp. 643-649, October 1962.

[Teoste, 1964] Teoste, R., "Digital Circuit Redundancy," IEEE Transactions on Reliability, Vol. R-13, pp. 46-61, June 1964.

[Tohma, 1974] Tohma, Y., "Design Technique of Fail-Safe Sequential Circuits Using Flip-Flop for Internal Memeory," IEEE Transactions on Computers, Vol. C-23, No. 11, pp. 1149-1154.

[Usas, 1975A] Usas, A. M., "A Totally Self-Checking Checker for the Detection of Errors in Periodic Signals," IEEE Transactions on Computers, Vol. C-24, No. 5, May 1975, pp. 483-489.

[Usas, 1975B] Usas, A. M., "Fail-Safe Circuits: A Means to Improve Reliability and Maintainabilty of I/O Subsystems," Technical Note no. 59, Digital Systems Laboratory, Stanford University, Stanford, California, June 1975.

[Usas, 1975B] Usas, A. M., "Fail-Safe Circuits: A Means to Improve Reliability and Maintainability of I/O Subsystems," Digest of IEEE Computer Society International Conference. Washington, D. C., September 11-12, 1975.

[Usas, 1975C] Usas, A. M., "Fault Detection and Diagnosis in Digital Computer Input-Output Systems," Ph.D. Thesis, Stanford University, in preparation.

[Wachter, 1975] Wachter, W. J., "System Malfunction Detection and Correction," Digest of Papers, 1975 International Symposium on Fault-Tolerant Computing, Paris, France, June 18-20, '1975, pp. 196-201.

[Wakerly, 1973)	Wakerly, J. F., "Low-Cost Error Detection Techniques for Small Computers," Technical Report no. 51, Digital Systems Laboratory, Stanford University, Stanford, California, December 1973.
[Wakerly, 1974]	Wakerly, J. F. and E. J. McCluskey, "Design of Low-Cost General-Purpose Self-Diagnosing Computers," Information Processing 1974, IFIP Congress 1974, August 5-10, Stockholm, Sweden, Vol. 1, pp. 108-111.
[Wakerly, 1974A]	Wakerly, J. F., "Checked Binary Addition Using Parity Prediction and Checksum Codes," Technical Report no. 39, Digital Systems Laboratory, Stanford University, Stanford, California, January 1974.
[Wakerly, 1974B]	Wakerly, J. F., "Partially Self-Checking Circuits and Their Use in Performing Logical Operations," IEEE Transactions on Computers, Vol. C-23, No. 7, July 1974, pp. 658-667.
[Wakerly, 1975]	Wakerly, J. F., "Detection of Unidirectional Multiple Errors Using Low-Cost Arithmetic Code," IEEE Transactions on Computers, Vol. C-24, No. 2, February 1975, pp. 210-212.
[Widdoes, 1975]	Widdoes, Lawrence C., Jr., "The Minerva Multi-Microprocessor," Technical Note no. 62, Digital Systems Laboratory, Stanford University, Stanford, California, July 1975.
[Wulf, 1974]	Wulf, W. A. et al., "Hydra: The Kernel of a Multiprocessor Operating System," Communications ACM, Vol. 17, No. 6, June 1974, pp. 337-347.

3. STUDY OF MAINTAINABLE COMPUTERS

Summary of Previous Results and Work in Progress

3.1 Applying Triple Modular Redundancy to Small Computers

The decreasing cost of computer hardware is increasing the feasibility of triple modular redundancy (TMR) as a means of providing fault tolerance in small computer systems. In a TMR system, each module is triplicated and majority voters are used at the interfaces between modules to mask the effects of single module failures.

In order to discover any special problems that might occur in applying TMR to small systems, we began the design of a TMR microcomputer system. Our first discovery was that applying TMR to arbitrary sequential modules requires special consideration of the effects of transient failures (section 3.1.1), Next we examined the reliability of triplicated memory systems and found that triplication is a better choice than coding for small systems (section 3.1.2). Finally we constructed models of several different TMR microcomputer system configurations and derived the reliability of each (section 3.1.3). We have shown that careful use of TMR can improve the mission time of small microcomputer systems by a factor of 3 to 10.

3.1.1 Effects of Transient Failures in Sequential Modules

Triple modular redundancy was first proposed by von Neumann [1956] as a means of masking the effects of transient component failures in a system. Applying redundancy at the component level was also proposed by Moore and Shannon [1956] as a means of masking transient failures in relay networks.

Subsequent researchers showed that TMR could be used to mask the effects of permanent failures in a system [Dickinson and Walker, 1958], and that TMR could be applied at higher levels than the component level [Flehinger, 1958; Lyons and Vanderkulk, 1962; Brown, Tierney and Wasserman, 1961].

Recent investigations into the reliability of TMR systems have concentrated on the effects of permanent failures [Gurzi, 1965, Rubin, 1967; Abraham and Siewiorek, 1974]. It has been assumed that most transients are masked and leave no permanent effect in TMR systems [Lyons and Vanderkulk, 1962]. That is, the effect of a transient failure is masked by the voter during the short period of time that it is present, and the effect disappears with the transient. Once a transient disappears, another can be tolerated. The only transients that were recognized to cause system failures were those that affected more than one member of a replicated module trio at the same time [Avizienis, 1967].

That multiple transients over a period of time could be tolerated was in fact shown to be true by von Neumann [1956] and by Moore and Shannon [1956] for redundancy schemes applied at the <u>component</u> level. In systems that apply redundancy at a higher level, we have demonstrated that this is not always true, that the effect of a single transient failure in one module can be permanent [Wakerly, 1975a]. The transient has a permanent effect when the affected module is a sequential machine that is never re-synchronized.

The need to re-initialize modules after transient failures has been long recognized for self-repairing systems with selective redundancy [Avizienis, 1967; Avizienis, 1971]. However, the re-synchronization problem in TMR systems has been neglected because redundancy has in the past been applied at a level low enough that the problem did not occur [von Neumann, 1956; Moore and Shannon, 1956; Dickinson, 1964]. The increasing complexity of integrated circuits is continually increasing the minimum level at which TMR may be applied in systems using standard MSI and LSI components. For example, in a microcomputer system, voting must be applied at the processor level. Reliability analysis of such systems may indicate that TMR should be applied at a level even higher than the minimum [Gurzi, 1965; Abraham and Siewiorek, 1974; Longden, 1966]. Yet as redundancy is applied at higher levels, transients become more likely to have permanent effects. Thus the designer must be aware of the effects of transients in specifying the application of redundancy to a system.

We have proved [Wakerly, 1975a] that multiple transients can be tolerated by a TMR system if and only if a synchronizing sequence is applied to the system periodically during normal operation. We have shown system structures that provide for easy synchronization, and we have suggested ways of modifying systems that do not normally receive synchronizing sequences.

3.1.2 Reliability of TMR Memory Systems

A TMR system can be partitioned into a number of cells so that the system reliability is the product of the cell reliabilities [Abraham, 1974]. The simplest type of cell, shown in Fig. 3.1, has one triplicated module with voters on each input. (There is a separate voter circuit for each input bit of the module.) The cell tolerates any single module or voter failure', since errors produced by such a failure will be corrected by the voters at the inputs of the next cell. Assuming for simplicity perfect voter reliability,

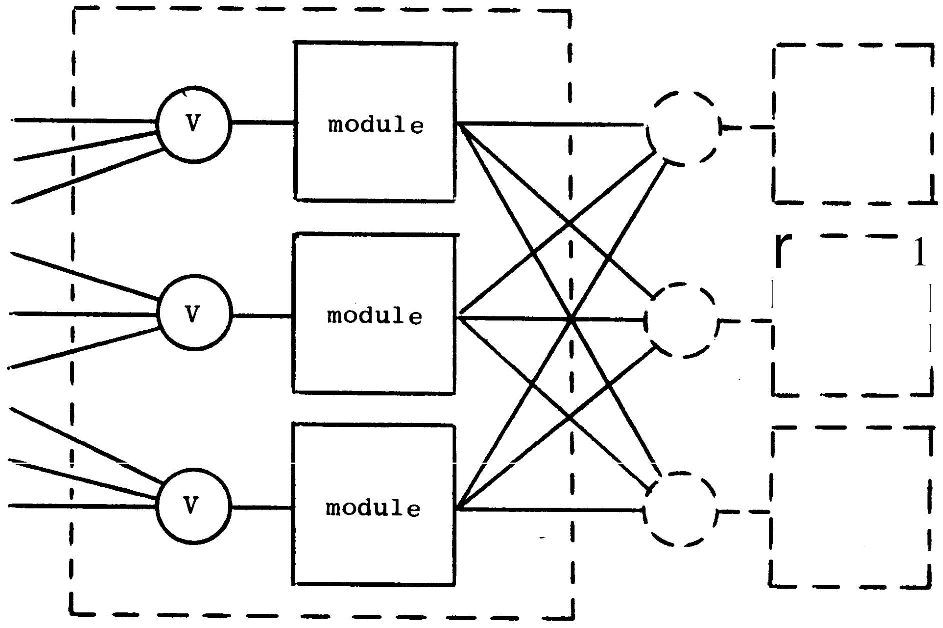


Fig. 3.1 A TMR voter-module cell.

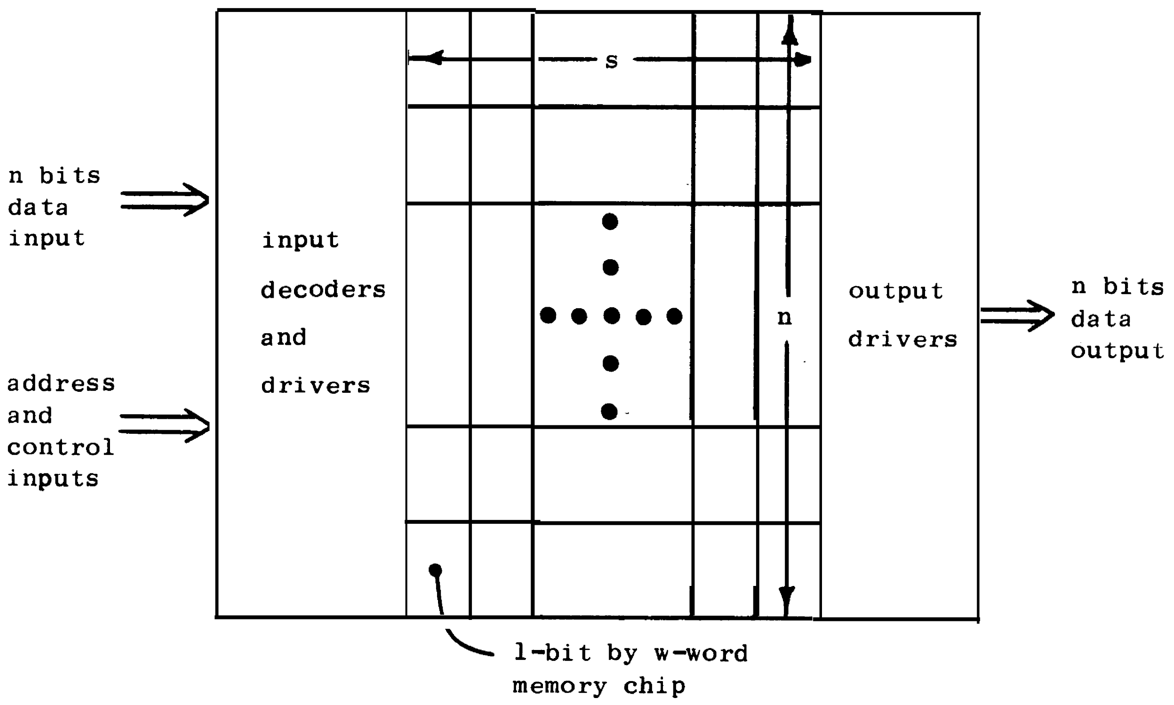


Fig. 3.2 An n-bit by ws-word semiconductor memory module.

the cell reliability R_{cell} is a function of the module reliability R_m,

$$R_{cell} = R_m^3 + 3R_m^2(1-R_m) \; . \qquad (3.1)$$

The semiconductor memory module of a small computer system can be modeled as shown in Fig. 3.2. There is some shared address decoding and driving circuitry, an array of memory chips, and perhaps some shared output circuitry. The memory array consists of ns 1-bit by w-word memory chips arranged in an $n \times s$ matrix to form the n-bit by ws-word array. If the memory chip reliability is R_c and the reliability of the common circuitry is R_d, then module reliability is $R_c^{ns} \cdot R_d$ and it would appear from (3.1) that the reliability of a TMR memory system is

$$R_{sys} = (R_c^{ns}R_d)^3 + 3(R_c^{ns}R_d)^2\,(1-R_c^{ns}R_d) \; . \qquad (3.2)$$

The above analysis neglects the organization of the memory array. Assuming that there is a voter for each bit of the memory output, the system fails only if there is a simultaneous error in a single bit position of two of the triplicated memory modules. Consideration of the memory array structure hence leads to the more accurate reliability formula,

$$R_{sys} = R_d^3(3R_c^2 - 2R_c^3)^{ns} + 3R_d^2(1-R_d)R_c^{2ns} \qquad (3.3)$$

The reliability expression above always produces a reliability value greater than or equal to (3.2) The improvement obtained by using (3.3) decreases as the reliability of the memory array (R_c) relative to the common circuitry (R_d) increases. For example, if R_c=1 the formulas are identical.

But for typical semi-conductor memory systems, the common circuitry comprises only about 10-15% of the total, and so the reliability value obtained by considering the structure of the memory array (3.3) is significantly higher than that obtained by simple analysis (3.2). A typical example is shown in Fig. 3.2.

The above discussion is intended only to give an indication of the nature of our results on memory systems. The actual memory system analysis is somewhat more complex, taking into account voter reliability, the placement of voters for the memory system inputs, the-possibility of having different chip types within the memory array, and a solution to the problem of multiple pattern-sensitive failures. The effectiveness of the TMR memory system organization and a system using a single-error-correcting code have been compared. While both systems are guaranteed to correct all single failures in the memory array, analysis has shown that the TMR system is more reliable because it corrects a larger number of multiple failures than the coded memory. Also the TMR system corrects all single failures in the common circuitry (Fig.3.2) while the coded system does not unless a copy of the common circuitry is provided for each of the memory bit slices. For an 8-bit memory system, coding requires 4 redundant memory bits per word while TMR requires 16. On the other hand, coding-requires a decoder that is more complex than the simple TMR voters, especially if the decoder itself is to be fault-tolerant. For small fault-tolerant memory systems that are to be interfaced to a TMR processor, TMR appears to be a much better choice than coding.

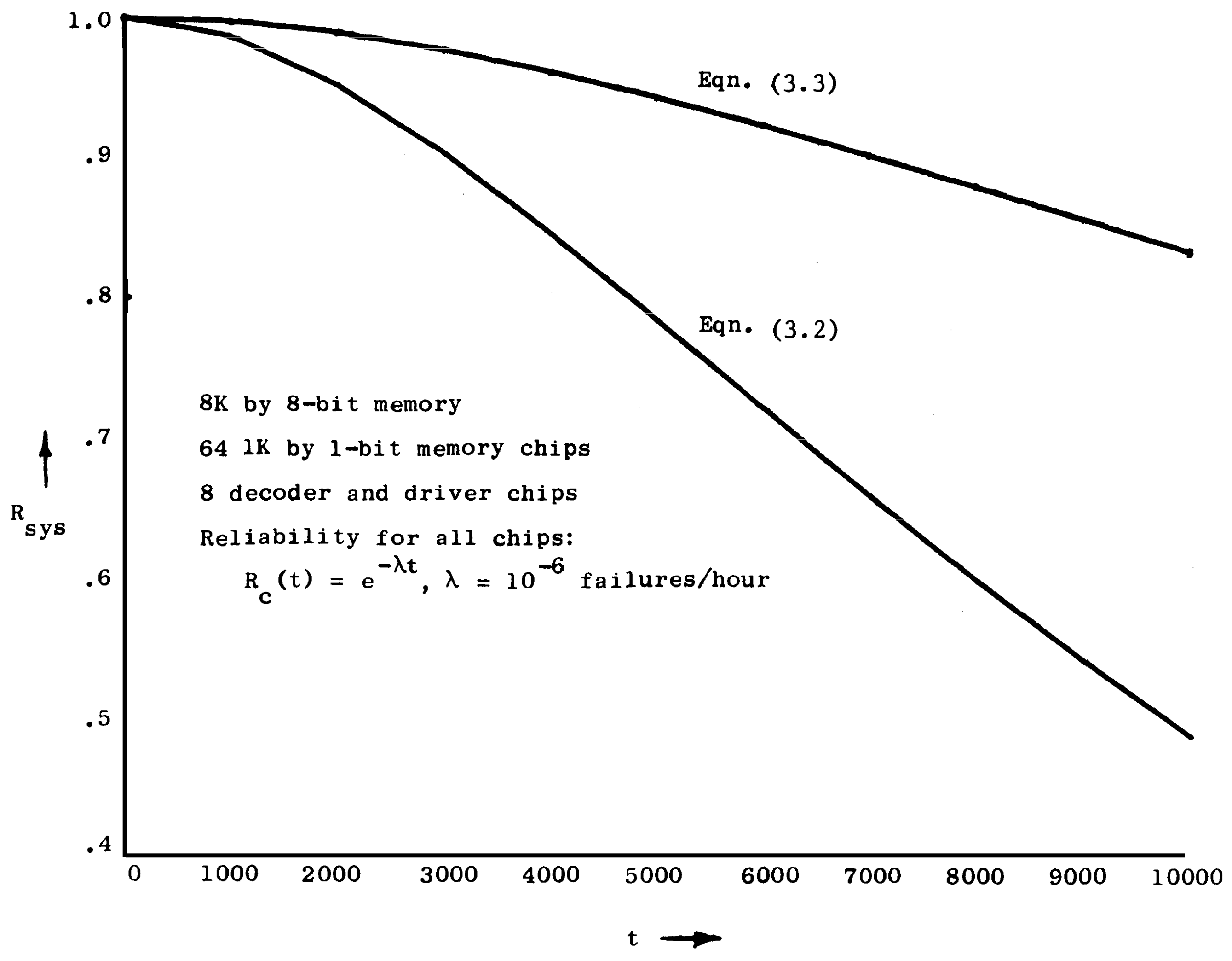


Fig. 3.3 TMR memory system reliability.

3.1.3 A TMR Microcomputer System

The thought of applying TMR to microcomputer systems raises some interesting questions. First of all, since a microprocessor is just a single chip, there is some question whether reliability can really be increased in a system that must use many voter chips constructed from the same unreliable technology as the microprocessor itself. Secondly, a microprocessor is a rather complex sequential machine with only limited access to its internal state, and so special care must be taken if the system is to tolerate multiple transient failures.

We will use the simple model of a microcomputer system shown in Fig. 3.4. The system consists simply of a microprocessor and memory, with data, address, and control lines going from the microprocessor to memory and data lines going from the memory to the microprocessor. Connections to peripherals are ignored; for the TMR system it is assumed that each peripheral interface has voters which monitor the I/O commands given by all three triplicated processors.

A typical LSI microprocessor is the Intel 8080 [Intel, 1974]. The 8080 is an 8-bit processor in a 40-pin package. It has 16 address lines, an 8-bit bidirectional data- bus, and 9 control lines entering and leaving the chip. The data bus must be externally split into two one-way buses for voting to be applied, and hence there are a total of 41 lines in an 8080 system that could be voted on. Since three voter circuits can be placed on a single 14-pin package, it is conceivable that a TMR 8080 system could have 3 8080 packages and 41 voter packages (triplicated voters) or 14 voter packages

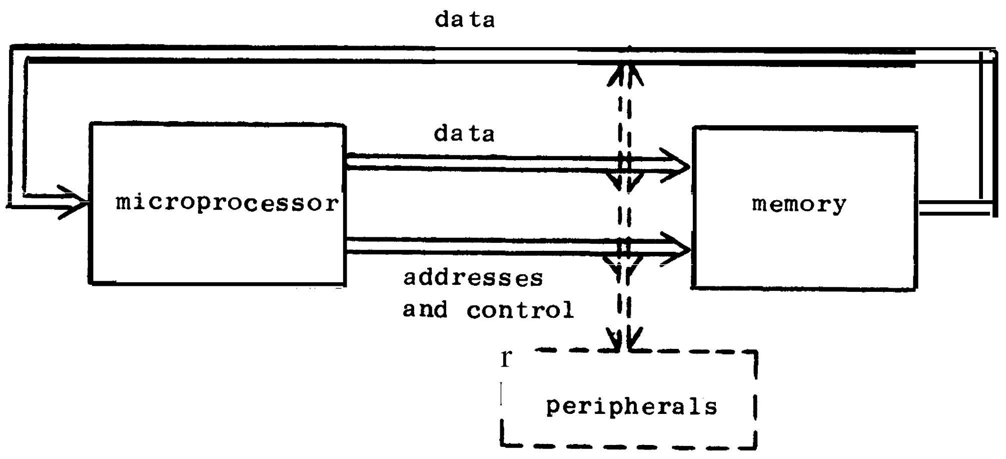


Fig. 3.4 Microcomputer system model.

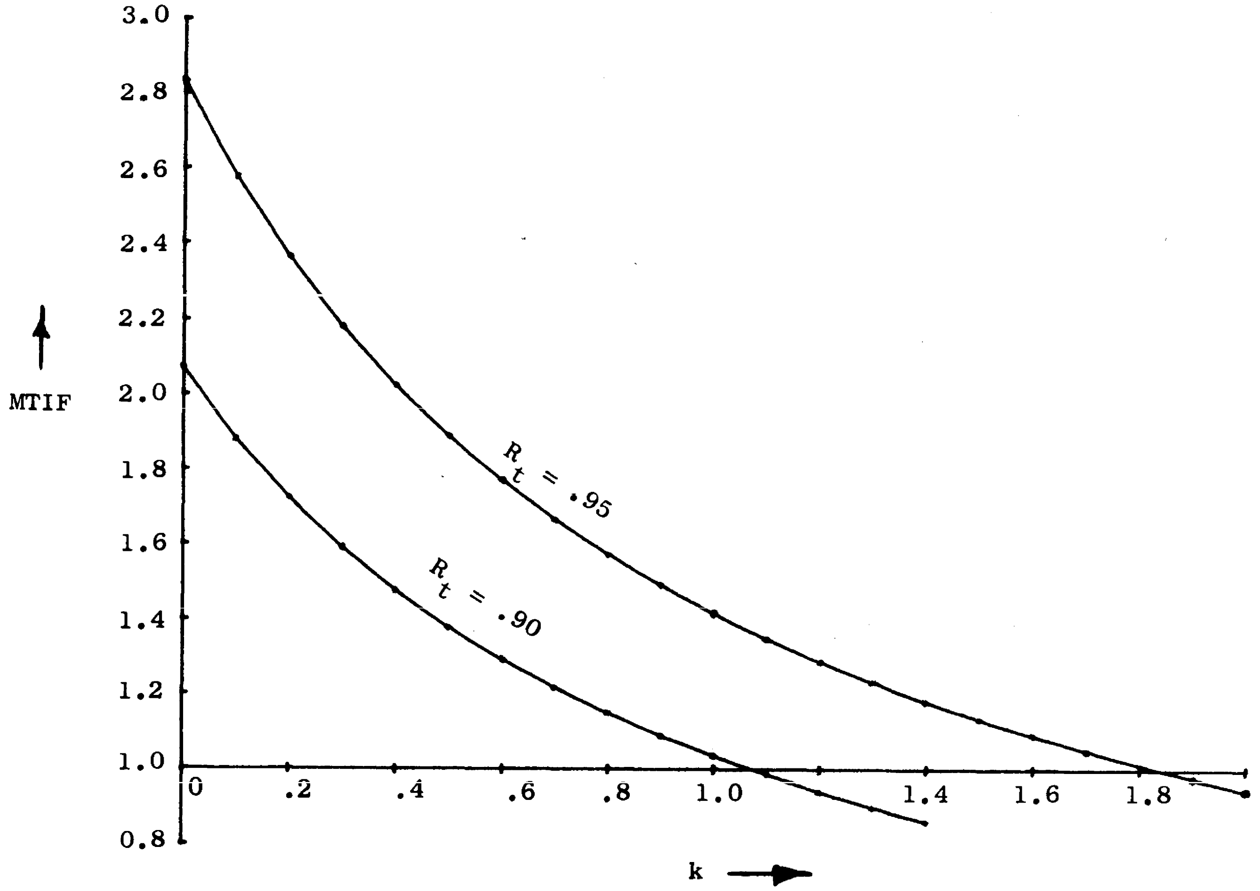


Fig. 3.5 Mission time improvement factor for a TMR microcomputer system.

(nontriplicated voters). Since a large percentage of integrated circuit failures are related to problems in packaging and I/O pins rather than circuit complexity, it is quite conceivable that the total voter unreliability in a TMR microcomputer system could approach or even exceed the microprocessor unreliability. In such a system the use of TMR could actually decrease the overall system reliability.

We can use a very simple model to justify the above thesis. Suppose that the reliability of a microprocessor module is R_m and the voter reliability is R_v. If n voters are required for each replicated module in a TMR system, then the total voter reliability is R_v^n. The total voter reliability can be related to the module reliability by a factor k such that $R_v^n = R_m^k$. The factor k can be interpreted to mean that the total failure rate of the voters is k times the failure rate of the microprocessor module. For a system with several voter packages for every microprocessor package, k could be in the range .1 (very reliable voters) to 2 or more (voter reliability comparable to microprocessor reliability). The reliability improvement obtained by using TMR for various values of k is shown in Fig. 3.5. The figure of merit used is the mission time improvement factor (MTIF), that is, the ratio of the mission times of the TMR system and the corresponding nonredundant system. (The mission time is the amount of time it takes for the system to degrade from its initial reliability of 1 to some terminal reliability R_t.) For the perfect voter case (k=0), the theoretical maximum MTIF is-obtained, 2.84 for R_t=.95 or 2.08 for R_t=.90; but for imperfect voters (k>0), the MTIF can be much less. If module and total voter reliabilities are equal (k=1), the MTIF is only 1.42 for R_t=.95, and about 1.0

for R_t=.90 (the mission times of the TMR and the nonredundant systems are equal). If the total voter failure rate is twice the module failure rate (k=2), the TMR system has a shorter mission time than the nonredundant system for either value of R_t.

Of course the above model is an oversimplification because it neglects the reliability of memory and other support circuits that are present in typical microcomputer systems. Including these components as part of the module would increase voter reliability relative to module reliability and hence reduce the value of k. However, the improvement is not dramatic and we have found that for practical systems it is often still worthwhile to try to increase total voter reliability by decreasing the number of voters.

Recalling that there are 41 lines that might be voted on in an Intel 8080 system, the most drastic reduction in the number of voters would be obtained by voting on none of the lines. The system would consist of three identical microcomputer/memory systems, each initialized to the same starting state and operating synchronously from a common fault-tolerant clock. Peripherals would have their own internal voters and they would perform operations as dictated by the majority of the replicated address, data, and control lines of the three identical systems. The problem with this scheme is that there is no mechanism for a microprocessor/memory system to be resynchronized after a transient failure, since there is no coupling among the replicated systems. Even if transient failures do not occur, we shall see that this system is less reliable than the next system we describe.

Suppose that voters are placed at the master reset input and the 8 data inputs of each microprocessor, as shown in Fig. 3.6. The address, data out, and control lines of each microprocessor go directly to the corresponding

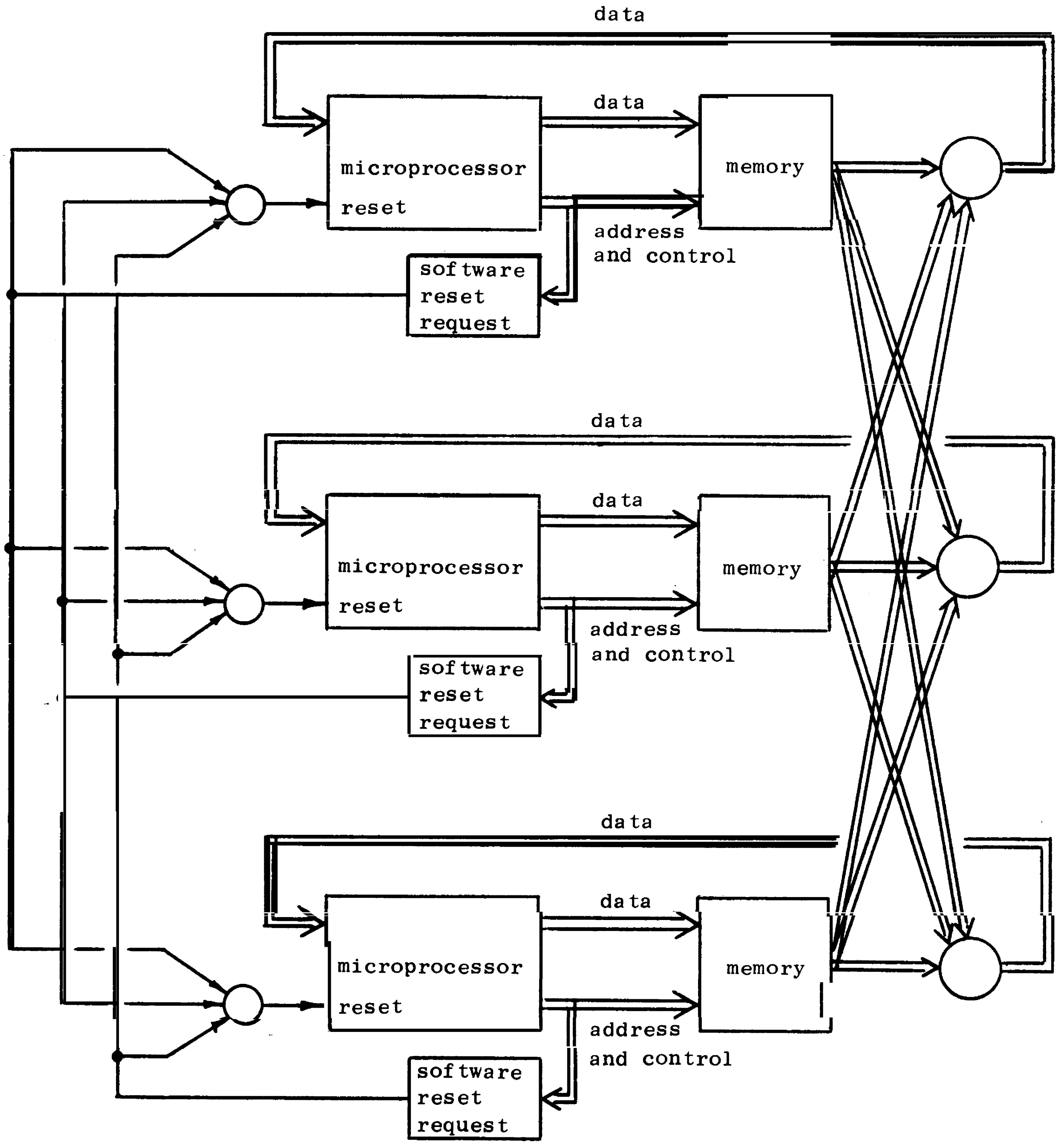


Fig. 3.6 Minimum TMR microcomputer configuration for resynchronization.

memory module without any voting. We have proved that this configuration has the minimum number of voters needed to provide re-synchronization after transient failures. For example, suppose a transient failure causes several registers of one microprocessor and several words in the corresponding memory module to contain incorrect data. Each of the incorrect registers is resynchronized with correct data if it is loaded from memory, since the voters insure correct memory output regardless of any possible errors in the state of one of the memories. Once the microprocessor is resynchronized, the memory is resynchronized by loading the incorrect memory words from the microprocessor.

Of course, it is possible that a transient failure can affect not only the register state but also the program state of a microprocessor. In general the microprocessor can attain any erroneous state and before being resynchronized it can create arbitrary errors in the corresponding memory module. It is this possibility that necessitates a voter on the master reset line of the microprocessors. Associated with each microprocessor is some interface circuitry that can be instructed by the software to initiate a hardware reset. Periodically the software would cause such a reset to occur, and since the reset line is voted on, a completely unsynchronized microprocessor must still obey the reset command. The reset command causes the microprocessor to begin executing a routine at some fixed location. The routine in this case is a synchronizing routine that first initializes all of the processor registers from memory, and then corrects any possible errors in a single memory module by sequentially reading and then rewriting every word in the memory.

Resynchronization after transient failures is also possible if voters are placed in the data lines going to the memory rather than coming out of the memory. However, placing the voters in the output lines results in better system reliability, since the memory reliability improvement for semiconductor memories discussed in section 3.1.2 is applicable. This improvement does not occur when voters are placed on the memory input only, since a single bit error in the memory output can produce several erroneous bits when the affected data is processed by the microprocessor.

The reliability analysis of the system in Fig. 3.6 is similar to the analysis of memory systems in section 3.1.2. In fact, equation (3.3) can be used to find the system reliability neglecting voters by simply considering the microprocessor to be part of the common circuitry of the memory module (R_d). In a similar way the reliability of the scheme with no voters and the scheme with voters on the data inputs only can be derived by use of equation (3.2). The exact reliability including voters for a nonredundant system and the three TMR system configurations is shown in Figs. 3.7 and 3.8 for two typical sets of parameters. It can be seen that for these two cases TMR increases the mission time by a factor of 3 to 4 for a terminal reliability of .95.

The work in this and the previous section is still in progress and a technical report is in preparation.

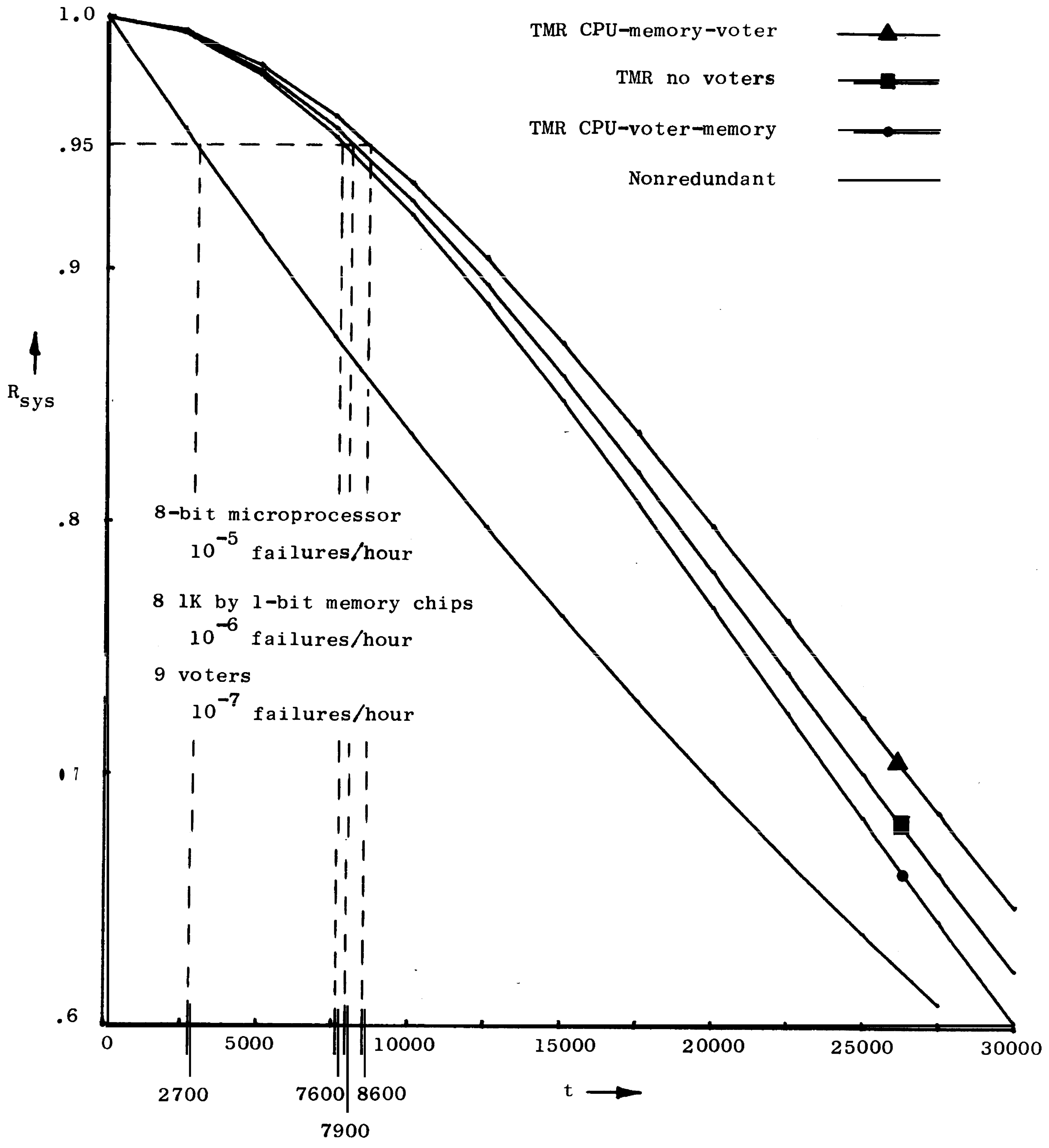


Fig. 3.7 TMR microcomputer system reliability.

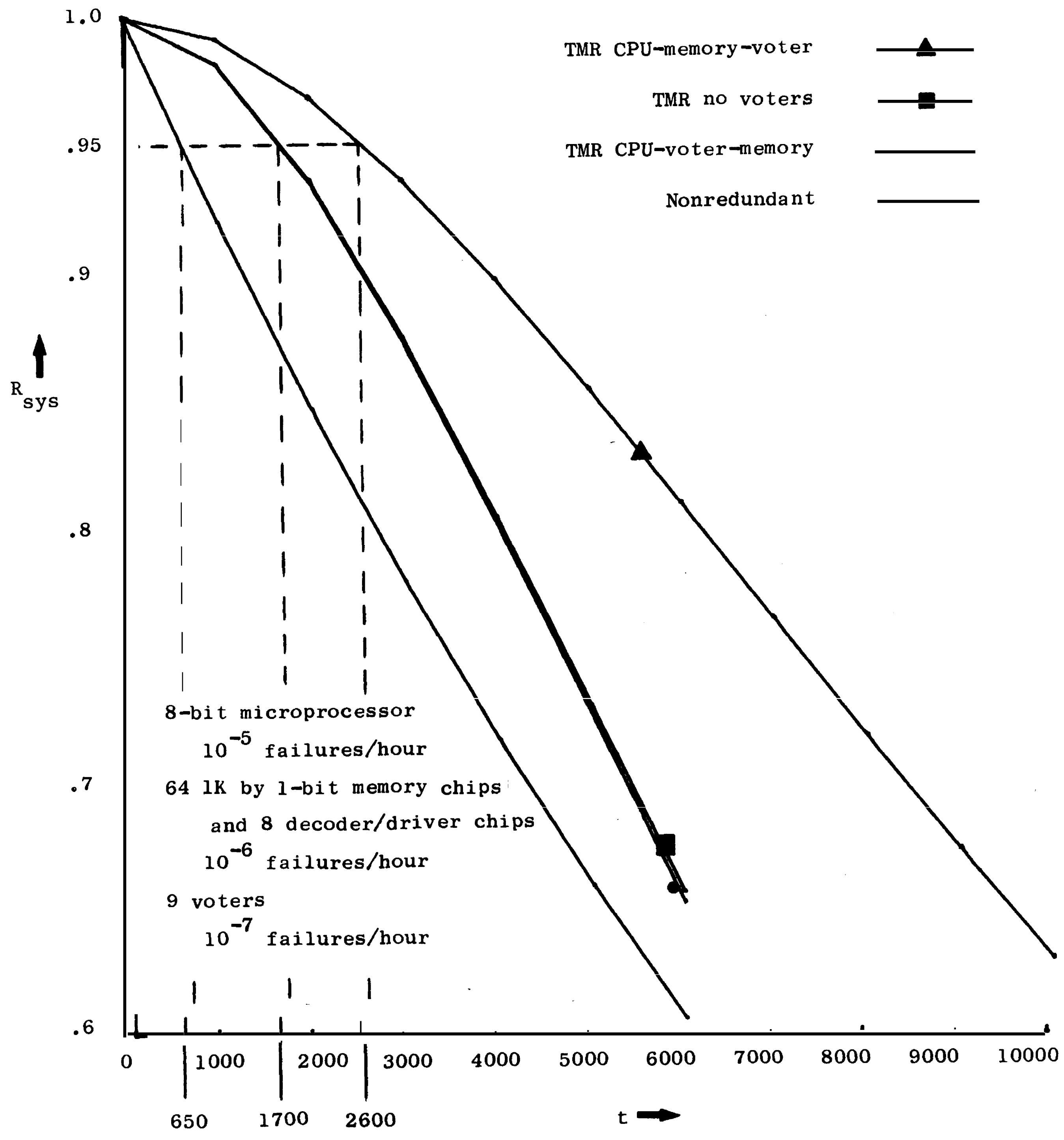


Fig. 3.8 **TMR microcomputer system reliability.**

3.2 Self-Checking Circuits*

The theory of self-checking combinational circuits was studied by the proposed principal investigator under previous NSF grants (GJ-27527 and GJ-40286). Recent work has focused on applications of self-checking combinational circuits and on the theory and applications of self-checking sequential circuits. Self-checking adders using checksum codes have been studied (section 3.2.1); a self-checking checker for periodic signals such as clocks has been designed (section 3.2.2); and a design approach for fail-safe sequential machines using realistic fault assumptions has been developed (section 3.2.3).

3.2.1 Checked Binary Addition Using Checksum Codes and Check Symbol Prediction

The code words of a checksum code are vectors of b-bit bytes with a single check byte that is the modulo 2^b sum of the data bytes. A checksum code can detect any error that affects only a single byte of a code word, and hence these codes are quite effective for detecting failures in data transmission and storage in byte-sliced systems. In such systems the circuits that handle the data are partitioned into b-bit byte slices, and hence a single component failure always results in a detectable single-byte error.

Checksum codes are not arithmetic codes, so that the check symbol of the sum of two code words cannot be derived from only the check symbols of the given code words. Hence, addition of two code words cannot be

* The work in this section has been supported by both NSF grants GJ-40286 and GK-43322.

checked by simply adding the check symbols, as in arithmetic codes. However, we have shown that check symbol prediction techniques can be used effectively in the design of self-checking adders for checksum code words. Such techniques had been developed earlier for simple parity-check codes [Sellars, 1968], requiring all of the inter-bit carry circuits of an adder to be duplicated. We have developed a similar technique for checksum codes which requires only the inter-byte carries to be duplicated [Wakerly, 1975b]. While duplicating the inter-bit carries requires an overhead of over 50% in the adder circuitry, duplicating the inter-byte carries requires only about 12% extra circuitry in a conventional MSI adder chip.

Although a self-checking adder for checksum code words is still slightly more expensive than a self-checking adder for words in a residue code (an arithmetic code), the checksum code still enjoys some advantages. As mentioned earlier, a checksum code detects all errors in a single b-bit byte. The residue code detects all single-byte errors except all 0's changed to all 1's or vice versa. Hence the single-byte error-detecting capability of a checksum code is superior to a residue code. For applications where single-byte error detection is of primary importance, a checksum code can be used to provide the required error detection capability; checked addition can still be performed at a cost only slightly higher than with residue codes, and much lower than with parity-check codes.

3.2.2 A Self-Checking Checker for Periodic Signals

Periodic signals have a known behavior, and deviations in their waveforms may indicate failures in the signal source. Monitoring these signals can be a valuable technique for detecting both hardware and software failures in a computer. Many circuits have been presented in the literature to check for errors in these signals (for example, see [Koczela, 1971] or [Chang, 1973]). These circuits, however, share a common weakness in that they are all susceptible to undetected internal failures; that is, they are not self-testing.

A totally self-checking periodic signal checker has been designed [Usas, 1975a]. As shown in Fig. 3.9, the checker consists of two monostable multivibrators M1 and M2. M1 is triggered on the leading edge of the input signal and produces an output pulse of fixed width equal to t_{on}, the expected on-time of the input signal. M2 is triggered on the falling edge of the input and produces a pulse of duration t_{off}, the expected off-time of the input.

The waveforms in Fig. 3.9 show that the output of M1 is a regeneration of the input while M2 produces the complement of the input. Hence during correct operation the checker output is always either 10 or 01. If the input signal becomes stuck on or off, the checker output becomes 00, and if the on-time or off-time vary more than a few percent from their expected values the checker output becomes either 00 or 11. It has been shown [Usas, 1975a] that failures in the monostables themselves produce either the 00 or the 11 output, and the checker is totally self-checking.

There still remains the issue of monitoring the checker output and sounding an alarm when the 00 or 11 output appears. This issue has been thoroughly discussed in [Usas, 1975d], and fail-safe solutions have been presented.

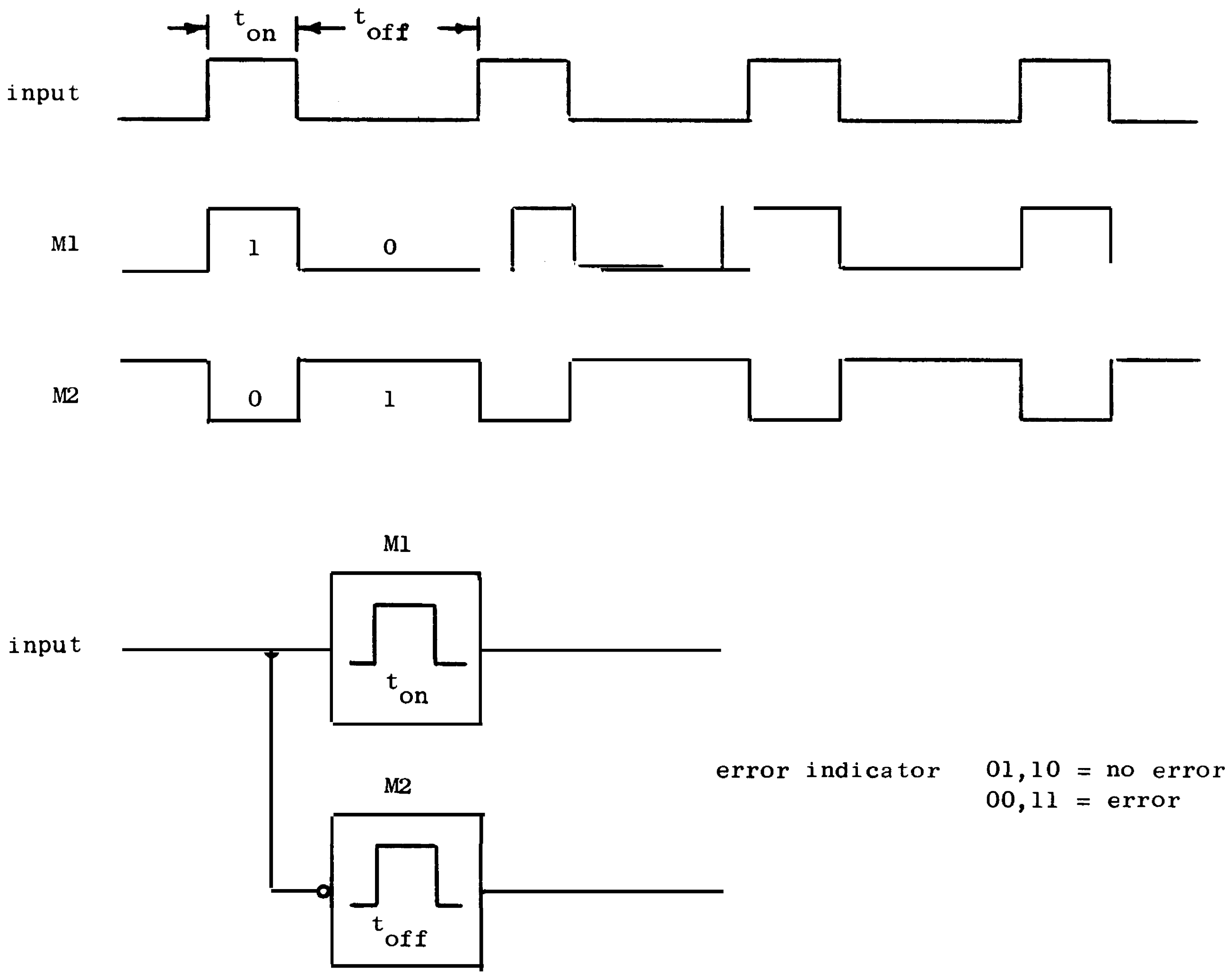


Fig. 3.9 Totally self-checking clock checker.

3.2.3 Fail-Safe Sequential Machines

Self-checking circuits have two properties, self-testing and fault-secureness. The self-testing property guarantees that all faults are tested in normal operation, while fault-secureness guarantees that any fault produces either the correct output or an error indication. The fault-secureness property for self-checking circuits is equivalent to the definition of "fail-safe" given by Tokaoka and Ibaraki [1972]. For historical reasons the name "fail-safe!' has been applied to fault-secure sequential machines.

A number of researchers have recently been concerned with the design of fail-safe (or fault-secure) sequential machines. Most of the contributions to a solution of the problem have assumed a simple delay element as the realization of the required memory function [Diaz, 1973; Sawin, 1974]. Tohma [1974] has suggested a design approach using more practical and familiar memory devices such as JK flip-flops. However, he assumes that the double-rail outputs of a faulty flip-flop are always complementary. That is, if the fault causes one output line to be an erroneous 1(O), then the other line is always an erroneous O(1). Unfortunately this is not the case if one considers the circuit implementation of a JK flip-flop and common integrated circuit failure modes. Usas [1975b] has presented a design approach that is based on the use of D flip-flops as memory elements and that assumes a more realistic model of the faulty behavior of the flip-flop.

Analyzing the circuit implementation of a conventional D flip-flop, Usas [1975b] shows that there do indeed exist faults such that the flip-flop outputs are identical, but that no fault results in both erroneous 1's and 0's on the flip-flop outputs. That is, if a fault f produces an erroneous 1(0) on a flip-flop output for some input value, then there is no input that will produce an erroneous 0(1) on that output in the presence of f. This permits the erroneous value to be held in the flip-flop by a suitable choice of next-state mapping for illegal states. The design procedure results in a machine such that any flip-flop failures cause the machine to go to and remain in a "trap" state. Trap states can easily be detected in order to produce an error signal.

For the detection of single flip-flop faults, the design technique requires only one extra flip-flop over the normal design. However, the combinational excitation circuitry is more complex because the technique requires each flip-flop to have a separate excitation circuit; sharing is not allowed. A second design procedure is given that uses more extra flip-flops, two for machines with up to 126 states and three for up to about 2^{37}. This procedure allows shared logic realizations of the flip-flop excitation functions and usually results in a lower overall cost for the machine.

3.3 Self-Diagnosing Computer Design

The object of this research has been to develop design principles for self-diagnosing computer processors, machines with extra hardware for automatic detection and diagnosis of internal failures. This work began under NSF grants GJ-27527 and GJ-40286, and since July 1974 has been supported by GK-43322.

Our previous efforts were documented by a report [Wakerly, 1973] and a summary [Wakerly and McCluskey, 1974]. A research monograph based on this work is also being written. Our current efforts are towards applying the general results of the previous studies to the design and actual implementation of a self-diagnosing computer processor. Several aspects of such a design were not dealt with by the previous studies and remain to be explored.

The planned processor has 32-bit data paths and a microprogrammed control unit. The instruction set has a full complement of arithmetic, logical, and shift operations, with both register-to-register and register-to-memory operations available. The execution time for a typical register-to-register instruction will be about 1.2 microseconds. The data paths and control unit of the processor have been designed and the microcode is currently being written.

There have been a number of previous efforts of self-checking computer processor designs like our self-checking design; these designs used error-detecting codes and some duplication to detect hardware failures. However, our design is different from previous designs in several respects. First of all, we are designing a general-purpose processor with an instruction set and execution times comparable to typical minicomputers. Our processor can be contrasted with the JPL STAR [Avizienis, 1971] which was a byte-serial machine, and Bell Labs ESS [Chang, 1973] which had no add instruction and required two microcycles for every register-to-register transfer. Secondly, the cost of checking in our design will be less than 25% of the total hardware cost for a 32-bit processor. Thirdly, our design makes efficient use of standard MSI integrated circuits, and the data paths and control unit are designed to allow straightfarward integration as LSI circuits comparable to those available commercially today. Finally, although previous machines claimed a high degree of self-checking for single gate faults, in our design we can guarantee detection of 100% of the failures that affect a single integrated circuit package.

3.3.1 Data Path Checking

The previous studies [Wakerly, 1973] showed how error-detecting codes and self-checking circuits could be used to detect errors in the data paths of a typical processor. The processor data paths are byte-sliced, so that each register, adder, multiplexer, etc.,- that handles a b-bit byte of data is a separate component. The data paths of an nb-bit processor can then be implemented by n b-bit slices operating in parallel, with appropriate interconnections for arithmetic carries, shifts, etc. In such a system, a single component failure affects only a single byte, and errors can be detected by the use of a b-bit residue code. The residue code associates with each datum in the processor a b-bit check symbol which is the residue of the datum modulo 2^b-1. An extra b-bit check slice is thus required to store and process the check symbols of the data residing on the data path slices. In our previous work we showed that the extra check slice is sufficient to detect all single byte errors regardless of the actual data path width, be it 3b bits, 8b bits, or more. However, it was not clear at that time whether the check slice could be the same as the data path slices in a practical design or whether the check slice would have to be different.

Our current work has shown that it is possible to design a standard byte slice that serves for both data and check symbols, at a cost comparable to a byte slice for an unchecked machine. Hence for an nb-bit machine using b-bit slices, the cost of data path checking is about 1/n, or 13% for a 32-bit machine using standard 4-bit components.

Our studies have shown that the only essential difference between a check slice and a data slice is that the check slice must have a provision for modifying the check symbol for certain arithmetic operations and reloading it for non-code-preserving operations. Except for this the check symbols are processed exactly the same as corresponding data in the data slices. Hence we have designed a 4-bit slice, a simplified version of which is shown in Fig. 3.10. The byte slice has a 16-word general register file, 8 scratch pad registers, memory address and data registers, shifting and byte-swapping logic, and ALU. A processor that uses this slice can efficiently perform all of the operations of a typical minicomputer: loading

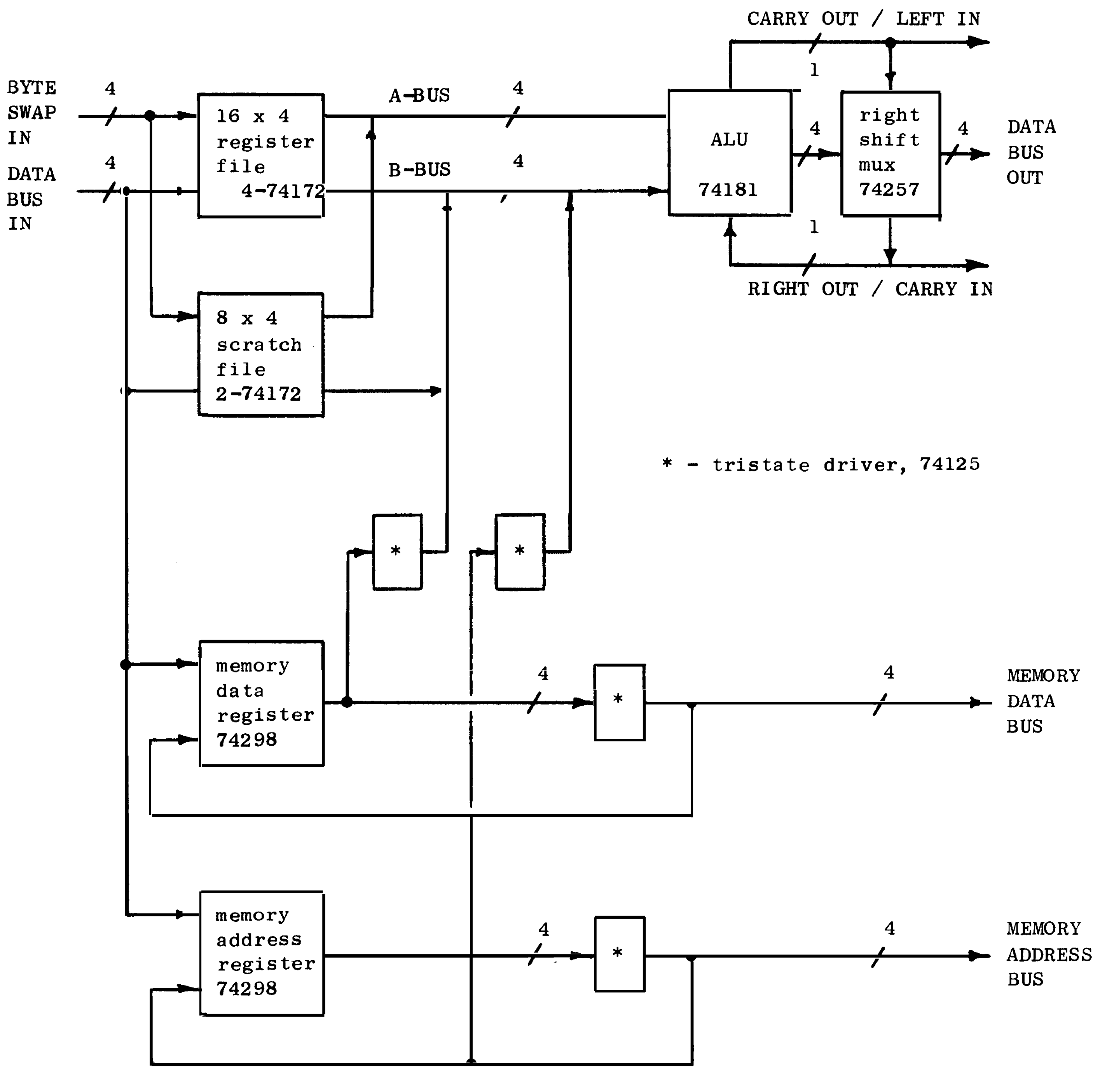


Fig. 3.10 **Processor data path slice.**

and storing data, arithmetic and logical operations, shifts and rotates, microprogrammed multiply and divide.

The only difference between the slice of Fig. 3.10 and a conventional slice is that the data path from the ALU output back to the register inputs is broken; in a conventional slice this connection is usually already made. The broken connection allows the system implementation shown in Fig. 3.11. For the data path slices, the ALU output is connected directly to the register inputs. For the check slice, the ALU output goes through combinational logic that modifies the check symbol as required before returning it to the registers. At the end of each machine cycle, the output of the data and check slices is loaded into a check register and checked for validity. For non-code-preserving operations it is possible to reload the check slice with a new check symbol generated from the non-code result.

The byte slice suggested by Fig. 3.10 has been designed in detail. Using conventional TTL integrated circuits, the slice can perform one operation every 200 nanoseconds. In our current design the combinational logic external to the check slice is implemented with a small, fast read-only memory and adds 30 nanoseconds to the maximum cycle time. Hence checking requires a 15% overhead in the cycle time of the machine.

The data path slice of Fig. 3.10 is similar to currently available LSI data path slices [Monolithic Memories, 1974; Rattner, 1974]. In fact, the slice has been designed with LSI in mind; it should be possible to fabricate the entire slice on a single integrated circuit chip. Our data path slice design requires no more circuitry and only four more pins than a comparable slice that cannot be used as a check slice.

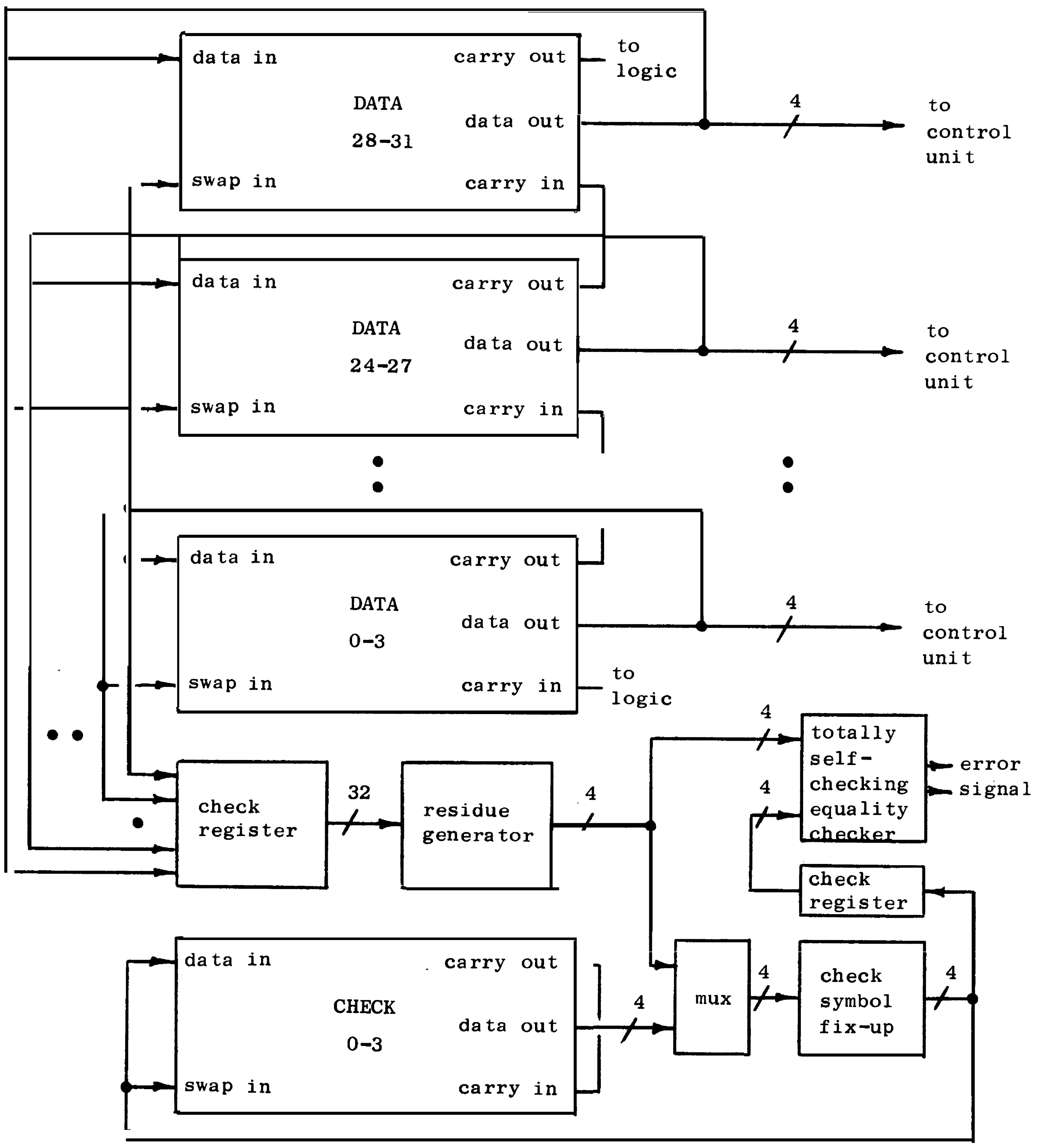


Fig. 3.11 Configuration of byte slices for 32-bit processor.

3.3.2 Control Unit Checking

In our previous work [Wakerly, 1973] we showed how error-detecting codes can be used to detect errors in the microprogram storage of a microprogrammed control unit. If the storage is implemented using b-bit by n-word ROM chips, then only one extra ROM chip is needed to hold b check bits for each word and insure detection of all single package failures. In our present processor design we expect to have 1K of 44-bit microprogram words implemented with 11 1K by 4-bit chips. One extra chip is required for checking and hence the overhead for checking the microprogram storage is 1/11, or about 9%.

In addition to microprogram storage a microprogrammed control unit requires registers to hold the instruction and the microprogram address and a fair amount of combinational logic for computing the next microprogram address. The next address could be computed for example as the next sequential address, as a subroutine return address, or as a function of internal flags (carry, overflow, etc.). In [Wakerly, 1973] we indicated that the only way to detect errors in some of the control unit functions was by duplication, while errors in other functions could be detected by clever use of the existing coding and by taking extra microprogram steps for checking. In our subsequent studies we have found that the "trick" methods are not sufficiently general for efficiently implementing a typical processor, and so we have rejected them in our design. Instead we plan to duplicate the instruction register, the microprogram address register, and the next address logic.

As shown in Fig. 3.12, there will be two identical "microprogram control" modules. The modules operate in parallel, so that both load their instruction register from the processor data bus, test processor flags, and sense microprogram jump conditions and addresses. The output of a module is a 10-bit microprogram address. The output of one of the modules is used to address the microprogram memory while the output of the second is compared with the first for equality.

Of course, the overhead for duplication of the control module is 100%, far from our overall system goal of 25%. However, the control module is only a part of the control unit, and the rest of the control unit (i.e., the microprogram memory) requires an overhead of only 9%. The fraction of the total control unit cost attributable to the duplicated control module depends on how that module is implemented. The present control module is designed with LSI in mind, so that the entire module could be fabricated on a single chip comparable to existing commercially available LSI control units chips [Rattner, 1974]. Only that single LSI package would be duplicated.

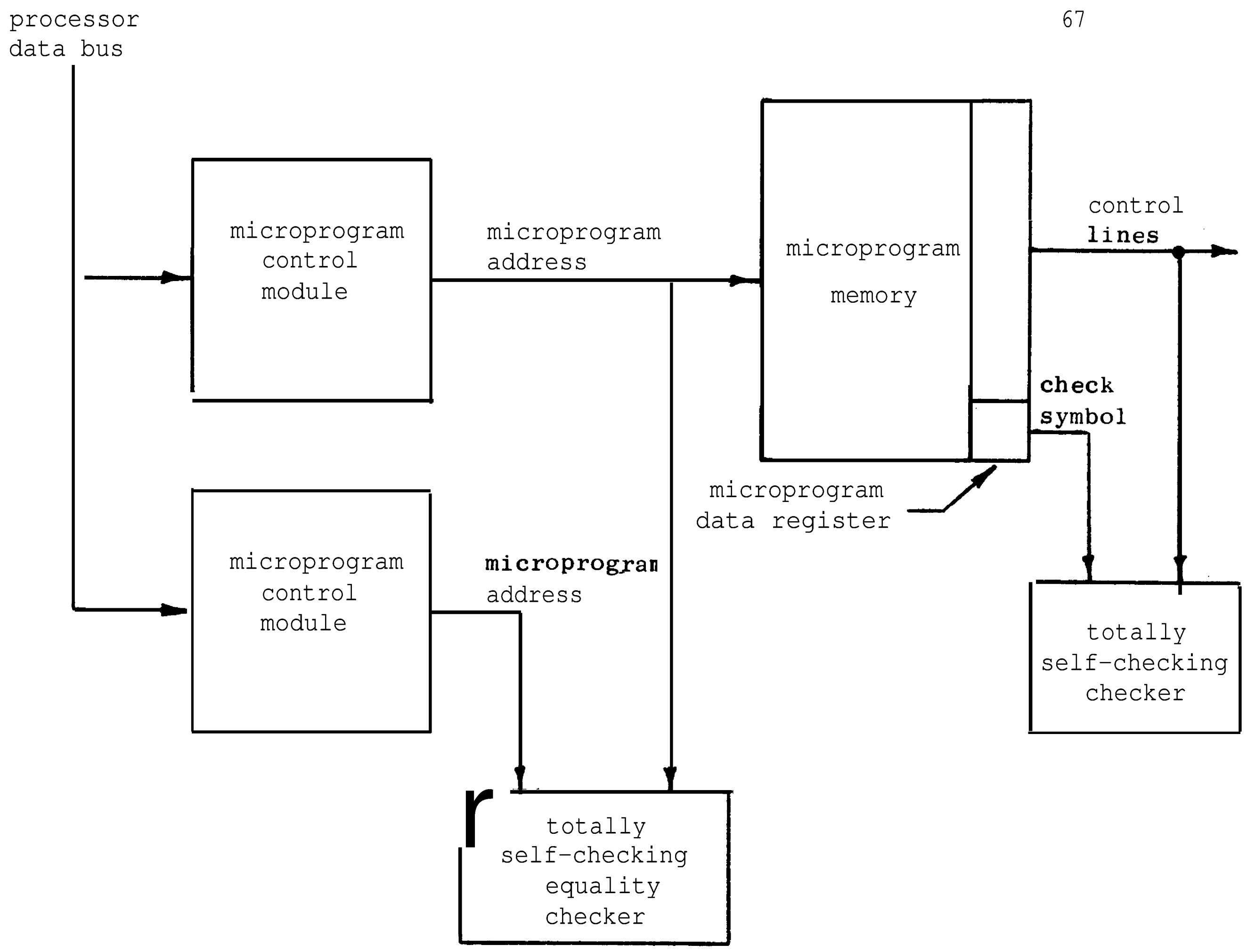

processor data bus
microprogram control module
microprogram address
microprogram memory
control lines
check symbol
microprogram data register
microprogram control module
microprogram address
totally self-checking checker
totally self-checking equality checker

Fig. 3.12 Microprogrammed control unit structure.

3.3.3 Diagnosis

After an error is detected in the self-checking computer processor, diagnosis must be initiated to isolate the failure to one replaceable module. Diagnosis will be performed by special microprograms called microdiagnostics. Although these microprograms have not been written, the hardware features necessary to support diagnosis have been considered in the present processor design. The goal is to minimize the amount of hardware that must be working for successful diagnosis to take place.

Assuming single package failures, any failure in the data paths of the machine can be easily isolated by microdiagnostics to one replaceable slice. Since there is one extra data path slice for checking, an operator can manually reconfigure the machine for continued operation by replacing the failed slice with the check slice and disabling data path checking until repairs are completed.

A failure in a single microprogram memory ROM package can be detected, but using the present single-error-detecting code diagnosis is not possible unless another copy of the ROM contents are available for comparison. This would mean duplicating the entire microprogram memory or having a copy available on auxiliary storage. If the ROM failure is not catastrophic, then it is possible to load a copy of the microprogram from external storage and perform the check. However, a catastrophic failure of a ROM package may affect all microinstructions and prevent this diagnosis from taking place. An alternative currently under study is to use a more sophisticated code to provide both detection and location of ROM failures. For memory words of up to 60 bits, two 4-bit check symbols in a 2-redundant b-adjacent code [Bossen, 1970] are sufficient to detect and locate (or even correct) all 4-bit errors.

A failure in one of the two duplicated control modules (Fig. 3.12) is detected by the equality checker monitoring their microprogram address outputs. Once a failure is isolated to one of the two modules, the other can manually be made the primary module (if it isn't already), and system operation can continue with checking disabled. The problem then is to determine which of the two modules is the faulty one. There are currently manual procedures for making this determination, but no automatic procedures short of triplication and voting have yet been devised.

The work in this and the previous two sections is still in progress and a technical report is in preparation.

3.3.4 Peripheral and Input/Output Checking*

This work has been directed toward the problems of detecting errors and performing fault diagnosis in the components associated with the input/output system of a computer. Such units as CPU interfaces, channels, bus controllers, device interfaces, device controllers, and devices have been studied with the goals of guaranteeing an indication of the occurrence of the error (self-checking) and providing expeditious repair (self-diagnosing).

Checking the validity of data transferred between a CPU and peripherals is fairly simple -- the same codes used to check memory data or CPU data

* The work in this section has been supported by both NSF grants GJ-40286 and GK-43322.

can be used to check the data transfers over I/O buses. Problems and solutions in interfacing between different codes for processors and peripherals have been given [Wakerly, 1973]. However, there remain unsolved problems in controlling the transfers of data between processors and peripherals, problems that do not occur in the design of a synchronous CPU. Since I/O buses usually have at least some asynchronous signals and some "hand-shaking", there are many new problems to consider. The sequential nature of I/O control led to the more fundamental studies of self-checking sequential machines described in sections 3.2.2 and 3.2.3. With these new techniques, self-checking controller designs are possible. In addition, specifications for signaling in digital buses to insure error detection have been outlined.

Error detection in peripheral devices is a difficult problem because of the diversity of devices in common use. However, the problem has been approached by considering device classes (disks, tapes, card equipment, etc.), and a list of applicable methods has been compiled. Also, the design of a CPU-controlled probe for peripheral diagnosis has been completed and operational guidelines are being assembled.

This first phase in research work on input/output fault detection and diagnosis is near completion and will be documented by a Ph.D. thesis [Usas, 1975c].

3.4 REFERENCES PREPARED UNDER PREVIOUS NSF GRANTS

* GJ-27527

† GJ-40286

+ GK-43322

* Abraham, J. A., and D. P. Siewiorek, 1974. "An algorithm for the accurate reliability evaluation of Triple Modular Redundancy networks," IEEE Trans. Comput. C-23(7): 682-692.

* Ogus, R. C., 1974. "Fault-Tolerance of the iterative cell array switch for hybrid redundancy," IEEE Trans. Comput. C-23(7): 667-681.

† Ogus, R. C., 1975. "Reliability analysis of hybrid redundancy systems with nonperfect switches," Technical Report 65, Digital Systems Laboratory, Stanford University, Stanford, California.

† Shedletsky, J. J., and E. J. McCluskey, 1974a. "The error latency of a fault in a combinational digital circuit," Technical Note 55, Digital Systems Laboratory, Stanford University, Stanford, Ca.; submitted to 1975 Int'l. Symp. on Fault-Tolerant Computing.

† Shedletsky, J. J., and E. J. McCluskey, 1974b. "The error latency of a fault in a sequential digital circuit," Technical Note 56, Digital Systems Laboratory, Stanford University, Stanford, Ca.; submitted to 1975 Int'l. Symp. on Fault-Tolerant Computing.

+† Usas, A. M., 1975a. "A totally self-checking checker for the detection of errors in periodic signals," IEEE Trans. Comput. C-24(5).

+† Usas, A. M., 1975b. "Design of fail-safe synchronous sequential circuits by explicit state trapping," Technical Note 52, Digital Systems Laboratory, Stanford University, Stanford, Ca.; submitted to 1975 Int'l. Symp. on Fault-Tolerant Computing.

+† Usas, A. M., 1975c. "Fault detection and diagnosis in digital computer input/output systems," Ph.D. thesis, Stanford University, Stanford, Ca. (in preparation).

† Usas, A. M., 1975d. "The detection of errors in periodic signals," Technical Note 45, Digital Systems Laboratory, Stanford University, Stanford, Ca.

*† Wakerly, J. F., 1973. "Low-cost error detection techniques for small computers," Ph.D. Dissertation, Department of Electrical Engineering, Digital Systems Laboratory, Stanford University, Stanford, Ca.

† Wakerly, J. F., and E. J. McCluskey, 1974. "Design of low-cost general-purpose self-diagnosing computers," Proc. IFIPS-74, Intl. Fed. Information Processing Societies, Stockholm, Sweden; Technical Note No. 38, Digital Systems Laboratory, Stanford University, Stanford, Ca.

+ Wakerly, J. F., 1975a. "Transient failures in triple modular redundant systems with sequential modules," IEEE Trans. Comput. C-24(5).

+† Wakerly, J. F., 1975b. "Checked binary addition using check symbol prediction and checksum codes," Technical Note 39, Digital Systems Laboratory, Stanford University, Stanford, Ca.; submitted to IEEE Trans. Comput.

3.5 ADDITIONAL REFERENCES

Avizienis, A., 1967. "Design of fault-tolerant computers," AFIPS Conf. Proc., 1967 FJCC 31: 733-743, Washington, D.C.: Thompson Books.

Avizienis, A., 1971. "The STAR computer: An investigation of the theory and practice of fault-tolerant computer design," IEEE Trans. Comput. C-20: 1312-1321.

Bossen, D. C., 1970. "b-Adjacent error correction," IBM J. Res. Develop. 14(7): 402-408.

Brown, W. G., J. Tierney, and R. Wasserman, 1961. "Improvement of electronic computer reliability through the use of redundancy," IRE Trans. Electron. Comput. EC-10: 407-416.

Chang, H. Y., et. al., 1973. "The design of a microprogrammed self-checking processor of an electronic switching system," IEEE Trans. Comput. C-22: 489-500.

Daly, W. M., A. L. Hopkins, and J. F. McKenna, 1973. "A fault-tolerant digital clocking system," Dig. 1973 Int'l. Symp. Fault-Tolerant Computing, IEEE pub. no. 73CH0772-4C: 17-21.

Diaz, M., J. C. Geffroy, and M. Courvoisier, 1973. "On-set realization of fail-safe sequential machines," Dig. 1973 Int'l. Symp. Fault-Tolerant Computing, 145-149.

Dickinson, W. E., and R. M. Walker, 1958. "Reliability improvement by the use of multiple-element switching circuits," IBM J. Res. Develop. 2: 142-147.

Dickinson, M. M., J. B. Jackson, and G. C. Randa, 1964. "Saturn V launch vehicle digital computer and data adapter," AFIPS Conf. Proc., 1964 FJCC 26: 501-516, Baltimore, Md., Spartan Books.

Flehinger, B. J., 1958. "Reliability improvement through redundancy at various system levels," IBM J. Res. Develop., 2: 148-158.

Gurzi, K. J., 1965. "Estimates for the best placement of voters in a triplicated network," IEEE Trans. Electron. Comput. EC-14: 711-717.

Intel Corporation, 1974. 8080 Users Manual, Santa Clara, California.

Johnson, A. M., 1971. "The microdiagnostics for the IBM 360/30," IEEE Trans. Comput. C-20: 798-803.

Koczela, L. J., 1971. "A three failure tolerant computer system," Dig. 1971 Int. Symp. Fault-Tolerant Computing: 101-104, Pasadena, California.

Longden, M., L. J. Page, and R. A. Scantlebury, 1966. "An assessment of the value of triplicated redundancy in digital systems," Microelectronics and Reliability 5: 39-55, Elmsford, N.Y.: Pergamon Press.

Lyons, R. E., and W. Vanderkulk, 1962. "The use of triple-modular redundancy to improve computer reliability," IBM J. Res. Develop. 6: 200-209.

Monolithic Memories, 1974. "5701/6701 4-bit expandable microcontroller," Sunnyvale, California.

Moore, E. F., and C. E. Shannon, 1956. "Reliable circuits using less reliable relays," Jour. Franklin Inst. 262: 191-208, 281-297.

Peterson, W. W., and E. J. Weldon, 1972. Error-Correcting Codes, Cambridge: MIT Press.

Plummer, W. W., 1972. "Asynchronous aribters," IEEE Trans. Comput. C-21(1): 37-42.

Ramamoorthy, C. V., and L. C. Chang, 1972. "System modeling and testing procedures for microdiagnostics," IEEE Trans. Comput. C-21: 1169-1183.

Rattner, J., J.-C. Cornet, and M. E. Hoff, 1974. "Bipolar LSI computing elements usher in new era of digital design," Electronics 47(18): 89-96.

Rubin, D. K., 1967. "The approximate reliability of triply redundant majority-voted systems," Dig. 1st Annu. IEEE Computer Conf. IEEE publ. 16051: 46-49.

Sawin, D. H., 1974. "Fail-safe synchronous sequential machines using modified on-set realizations," Dig. 1974 Intl. Symp. Fault-Tolerant Computing session 3: 7-12.

Sellars, F. F., M.-Y. Hsaio, and L. W. Bearnson, 1968. Error Detecting Logic for Digital Computers, New York: McGraw-Hill.

Tohma, Y., 1974. "Design technique of fail-safe sequential circuits using flip-flop for internal memory," IEEE Trans. Comput. C-23(11): 1149-1154.

Tokaoka, T., and T. Ibaraki, 1972. "N-fail-safe sequential machines," IEEE Trans. Comput. C-21(11): 1189-1196.

von Neumann, J., 1956. "Probabilistic logics and the synthesis of reliable organisms from unreliable components," Automata Studies 43-98, Princeton, N. J.: Princeton University Press.

Wensley, J. H., 1972. "SIFT-Software implemented fault tolerance," AFIPS Conference Proceedings, Fall Joint Computer Conference, 41(1): 243-253.

4. INVESTIGATION AND EVALUATION OF DUAL COMPUTER CONFIGURATIONS

4.1 Introduction

This report summarizes the current status of the project which is concerned with the investigation and evaluation of dual computer configurations.

For the past two years, work has been underway under the sponsorship of NASA Ames Research Center to study reliable computer systems for use in Short Takeoff and Landing (STOL) aircraft. A prototype system has been developed by the Charles Stark Draper Laboratories (CSDL) in Cambridge, Mass., for this purpose, and was delivered to the NASA Ames Research Center in mid-1975 for a series of flight tests under the direction of personnel from NASA. This system, known as the SIRU system [1], consists of a set of navigational instruments [the Strapdown Inertial Reference Unit (SIRU)], and a digital computer complex to process information from the SIRU and display this information to the pilot of the aircraft. The SIRU navigation package is fault-tolerant and it was desirable to develop a reliable computer system that would match the high reliability of the SIRU instrument system. The computer complex used is a dual processor, real-time system using two Honeywell H316 central processing units, with a special purpose arbiter component to evaluate the operation of the two Honeywell processors. Both processors execute the same algorithms, and the arbiter designates one processor as master and one as backup for each basic operational cycle.

The initial work on the project involved the study of the actual SIRU system itself. Research work is ongoing in this area, and we hope to use the SIRU system as a basis for a more general study of dual computer configurations. Theoretical studies have been carried out of both the hardware and software aspects of the dual computer system, and these investigations are continuing. A second thrust of the present research is to thoroughly evaluate the prototype system both using the theoretical results as guidelines, as well as through experimentation on the actual system. Based on these studies we hope to make recommendations which could improve the prototype system performance.

4.2 Summary of Technical Areas in the Project

4.2.1 The Siru computer system

CSDL carried out the initial design of the computer system [1], and the design was studied by the Digital Systems Laboratory (DSL). It was decided that closer collaboration between DSL and CSDL would prove fruitful and, as a result, meetings were held at CSDL in November 1973, May 1974 and December 1974 during which the progress of the design was discussed and reviewed. Thereafter, the final system design was decided upon, and CSDL carried out the actual construction of the system.

A very important component of the SIRU dual computer system is the arbiter unit. In the prototype system, the arbiter forms an opinion as to the functional status of the system and communicates this information to the pilot; in later versions of the system, however, the arbiter might be responsible for the control of actuators in the aircraft and thus would become a very critical unit. In the latter case, the arbiter would have to be a highly reliable module.

A reliability model has been developed to study the effect of the actual arbiter reliability on the overall system reliability. In the prototype system the arbiter is the only component which is not redundant and the model assumes that an arbiter failure will result in system failure. Results have been obtained which indicate the minimum reliability value the arbiter could have before the redundant system becomes less reliable than the nonredundant (simplex) system. Other aspects, such as the effect of the arbiter on the system mission times, have been studied, and we hope to generalize these studies to enable more general systems to be modeled.

We are also studying the possiblity of improving the arbiter for future systems. The first aspect is the improvement of the arbiter's reliability. Since the arbiter has an important role in the system and will become even more crucial in later versions of the system, it will become necessary to realize the unit using some type of redundant structure. Redundant implementations of the arbiter are being studied to determine which redundancy structure would be the most suitable for the arbiter from cost, reliability and speed viewpoints.

A second aspect is the increasing of the arbiter capabilities. A more powerful arbiter would be able to form a more accurate evaluation of the two processors, and make a better decision as to their possible malfunctioning. The idea of using a programmed microprocessor seems to be an attractive way of achieving this goal. The microprocessor would not only be more powerful than the present hardwired arbiter, but would also have an increased amount of flexiblity. In addition, improving the reliability of microprocessor systems through redundancy is at present being studied elsewhere in DSL, and promising results have been forthcoming [6].

An important aspect of the use of microcomputers in aviation systems is the necessity of obtaining microprocessors having military specifications. At present only one such device seems to be commercially available [7].

A final area of consideration is the possibility of replacing the H316 processors themselves with microprocessors. This would greatly reduce the size and weight of the system, factors which are important in aviation systems. The speed and capabilities of current microprocessors and semiconductor memory systems are presently being studied to determine the feasibility of their replacing the H316 processors.

4.2.2 Error recovery techniques in computer systems

A survey was carried out of the various types of error recovery techniques employed in current computer systems [2]. Some of the principal systems which use these techniques were described [2], and a bibliography containing some of the principal papers in the area of recovery techniques was also included in [2].

A significant aspect of highly reliable computer systems is their recovery capabilities. It is very important that some type of automatic recovery procedure must be implemented in these systems.

Failures in digital systems are generally classified as being solid or transient. The effects of solid faults remain in the system until removed by the repair facility, which is very often a manual operation; the effects of transient failures may disappear from the system. Recovery procedures tend to differ for transient and solid faults, and several techniques have been characterized. Generally, transient failures can be resolved by retry or refresh methods; solid faults usually involve some type of reconfiguration to nullify their effects, and this causes some degree of system-performance degradation. A more comprehensive description of these points can be found in [2].

4.2.3 System reliability modeling

The SIRU system is one example where duplication is used to achieve higher computer system reliabilities. Such dual redundant systems depend upon a scheme for detecting a faulty module and giving control to the properly functioning unit. Various techniques exist to allow system recovery from transient errors. In order to determine the effectiveness of these error detection and recovery schemes, mathematical models of the computer systems are used.

These mathematical techniques are particularly applicable to the analysis of dual redundant computer systems such as SIRU. Such systems have a small number of states, making the mathematical manipulations feasible. The error detection and recovery methods can be modeled to first order by a simple Bernoulli process with probability of success, p. Such a model incorporates a state for the error detection or recovery technique which has probability p of returning to non-degraded system performance and probability 1-p of entering some degraded (or failure) state. A refinement of such a model incorporates:

(1) the probability of successful recovery without further system degradation, i.e., the recovery effectiveness; and

(2) the probability that the system does not immediately enter a

failure state given that a fault occurs, i.e., coverage. As an example of such an analysis, consider the four-state Markov chain with states:

1 -- 2 computers functioning (full performance)

2 -- error recovery / one computer failed

3 -- 1 computer functioning / 1 computer failed (degraded performance)

4 -- both computers failed (system failure).

Analysis of this Markov model can determine values of recovery effectiveness and coverage necessary to achieve desired reliabilities for the computer system [3].

It is also possible to extend the model of the error detection or recovery process to include several states, thus realizing a more accurate analysis of the merits of such a process. This is especially useful when these processes are implemented with arbiters and/or microprocessors.

4.2.4 Signal reliability of a circuit

The fact that faults present in a circuit will not always cause the output of a circuit to be incorrect leads to interest in the probability that the output of a circuit is correct. This measure is called the signal reliability of the circuit output. Two methods for evaluating signal reliability have been devised [4]; the first uses the concept of fault-equivalence classes, and the second employs the probabilistic model for combinational circuits.

The measure has application in fault-latency studies [5]. It is known that faults occurring in a circuit may not immediately manifest themselves as errors on the output of the circuit but that, generally, some time elapses before this manifestation takes place. This interval between the occurrence of a fault and its manifestation as an error on the output is called the "latency" of a fault, and this latency is an important parameter because it provides an indication as to how far to roll back a program when an error occurs. In addition, if the latency of a fault is large, a second fault may occur before the first is detected as an error, and this would violate the often-used single-fault assumption.

The signal-reliability measure is also useful in modeling intermittent faults in a circuit because it measures the instantaneous probability of a correct output of the circuit.

4.2.5 Software reliability

This research has been concerned with methodologies for testing and proving correct parallel software systems. Abstract modeling formalisms such as Petri nets, vector addition systems, Presburger logic and first order logic have been studied as possible representations of component interactions in parallel systems, and their ability to express correctness criteria has been investigated. A simple 3-process single operating system has been modeled as a Petri net, and by means of a reduction procedure several correctness questions concerning activity and history sequences in that system have been answered. In relation to the testing of parallel systems, formalisms for error description, detection and injection of faults into these systems are being investigated.

As a case study for applying the results of this research, the SIRU inertial navigation computer, a dual redundant real time system, has been chosen. A low level description of the system software in a PL360-like language has been obtained, and is presently being used in studying the robustness of the current software, as well as techniques for fault injection and detection in real time operating systems. A high level representation in PASCAL of an operating system for the SIRU dual redundant computer that incorporates the recommendations from the above study will be obtained. Hopefully the techniques obtained in dealing with real time systems might be extended to more complex operating systems.

4.3 References

[1] B. E. Ressler, "Design of a Dual Computer Configuration for Redundant Computation", M.S. Thesis, MIT, Cambridge, Mass., June 1973.

[2] E. Fregni and R. C. Ogus, "Error Recovery Techniques in Computer Systems- A Survey", Tech. Note no. 42, Digital Systems Laboratory, Stanford University, Stanford, California, June 1974.

[3] E. Fregni, D. Beaudry and R. C. Ogus, "A Markov Model for Reconfigurable Computer Systems", Tech. Note no. 43, Digital Systems Laboratory, Stanford University, Stanford, California, August 1974.

[4] R. C. Ogus, "The Probability of a Correct Output from a Combinational Circuit", IEEE Trans. on Computers, Vol. C-24, pp. 534-544, May 1975.

[5] J. J. Shedletsky and E. J. McCluskey, "The Error Latency of a Fault in a Combinational Digital Circuit", Digest of 5th Int'l. Symposium on Fault-Tolerant Computing, Paris, June 1975.

[6] J. F. Wakerly, "Reliability of Microcomputer Systems Using Triple Modular Redundancy," Tech. Note no. 61, Digital Systems Laboratory, Stanford University, Stanford, California, April 1975.

[7] Monolithic Memories, Inc. Data Sheet, "4-Bit Expandable Bipolar Microcontroller - 5701/6701," August 1974.

5. BIOGRAPHICAL SKETCHES OF PRINCIPAL RESEARCHERS IN CENTER FOR RELIABLE COMPUTING (CRC)

Thomas H. Bredt, Assistant Professor of Electrical Engineering (on leave), Ph.D. (CS), Stanford University, 1970, is currently performing research on parallel computer systems including operating systems and asynchronous logic networks. Studies of operating systems include resource allocation design methodologies, system structure, logical correctness, and error detection and recovery. Hazards in asynchronous systems and mathematical models of parallel systems are also being investigated.

Jacques Losq, Research Associate in Electrical Engineering, Ph.D. (EE), Stanford University, 1975, has been working on modelling and analyzing redundant digital systems. His general interest is focused on the determination of the degree and type of redundancy that provide the best trade-off between cost and performance for specific missions. This includes the study of massive, stand-by and mixed redundancy along with careful analysis of the more general survivable systems, in particular multi-micro (or mini) computers.

Edward J. McCluskey, Professor of Computer Science and Electrical Engineering, Sc.D. (EE), Massachusetts Institute of Technology, 1956, is Director of the Digital Systems Laboratory. He is widely known for his contributions to switching theory, and is the author of a standard text on the subject. His current research efforts are directed toward the area of fault-tolerant computing, including testing and diagnosis of digital circuits, redundancy schemes, and reliability modeling and evaluation. He is also involved in investigating multiprocessor systems, particularly those using microprocessors.

Roy C. Ogus, Research Associate in Electrical Engineering, Ph.D. (EE), Stanford University, 1975, has been project leader of the dual computer systems project, which is involved with the evaluation of a dual computer system used in a guidance and navigation application. His current research interests include the areas of computer reliability, fault-tolerant computing, computer architecture and microprocessor systems.

John F. Wakerly, Assistant Professor of Electrical Engineering, Ph.D. (EE), Stanford University, 1974, has been active in the development of the Digital Systems Laboratory's hardware laboratory. His research interests are in the areas of computer hardware reliability and maintainability, computer architecture, and microcomputer systems. His current projects include the design and construction of an ultra-reliable triplicated microcomputer system, a multi-microprocessor network, and a self-diagnosing minicomputer system.

6. LISTING OF PERSONNEL IN CENTER FOR RELIABLE COMPUTING

DIRECTOR:	Edward J. McCluskey
ASSISTANT DIRECTORS:	Thomas H. Bredt
	John F. Wakerly
	Roy C. Ogus
SENIOR RESEARCH STAFF:	Jacques Losq
	Andre Verdillon
STUDENT RESEARCHERS:	R. Betancourt
	D. M. Beaudry
	M. L. Blount
	D. Davies
	M. Hadidi
	A. F. Hunter
	S. G. Kolupaev
	P. J. LeVine
	D. J. Lu
	J. R. McClure
	K. P. Parker
	J. V. Phillips
	J. Savir
	J. J. Shedletsky
	P. A. Thompson
	W. A. Wallach

7. PREVIOUS RESEARCHERS IN CRC

Previous researchers in the Center for Reliable Computing together with their current locations and Ph.D. dissertation titles are listed below.

Thomas H. Bredt - Hewlett-Packard Computer Division, Cupertino, CA.
Ph.D. 1970, "Control of Parallel Processes."

Frederick W. Clegg - Hewlett-Packard Computer Division, Cupertino,CA.
Ph.D. 1970, "Algebraic Properties of Faults in Logic Networks."

Donald J. Chesarek - IBM Systems Development Division, Los Gatos, CA.
Ph.D. 1972, "Fault Detecting Experiments for Sequential Machines."

Donald P. Siewiorek - Carnegie-Mellon University, Departments of Computer Science and Electrical Engineering, Pittsburgh, Pa.
Ph.D. 1972, "Fault-Tolerant Computers Using Self-Diagnosis and Hybrid Redundancy."

Raymond T. Boute - Bell Telephone Manufacturing Co., Antwerp, Belgium.
Ph.D. 1973, "Faults in Sequential Machines: Algebraic Properties and Defection Methods."

Jacob A. Abraham - University of Illinois, Department of Electrical Engineering, Champaign, IL.
Ph.D. 1974, "Reliability Analysis of Digital Systems Protected by Massive Redundancy."

Hajime Mitarai - Canon Industries, Tokyo, Japan.
Ph.D. 1974, "The Use of Semiconductor Read-Only Memory for Logic."

John F. Wakerly - Digital Systems Laboratory, Stanford University, Stanford, CA.
Ph.D. 1974, "Low-Cost Error Detection Techniques for Small Computers."

David T. Wang - IBM Systems Products Division, Endicott, NY.
Ph.D. 1974, "An Algorithm for the Generation of Test Sets for Combinational Logic Networks."

Jacques Losq - Digital Systems Laboratory, Stanford University, Stanford, CA.
Ph.D. 1975, "Modelling and Reliability of Redundant Digital Systems."

Kenyon C. Y. Mei - Hewlett-Packard Computer Division, Cupertino, CA.
Ph.D. 1975, "Dominance Relations of Stuck-at and Bridging Faults in Logic Networks."

Roy C. Ogus - Digital Systems Laboratory, Stanford University, Stanford, CA.
Ph.D. 1975, "Design and Evaluation of Ultra-Reliable Hybrid Redundant Digital Systems."

Francisco J. O. Dias - University of Sao Paulo, Sao Paulo, Brazil.
Ph.D. 1975, "Multiple Fault Analysis in Combinational Logic Circuits."

In addition, the following scholars have visited the CRC and performed research in the listed areas.

Pawel N. Kentopf - Polish Academy of Sciences, Warsaw, Poland.
Design of universal logic modules.

Yoshihiro Tohma - Tokyo Institute of Technology, Tokyo, Japan.
Redundancy techniques for constructing ultra-reliable systems.

Sándor Várszegi - Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, Hungary.
Testing of digital networks.

Andre Verdillon - University of Grenoble, Grenoble, France.
Fault properties.

8. CENTER FOR RELIABLE COMPUTING BIBLIOGRAPHY

8.1 JOURNAL PAPERS

Prepared Under Previous NSF Grants

†**Abraham**, J. A. and Siewiorek, D. P., "An Algorithm for the Accurate Reliability Evaluation of Triple Modular Redundancy Networks," IEEE Transactions on Computers, Vol. C-23, No. 7, July, 1974, pp. 682-693.

*Abraham, J. A., "A Combinatorial Solution to the Reliability of Interwoven Redundant Logic Networks," IEEE Transactions on Computers, Vol. C-24, No. 5, May, 1975, pp.578-584.

***Betancourt**, R., "Derivation of Minimum Test Sets for Unate Logical Circuits," IEEE Transactions on Computers, Vol. C-20, No. 11, November, 1973, pp. 1264-1269.

†**Boute**, R. T., "Distinguishing Sets for Optimal State Identification in Checking Experiments," IEEE Transactions on Computers, Vol. C-23, No. 8, August, 1974, pp. 874-878.

†**Boute**, R. T., "Optimal and Near-Optimal Checking Experiments for Output Faults in Sequential Machines," IEEE Transactions on Computers, Vol. C-23, No. 11, November 1974, pp. 1207-1213.

Clegg, F. W., "Use of SPOOF's in the Analysis of Faulty Logic Networks," IEEE Trans. on Computers, Vol. C-22, No. 3, March, 1973, pp. 229-234

‡**Dias**, F. J. O., "Fault Masking in Combinational Logic Circuits," IEEE Transactions on Computers, Vol. C-24, No. 5, May, 1975, pp. 476-482.

*McCluskey, E. J., "Test and Diagnosis Procedures for Digital Networks," Computer, Vol. 4, No. 1, January/February, 1971, pp. 17-20.

*McCluskey, E. J. and F. W. Clegg, "Fault Equivalence in Combinational Logic Networks," IEEE Transactions on Computers, Vol. C-20, No. 11, November 1971, pp. 1286-1293.

†**Mei, K.** C. Y., "Bridging and Stuck-At Faults," IEEE Transactions on Computers, Vol. C-23, No. 7, July, 1974, pp. 720-727.

†**Ogus**, R. C., "Fault-tolerance of the Iterative Cell Array Switch for Hybrid Redundancy," IEEE Transactions on Computers, Vol. C-23, No. 7, **July**, 1974, pp. 667-682.

- - - - - -

* NSF grant GJ 165

† NSF grant GJ 27527

‡ NSF grant GJ 40286

ψ**Ogus**, R. C., "The Probability of a Correct Output from a Combinational Circuit, IEEE Transact-ions on Computers, Vol. C-24, No. 5 May, 1975. pp. 534-544.

ψ**Parker**, K. P. and E. J. McCluskey, "Analysis **of** Logic Circuits with Faults Using Input Signal Probabilities," IEEE Transactions on Computers, Vol. C-24, No. 5, May, 1975, pp. 573-578.

ψ**Parker**, K. P. and E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks," IEEE Transactions on Computers, Vol. C-24, No. 6, June, 1975, pp. 668-670.

†**Siewiorek,** D. P. and E. J. McCluskey, 'An Iterative Cell Switch Design for Hybrid Redundancy," IEEE Transactions on Computers, Vol. C-22, No. 3, March, 1973, pp. 290-298.

†**Siewiorek, D.** P. and E. J. McCluskey, "Switch Complexity in Systems with Hybrid Redundancy,' IEEE Trans. on Computers, Vol. C-22, No. 3, March, 1973, pp. 276-282.

†**Siewiorek,** D. P., "Reliability Modeling of Compensating Module Failures in Majority Voted Redundancy," IEEE Transactions on Computers, Vol. C-24, No. 5, May 1975, pp. 525-533.

ψ**Usas,** A. M., "A Totally Self-Checking Checker Design for the Detection of Errors in Periodic Signals," IEEE Transactions on Computers, Vol. C-24, No. 5, May, 1975, pp. 483-489.

†**Wakerly,** J. F., "Partially Self-Checking Circuits and Their Use in Performing Logical Operations,' IEEE Transactions on Computers, Vol. C-23, No. 7, July, 1974, pp. 658-667.

†**Wakerly,** J. F., "Detection of Unidirectional Multiple Errors Using **Low-**Cost Arithmetic Code," IEEE Transactions on Computers, Vol. C-24, No. 2, February, 1975, pp. 210-212.

⊗**Wakerly,** J. F., "Transient Failures in Triple Modular Redundancy Systems with Sequential Modules," IEEE Transactions on Computers, Vol. C-24, No. 5, May, 1975, pp. 570-573.

†**Wang,** D. T., "An Algorithm for the Generation of Test Sets for Combinational Networks," IEEE Transactions on Computers, Vol. C-24, No. 7, **July,** 1975, pp. 742-746.

†**Wang,** D. T., "Properties of Faults and Criticalities of Values **under** Tests for Combinational Networks," IEEE Transactions on Computers, Vol. C-24, No. 7, July, 1975, pp. 746-750.

- - - - - - -

† NSF grant GJ 27527

ψ NSF grant GJ 40286

⊗ NSF grant GJ 43322

8.2 CONFERENCE PAPERS

Prepared Under Previous NSF Grants

†**Abraham**, J. A., "An Algorithm for the Accurate Reliability Evaluation of TMR Networks," Digest of 1973 International Symposium on Fault-Tolerant Computing, Palo Alto, California, June 20-22, 1973, **pp.** 119-125.

†**Boute**, R. and E. J. McCluskey, "Fault Equivalence in Sequential Machines," Symposium on Computers and Automata, Polytechnic Institute of Brooklyn, April 13-15, 1971, pp. 483-507.

†**Boute**, R. T., "Algebraic Properties of Testing and Diagnosing Sequences," Digest of 1975 International Symposium on Fault-Tolerant Computing, Paris, France, June 18-20, 1975, p. 242.

***Clegg**, F. W., "Use of SPOOFs for Faulty Logic Network Analysis," Digest of 1972 International Symposium on Fault-Tolerant Computing, Newton Massachusetts, June 19-21, 1972, pp. 143-148.

***Clegg**, F. W. and E. J. McCluskey, "The Algebraic Approach to Faulty Logic Networks," Digest of 1971 International Symposium on Fault-Tolerant Computing, Pasadena, California, March 1-3, 1971, pp. 44-46.

ψ**Dias**, F. J. O., "Fault Masking in Combinational Logic Circuits," Digest of 1974 International Symposium on Fault-Tolerant Computing, Urbana, Illinois, June 19-21, 1974, pp. 1.23-1.30.

ψ**Losq, J.**, "Influence of Fault-Detection and Switching Mechanisms on the Reliability of Stand-By Systems," Digest of 1975 International Symposium on Fault-Tolerant Computing, Paris, France, June 18-20, 1975, pp. 81-86.

j-McCluskey, E. J., "Probability Models for Logic Networks," **Proc.** of the Fourth Manitoba Conference on Numerical Mathematics, Winnipeg, Canada, October 2-5, 1974, pp. **21-28.**

@McCluskey, E. J., "**Micros**, Minis and Networks," Proceedings of the Meeting on Twenty Years of Computer Science, Pisa, Italy, June 16-19, 1975, pp. 23-33.

†**Mei**, K. C. Y., "Bridging and Stuck-At Faults," Digest of 1973 International Symposium on Fault-Tolerant Computing, Palo Alto, California, June 20-22, 1973, pp. 91-95.

- - - - - -

* NSF grant GJ 165

† NSF grant GJ 27527

ψ NSF grant GJ 40286

†Mitarai, H., "Design of a Parallel Encoder/Decoder for the Hamming Code, Using ROM," First USA-Japan Computer Conference, Tokyo, Japan, October, 1972.

†Mitarai, H., "ROM Micro-Reduction Techniques," Second USA-Japan Computer Conference, Tokyo, Japan, August 1975.

ψOgus, R. C., "The Probability of a Correct Output from a Combinational Circuit," Digest of 1974 International Symposium on Fault-Tolerant Computing, Urbana, Illinois, June 19-21, 1974, pp. 1.13-1.20.

†Ogus, R. C., "Fault-Tolerance of the Iterative Cell Array Switch for Hybrid Redundancy," Digest of 1973 International Symposium on Fault-Tolerant Computing, Palo Alto, California, June 20-22, 1973, pp. 107-113.

†Ogus, Roy c., and J. F. Wakerly, "Fault-Tolerant Design of Minicomputers," Proc. Symp. on Minicomputers, South African Council for Automation and Computation, Pretoria, South Africa (September 1973).

ψParker, K. P. and E. J. McCluskey, "Analysis of Logic Circuits with Faults Using Input Signal Probabilities," Digest of 1974 International Symposium on Fault-Tolerant Computing, Urbana, Illinois, June 19-21, 1974, pp. 1.8-1.13.

†Reese, R. D. and E. J. McCluskey, "A Gate Equivalent Model for Combinational Logic Network Analysis," Digest of 1973 International Symposium on Fault-Tolerant Computing, Palo Alto, California, June 20-22, 1973, pp. 79-85.

ψShedletsky, J. J. and E. J. McCluskey, "The Error Latency of a Fault in a Combinational Digital Circuit," Digest of 1975 International Symposium on Fault-Tolerant Computing, Paris, France, June 18-20, 1975, pp. 210-214.

†Siewiorek, D. P. and E. J. McCluskey, "An Iterative Cell Switch Design for Hybrid Redundancy," Digest of 1972 International Symposium on Fault-Tolerant Computing, Newton, Massachusetts, June 19-21, 1972, pp. 182-189.

†Siewiorek, D. P., "Reliability Modeling of Compensating Module Failures in Majority Voted Redundancy," Digest of 1974 International Symposium on Fault-Tolerant Computing, Urbana, Illiois, June 19-21, 1974, pp. 2.14-2.20.

⊗Usas, A. M., "Fail-Safe Circuits: A Means to Improve Reliability and Maintainability of I/O Subsystems," Digest of Fall, 1975 CompCon Conference, Washington, D.C., September 9-11, 1975,

- - - - - -

† NSF grant GJ 27527

ψ NSF grant GJ 40286

⊗ NSF grant GJ 43322

ψUsas, A. M. and E. J. McCluskey, "Design and Application of a Self-Checking Periodic Signal Checker," *Digest of Fall, 1974 CompCon Conference*, Washington, D.C., September 10-12, 1974, pp. 83-91.

ψWakerly, J. F. and E. J. McCluskey', "Design of Low-Cost General-Purpose Self-Diagnosing Computers," *Proceedings of the IFIP Congress 1974*, Stockholm, Sweden, 1974, pp. 108-111.

†Wakerly, J. F., "Partially Self-Checking Circuits and Their Use in Performing Logical Operations," *Digest of 1973 International Symposium on Fault-Tolerant Computing*, Palo Alto, California, June 20-22, 1973, pp. 65-73.

- - - - - -

ψ NSF grant GJ 40286

† NSF grant GJ 27527

8.3 TECHNICAL PAPERS

Prepared under Previous NSF Grants

[TR 4] ***Clegg,** F. W. and E. J. McCluskey, "Algebraic Properties of Faults in Logic Networks," Tech. Rpt. no. 4, **SU-SEL-69-078**, March 1970. 140 pages.

[TR 11] *Clegg, F. W., "The SPOOF: A New Technique for Analyzing the Effects of Faults on Logic Networks," Tech. Rpt. no. 11, SU-SEL-70-073, August 1970. 40 pages.

[TR 15] **†Boute,** R. and E. J. McCluskey, "Fault Equivalence in Sequential Machines," Tech. Rpt. no. 15, SU-SEL-71-038, June 1971. 48 pages.

[TR 20] **†Siewiorek,** D. P. and E. J. McCluskey, "An Iterative Cell Switch Design for Hybrid Redundancy," Tech. Rpt. no. 20, SU-SEL-71-064, December 1971. 60 pages.

[TR 21] **†Siewiorek,** D. P. and E. J. McCluskey, "A Measure of Switch Complexity in Systems with Standby Spares," Tech. Rpt. no. 21, SU-SEL-71-065, December 1971. 51 pages.

[TR 22] **†Siewiorek,** D. P., "Models of Self-Diagnosable Systems," Tech. Rpt. no. 22, SU-SEL-71-066, December 1971. 38 pages.

[TR 23] **†Siewiorek,** D. P., "An Improved Algorithm for Selecting a Set of Diagnostic Tests," Tech. Rpt. no. 23, SU-SEL-71-067, December 1971. 17 pages.

[TR 24] **†Siewiorek,** D. P., "An Improved Reliability Model for NMR," Tech. Rpt. no. 24, **SU-SEL-72-004,** December 1971. 35 pages.

[TR 30] **†Boute,** R. T., "Adaptive Design Methods for Checking Sequences," Tech. Rpt. no. 30, SU-SEL-72-034, July 1972. 31 pages.

[TR 35] **†Kolupaev,** S. G., "Separate Non-Homomorphic Checking Codes for Binary Addition," Tech. Rpt. no. 35, SU-SEL-72-033, July 1972. 24 pages.

[TR 36] **†Mitarai,** H. and E. J. McCluskey, "Design of a Parallel Encoder/Decoder for the Hamming Code Using ROM," Tech. Rpt. no. 36, SU-SEL-72-030, June 1972. 25 pages.

- - - - - -

* NSF grant GJ 165

† NSF grant GJ 27527

[TR 37] †Boute, R. T., "Algebraic Properties of Test Sequences and Fault Relations," Tech. Rpt. no. 37, SU-SEL-72-051, November 1972. 45 pages.

[TR 38] †Boute, R. T., "Equivalence and Dominance Relations between Output Faults in Sequential Machines," Tech. Rpt. no. 38, SU-SEL-72-052, November 1972. 43 pages.

[TR 39] †Boute, R. T., "Properties of Memory Faults in Sequential Machines," Tech. Rpt. no. 39, SU-SEL-72-053, November 1972. 42 pages.

[TR 403 †Boute, R. T., "Checking Experiments for Output Faults," Tech. Rpt. no. 40, SU-SEL-72-054, November 1972. 45 pages.

[TR 41] †Boute, R. T., "Fault Detection in Fundamental-Mode Circuits," Tech. Rpt. no. 41, SU-SEL-72-055, November 1972. 76 pages.

[TR 49] †Kolupaev, S. G., "Self-Testing Residue Trees," Tech. Rpt. no. 49, SU-SEL-73-030, August 1973. 47 pages.

[TR 50] †Wakerly, J. F., "Partially Self-Checking Circuits and Their Use in Performing Logical Operations," Tech. Rpt. no. 50, SU-SEL-73-039, August 1973. 41 pages.

[TR 51] †Wakerly, J. F., "Low-Cost Error Detection for Small Computers," Tech. Rpt. no. 51, SU-SEL-74-007, December 1973. 221 pages.

[TR 55]†Ogus, R. c., "Fault-Tolerance of the Iterative Cell Array Switch for Hybrid Redundancy through the Use of Failsafe Logic," Tech. Rpt. no. 55, SU-SEL-73-031, August 1973. 66 pages.

[TR 56] †Abraham, J. A. and D. P. Siewiorek, "Reliability Modeling of NMR Networks," Tech. Rpt. no. 56, June 1974. 71 pages.

[TR 58]ǂLosq, J., "Modeling and Reliability of Redundant Digital Systems," Tech. Rpt. no. 58, February 1975. 130 pages.

[TR 62]ǂLosq, J., "A Highly Efficient Redundancy Scheme: Self-Purging Redundancy," Tech. Rpt. no. 62, July 1975. 34 pages.

[TR 65] †Ogus, R. C., "Reliability Analysis of Hybrid Redundant Systems with Nonperfect Switches," Tech. Rpt. no. 65, SU-SEL-74-052, November 1974. 101 pages.

[TR 75] ǂLosq, J., "Influence of Fault-Detection and Switching Mechanisms on the Reliability of Stand-by Systems," Tech. Rpt. no. 75, July 1975. 35 pages.

- - - - - -

† NSF grant GJ 27527

ǂ NSF grant GJ 40286

[TR 93] ψ**Parker,** K. and E. J. McCluskey, "Sequential Circuit Output Probabilities from Regular Expressions," Tech. Rpt. no. 93, SU-SEL-75-023, June 1975. 31 pages.

[TR 94] ψ**Dias,** F. O., "Truth-Table Verification of an Iterative Logic Array," Tech. Rpt. no. 94, SU-SEL-75-024, June 1975. 31 pages.

- - - - - -

ψ NSF grant GJ 40286

8.4 TECHNICAL NOTES

Prepared under Previous NSF Grants

[TN 2] ***Mei,** K. C. Y., "Fault Dominance in Combinational Circuits," Tech. Note no. 2, August 1970. 24 pages.

[TN 31 *Betancourt, R., 'Derivation of Minimum Test Sets for Unate Logical Circuits," Tech. Note no. 3, August 1970. 24 pages.

[TN 51 *Siewiorek, D. P., "On Rapid Calculating Techniques for the Reliability of Serial Triple-Modular Redundancy," Tech. Note no. 5, October 1970. 14 pages.

[TN 8] *Siewiorek, D. P., "A Re-evaluation of the Classical Model for NMR Reliability, Tech. Note no. 8, March 1971. 13 pages.

[TN 9] **†Boute,** R., "Algorithms for Combinational Fault Equivalence Using LISP," Tech. Note no.9, September 1971. 26 pages.

[TN lo] *McCluskey, E. J. and F. W. Clegg, "Fault Equivalence in Combinational Logic Networks," Tech. Note no. 10, March 1971. 34 pages.

[TN 13] **†Siewiorek,** D. P. and E. J. McCluskey, 'Switch Designs for Hybrid Redundancy," Tech. Note no. 13, December 1971. 22 pages.

[TN 14] **†Siewiorek,** D. P., "A Unifying Perspective of Fault Tolerant Computer Techniques," Tech. Note no. 14, December 1971. 26 pages.

[TN 18] **†Parker,** K., "Probabilistic Test Generation," Tech. Note no. 18, January 1973. 13 pages.

[TN 19] **†Kolupaev,** S. G.,. "Self-Testing Modulo 3 Residue Tree," Tech. Note no. 19, July 1972. 12 pages.

[TN 20] **ᛉParker,** K. P. and E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks," Tech. Note no. 20, November 1973. 7 pages.

[TN 21] **ᛉParker,** K. P. and E. J. McCluskey, "Analysis of Logic Circuits with Faults Using Input Signal Probabilities," Tech. Note no. 21, January 1974. 19 pages.

- - - - - -

* NSF grant GJ 165

† NSF grant GJ 27527

ᛉ NSF grant GJ 40286

[TN 26] †**Wakerly**, J. F., "Detection of Unidirectional Multiple Errors Using Low-Cost Arithmetic Codes," Tech. Note no. 26, May 1973. 10 pages.

[TN 28] †**Reese**, R. D. and E. J. McCluskey, "A Gate Equivalent Model for Combinational Logic Network Analysis," Tech. Note no. 28, January 1973. 21 pages.

[TN 31] ☨**Dias**, F. J. O., "Fault Masking in Combinational Logic Circuits," Tech. Note no. 31, January 1974. 17 pages.

[TN 32] *Abraham, J. A., "A Combinatorial Solution to the Reliability of Interwoven Redundant Logic Networks," Tech. Note no. 32, January 1974. 22 pages.

[TN 33]☨**Losq**, J., "Redundancy Scheme for Optimum Multiple Fault Tolerance," Tech. Note no. 33, January 1974. 27 pages.

[TN 34]☨**Losq**, J., "Computer Networks with Constant Maximum Delay Under Communication Line Failures," Tech. Note no. 34, January 1974. 33 pages.

[TN 35] ☨**Ogus**, R. C., "The Probability of a Correct Output from a Combinational Circuit," Tech. Note no. 35, January 1974. 34 pages.

[TN 38] ☨**Wakerly**, J. F. and E. J. McCluskey, "Design of Low-Cost General-Purpose Self-Diagnosing Computers," Tech. Note no. 38, January 1974. 18 pages.

[TN 39] ☨**Wakerly**, J. F.,"Checked Binary Addition Using Parity Prediction and Checksum Codes," Tech. Note no. 39, January 1974. 16 pages.

[TN 45] ☨**Usas**, A. M., "The Detection of Errors in Periodic Signals," Tech. Note no. 45, April 1974. 33 pages.

[TN 55] ☨**Shedletsky**, J. J. and E. J. McCluskey, "The Error Latency of a Fault in a Combinational Digital Circuit," Tech. Note no. '55, November 1974. 21 pages.

[TN 56] ☨**Shedletsky**, J. J. and E. J. McCluskey, "The Error Latency of a Fault in a Sequential Digital Circuit," Tech. Note no. 56, December 1974. 23 pages.

[TN 58] *McCluskey, E. J., "Micros, Minis and Networks," Tech. Note no. 58, June 1975. 10 pages.

- - - - - -

† NSF grant GJ 27527

☨ NSF grant GJ 40286

[TN 59] Usas, A. M., "Fail-Safe Circuits: A Means to Improve the Reliability and Maintainability of I/O Subsystems," Tech. Note no. 59, June, 1975. 4 pages.

[TN 60] *McCluskey, E. J., K. P. Parker and J. J. Shedletsky, "Boolean Network Probabilities and Network Design," Tech. Note no. 60, July 1975. 5 pages.

- - - - - -

* NSF grant GJ 40286