Final Report – Phase I

TECHNIQUES FOR THE REALIZATION OF ULTRA-RELIABLE SPACEBORNE COMPUTERS

By: J. GOLDBERG K. N. LEVITT R. A. SHORT

Prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
ELECTRONICS RESEARCH CENTER
575 TECHNOLOGY SQUARE
CAMBRIDGE, MASSACHUSETTS 02139

CONTRACT NAS 12-33

STANFORD RESEARCH INSTITUTE

MENLO PARK, CALIFORNIA

STANFORD RESEARCH INSTITUTE

MENLO PARK, CALIFORNIA

September 1966

Final Report – Phase I

TECHNIQUES FOR THE REALIZATION OF ULTRA-RELIABLE SPACEBORNE COMPUTERS

By: J. GOLDBERG K. N. LEVITT R. A. SHORT

Prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
ELECTRONICS RESEARCH CENTER
575 TECHNOLOGY SQUARE
CAMBRIDGE, MASSACHUSETTS 02139

CONTRACT NAS 12-33

SRI Project 5580

Approved: D. R. BROWN, MANAGER
COMPUTER TECHNIQUES LABORATORY

J. D. NOE, EXECUTIVE DIRECTOR
ENGINEERING SCIENCES AND INDUSTRIAL DEVELOPMENT

Copy No. 30

ABSTRACT

This is a report of a study of techniques for the realization of ultrareliable, high-performance, spaceborne computers. The study included the evaluation of, and several new contributions to, the most significant known techniques and the proposal and investigation of several promising new techniques. The state of the art of existing redundancy techniques for fault-detecting and fault-masking is assessed, with special emphasis on multiple-line voting redundancy, error-correcting codes, and redundant-state schemes for sequential networks. A number of directions for the improvement of these techniques are described. Significant potential improvements in reliability are available in designs allowing for a high degree of reconfigurability in structure and programs, and system schemes and design techniques needed for such behavior are proposed and investigated. In particular, we discuss the design of minimal test schedules for fault detection and diagnosis, the design of highly modular processing networks and of programmable interconnection networks, and the overall organization of maintenance and computation functions in a computer system. The application of error control techniques to memory systems and to power supplies is considered, and the possible use of all-magnetic logic networks is examined. Included in the report is a critical and selective survey of the literature that is relevant to the attainment of reliable systems and networks through the judicious use of redundant structures. Finally, recommendations are made for further research into the development of techniques for ultrareliable system design.

FOREWORD

This is a report of a one-year research study of techniques for the realization of ultrareliable spaceborne computers. This study was conducted in the Computer Techniques Laboratory of Stanford Research Institute, under the sponsorship of the Electronics Research Center of the National Aeronautics and Space Administration.

The major objective of the study was to provide guidelines for the design of computers intended to function reliably under the severe conditions imposed by spaceborne missions. It is clear that the spaceborne requirement introduces design difficulties which are not attendant to other applications. For example, the possibility of unprogrammed maintenance and inspection routines is severely limited; the successful use of a radio link cannot always be assured; the computations are complex and highly varied; the performance requirements are very high; and there are special physical constraints on the construction and operation of the computer. In order to attain an acceptably reliable system the judicious use of redundant structures is mandatory. Of course, the observation that redundancy is required to improve reliability is not unique to this study, and hence in the course of our investigation we utilized many well-established (at least in principle) techniques, e.g., fault masking by multiple-line voting and adaptive replacement of faulty subsystems with standby units. In order to properly assess the myriad of proposed redundancy measures, a considerable portion of the report is devoted to commentaries on state-of-the-art developments. This inclusion of review material enables an engineer who is related to this area solely as a user to satisfy his requirements with minimal recourse to other documents.

However, many novel developments are reported herein, and our concern in this Foreword is to guide the reader--whether he is a research specialist or one with little prior knowledge of the subject matter--to those sections which are of most interest to him.†

The report is organized into four chapters and four appendices. Each major section of the chapters contains conclusions and a detailed listing of outstanding research problems; the major conclusions of the research study and recommendations for further research are presented in Chapter IV. The first chapter serves as an overall introduction to the report. It contains (1) the statement of the problem--in particular a detailed discussion of the characteristics of advanced spaceborne computers; (2) the goals, methods and assumptions of the study; (3) the criteria of performance, including a discussion of relevant reliability and cost measures; and (4) the organization of the report.

The second chapter is concerned with those logical design techniques for fault masking and error detection in which the error control is passive. In Sec. II-A-1* we present a detailed (historical) review of the most important known fault-masking techniques. Specific combinational fault-masking design techniques are given in Sec. II-A-2. Included herein are reviews of the multiple-line voting approaches [Sec. II-A-2-a-2)]* as applied to such simple network models as cascades and trees, and also some embellishments of known techniques [Sec. II-A-2-a-3)] for the analysis of arbitrary replicated networks. Some new results are presented on bounds on the reliability of arbitrary replicated networks [Sec. II-A-2-a-4)] and also on optimum techniques for the realization of multiple-output networks which are replicated [Sec. II-A-2-a-5)]. In Sec. II-A-2-b we present some unique digital realizations of the extremely powerful adaptive-voting scheme, along with a detailed examination of schemes which combine fault masking and network replacement. Such schemes have heretofore not been reported in the literature. In Sec. II-A-2-c* a review is

† Sections containing reviews of prior techniques are marked, in this Foreword, by an asterisk.

presented of the known techniques for the realization of voting networks--much of the prior work has related to networks which realize the majority function of 3 or 5 inputs--and also some novel designs (which are approximately minimal) are given for majority-function networks with an arbitrary number of inputs, suitable for different technologies. A review of the known techniques of fault control for sequential networks is presented in Sec. II-A-3* along with some (apparently) novel fault-detection schemes relying upon state-parity checking and state-weight checking. Chapter II concludes with Sec. II-B,* which surveys in detail the known coding techniques which appear to be appropriate for checking computer operations.

Chapter III is concerned with techniques for dynamic error control, i.e., ways in which the logical interconnections among the components of the computer may be altered. In Sec. III-A we discuss the particular system organization features that facilitate dynamic maintenance processes. Section III-B is concerned with the design of test schedules of minimal length, for the fault diagnosis of combinational networks. Included here is a review of the state of the art of diagnosis (Sec. III-B-3)* along with a discussion of some novel techniques for fixed-schedule and serial-schedule types of tests based upon reduction of a fault table. In Sec. III-C we consider the design of networks for a reconfigurable network--an area which has received little prior attention. The design of commutation networks--networks whose function is to provide interconnection between operating modules and to disconnect faulty modules for the system--is discussed in Sec. III-C-2. Two types of structures are presented--a unique sequential network and a combinational type of network which is somewhat suggestive of the central telephone exchange. In Sec. III-C-3 we consider the design of a modular arithmetic processor wherein the functions of computation, storage, and primitive control are all combined in an iterated set of replaceable modules. The chapter concludes with brief descriptions of some novel techniques for realizing programmable control units. This remains a major area for further research.

Appendix A* emphasizes the practical problems of applying redundancy to spaceborne memories. Included here is an evaluation and comparison of several state-of-the-art schemes, e.g., the use of codes to protect the

data channels and the access circuits, in addition to some suggestions for future research such as a consideration of those reliability techniques which relate to special memory types.

Appendix B is a detailed discussion of the reliability problems peculiar to power supplies. Suggestions are given for novel means of error control, including considerations of weight and volume.

Appendix C is an examination of the possible role of magnetic logic for attaining ultrareliable operation, with special attention to applications wherein the low speed of operation attendant to magnetic logic does not limit overall computation speed.

Appendix D* is a critical and selective survey of the literature that is relevant to the attainment of reliable systems and networks through the judicious use of redundant structures. Although several complete bibliographies of the literature have appeared previously, no surveys were available which could be used to quickly distinguish those contributions which are concerned with tactical expositions, applications, or advanced mathematical theories.

The technical studies reported here are the work of the following members of the Computer Techniques Laboratory:

Mr. J. A. Baer
Mr. C. B. Clark
Dr. B. Elspas
Mr. J. Goldberg
Dr. W. H. Kautz
Dr. K. N. Levitt
Mr. S. W. Miller
Dr. R. A. Short
Dr. H. S. Stone.

All of these individuals contributed to the writing of the various sections of the report. The report was organized and edited by Mr. J. Goldberg, who was Project Leader, Dr. K. N. Levitt, and Dr. R. A. Short.

CONTENTS

ILLUSTRATIONS

ILLUSTRATIONS (Continued)

TABLES

I OBJECTIVES AND APPROACH

In this chapter we shall discuss the problem of realizing ultra-reliable spaceborne computers and explain the method of approach taken in the study of the problem. We first discuss the basic characteristic of an advanced spaceborne computer and the problems of design that result from the novel operational and technological requirements involved. We then discuss the particular goal of the study, its technical scope, and the criteria employed.

A. Statement of the Problem

In this part we consider the basic characteristic of a future advanced spaceborne computer, and the problems of design that arise from the requirements of performance, the constraints on construction and operation, and the unreliability of components and assembly.

1. Basic Characteristics of an Advanced Spaceborne Computer

a. Special Requirements and Constraints in Computation, Maintenance, and Construction

The range of computational problems and the rates and capacities of computation that will be required in the coming generation of space computers are currently under study by NASA.*[114] Some qualitative statements may be made at this time:

First, the computations will be complex and highly varied.[49] They may be expected to include the checkout, monitoring and control of other spacecraft subsystems; guidance; navigation; maneuvering; the control of communications and of experiments; the processing of data from experiments and photographs; and display. If the scope of the missions is extended to operations on the surface of a moon or planet, the list may be

---

* References are listed alphabetically at the end of this report.

extended to include control of complex stationary or non-stationary mechanisms. The computations may thus be expected to be of the general scientific type, possibly including heuristics.

Second, the performance requirements will be very high. For the foregoing tasks, large memory capacity would be required for storage of constants and intermediate variables and for storage of the many complex programs required, both for the objective computations and for executive and maintenance computations. The input signals for such applications would range widely in form and rate and might be synchronized with the computer's cycle of operation. Thus the computer must be capable of sustaining a number of active inputs at once, and it must be interruptible by external command.

Certain tasks, such as guidance in the vicinity of a planet, require very high computation rates.

Some of the computations, such as pattern processing and coordinate transformations, have functions that may be evaluated with a high degree of parallelism; hence not only is general high-speed arithmetic needed, but some kind of bulk-parallel processing may also be very useful.

Third, the computations will have a range of priorities. In complex missions of long duration it is natural to include as many activities as are permitted by constraints of weight and power. The activities will no doubt have widely differing values to the overall mission, and there will no doubt be a complex set of interdependencies among the activities. Also, some computations will have a range of acceptable precision, with a corresponding range of values to the mission.

Fourth, there will be special physical constraints on the construction and operation of the computer. In construction, there will be severe limitations on weight, volume and allowable power dissipation. Also, it is likely that physical accessibility to components will be very restricted. In the operation of the computer, there may be occasional interruptions in power, either planned or unplanned. Restarting and recovery of a computation after such interruptions must, of course, be automatic.

The most important physical constraint is that the components available for construction are not perfectly reliable. In fact, for the number of components needed and for the length of time of operation of the missions of interest, not only is the probability of error-free operation unacceptably low, but it is extremely expensive in time and equipment to test a computer* so as to estimate its reliability accurately.

Fifth, the amount of available human intervention will be very limited. The computers of interest to NASA for the present study include those for manned and for unmanned missions. The main functions that will be affected by the presence of a man are the executive control of the various phases of computation, the checking and repairing of the computer, and the peripheral functions of input and output. Even with a man present, there will be some limitations on control due to restrictions on time, accessibility, or technical knowledge. Some radio communication with a manned maintenance facility should be possible in many missions.

b. Design Consequences of the Special Requirements and Constraints

The requirements and constraints described have special significance in the design of a computer system. The complexity and variety of computations require that the computer be general-purpose programmable. The high performance requirements call for large memory capacity, high processing speed, and input-output facilities with elaborate signal-processing and control features. Thus a large number of components will be needed for logic and for storage functions. The variability in value among the computations requires that the computer be capable of altering the scheduling of tasks to match the available performance capability, in the event that failures in equipment reduce that capability below its nominal value.†

---

* We emphasize the computer as a whole, since the reliability of the assembly of components is as significant a factor in the reliability of modern systems as is the reliability of the components themselves.

† Such behavior is known colloquially as "graceful degradation" and "failing soft."

The limitation on human intervention for executive control requires that there be some degree of built-in capability for the basic executive functions, as follows:

- For organizing the equipment and the programs so as to achieve the highest possible level of service on the tasks of a mission, according to the value of the tasks.
- For organizing the equipment and the programs to avoid errors, by avoiding the use of faulty equipment or by performing computations redundantly.
- For detecting errors and correcting them by recomputation.

In addition to this built-in capability, any radio communication that is available should be exploited to its limit, because the reliability and depth of analysis in a manned facility will be superior to that available in an on-board program. However, for deep-space missions, the data rate and delay time and the reliability of such communications may well be inadequate to ensure the reliability of some real-time computations.

The special physical constraints on construction and operation have a number of implications for the logical organization of the computer. In modern semiconductor technology the weight and volume of a computer is predominantly that of the packaging and the interconnections. These are strongly influenced by factors of logical organization, such as the degree of parallelism of logical operations, the size of the basic packages, and the way in which logic functions are divided among the packages. The limitations on power dissipation are significant in several ways. First, there is a complex interaction between system speed and power dissipation; i.e., use of slow circuits reduces the power cost per circuit, but it requires use of more circuits, operating with a higher degree of parallelism, in order to achieve a given computation speed. Second, in a redundant

system, it would be advantageous to be able to remove power from inactive components, both to reduce power consumption and to increase the life of components.*

Finally, the fact that it is impossible to ensure perfect operation with adequate confidence for large computers over the period of time appropriate to sustained space missions, requires that the computer have the capability of accommodating failures among its component parts. Since modern component parts do not have the capability for physical self-repair, some form of redundancy of components is clearly required.

To summarize, an advanced spaceborne computer for future deep-space missions must be general-purpose programmable; it must have high memory capacity, high computation speeds and complex input-output facilities; it must have some combination of remote and local control of error-accommodation processes; and it must employ some form of logical redundancy in its construction.

2. Problems of Design for Reliability

In the foregoing part, it was concluded that it will be necessary to employ logical redundancy in high-performance, long-duration spaceborne computers. A number of redundancy schemes have been described and employed in practice, but a generally-accepted design art for such redundancy does not exist. Recent advances in device technology are making it possible to apply redundancy with much greater effectiveness than in the past; in particular, microelectronic fabrication has lowered the weight, size, and cost of logic elements, permitting the use of high orders of redundancy; and continuing refinements in production have increased the inherent reliability of components, thus making a given order of redundancy more effective in extending component life.

---

* Knowledge as to the effect of removing power from modern digital components on their life is not well substantiated. Various authorities estimate an increase in the mean time to failure of from 50% to 300%.

The components and assemblies produced by employing micro-electronic fabrication have cost and reliability factors that differ from those of previous fabrications. Several examples may be given.

First, recent reliability reports[345] indicate that failures in interconnections are as significant as failures in active elements. Among various kinds of connections, those within a monolithic circuit may be substantially more reliable than those between the circuit and the external connection system, although as connections within arrays are made more complex (e.g., by using two or more layers of connections) this may not be so.

A second new factor is standardization. If the logic networks of a conventionally organized computer are simply partitioned and realized as monolithic arrays, a great number of different kinds of arrays will result--perhaps as many as there are arrays. The number of different array types influences the initial reliability of the arrays and the techniques of diagnosis and replacement in service, as may be seen by the following considerations.

Using many array types tends to reduce initial reliability, since it is generally accepted that the reliability of a product increases with the accumulated experience in producing and using it. Using many array types requires that many different sets of diagnostic tests be kept available within the computer memory. Finally, using many array types reduces the effectiveness of a set of spare parts, since a given spare may be employed only in a few positions.

The desirability of using large monolithic arrays of semiconductors for low weight and high reliability (primarily due to the minimal use of unreliable kinds of connections), is in conflict with the desirability of standardizing the arrays, and new schemes of logical organization are clearly needed to achieve a good balance among the various factors.

Other new criteria for the logical design also apply. Thus, networks should be designed so that they are easy to diagnose, so that a failure at a given point does not propagate very far in either direction of signal flow; so that they are as multifunctional as is practical; and in general,

so that they are well suited to various modes of system redundancy such as error detection, fault masking, and replacement. The conventional criterion of minimality of the number of active elements is clearly not a major one in itself.

On the system level, the key new criteria that have not applied with great strength in previous computers are autonomy and flexibility. It is not sufficient, as in usual applications of redundancy, to have errors indicated, but it is necessary to have the capability to accommodate them incorporated in the system; furthermore, such accommodation should be accomplished with great flexibility, both in programming and in hardware, so that the redundancy of equipment is employed to realize the utmost in performance.

In summary, the new cost criteria and failure characteristics of advanced devices and the special logical requirements of fault accommodation present novel problems and opportunities for logical design. These problems and opportunities apply both in the realization of existing error-control techniques and in the design and realization of advanced error-control techniques.

B. Goals, Methods, and Assumptions of the Study

In this section we state the goals of the study, describe the method of approach taken, and state the assumptions that were made in the study, including the scope of the systems of interest and the criteria of evaluation.

1. Goals of the Study

In view of the crucial need for powerful techniques of reliability design for advanced spaceborne computers, the goals of the study have been as follows:

(1) To survey the state of the art of logical design of spaceborne computers as it pertains to the enhancement of reliability; in particular, to examine the various known techniques so as to determine their adequacy for the expected mission requirements, and to determine their mutual compatibility when applied in a computer system

(2) to conceive and evaluate new schemes of system design and operation that offer promise of advancing the state of the art

(3) to recommend further directions of research that will aid in the improvement of present techniques, the evaluation and realization of the new schemes conceived, and the conception of further advanced schemes.

2. Method of Approach of the Study

a. Survey of Known Techniques

The first task of the study was to survey relevant, known techniques. The approach taken to this task was to distinguish those sections of a hypothetical spaceborne computer to which distinctive problems of design apply, to survey the literature for design techniques appropriate to those sections, and to evaluate the merits of the techniques that are deemed most useful for the application.

The major sections of a hypothetical spaceborne computer that were distinguished were the general logic networks, the arithmetic section, and the memory system. The reliability techniques that were considered relevant to the study included the encoding of information for transfer, storage, arithmetic, and control function and for error control; the logical structuring and testing of networks; and those aspects of circuit fabrication that bear on logic design. The literature surveyed included books, professional journals, conference proceedings, and unclassified research reports; in addition, several conferences were attended which were in part or in whole devoted to problems of reliable computer design.*

The evaluation of a given technique was concerned both with the state of the engineering art for its application and with its intrinsic or potential value for the application. The state of the engineering art was taken to include the accuracy and convenience of known methods of analysis of systems employing the technique, and the difficulty of

---

* A detailed survey of the literature on the application of redundancy techniques to reliable computer design is presented in Appendix D of this report.

applying the technique in practical system design. In some cases where a promising technique appeared to need further development, effort was made to solve some of the outstanding problems in order to contribute to the art and to help assess necessary directions of development. The choice of criteria of evaluation will be considered in Sec. I-B-3.

b. Conception of New Schemes

The second task was to conceive new schemes of system design and operation that offer promise of advancing the state of the art. Serious exploration of the basic concepts of redundancy design date back at least to the work of von Neumann in 1952;[318] the concept of a highly reconfigurable modular computer dates back at least to the work of Holland in 1959.[133] Fundamentally new notions (e.g., Moore and Shannon's scheme for recursive construction of relay nets[214] and Pierce's scheme for adaptive voting)[240] have been rare. New schemes have generally been ingenious implementations of known principles, or means for exploiting some special circuit characteristics such as asymmetries in fault types. Some new schemes of this type were developed during the study for the realization of adaptive circuits using only digital switching elements.

Although fundamentally new schemes are to be desired, the study has revealed that there is a substantial lack of design knowledge appropriate to the practical realization of a computer having a high degree of flexibility and autonomy and employing modern device fabrication. Such realization calls for the creation of particular schemes for the overall distribution of functions in such a system, for the realization of particular functions, and for the integration of various kinds of error-control processes. A number of such schemes will be described in this report.

c. Recommendations for Directions of Further Research

In the course of the study a number of significant problems were uncovered. Some were concerned with the advancement of a known technique--e.g., improvement of the facility and accuracy of the analysis and synthesis of "restoring" type redundancy (Sec. II-A-2-a); some with new kinds of networks--e.g., "commutation networks" (Sec. III-C-2); and

some with basic design problems, such as the incorporation of error-control criteria (e.g. ease of diagnosis of faults) in general network synthesis. This report presents the results of some original work on these problems. Recommendations for future work are also included in this report.

3. Technical Assumptions of the Study; Definition of Terms

In this part we shall discuss the technical assumptions that guided the study; in particular, the scope of the design techniques considered, the choice of device technologies, the level of reliability of interest, and the criteria for evaluating reliability techniques.

a. Definition of Terms

We shall first define a number of terms that will be used repeatedly in the report; we have attempted to be consistent with established usage. The definitions are as follows:

Fault: A physical condition of a component that prevents the system of which it is a part from completely performing its specified function. A system within such a state will be called faulty; otherwise it will be called perfect.

Error: An incorrect information state (this state can, of course, appear at the output of a perfect network as a result of an erroneous input).

Fault Masking: A property of a system such that it is perfect even though some of its subsystems may be faulty (see Fault Accommodation).

Fault Detection: Determination as to whether or not a system is faulty.

Fault Location: Determination as to which subsystem in a system is faulty.

Fault Characterization: Determination of the subset of functions of a system that are improperly performed. For each output, the precise characterization will be that subset of inputs resulting in erroneous output. A simpler characterization might be simply a distinction as to which outputs of a system are perfect.

Fault Diagnosis: Techniques for fault detection, location, or characterization.

Fault Correction: Alteration of the physical condition causing a fault so as to restore the system to perfect operation (this could include changing operating conditions such as voltages and frequency).

Fault Accommodation: Essentially the same as fault masking, but masking usually refers to an instantaneous process, while accommodation may also include a sequential process.

Error Detection: Determination that the information state of a signal or a set of signals is in error. This determination may be a computation on the set of signals itself, or in reference to another set of signals from which the set is derived.

Error Correction: A computation upon a set of data, perhaps including other related data, that corrects an error in the set.*

Error Accommodation: (not a widely used term): Comprehends both error correction and alteration of system behavior so as to achieve some modified objective.

Error Control: A general term, including error accommodation and fault accommodation.

Static (or Passive) Error Control: Error-control processes in a system that do not involve changes in the functions of its subsystems or their interconnections.

Dynamic (or Active) Error Control: Error-control processes in a system that involve changes in the functions of its subsystems or their interconnections.

b. Scope of Design Techniques

The scope of the error-control techniques studied was centered about the logical behavior of a computer and its component subsystems. Questions of good design practices for devices and circuits, on the one hand, or on the other hand for computational programs, were excluded.

---

* The distinction between error correction and fault masking is usually clear, but it is often dependent upon how a system is considered to be partitioned.

However, the interactions between logical and circuit design, and logic and program design, that affected the implementation of an error-control technique were of great interest. Thus it is of interest to determine what special constraints upon logical design result from limitations on devices and programs, and also what special device types or program functions would aid the effectiveness of a logical design scheme.

c. Device Technologies

Because of the requirement for high performance, it was assumed that the major devices technology employed will be that based on modern high-speed semiconductor devices. Special emphasis was given to the use of integrated circuits, and in particular it was assumed that the use of large-array monolithic circuits would be very significant in the realization of future spaceborne computers.

The study of memory systems was primarily on the logical level, so that choice of device type was not important. However, it was assumed that the memory performance is consistent with high-speed bit-parallel computation. In the report on that study (Appendix A) it is noted that monolithic semiconductor memory arrays have some attractive features for the spaceborne computer application.

A study was also made of the possible benefits of using magnetic logic devices for special functions within a spaceborne computer (Appendix C).

C. Criteria of Performance

The major criteria used in evaluating an error-control technique were the increases in the measures of reliability, weight, volume, and power consumption of a functional unit employing that technique, relative to the measures of those parameters in a unit having the same processing function but without special error-control features.

There are a number of factors that complicate both the absolute and relative estimates of these measures. In the case of the reliability measure, modern logic components have a high reliability but the time-failure distribution of a given product or assembly method is usually not known; hence it is extremely costly or in many instances impossible,

to estimate absolute reliability. Since it has been established that redundancy is essential to the task, and since one of the ultimate goals of the study is to find the most effective means for employing redundancy, it is sufficient to be able to compare alternative schemes on a relative basis. Thus the major reliability criterion used in the study was comparison of the reliabilities of alternative networks as analytic functions of the reliabilities of their components, where reliability is defined, as usual, as the probability that a component remains perfect for a specified operating time. In some examples, the familiar exponential failure-distribution law was assumed, and various published failure-rate values were employed, in order to give some engineering "feeling" of real time. The resulting probability and time values should be taken very cautiously.

A comment on time measures of reliability is appropriate at this point. A common time measurement is the "mean time to failure" (MTF) which in the literature is almost invariably computed on the assumption of an exponential failure law. It should be noted that for such a failure law the reliability of a system for the time base equal to the MTF is 1/e. This is too small a value for an expensive mission such as a deep-space probe; hence if MTF is used as a measure, values substantially greater than the mission life must be considered. Comparisons of schemes on the bases of their MTF values must consequently deal with time values that have only weak intuitive significance. Thus for $1 - P(t) = 1 - \exp(-T_{Mission}/MTF)$ very small [where $P(t)$ is the probability of perfect system operation at time t] MTF is approximately $T_{Mission}/[1 - P(t)]$. For example for $P(t) = 0.999$, MTF = $T_{mission} \times 1000$. A measure having much greater engineering significance, proposed by Knox-Seith[164] and Angell[8] is the "useful life," defined to be the longest mission time for which the probability of failure is no greater than Δ, where Δ is usually much less than one. Not only does this measure have greater heuristic significance for the missions of interest, but it is also more sensitive to variations in redundancy than the MTF measure. This measure is discussed further in Sec. II-A-2-a-1).

Measures of weight and volume are complicated by packaging considerations, since the weight of the active components in future technologies may be considered to be almost negligible compared to that of the packaging and interconnections. For nonintegrated circuits, packaging costs are approximately proportional to component count; but for integrated circuits, packaging costs depend upon the component count and the number of components that may be incorporated within a package. The latter number may be expected to increase within a range of two orders of magnitude over present integrated-circuit values (which typically provide the equivalent of one flip-flop per package). Hence comparisons of the weight and volume of realizations of alternate schemes must consider the effects of a rapidly developing technology. Thus, if one scheme lends itself better to larger array realization (by reason, for instance, of greater modularity), the packaging cost may actually be less than for a scheme whose count of logic circuits is lower.

Measures of power are somewhat simpler. The obvious factors of significance are the number of logic circuits and the fraction of those that may be in a power-on status. One indirect factor that is sensitive to logical organization is the degree to which circuit speed may be exchanged for number of parallel-acting circuits. Thus if such an exchange may be made in direct ratio, it would permit a reduction in total power consumption to be realized by the use of devices having low values of switching time--power consumption product.* Increases in parallelism adequate for significant power savings may not be feasible because of inherent serialism within the computations of interest, but the possibility of such savings should not be overlooked.

---

* To illustrate, if in a system with n circuits, each with circuit switching time t and power consumption p, computation speed s is proportional to t/n, then total power = p n $\propto$ pt/s. For a given system speed, devices having lower values of pt would take less total power.

D. Organization of the Report

Chapter I has been a review of the operational and technical problems of realizing reliable spaceborne computers and an explanation of the method of approach of the present study. The technical analysis of reliability techniques is presented in Chapter II, which is concerned with techniques for fault masking and error detection, and Chapter III, which is concerned with techniques for automatically controlled fault diagnosis and reconfiguration.

The techniques of Chapter II are moderately well included in the present state of the art of reliability design; but, as is seen, by no means are they adequately understood. Most of these are forms of passive error control, but several elementary active schemes are also included. For example, the use of codes for error detection is a component of dynamic error control, but it is included in Chapter II because it is a fairly well-studied technique, and because the coding approach is helpful in describing certain fault-masking schemes.

The techniques of Chapter III are all components of dynamic error control, and they are all either new techniques, or practical implementations of hitherto "ideal" schemes.

In Chapter IV we present the conclusions of the study and recommendations for a program of further research.

There are four appendices. The first is concerned with the application of error-control techniques to memory systems. The second is concerned with the design of modular distributed power supplies in relation to the modularization of the logic of a computer. The third considers the possible applications of magnetic-logic devices. Since such devices are slow, the study concentrated on those functions for which their use would not substantially slow down a computer's basic cycle of operation.

The fourth appendix is an extensive guide to the literature that is directly pertinent to the design of reliable spaceborne computers. It is intended that this guide be directly usable as an introduction to the literature; hence there is some overlap between it and the comments on the literature found in the main text.

II TECHNIQUES OF LOGICAL DESIGN FOR FAULT MASKING AND ERROR DETECTION

This chapter is concerned with techniques of logical analysis and design that are needed for the realization of computer functions in which the control of errors is static; i.e., in which the error state of the computer is a subject of concern only to the local areas.

In this chapter we will discuss refinements of basic techniques that are well known. Our goal is to distinguish those methods which are particularly suited for the achievement of high reliability for specific functions, at minimum cost, and also to indicate algorithm-like methods, wherever possible, for the optimum application of the techniques. Section II-A is concerned with fault masking as applied to general logic functions, both combinational and sequential. A comprehensive review is first presented of the known fault-masking techniques, followed by a detailed discussion of the voting-type restoration scheme. Here several techniques are presented concerning the analysis of arbitrary restored networks; in addition, several novel implementations are presented of the powerful adaptive-restoration scheme. Several schemes are discussed for detecting failures in sequential networks.

The chapter concludes with Sec. II-B which discusses the status of error-correction coding techniques for passive error control, in particular as applied to arithmetic and storage operations.

In the attempt to distinguish the optimum applications of the logical design techniques many problems were uncovered, both analytical and of an engineering nature. In some instances detailed solutions were examined, while in other cases rough designs were presented with conjectures relating to the optimum solution, thus providing a framework for future research.

A. Fault-Masking Techniques for General Logic Functions

In this section we consider logical techniques for the masking of faults in networks realizing general logic functions. In the first part we review the most significant known techniques. In the second part we examine two of the most important techniques in detail--the multiple-line voting scheme and the combined fault-masking and replacement scheme--in order to assess the state of the art of their application. Also in the second part we examine some realizations of voting networks. The first two parts are concerned essentially with combinational networks; in the third part we examine the state of the art of techniques for error control for sequential networks, with special emphasis on error detection.

1. Review of Significant Techniques

In this part we attempt to assess the applicability to spaceborne computers of the most significant known techniques for designing logic networks that can mask internal faults. A number of the most attractive techniques are distinguished, and reference is made to the sections of this report that consider the techniques in greater detail.*

Many schemes have been described for designing logic networks having fault-masking capability. The well-known schemes exhibit a great deal of ingenuity, but only a few have combinations of features that make them practical for application to present-day digital networks.

The tree of Fig. II-A-1 has been constructed in order to display the most significant techniques in an orderly perspective. Only those techniques have been included that have been described in sufficient detail that functional networks could be designed, and for which circuit and design techniques are currently available. Several other interesting schemes, for which some circuit or logical action has been hypothesized but for which no practical designs have been given, will be mentioned separately.

---

* Several reviews that may be of interest to the reader are those by Teoste,[300] Pierce,[243] and Garner et al.[92]

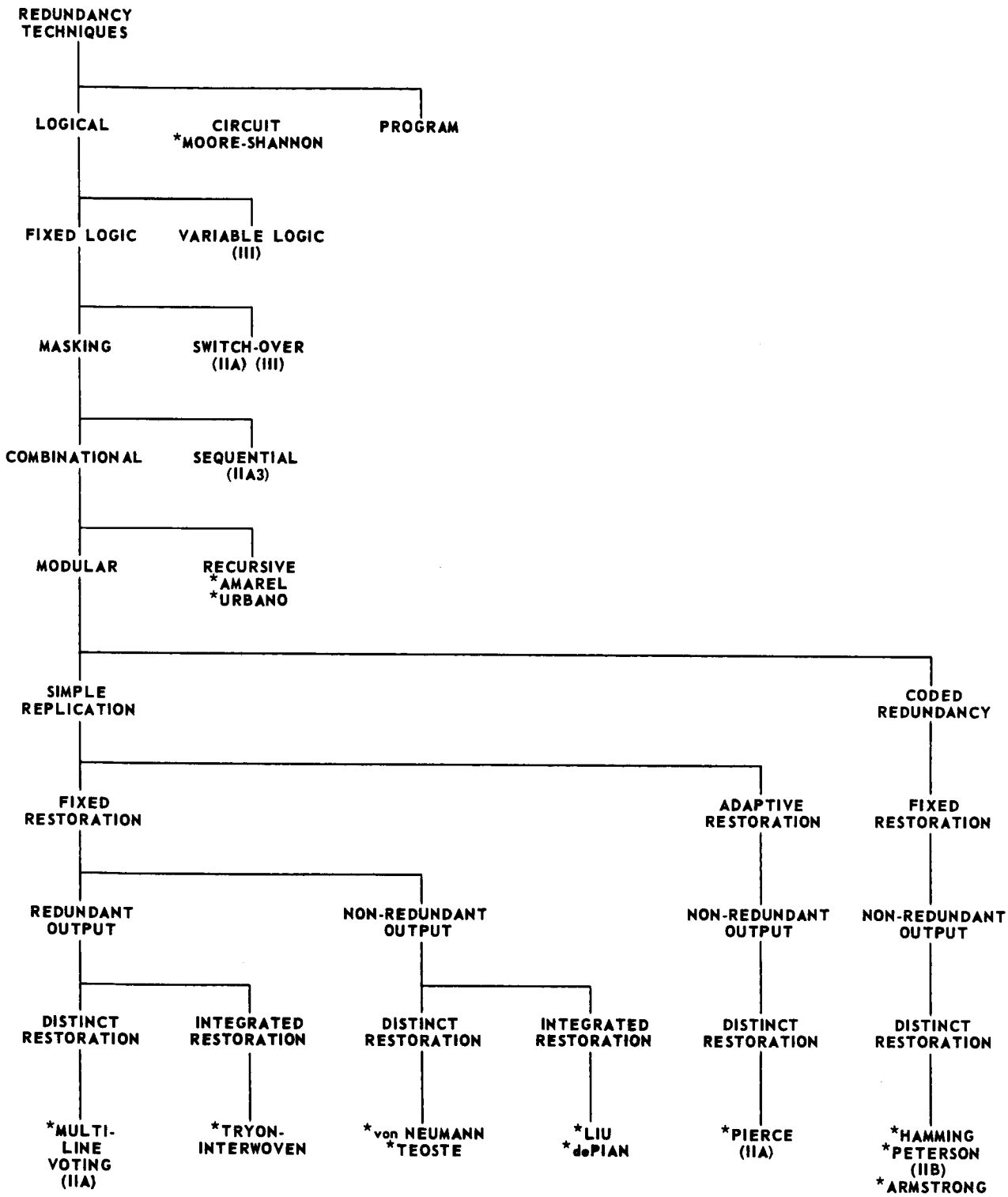


FIG. II-A-1 A CLASSIFICATION TREE FOR REDUNDANCY TECHNIQUES

The first (top) level distinguishes these major classes of redundancy techniques: circuit, logical, and programming. The standard method of circuit redundancy is the replacement, within a circuit, of an unreliable circuit component (e.g., a diode or a resistor) by a cluster of components whose net circuit impedence or transfer function changes within limits that are acceptable for correct circuit operation when some number, or fewer, of the components fail. General methods for organizing such networks have been given by Shannon and Moore.[214] In practice, this approach is useful for special circuits within a computer, in which a given component is under heavy electrical stress (e.g., in power supplies or high-current pulse drivers), or in which the circuit has very generous operating margins. It has been found that application of the technique to low-level logic circuits or memory sense amplifiers results in circuits that have substantially poorer margins with respect to component aging, noise, and variations in operating voltages and temperature than equivalent non-redundant circuits; hence there is likely to be a reduction in reliability if the technique is applied to low-level circuitry. Programming redundancy is a very important technique for spaceborne missions; it is discussed in Sec. III-A of this report. The remainder of the tree is concerned with techniques of redundancy at the logic level.

The second level distinguishes techniques in which the logical structure of a system is variable or fixed. Provision of a variable structure in a system that is part of a larger system increases the flexibility with which the larger system may accommodate faults; thus, not only may the number of tolerable fault conditions be increased, but the system may permit tradeoffs in performance that may enable critical functions to be performed. This technique is discussed at length in Chapter III of this report.

The third level distinguishes techniques of fixed logic function in which faults are accommodated by masking or by switchover. Automatic switchover of spare parts to replace faulty parts is commonplace in general electrical and electronic practice, and is an established practice in space vehicles.[265] Although many computer-oriented reliability

analyses have been made of abstract models of this method (e.g., by Kruus[171] and Muth,[225]), practical applications to computers has been rare. Some recent examples are the Bell System ESS-1 Central Switching System[39] and the memory system of the Saturn Guidance Computer[61] (anticipated by Kemp,[157]) both of which use duplex switching. Recently, descriptions of techniques for the practical logical design of computers, in which switch-over would be applied at relatively low levels within the computer, are appearing in the literature (e.g., Terris[303] and Agnew et al.[1]). In this report, techniques relating to this method are discussed in Sec. II-A-2-b and in Chapter III. All the remaining schemes are forms of fault masking within a network of fixed structure; i.e., at no time may a part of the network be blocked from contributing to the network output.

The fourth level distinguishes techniques that deal with networks essentially as sequential or as combinational networks. Although a computer is ultimately a sequential network, it is helpful to approach the design of various subsystems with emphasis either on their state-sequence behavior or on the classes of combinational functions that are realized in the network. Error-control techniques for sequential networks are discussed in Part 3 of this section.

The fifth level distinguishes techniques in which the process for the construction of a fault-masking network is either recursive or modular. In a recursive process, first employed by Shannon and Moore, a given subsection of a network is replaced by a cluster of elements in a way that preserves the structure of the network, and this process is repeated for all subsubsections of the subsection, until a desired improvement in reliability is achieved. Amarel and Brzozowski[6] described an approach in which the unit of recursion is a single gate (producing so-called "triangular nets") and Urbano[312] extended their scheme by making the unit of recursion a network (producing so-called "iterated neural nets"). These approaches are of considerable theoretical interest in relation to the question of limits to achievable reliability, but the schemes lead to networks that are so large, or that have such drive requirements that they may be considered impractical for present-day realizations. Modular schemes proceed by adding functional networks to

one or more nonredundant functional networks, and combining their outputs in some special manner. The remaining schemes to be discussed here may all be considered to be modular.

The sixth level distinguishes between simple replication and coded redundancy. In coded redundancy, the functions provided by the resultant nets are not identical to those of the original nets, but are related in ways that may usually be described conveniently in terms of a code. Code redundancy may be more or less effective than replication, depending upon the logical functions performed. Coding redundancy has been demonstrated to be either superior to or competitive with replication redundancy for special functional areas within a computer that are characterized by a high degree of uniformity of structure. These are data storage, arithmetic, and analog-digital conversion. Methods for these functions have been highly developed, following the early work on the different subjects by Hamming,[117] Peterson,[239] and Kautz,[151] respectively.* Recent developments are discussed at some length in Sec. II-B of this report. Coding methods for sets of general, complex logic functions have been described by Lofgren,[184] and presented (more comprehensively) by Armstrong.[10] The value of these methods for general logic functions has not been demonstrated, and has been doubted by several authorities (e.g., Pierce,[243] pp. 132-145). The two factors that lead to low efficiency are the need to apply fault masking to the decoding logic and the possible high cost of producing the redundant checking functions independently of the nonredundant functions. Although the approach is not attractive as a general method, logical designers would be well advised to be aware of its possible value for particular applications.

The seventh level distinguishes between schemes in which the restoration of the desired function from the possibly imperfect set of functions produced is performed according to an adaptive rule or a fixed rule. The

---

* In citing early authorities, we attempt to specify those references which first developed a concept with some generality; these have often been preceded by disclosures of particular schemes that embody the general principles.

use of an adaptive rule was proposed by Pierce, who demonstrated a rule which, if implemented reliably, would achieve higher reliability than a fixed rule of the same order of redundancy. The method has not been applied, because of the unavailability in modern technology of circuit elements that would implement the rule reliably. With the miniaturization of logic elements, it appears to be increasingly feasible to implement an adaptive rule in an all-digital circuit. Several schemes for such implementation are described in Sec. II-A-2 of this report.

The remaining schemes are distinguished by the criteria of levels 8 and 9. These are, respectively, whether or not the fault-masked network produces redundant output functions, and whether the restoration function and the basic logic functions are accomplished in distinct logic networks or are integrated in a single network.

The scheme of Liu and Liu[183] (with its derivatives) is the only example found in this study of a general treatment of single-output, integrated restoration. It employs a network of redundant-input threshold logic elements. In these elements, an output is produced if the linear sum of weighted inputs exceeds a threshold, and the weight and the threshold are chosen so that if an input is in error, the sum is still on the correct side of the threshold. A serious disadvantage of this scheme is that the sum of the weights required for the many inputs to an element is so great, even for simple functions, that presently designable threshold logic circuits would have unacceptably low margins. The scheme as it stands must thus be considered impractical. It is clearly possible to transform the threshold-element networks into networks of simpler (e.g., NOR) elements, but it is not known how such networks would compare in size with those realized by nonintegrated schemes. This possibility has not been investigated during the present study, but some consideration is recommended. De Pian and Grisamore[59] have noted the possible merits of this approach, and have given an illustration that is too simple to permit a general evaluation.

Networks built according to the schemes of von Neumann ("single-vote-taker redundancy")[318] and Teoste ("gate-connector" redundancy)[300]

have single outputs and distinct sections for corrections and for the basic logic. The scheme of Teoste may be considered as an adaptation of the von Neumann scheme for even-order redundancy. Although its description supposes the availability of three-terminal branch-type (relay-like) logic for the restoration circuit, there is a simple gate-type equivalent. In the very well-known von Neumann scheme, the restoration logic is the majority function. A number of significant, good features of these schemes may be enumerated as follows.

(1) The scheme is equally effective for both $0 \rightarrow 1$ and $1 \rightarrow 0$ errors.

(2) The correction logic may be realized by the same kind of digital circuitry as the functional logic; i.e., no special elements are needed.

(3) The size of the functional module is unlimited; i.e., it may range from a single gate to a whole computer.

(4) No modifications are required to the method of realizing the functional logic, either in network structure or in factors of element usage, such as fan-in or fan-out.

(5) The scheme is extendable to high orders of redundancy, and a system may employ different orders at various points without causing any special problems in design.

A significant limitation of the scheme is its sensitivity to faults within the restoring network. For the ultimate output of a computer, if there is a single receiver, the reliability of the element producing the output must affect the reliability of the output; also, there is no benefit in cascading vote-takers upon vote-takers. Hence the reliability of the restoring network is an ultimate limit to the reliability of the system. When the receiver of a system output may be replicated itself, then limitation due to the restorer can be reduced significantly, as in the following schemes.

The multiple-line voting scheme (also called multiple vote taking, triple modular redundancy, etc.) derived by a number of workers from von Neumann[318] and the scheme of Tryon[305] (also called quadded logic or interwoven logic) employs a redundant encoding on the output of a functional network. Thus, a number of versions of a logic function are generated, and all of these are employed as inputs to successive logic networks (the order of redundancy of output lines is often made the same

as the order of redundancy of the functional modules, but this is not essential). This approach is beneficial because it reduces the dependency of the system's reliability upon the reliability of the elements in the restoring circuit.

Tryon's scheme is effective and simple to apply (see Pierce,[243] Chapter V). The following are some of its weaknesses:

(1) Fan-in and fan-out requirements for all logic elements are increased significantly; e.g., for fourfold redundancy both fan-in and fan-out are doubled. This tends to decrease the basic reliability of the elements, and if more elements are used to handle the increased input and output loading, new sources of error are introduced. Moreover, for quadding the redundancy in elements is approximately eightfold.

(2) Cross connections among logic elements (which are essential for the fault-masking action) must be made at every level of logic. This tends to increase the number of interconnections and the weight and volume of the interconnection scheme. Also, it does not permit variation in redundancy ratio except by changing whole orders of redundancy.

(3) Reliability analysis of the networks constructed is extremely complex; hence it is difficult to design for good reliability improvement under weight and power constraints.

(4) Design of networks for ease of preflight testing* appears to be difficult; for example, in order to isolate a set of elements for testing, all untested gate outputs must be set to a constant 1 or a constant 0, depending on the position of a gate within a network.

Because of these weaknesses, the Tryon scheme has not been employed extensively in practice. It is an elegant scheme of substantial theoretical interest, and it is almost practical, but it appears to be generally inferior to the multiple-line voting scheme.

The multiple-line voting scheme has all the advantages enumerated for the single-line voting scheme, in addition to the reduction of the sensitivity of the system to vote-taker unreliability. Also, preflight testing is reasonably straightforward, given independent control of the

---

* Such testing is desirable in order to expose faults that would otherwise be masked, so that the computer may be started in as perfect an initial state as possible.

power supply for the individual ranks of logic. This scheme appears to be the most attractive way of accomplishing fault masking for general logic functions presently known. Furthermore, it is applicable to networks for special functions such as arithmetic and storage, and it is in fact quite competitive with schemes based on coding. A number of important problems of analysis and optimum designs remain to be solved. These are discussed in detail in Sec. II-A of this report.

Several concepts for fault masking that have been described in the literature have been omitted from the above classification and discussion. Some of these are the multivalue logic schemes of Lowenschuss[185] and the Transor and Quantile schemes by Mann[195]. These have been omitted because they require some hypothetical circuit element or circuit-design scheme, the feasibility of which has not been evaluated. It is not known that any of these schemes have been developed since their description.

Inspection of the classification tree of Fig. II-A-1 suggests a number of possible variations in the approaches of adaptive restoration and coded redundancy, based on the use of either redundant outputs or integrated restoration or both. Some of these variations may provide bases for useful schemes.

It should be reiterated that all the schemes described here accomplish fault masking locally within a system; i.e., there is no provision for reallocation of redundant equipment among different functions. It should be noted that those schemes in which basic logic function and restoration are distinct permit an easy augmentation in logic to provide, concurrently with the fault-masked data output, a separate signal that may indicate a fault condition; and further, to provide indication as to which replica is faulty. This feature is clearly useful in a replacement-switching system.

2. Combinational Techniques for Fault Masking in General Logic Networks

In this section we discuss techniques for the design of networks for realizing general logic functions that can mask internal faults. The techniques to be considered do not depend on the history of the

network, so although the network may contain sequential elements, it is convenient to visualize the networks to be discussed in this section as strictly combinational networks. Fault-masking schemes that are based on the sequential nature of a network are discussed in Sec. II-A-3. In part a we review the state of the art for the analysis and design of multiple-line voting-restoration networks, which is deemed to be the most important available technique. A number of improvements to existing methods of application are offered, and a number of suggestions are made for further points of improvement. In part b we consider several new schemes for the combination of fault-masking and switch-over redundancy. Detailed schemes for networks having such features are presented, and a number of problems that require further development are indicated. In part c we present a number of particular designs for restoring elements; included are single-output majority-function nets which are suitable for passive fault masking, and multiple-output threshold-function nets, which are useful in schemes for combined fault masking and switch-over.

a. Techniques for the Design of Multiple-Line Fault-Masking Networks

1) Introduction and Summary of Prior Work

The preceding discussion has served to introduce the various known techniques of digital logical design for error detection, fault masking, and repair.

In the present section a detailed description will be presented of the passive fault-masking technique wherein redundant circuits are used in connection with selectively placed restoring organs (voters) throughout the system. Our ultimate aim in this study is to distinguish those portions of a spaceborne computer which can most benefit from the application of the restoration technique, and also to provide computer designers with an indication of the reliability improvement that can be anticipated and specific rules for designing redundant networks.

Since von Neumann's initial paper proposing the restoration scheme, many subsequent papers have appeared discussing various aspects of

this technique. Each of these studies can be considered as relating to one or more of the following three categories:

(a) Qualitative description of the advantages of using redundancy and restoring organs.[318, 188, 33, 333] These studies were primarily concerned with establishing the basic theory embodying the restoration scheme, and also with distinguishing the overall improvement in reliability for simple digital models incorporating restoration.

(b) Quantitative discussion of simple models. These studies were concerned with the extension of previous results to provide a quantitative evaluation of the improvement in reliability which can be achieved with the restoration scheme, and also with methods of determining for a given system "cost," the optimum allocation of redundancy, i.e., the evaluation of the order of replication and the placement of voters so as to minimize the probability of system failure. Unfortunately these studies pertained only to simple models of digital systems--namely, the visualization of a computer as a cascade of single-input, single-output blocks,[164, 115] or as a tree network of double-input, single-output blocks.[69]

(c) Analyses of complex system models. The recognition of the shortcomings of the simple cascade or tree models led to the search for techniques permitting the analysis of arbitrary restored systems. These studies culminated in Monte Carlo simulation programs for the analysis of specific systems[75, 188, 141] and also in an approximate analytical approach based upon determining the set of network cut-sets which when individually faulty will result in system failure.[140, 141] With either analysis technique, it is possible to determine the optimum placement of voters (given a maximum permitted number of voters) by, in effect, evaluating the probability of system failure for each possible set of voter locations, and then choosing that set which minimizes the failure probability. A dynamic programming approach[272] to the voter placement problem has been proposed along with several other approaches[141] which are not as lengthy as the exhaustive procedures, although their application has not yet been established.

Although this prior work provides an adequate framework for determining, for certain systems, the expected gain in reliability

from the use of the restoration technique, several important questions have remained unanswered. These include the following.

(a) For a given logical network, and a maximum permitted overall redundancy, how can one estimate the expected reliability, assuming optimum allocation of voters? This question has remained unanswered in spite of the abundance of research. For example, a reliability simulation was conducted[75] of an arithmetic and control section of an airborne digital computer, which as a nonredundant unit required 300 4-input, single-output gates and exhibited a probability of failure of 0.095. With triple replication and the use of $3 \times 246 = 738$ voters it was found that the probability of failure was decreased to 0.004. On the basis of visualizing the computer section as a cascade of 300 triplicated gates with 3×246 voters placed optimally in the network, it can be shown by the analysis technique discussed in Ref. 164 that the failure probability is 0.0001. Thus it is indicated that extreme care should be exercised in applying results derived from consideration of the simple model.

(b) What logical design techniques, if any, relating to the logical dependence of the outputs of a multiple-output network, yield replicated networks with minimum probability of failure?

(c) For a given logical network, and a maximum permitted redundancy, what are "simple" techniques for determining the optimum allocation of the available redundancy?

Question (a) is discussed in Sec. II-A-2-a-3), where several simple techniques are presented for the analysis of given replicated logical networks with arbitrary placement of voters. These analysis techniques are then used in Sec. II-A-2-a-4) to derive upper and lower bounds on the redundant network reliability (assuming that the initial network has a high circuit reliability) as functions of the number of gates in the network and the maximum fan-in and fan-out. It is shown that the simple cascade model provides an optimistic estimate of the probability of failure.

Question (b) pertains to optimum logic design techniques for multiple-output networks which are replaced by redundant versions of the network with a restoring organ for each output. The major query is

whether, on the basis of minimum probability of failure, the network should be realized in a minimal-gate manner--with, of course, a dependency in the outputs--or whether it should be implemented with a distinct independent block for each output. In Sec. II-A-2-a-5) it is shown that for a complete decoding tree the minimal dependent output implementation is to be preferred, although at this time we have not yet established that the minimal-gate solution is to be preferred in general.

Question (c) concerns simple techniques for the determination of the optimum replication order and the optimum placement of voters given a fixed, but arbitrary, amount of available redundancy. Except for the simple cascade and tree models this question has remained unanswered, although we have formulated several conjectures relating to heuristic programming techniques, which are discussed in Sec. II-A-2-a-6).

Before proceeding to the development of analytical techniques for answering the three questions, it is convenient to define seven assumptions concerning circuit failures.

(1) A nonredundant network is composed of many circuit blocks; the operation of each circuit block is essential for overall network operation.*

(2) A circuit under consideration either is working properly or has failed completely.

(3) If a circuit has failed, its output will always be in error, regardless of the condition of the input variables. This is tantamount to assuming that the period of occurrence of those inputs for which the output is in error due to the failure is low compared to the mission life.

(4) The failure of a circuit will not affect the input to the circuit.

(5) If a circuit has failed, it can only be restored to proper operation by being repaired.

---

* This assumption negates the possibility of inherent logical redundancy which can effect fault masking.

(6) Circuit failures are independent and are due primarily to random component failures, as contrasted to predictable component wear.

(7) The probability of circuit failures is small, throughout the mission time of interest.

In the discussion to follow we will determine the performance of complex redundant systems as a function of the failure probabilities which are assigned to each of the circuit blocks of the network. Although it is understood that the overall failure probability of the network will be a function of time, the exact functional dependency can be determined only if a temporal function relating the occurrence of circuit failures is known. It has been shown[66] that for the assumption of random circuit failures, the number of circuit failures in a given period of time will have a Poisson distribution if each circuit is repaired or replaced after it has failed, and also if several other weak conditions are satisfied. The replacement (or renewal) assumption is valid for redundant systems with high values of replication, in which case for the purposes of calculation it can be assumed that the probability of an individual circuit block operating correctly decreases exponentially with time.

It is not immediately clear that the exponential failure law applies to the circuit blocks when low orders of replication (e.g. 3) are used, but because of the unavailability of accurate data on the distribution of component life we will assume that the exponential law, with published parameters for components, can be applied.

In the consideration of an appropriate measure of reliability it is important to weigh the intended application of the system. Clearly all of the pertinent information concerning system reliability is contained in the function $P(t)$, which is defined as the probability that the system is operating correctly at time t. The system behavior could also be expressed in terms of the function $Q(t) = 1-P(t)$--the

probability that the system will fail during the time interval (0,t).* The relative improvement in reliability obtained by using redundancy can be expressed as

$$S = \frac{P(t) \text{ for redundant system}}{P(t) \text{ for nonredundant system}} = \frac{P_R(t)}{P_0(t)} \qquad \text{(II-1)}$$

In many cases it is difficult to derive explicit functions P(t), Q(t), and moreover it has been the practice of reliability engineers to refer to one parameter as a means for describing performance. The commonly used parameter is the mean time to failure (MTF), defined as

$$MTF = \int_0^\infty P(t)dt \quad . \qquad \text{(II-2)}$$

The corresponding improvement factor is expressed as

$$I = \frac{MTF \text{ for redundant system}}{MTF \text{ for nonredundant system}} \qquad \text{(II-3)}$$

Assuming that the exponential failure law is satisfied for the system, then the MTF expresses the time at which the probability of the system operating correctly is 1/e = 0.37. If a single parameter is used to describe the system performance for such applications as manned space missions, it is evident that the term MTF is not appropriate since it is unlikely that a mission would be permitted to progress to the point where the probability of success is as low as 0.37. Perhaps a more appropriate single term for describing the performance of a system is the useful life[164,8] T_Δ, which is defined to be the longest mission time for which the probability of failure is no greater than Δ, where Δ is usually much less than one.

---

* In the following sections it will at times be convenient to discard the argument t in referring to success or failure probabilities. It is understood, however, that these probabilities indeed are temporal functions.

It then follows that

$$Q(T_\Delta) = \Delta \quad , \tag{II-4}$$

and the corresponding improvement factor is defined as

$$R(\Delta) = \frac{T_\Delta \text{ for redundant system}}{T_\Delta \text{ for nonredundant system}} = \frac{T_{\Delta R}}{T_{\Delta 0}} \tag{II-5}$$

In the following section a brief review is presented of the simple models which visualize a complex network as a cascade or tree of simple logic networks.

2) Techniques for the Analysis of Simple Models*

In Fig. II-A-2 the use of redundant binary circuits followed by a redundant set of majority vote takers is illustrated.

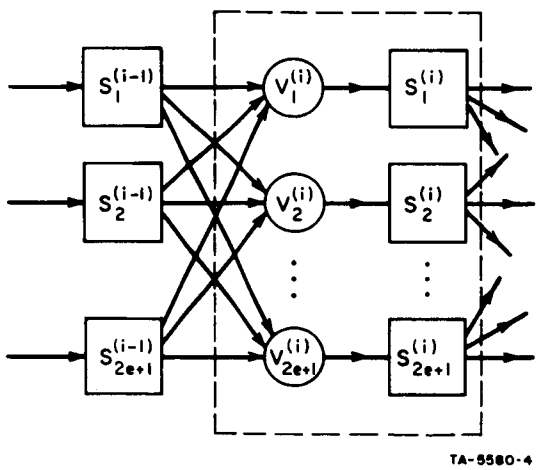


FIG. II-A-2 STAGE FOR CASCADE NETWORK

If voters are spaced throughout a large redundant logical network it is not difficult to visualize the network as consisting of sets of redundant

* Portions of this section incorporate some of the analytical techniques presented in Refs. 164, 69, and 188.

circuit blocks surrounded by redundant voters. It will be convenient to define a stage in such a redundant network as a portion of the network including a set of input voters and all of the circuit blocks between the input and output voters. The stage concept, which is admittedly ambiguous at this point, will be clarified in Sec. II-A-2-a-3) when arbitrary networks are discussed. At present it will suffice to refer to a stage, shown enclosed in the dotted lines of Fig. II-A-2, in a simple cascade of redundant, single-input, single-output circuit blocks with spaced voters. (It will at times be convenient to represent the replicated networks with voters as a nonredundant network with circles placed at locations where the set of replicated voters would appear. This simplified representation is illustrated in Fig. II-A-3 for the cascade stage.)

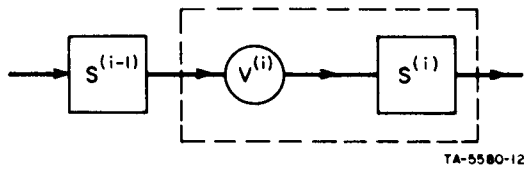


FIG. II-A-3 SIMPLIFIED REPRESENTATION OF A RESTORED NETWORK STAGE

In this case the order of replication is $2e + 1$, $e = 1, 2, \ldots$. We will characterize the stage (distinguished as the i^{th} stage) as operating (correctly) if at least $e + 1$ of the circuit blocks $S_1^{(i)}, S_2^{(i)}, \ldots, S_{2e+1}^{(i)}$ provide correct outputs, given that at least $e + 1$ of the previous circuit blocks (contained in the $i - 1$ stage) $S_1^{(i-1)}$, $S_2^{(i-1)}, \ldots, S_{2e+1}^{(i-1)}$ provide correct outputs. Similarly the i^{th} stage is characterized as not operating if at least $e + 1$ of the pertinent circuit blocks do not provide correct outputs. With these definitions of "stage" and "operating stage," an entire network will be operating if all of its stages are operating.

Referring to Fig. II-A-2 the probability $q^{(i)}$ that the i^{th} stage is not operating is

$$q^{(i)} = \sum_{j=e+1}^{2e+1} \binom{2e+1}{j} (pp_v)^{2e+1-j} (1 - pp_v)^j \qquad \text{(II-6)}$$

where

p = probability that a circuit block is operating

p_v = probability that a voter is operating

$$\binom{2e+1}{j} = \frac{(2e+1)!}{j!(2e+1-j)!} \qquad \text{(II-7)}$$

If Eq. (II-6) is expanded in a series involving powers of the parameters $q = 1 - p$ and $q_v = 1 - p_v$, and all higher-order terms are discarded, it is noted that

$$q^{(i)} \approx \binom{2e+1}{e+1} (q + q_v)^{e+1} \text{ when } q, q_v \ll 1 \quad . \qquad \text{(II-8)}$$

It is of interest to augment the conditions $q, q_v \ll 1$, for which the above approximate equation is valid, with more precise conditions. It can be shown that the term of the expansion of Eq. (II-6) in which q, q_v appear with exponents summing to $e + 2$ is

$$\binom{2e+1}{e+2}[q + q_v]^{e+2} - \binom{2e+1}{e+1}\left\{(e+1)[q^{e+1}q_v + q_v{}^{e+1}q] + eq^{e+2} + eq_v{}^{e+2}\right\} \qquad \text{(II-9)}$$

The ratio of (II-9) and Eq. (II-8) is, for $q \approx q_v$, of the order of q. This indicates that the error introduced by using the approximation (II-6) is negligible for practical systems. Also, for $e = 1$, (II-9) is negative indicating that (II-8) provides an upper bound on the stage failure probability.

Equation (II-6) can be derived in an alternate manner. Let us say that the stage of Fig. II-A-2 does not operate correctly if and only if the proper combinations of $e + 1$ failures occur covering

voters and circuit blocks. (Under this failure condition we ignore the occurrence of more than e + 1 failures.) As an illustration of the procedure for the counting of the proper failure combinations, let e - 1 of the total of e + 1 permitted failures occur in the voters, and 2 failures occur in the circuit blocks. There are then $\binom{2e+1}{e-1}$ combinations of e - 1 voter failures, and $\binom{e+2}{2}$ circuit-block failures* which can result in stage failure. Hence the probability of stage failure due to e - 1 voter failures and 2 circuit-block failures is

$$\binom{2e+1}{e-1}\binom{e+2}{2}q_v^{e-1}q^2p_v^{e+2}p^{2e-1}$$

$$\approx \binom{2e+1}{e-1}\binom{e+2}{2}q_v^{e-1}q^2 \quad \text{for } q,\ q_v \ll 1 \quad .$$

Considering, then, all proper combinations of voter and circuit-block failures, the expression for the probability of stage failure is approximated by

$$q^{(c)} \approx \sum_{j=o}^{e+1}\binom{2e+1}{e+1-j}\binom{e+j}{j}q_v^{e-j}q^j = \binom{2e+1}{e+1}(q_v+q)^{e+1} \quad . \tag{II-10}$$

This failure-counting technique will be exploited further in Sec. II-A-2-a-3) in the consideration of arbitrary networks.

When a network is formed by connecting N identical stages of the type shown in Fig. II-A-2 in a simple cascade, the resultant expression for the probability of network failure is

$$Q_R = 1 - (1 - q^{(i)})^N \quad . \tag{II-11}$$

---

* Once the positions of the voter failures have been established there are only e + 2 circuit block locations remaining for which combinations of failures can result in stage failure.

Equation (II-11) can be approximated as

$$Q_R \approx N\binom{2e+1}{e+1}(q_v + q)^{e+1} \qquad \text{(II-12)}$$

with an error term on the order of the value specified by Eq. (II-9) but increased by a factor of N.

Equation (II-12) reflects the probability of overall network failure based upon the assumption that only the proper combination of e + 1 failures occurring in the N(2e + 1) circuit blocks and voters can result in failure.

As a final remark concerning the reliability of simple cascades, consider the problem of determining the optimum number of voters (and also, as a trivial question for the cascade, the placement of the voters) so as to minimize the failure probability. Assume that in the cascade of N replicated circuit blocks, a set of voters is placed after every N/N' circuit blocks.* We then find that the expression for network failure probability Q_R' reduces to the following Eq. (II-13), where it is noted that the network, whose performance is described by Eq. (II-12), now consists of a cascade of N' stages and the probability of circuit-block failure in each stage is $(N/N')q$.

Then

$$Q_R' \approx N'\binom{2e+1}{e+1}\left[q_v + (N/N')q\right]^{e+1} . \qquad \text{(II-13)}$$

We find that Q_R is minimized (performing the minimization over the variable N') for

$$N'q_v = eNq . \qquad \text{(II-14)}$$

---

* It is assumed that N' divides N.

For e = 1--which, it is recalled, is tantamount to triple replication--the optimum stage division of the network is such that the failure probability of the voter is equal to the combined (nonreplicated) failure probability of the circuit blocks in each stage. It is noted that with the optimum stage division, for triple replication, the amount of equipment required is on the order of six times the nonredundant equipment, assuming that circuits exhibiting comparable failure probabilities are of similar complexity. If fewer stage divisions than specified by Eq. (II-14) are employed in order to reduce the overall redundancy level, then the optimum placement of the available voters is such that all stages exhibit equal (or nearly equal) failure probabilities.

It is of interest to determine the tradeoffs between reliability improvement achieved through the use of redundancy as a function of the cost of implementation, and the cost of the nonredundant network.* In this instance the cost factor will simply reflect the ratio of the number of components in the redundant and nonredundant networks. The analytical procedure, however, can be easily modified to include other cost factors.

Consider the simple cascade of N circuit blocks each with failure probability q = 1 - p. The probability of failure for the resultant restored network, with voters placed after every N/N' circuit blocks, can be derived in a conventional manner by application of simple modifications of Eqs. (II-6) and (II-11)† accounting for the existence of N' stages as opposed to N stages. The following assumptions are pertinent to the analysis:

---

* The following discussion related to reliability as a function of cost was adapted from notes of E. K. Van De Riet. The results reported herein are an extension of the results presented in Ref. 164 to include the effect of realistic cost and reliability measures for the voters.

† The exact formulas for failure probability rather than the approximate formula (II-12), were employed because some consideration was given to circuit-block and voter failure probabilities which were not low enough to warrant the use of the approximation.

(a) There are an equal number of components in a single-circuit block and a 3-input voter circuit.

(b) The complexity of a voter circuit, for more than 3 inputs, is proportional to 3^{e-1}. (This complexity assumption is based upon a component count of non-minimal realizations of majority gates. The majority-gate implementation presented in Sec. II-A-2-c indicates that perhaps a more realistic complexity factor would be 2^{e-1}.)

(c) We can define the following overall restored-network cost ratio C, based upon the above assumptions, and also contingent upon the assumption of component count as the primary cost factor.

$$C = (2e + 1) + \frac{N'}{N}(2e + 1)3^{e-1} \quad . \tag{II-15}$$

Also the probability of voter failure can be expressed as

$$q_v = 3^{e-1}q \quad . \tag{II-16}$$

It will be convenient to express all failure probabilities in terms of the failure probability of the nonredundant network, $Q_0 = 1 - P_0$. Thus the probability of a circuit block operating is given by

$$p = (1 - Q_0)^{1/N} \quad .$$

Then the expression for network failure probability becomes

$$Q_r = 1 - \left[1 - \sum_{j=e+1}^{2e+1} \binom{2e+1}{j} \left\{(1 - Q_0)^{1/N'} 3^{e-1} \left[1 - (1 - Q_0)^{1/N'}\right]\right\}^{2e+1-j} \times \left(1 - \left\{1 - (1 - Q_0)^{1/N'} 3^{e-1} \left[1 - (1 - Q_0)^{1/N'}\right]\right\}\right)^{j}\right]^{N'} \quad . \tag{II-17}$$

Figure II-A-4 shows plots of the ratio of redundant failure probability and nonredundant failure probability, Q_R/Q_0, as a function of the cost ratio and the nonredundant-network failure probability. The curves provide an indication of the cost of achieving failure-probability improvements corresponding to $Q_R/Q_0 = 10^{-1}$, 10^{-2}, 10^{-3}, 10^{-4}. The discontinuities in the curves for the latter three values of Q_R/Q_0 indicate where the particular reliability improvement can be achieved by changing the order of replication.

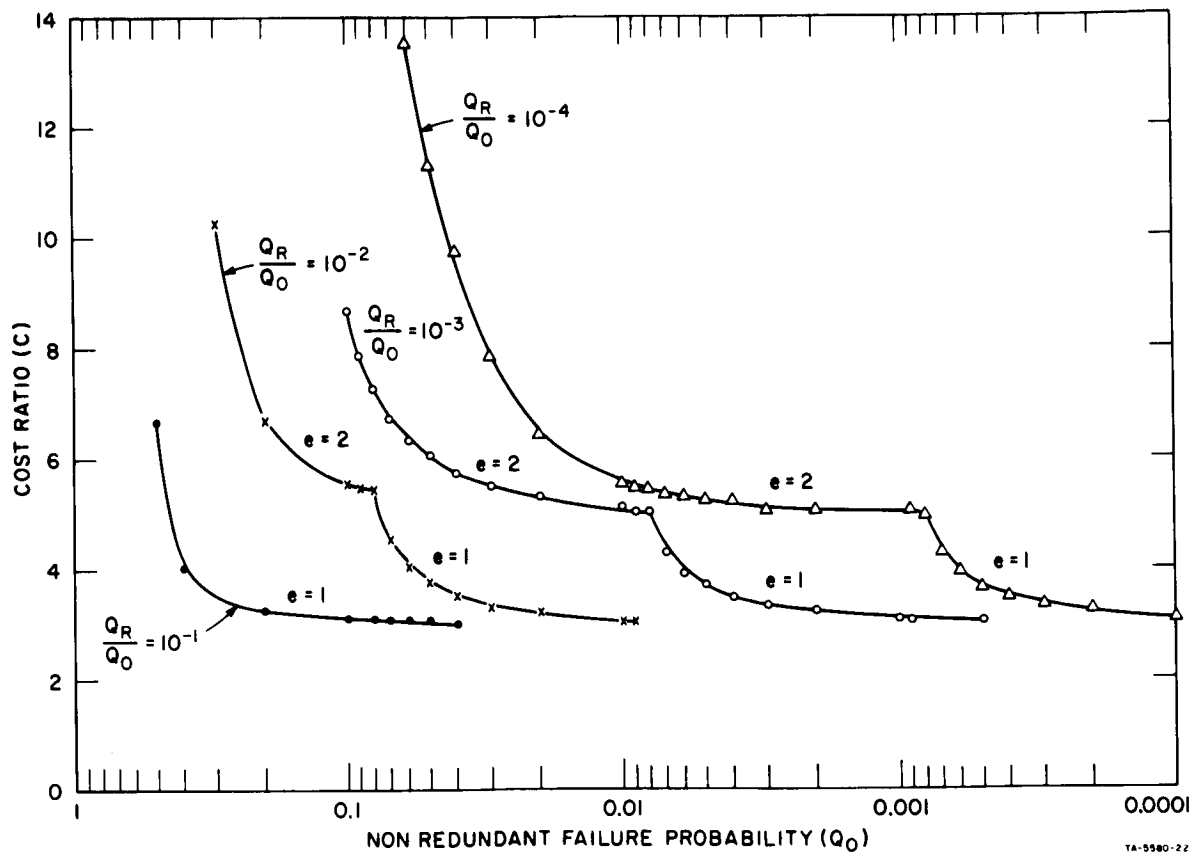


FIG. II-A-4 RELIABILITY IMPROVEMENT AS A FUNCTION OF COST FOR CASCADE MODEL

The preceding discussion is concerned with the effect of voting-type redundancy in that model wherein a computer network is visualized as a cascade of identical single-input, single-output circuit blocks. The analytical techniques employed therein can be directly

extended to a case where a uniformly converging tree is realized from a basic primitive fan-in circuit block. A redundant fan-in stage with f inputs and one output is shown in Fig. II-A-5. As with the cascade stage of Fig. II-A-2, it is convenient to visualize the set of input voters as associated with the stage. We find that the probability of the fan-in stage not operating is approximated by

$$q^{(i)} \approx \binom{2e+1}{e+1}(q + fq_v)^{e+1} \quad , \text{ for } q,\ q_v \ll 1 \quad . \qquad \text{(II-18)}$$

The error term is evaluated from Eq. (II-9) by replacing q_v with fq_v. A uniform tree is easily formed from the fan-in stages; in this case voters are placed on f input lines to each circuit block. For example, a 4-level tree is shown in Fig. II-A-6, where the dotted lines indicate the division of stages (of which there are 15). The network failure probability can be approximated by multiplying the number of stages formed by the value of $q^{(i)}$ derived from Eq. (II-18). If, in the uniform tree, voters do not appear on the inputs to each circuit block--as shown also in Fig. II-A-6 where voters appear every other level to form 5 stages (the stages are enclosed in dashed lines)--then the application of the stage failure probability expression, Eq. (II-18), must be altered. This is because for the case considered here each stage encloses three circuit blocks* and the fan-in to each stage is increased to 4. Thus for each of the "dashed" stages of Fig. II-A-6 the probability of failure can be expressed as

$$q^{(i)} \approx \binom{2e+1}{e+1}(3q + 4q_v)^{e+1} \quad . \qquad \text{(II-19)}$$

The relative values of q and q_v which minimize the failure probability of a tree can be easily determined in a manner similar to that used for the cascade. Curves for reliability improvement as a function of cost can also be easily derived for the tree model. The

---

* Clearly deleted from the stage are the two voters connected to the output circuit block.

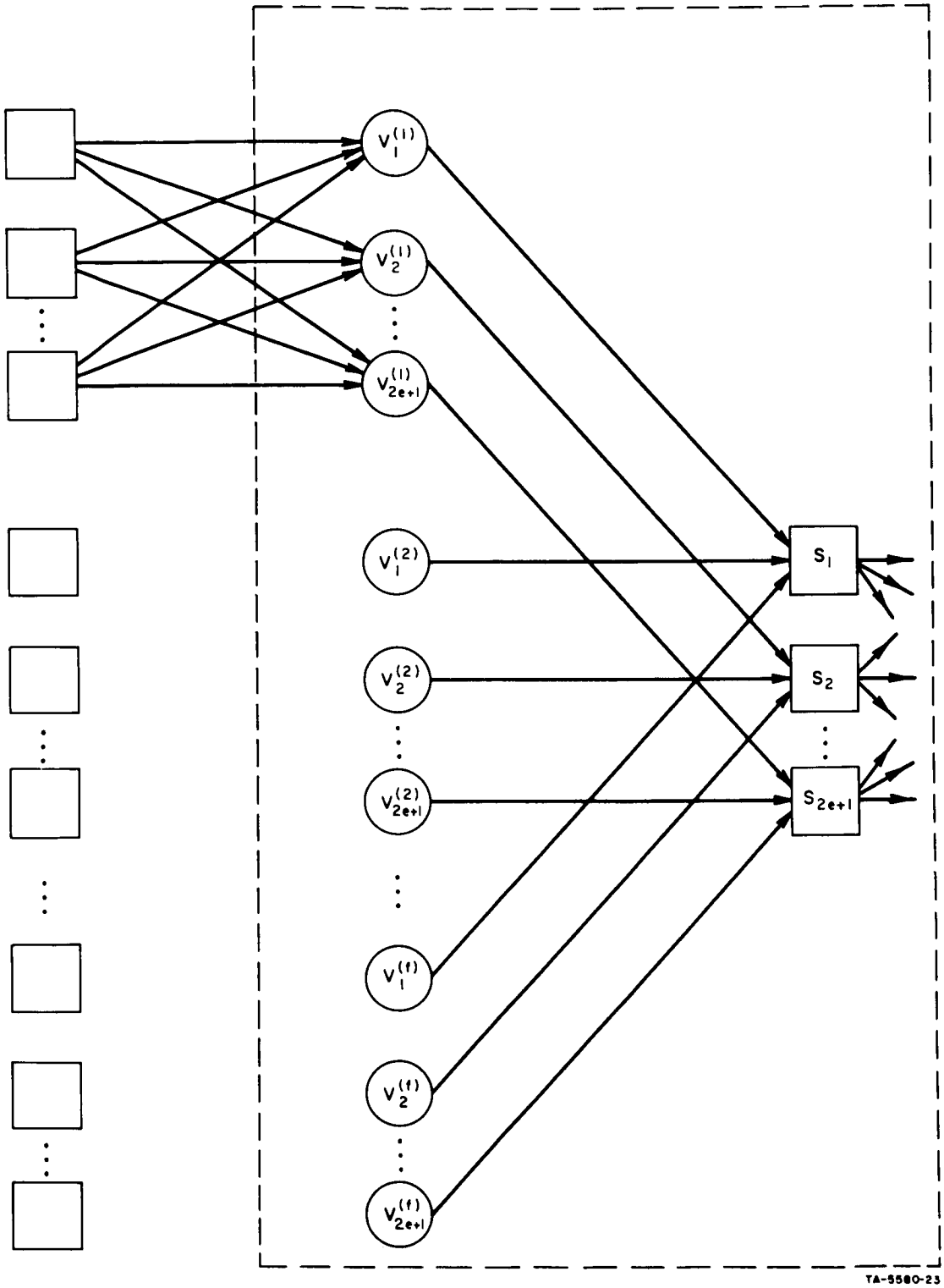


FIG. II-A-5 FAN-IN STAGE FOR TREE NETWORK

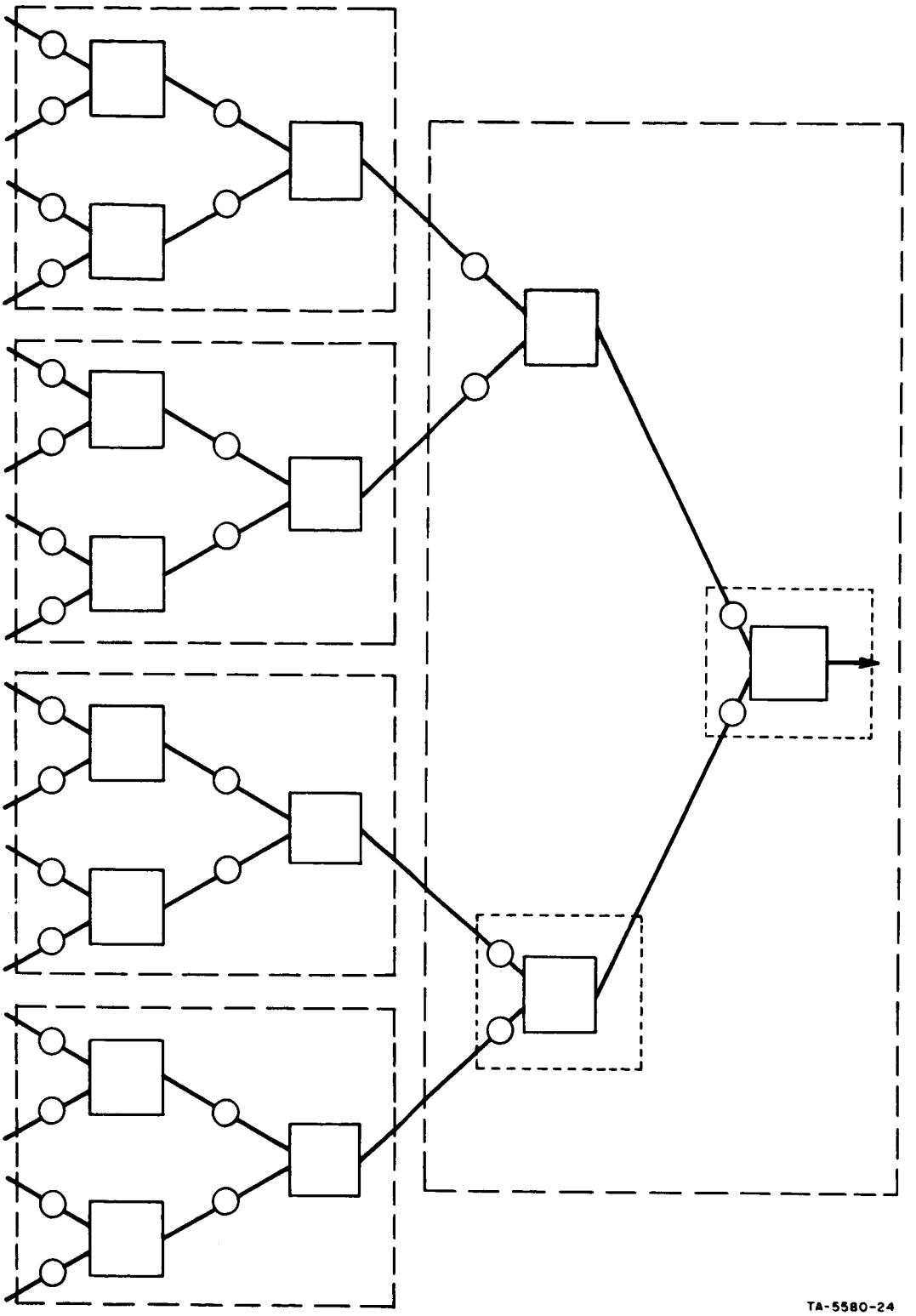


FIG. II-A-6 STAGE PARTITIONING OF UNIFORM TREE

results for the tree case are not presented here since it is felt that the corresponding results for the cascade model provide a sufficient qualitative measure of the utility of the restoration technique for a given network size. In the following section techniques are discussed for the analysis of specific arbitrary networks and the utility of the simple cascade model is discussed.

3) Techniques for the Analysis of Arbitrary Triplicated Networks

In the preceding section it was shown that the probability of failure of a cascade-type restored stage (Fig. II-A-2) could be approximated by determining the number of sets of e + 1 failures, e + 2 failures, etc., which resulted in stage failure. A similar analysis of determining the occurrence of the "most-likely" failures will be used to evaluate the failure probability of arbitrary networks, and the analytical method will be illustrated with an example. Consider the "arbitrary" triplicated* stage of Fig. II-A-7, where it is assumed that the failure probability of each voter is q_v and the failure probability of each circuit block S_{ij}, $i = 1, \ldots, 4$, $j = 1, \ldots, 3$ is q. (This latter assumption concerning the equality of circuit-block failure probabilities is based upon our supposition that logical designers will probably attempt to employ large sets of nearly identical blocks in designing combinational and sequential nets. If this assumption is not valid in certain applications, then the analytical method to be discussed is still applicable in theory, but the "bookkeeping" operations become somewhat involved.)

As a first step in the analysis of this stage let us count the number of occurrences of two failures which result in stage failure, where the stage (of Fig. II-A-7) is defined as operating correctly if and only if at least two of the outputs designated A and at least two of

---

* Unless otherwise stated we will, in the remainder of this section, consider only triple-order replication; but it should be noted that the results reported herein can be extended to higher-order replication values.

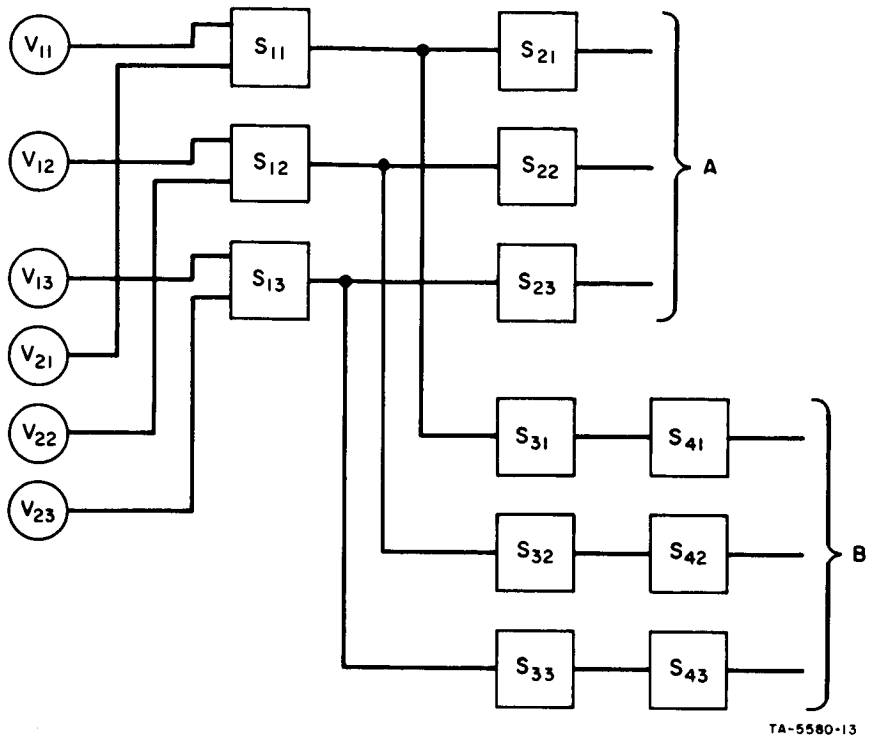


FIG. II-A-7 "ARBITRARY" TRIPLICATED STAGE

the outputs designated B are correct. For example, failures of the elements V_{11} and S_{42} will result in two errors in the output B (and hence will result in stage failure), while failures of the elements S_{21} and S_{42} will not result in stage failure. In the terminology of Ref. 141 we say that groups V_1 and S_4* are linked† and groups S_2 and S_4 not linked, where two groups are defined as linked if two failures, one occurring in each group, can result in stage failure. It is noted that

---

* Only a single subscript is used if the purpose is to identify a group of 3 replicated elements (voters or circuit blocks).

† The analytical approach based on linked groups is somewhat related to previous investigations (Refs. 140, 141), but these prior studies give accurate results only when the probability of occurrence of more than two failures is minimal or when the network structure closely approximates either of the aforementioned simple models. We have modified the procedure so as to account for the occurrence of more than two failures.

all groups are trivially linked to themselves. In the consideration of the linked groups S_i and $S_{i'}$*, $i \neq i'$, it is easily shown that there are 6 sets of double failures (one failure occurring in each group) which can result in stage failure. In the consideration of the trivially linked groups S_i and S_i it is verified that there are 3 sets of double failures which result in stage failure. If two groups are not linked then there are no double failures covering these 2 groups which can result in stage failure.

The "bookkeeping" procedure for the double-failure patterns is facilitated by a table such as Table II-A-1, relating to the stage of Fig. II-A-7 and reflecting the linked groups in the stage. An entry in

Table II-A-1

DOUBLE FAILURE PATTERNS BY LINKING PROCEDURE

		V_1	V_2	S_1	S_2	S_3	S_4	
Region I	V_1	3	3	3	3	3	3	Region II
	V_2	3	3	3	3	3	3	
	S_1	3	3	3	3	3	3	
	S_2	3	3	3	3	0	0	
Region II	S_3	3	3	3	0	3	0	Region III
	S_4	3	3	3	0	0	3	

the table is either 3 or 0, according to whether the groups corresponding to the respective row and column headings are linked or not linked. The table is partitioned into three regions, each reflecting the element types in which the failures have occurred. Region I relates to the occurrence of failures exclusively in voters; region II relates to the occurrence of

* We could have equivalently selected the linked groups V_i and $V_{i'}$ or the groups S_i and V_i.

one voter failure and one circuit-block failure; and region III relates to the occurrence of failures exclusively in the circuit blocks. The probability of stage failure resulting from the occurrence of two element failures is then derived from the double-failure-pattern table as

$$\begin{aligned}\text{Prob. of stage failure from 2 element failures} = \\ (\text{Sum of entries in Region I})\left[q_v^2(1-q_v)^{N_v^{(r)}-2}(1-q)^{N_s^{(r)}}\right] \\ + (\text{Sum of entries in Region II})\left[q_v q(1-q_v)^{N_v^{(r)}-1}(1-q)^{N_s^{(r)}-1}\right] \\ + (\text{Sum of entries in Region III})\left[q^2(1-q_v)^{N_v^{(r)}}(1-q)^{N_s^{(r)}-2}\right]\end{aligned} \tag{II-20}$$

where $N_v^{(r)}$ and $N_s^{(r)}$ are the number of voters and the number of circuit blocks (replicated), respectively, in the stage. It will be convenient to approximate Eq. (II-20) as

$$\begin{aligned}\text{Prob. of stage failure from 2 element failures} = \\ f_2(q,q_v) + f_3(q,q_v)\end{aligned} \tag{II-21}$$

where $f_2(q,q_v)$ and $f_3(q,q_v)$ are respectively the terms in the expansion of Eq. (II-20) in which the sum of the exponents of q and q_v are 2 and 3. We can visualize $f_2(q,q_v)$ as representing an approximation to the failure probability and $f_3(q,q_v)$ as representing a contribution to the dominant error term.

Hence for the stage being considered the expression for the probability of failure becomes

$$\begin{aligned}&12q_v^2(1-q_v)^4(1-q)^{12} + 24qq_v(1-q_v)^5(1-q)^{11} + 30q^2(1-q_v)^6(1-q)^{10} \\ &\approx (12q_v^2 + 24qq_v + 30q^2) - (48q_v^3 + 264q_v^2q + 444q_vq^2 + 300q^3) \quad .\end{aligned}$$

It is of interest to consider the effect of error patterns of weight greater than 2 on the failure probability, to augment the error term $f_3(q,q_v)$. We have developed a systematic procedure for determining the contribution to the failure probability from triple element failures, and this procedure can be generalized for the consideration of an arbitrary number of element failures. As might be expected, the complexity of the bookkeeping operations increases as additional failure combinations are considered.

In the consideration of triple-element failures, six cases completely cover all combinations of group linkings of circuit blocks (or voters). For example, if we are concerned with the number of triple failure patterns of the groups S_i, $S_{i'}$, $S_{i''}$, (one failure occurring in each group) which will result in stage failure, then the counting procedure simply reflects the manner in which the three groups are linked, and also the equality of i, i′ and i″. As a specific case consider (for the stage of Fig. II-A-7) the 3 failures as occurring in the groups S_1, S_2, and S_3. In this case one group (S_1) is linked to each of the other groups, and also $i \neq i'$, $i \neq i''$, $i' \neq i'$, where $i = 1$, $i' = 2$, $i'' = 3$. There are 27 triple error patterns covering the 3 groups, and 24 of these--all except the error patterns $S_{1j}\ S_{2j}\ S_{3j}$, $j = 1, 2, 3$--will result in stage failure. The six cases are summarized below.

Case a: $i = i' = i''$--1 failure pattern

Case b: $i = i' \neq i''$; $S_{i'}$ and $S_{i''}$ not linked--9 failure patterns

Case c: $i = i' \neq i''$; S'_i and S''_i linked--9 failure patterns

Case d: $i \neq i'$, $i \neq i''$, $i' \neq i''$; all group pairs not linked--0 failure patterns

Case e: $i \neq i'$, $i \neq$ ″, $i' \neq i''$; S_i and S'_i linked, remaining two group pairs not linked--18 failure patterns

Case f: $i = i'$, $i \neq i''$, $i' \neq i''$, S_i linked to S'_i and S''_i--24 failure patterns.

As was the case with the double failure patterns, the failure probability calculation resulting from the consideration of triple

error patterns is greatly facilitated with the use of a failure-pattern table, as illustrated for the stage of Fig. II-A-7 by Table II-A-2. The row headings, as in the table for the double failure patterns, are a listing of the N_v voter groups followed by the N_s circuit-block groups. However, the column headings are a listing of the $\binom{N_v + N_s}{2} + N_v + N_s$ combinations of the groups taken two at a time. The entries in the table relate to the number of failure patterns corresponding to each of the 6 above cases, where the particular case is distinguished by the corresponding row-column headings. For example, the entry corresponding to row S_3 and column S_2S_4 is defined by Case e, in that S_3 and S_4 are linked, and the resultant entry is 6.*

The table is partitioned into 4 regions, each reflecting the element types in which the failures have occurred. Then the contribution to the stage failure probability from the occurrence of triple failures is given by the following, where only the lowest-order terms are retained.

$$\begin{aligned}
&\text{Prob. of stage failure from triple failures} \approx \\
&(\text{sum of entries in Region I})q_v^3 \\
&+ (\text{sum of entries in Region II})q_v^2 q \\
&+ (\text{sum of entries in Region III})q_v q^2 \\
&+ (\text{sum of entries in Region IV})q^3 \quad . \qquad \text{(II-22)}
\end{aligned}$$

Combining the results of Eqs. (II-21) and (II-22) we can then represent the failure probability of a stage by the term $f_2(q,q_v)$

---

* The entries corresponding to Cases d, e, and f are 0, 6, and 8 respectively or one-third of the number of failure patterns listed for these cases, since there are 3 locations in the table in which entries appear for S_i, S_i', S_i'', $i \neq i'$, $i \neq i''$, $i' \neq i''$. Those entries corresponding to Cases b and c are 9/2 or one-half of the number of failure patterns, since there are 2 locations where $i = i' \neq i''$. The entry corresponding to Case a is 1.

Table II-A-2

TRIPLE FAILURE PATTERNS BY LINKING PROCEDURES

Region I

	V_1V_1	V_1V_2	V_2V_2	V_1S_1	V_1S_2	V_1S_3	V_1S_4	V_2S_1	V_2S_2	V_2S_3	V_2S_4	S_1S_1	S_1S_2	S_1S_3	S_1S_4	S_2S_2	S_2S_3	S_2S_4	S_3S_3	S_3S_4	S_4S_4
V_1	1	9/2	9/2	9/2	9/2	9/2	9/2	8	8	8	8	9/2	8	8	8	9/2	8	8	9/2	8	9/2
V_2	9/2	9/2	1	8	8	8	8	9/2	9/2	9/2	9/2	9/2	8	8	8	9/2	8	8	9/2	8	9/2
S_1	9/2	8	9/2	9/2	8	8	8	9/2	8	8	8	1	9/2	9/2	9/2	9/2	8	8	9/2	8	9/2
S_2	9/2	8	9/2	8	9/2	8	8	8	9/2	8	8	9/2	9/2	8	8	1	9/2	9/2	9/2	6	9/2
S_3	9/2	8	9/2	8	8	9/2	8	8	8	9/2	8	9/2	8	9/2	8	9/2	9/2	6	1	9/2	9/2
S_4	9/2	8	9/2	8	8	8	9/2	8	8	8	9/2	9/2	8	8	9/2	9/2	6	9/2	9/2	9/2	1

Region II

Region III

Region IV

plus an error term given by the sum of $f_3(q,q_v)$ and the result of Eq. (II-22) above. It appears that the error term (for triple replication) is always negative, but no proof of this conjecture has as yet been formulated. It also appears, for $q \approx q_v$, that the ratio of the error term to $f_2(q,q_v)$ is on the order of the product of qN_s.

For the stage we have been considering, it is easily verified that the error term is

$$20q_v^3 + 96q_v^2q + 84q_vq^2 + 98q^3 \quad .$$

Prior to this point in the discussion of arbitrary restored networks we have been concerned with finding the failure probability of an artefact unit called a "stage." It is noted, by referring to the stage of Fig. II-A-7, that voters only appear on the input lines. However, the techniques illustrated for the stage can be applied to any arbitrary restored network with arbitrary placement of voters. For arbitrary networks, the failure-pattern tables for double and triple failures are applicable, but appropriate row and column headings must be present for each group of circuit blocks or voters. In the compilation of this table we note that two groups (circuit blocks or voters) are linked if there exists at least one path (in the direction of normal signal flow) connecting the output of either group with the output of the other group which does not include a voter, or if there are two paths, one emanating from the output of each group, which intersect at a circuit block without including a voter.* For example, referring to the network of Fig. II-A-8 (in simplified representation), it is seen that V_3 and S_4 are linked and S_2 and V_4 are linked, but S_1 and S_2 are not linked.

---

* It is easily verified that this property of linked groups is consistent with the definition presented previously in this section in terms of the effect of failures in the respective groups. It is also noted that a circuit-block group followed by a voter group is not linked to it (in the absence of feedback) since there is a voter in the path connecting the outputs.

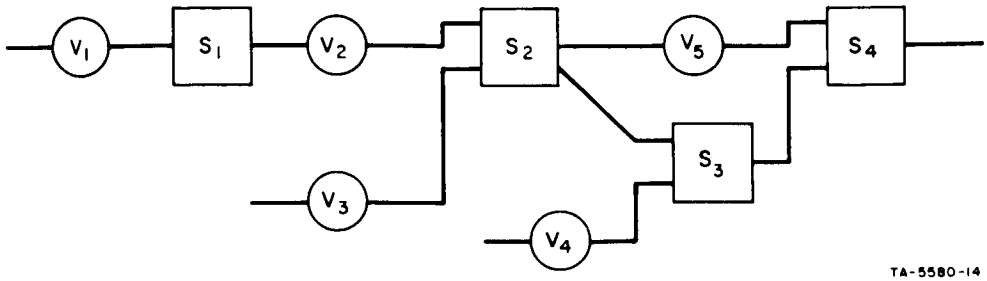


FIG. II-A-8 NETWORK TO ILLUSTRATE LINKING

The approximation to the failure probability of a network as the probability of only two or three linked-circuit-block failures is only valid if the product of the failure probability of a circuit block and the number of circuit blocks in the nonreplicated network is much less than unity.* If this assumption is not valid (although for most networks in a spaceborne computer the assumption indeed appears to be satisfied) then an accurate analysis can be performed by visualizing the network as a composition of stages and utilizing the analytical techniques discussed in order to determine the reliability of each stage.

In Sec. II-A-2-a-2) an ad hoc definition of a stage was given. At this time we can formulate the definition in more precise terms relating to linked groups.

> Definition: A stage of an arbitrary restored network is a collection of circuit-block and voter groups with voters only on external input lines and also with the restriction that no group within the stage is linked to a group external to the stage.

The probability of the network operating correctly is then given by the product of the operating probabilities of the respective stages, where the expression for the probability of a particular stage operating is derived by the methods of this section.†

---

* An illustrative example is discussed in Sec. II-A-2-a-4).

† The circuit-block failure probabilities will probably be low enough to permit the evaluation of the stage operating probability in terms of the probability of 2 or 3 element failures.

It has been noted previously[272] that for certain network topologies there is some difficulty attendant to the problem of revealing the stages. As an example, consider the restored network of Fig. II-A-9.

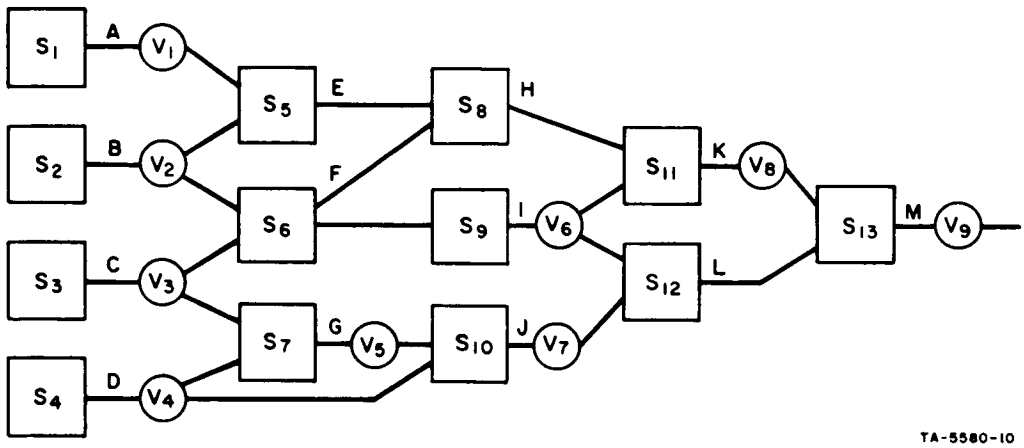


FIG. II-A-9 ARBITRARY NETWORK TO BE SUBDIVIDED INTO STAGES

The subdivision of the network into stages is not possible with the present connection. However, the subdivision can be performed by considering the voter V_6 as two voters V_6' and V_6'' appearing on the appropriate input lines of circuit blocks S_{11} and S_{12} respectively, and also considering the voter V_4 as two voters V_4' and V_4'' appearing on the appropriate lines of circuit blocks S_7 and S_{10}. Three stages, as shown in Fig. II-A-10, can then be formed; it can be shown that the failure probability of the original network is slightly lower than the failure probability of the modified network with the additional voters. Thus, in the analysis, the error attendant to the modification is in the direction of a conservative estimate of reliability.

In the present section techniques have been given for the analysis of arbitrary restored networks. In many applications it is important to estimate the reliability of a given network with an assumed number of available voters, without going through a lengthy analysis. A method for the evaluation of upper and lower bounds on network failure probability is discussed in the following section.

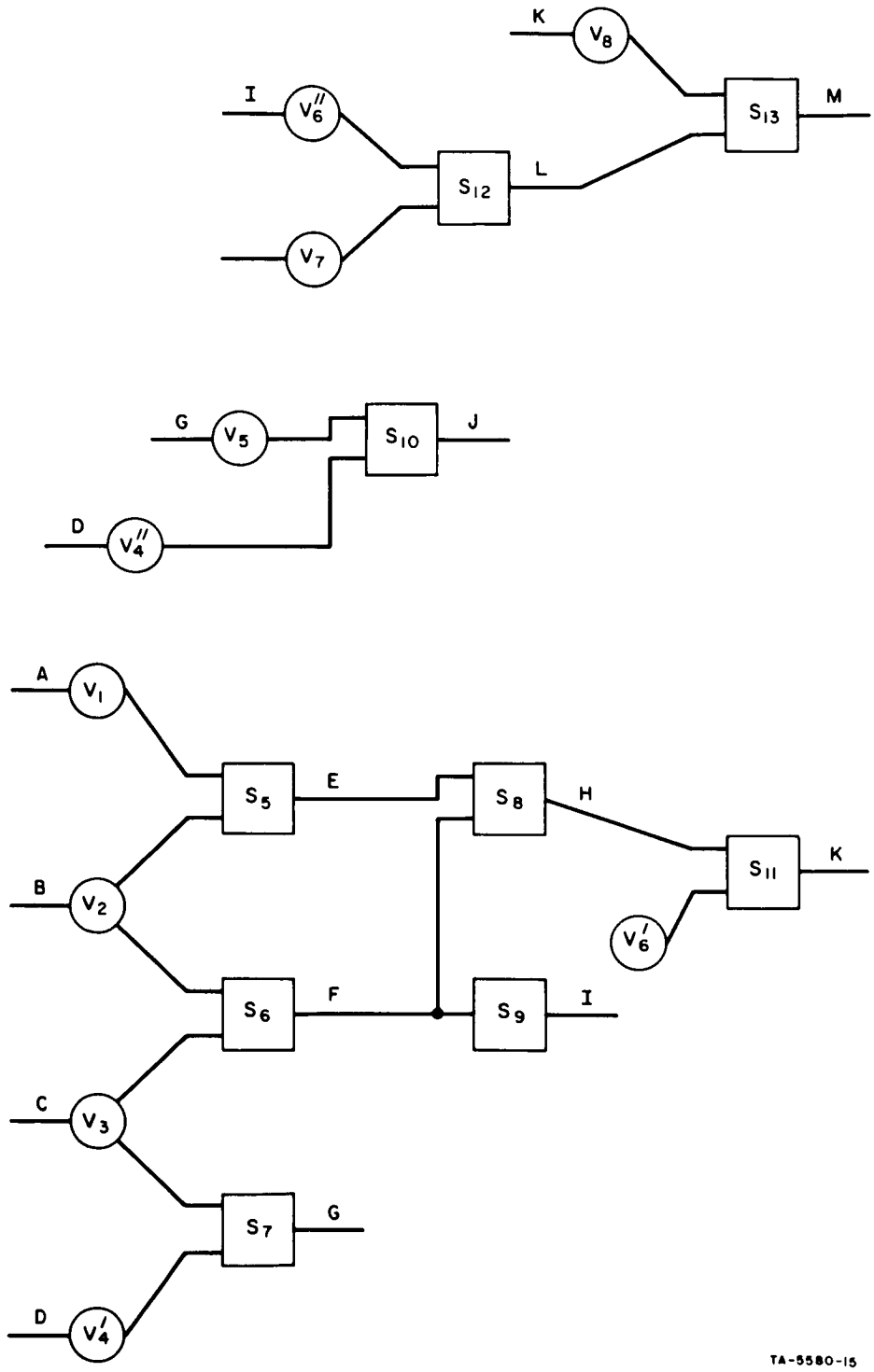


FIG. II-A-10 STAGE SUBDIVISION OF NETWORK

4) Bounds on Network-Failure Probability

In this section we will present upper and lower bounds on the failure probability which can be expected from triplicated restored networks of a given "size" with a given density of voters. Our major motivations for considering this problem are to discover the relevance of describing the performance of arbitrary networks in terms of the simple cascade model, and also to present easily applied methods for evaluating a rough measure of the reliability of redundant networks. It is shown below that visualizing a complex network as a simple cascade gives an optimistic measure of the reliability.

In considering the derivation of upper and lower bounds on the expected failure probability of networks we have assumed the following constraints, in addition, of course to the assumptions discussed in Sec. II-A-2-a-1).

(a) The (nonredundant) network is composed entirely of one primitive element type; in the analysis to follow we assume the primitive element to be a 3-input, single-output circuit block where all possible interconnections are permitted (provided the fan-in and fan-out do not exceed 3). The assumption of this primitive element type is not restrictive, and the results can be readily generalized.

(b) The voters are dispersed throughout the network so that the overall failure probability is close to minimal. It is not known how to optimally place the voters in an arbitrary network, so we have operated on the premise that the restored network will be close to optimal if the voters are placed so that the maximum number of circuit blocks traversed on any path between voters is minimized. (For regularly structured networks this placement criterion appears to be optimal, but it is not difficult to visualize topologies for which the failure probability is not minimized by such a technique.) In order to facilitate the calculations we assume that the maximum number of circuit blocks on nonrestored paths could assume, for different cases studied, the values 1, 2,.... This latter assumption enables the determination of bounds on the failure probability for different numbers of available voters.

(c) Voters are placed on the output lines of circuit blocks.

(d) The circuit-block and voter failure probabilities are low enough that the analytical techniques of the previous section, based upon only double element failures resulting in network failure, are applicable.

First we will consider the determination of the greatest lower bound on the failure probability, $Q_R^{(L)}$. Assume that we have a network composed of interconnections of N_s 3-input single-output circuit blocks, and that voters appear following every N_s/N' blocks.* Hence for triple replication $3N'$ voters are required. We recall that the failure probability is minimized by minimizing the number of linked circuit blocks and voters. Since there are at least N_s/N' circuit blocks between voters, the failure probability is minimized if each circuit block is linked to a total of N_s/N' circuit blocks, each voter is linked to N_s/N' circuit blocks, and each voter is only linked to itself. The network which satisfies these linking conditions is the simple cascade shown in Fig. II-A-11; this cascade is admittedly artificial since all of the inputs to a circuit block originate at the same source.

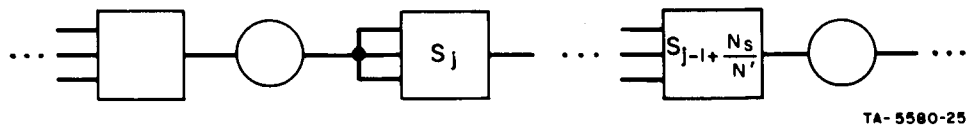


FIG. II-A-11 CASCADE NETWORK WHICH MINIMIZES FAILURE PROBABILITY

The failure probability of the network is easily determined as

$$Q_R^{(L)} = 3N'\left(q_v + \frac{N_s}{N'} q\right)^2 . \qquad \text{(II-23)}$$

Now consider the upper bound on failure probability which is realized for a topology for which the number of circuit-block and

* It will be assumed that N_s and N_s' are such that $N' | N_s$.

voter groups which are linked is maximized. Consider a 3-input, single-output circuit block whose output is connected to a voter. Distinguish this circuit block as a first-order block. Since there is a voter on the output of the first-order block, only self-linkings and linkings by means of the 3 inputs are possible. Assume that the sources of these three inputs, called second-order blocks, are all distinct. In turn, each of these second-order blocks is a sink for 3 third-order blocks, etc. (Since in the network as visualized the number of circuit blocks between voters does not exceed N_s/N', the maximum "order" of a block is N_s/N'.) It is then seen that the maximum number $\eta_{\alpha\beta}$ of α-order blocks with which a β-order block is linked is given by

$$\eta_{\alpha\beta} = 3^{[\max(\alpha,\beta)-1]} \quad . \tag{II-24}$$

The maximum number of γ-order blocks which can be linked to a γ-order block is

$$\eta_{\gamma} = 3^{\gamma-1} \quad . \tag{II-25}$$

In a network with N' voters we can treat the voters as equivalent, on the basis of the calculation of linkings, to $(N_s/N' + 1)$-order blocks. A portion of the network which exhibits the maximum possible linkings among the various circuit blocks and voters, for $N_s/N' = 2$, is shown in Fig. II-A-12.* It is seen that all 9 voters are linked and also that each voter is linked to each of the 18 circuit blocks. Each second-order circuit block is linked to 3 second-order circuit blocks (including itself), and is also linked to 3 first-order blocks. Each first-order block is linked to itself, and also to 3 second-order blocks.

---

* The overall network, which is a cascade of $N' \Big/ \left(3^{N_s/N'}\right)$ stages of the type shown in the figure, contains $3^{N_s/N'}$ inputs and $3^{N_s/N'}$ outputs.

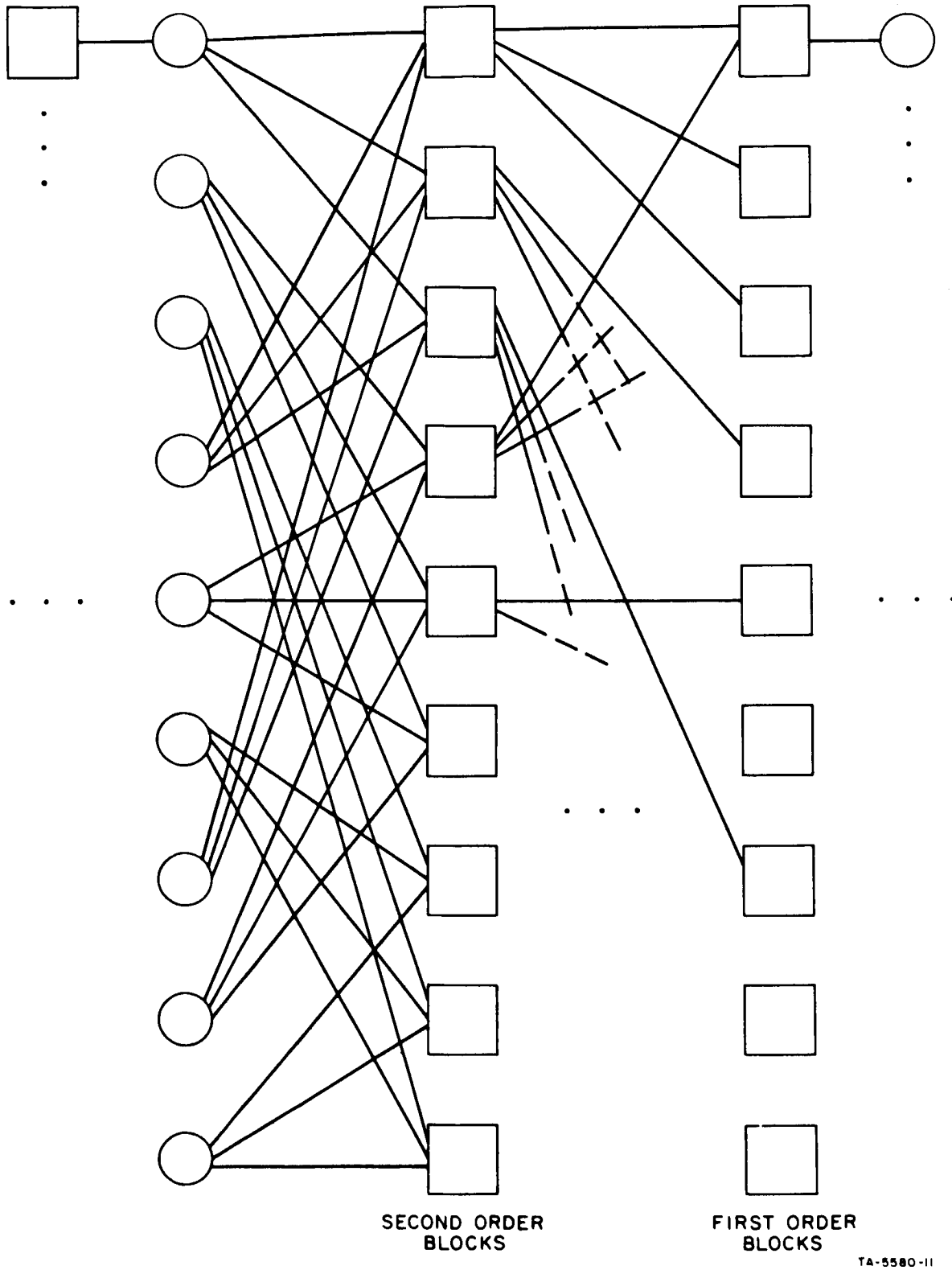


FIG. II-A-12 NETWORK WHICH MAXIMIZES FAILURE PROBABILITY

The expression for the upper bound on the failure probability, $Q_R^{(u)}$, for arbitrary N_s and N', is derived as follows.

Consider, first, the contribution to the failure probability from the linking of circuit blocks. A block in order i is [from Eq. (II-24)] linked to a maximum of 3^{i-1} blocks, contained in orders 1, 2, ..., i - 1. This block in order i is also linked to 3^{j-1} blocks contained in orders for which j satisfies j = i, i + 1, ..., N_s/N'. Thus the total number of circuit blocks to which each block in order i is linked is given by

$$(i - 1)3^{i-1} + \sum_{j=1}^{N_s/N'} 3^{j-1} \quad . \tag{II-26}$$

However we note that, assuming an equal number of blocks of each order in the network, there are a maximum of N' blocks of order i; and also, from Eq. (II-20) the contribution to the failure probability from each linking between circuit blocks is $3q^2$. Thus the contribution to the network failure probability from circuit-block linkings is given by

$$3N'q^2 \sum_{i=1}^{N_s/N'} \left[(i - 1)3^{i-1} + \sum_{j=1}^{N_s/N'} 3^{j-1}\right] \quad . \tag{II-27}$$

Each voter is linked to a maximum of $3^{N_s/N'}$ voters, and since there are N' voters in the overall network, the contribution to the failure probability from voter linkings is given by

$$3N'q_v^2 3^{N_s/N'} \quad . \tag{II-28}$$

Similarly, each of the N' voters is linked to $(N_s/N')3^{N_s/N'}$ circuit blocks (and of course each of the N_s circuit blocks is linked to $3^{N_s/N'}$ voters), thus indicating that the contribution to the failure probability from voter-circuit block linkings is given by

$$3 \times 2N_s q q_v 3^{N_s/N'} \quad . \tag{II-29}$$

Simplifying Eq. (II-27) and summing the resultant expression with Eqs. (II-28) and (II-29) we find that $Q_R^{(u)}$ becomes

$$Q_R^{(u)} = 3N'q^2\left[\left(N_s/N'\right)3^{N_s/N'} + 1 - 3^{N_s/N'}\right] + 3N'q_v^2 3^{N_s/N'} + 6N_s q q_v 3^{N_s/N'} \quad . \qquad \text{(II-30)}$$

It is of interest to compare the lower and upper bounds as specified by Eqs. (II-23) and (II-30). We can visualize the two networks which satisfy the lower and upper bounds as each containing N_s circuit-block groups and N' and $N'/3^{N_s/N'}$ stages respectively. A meaningful comparison is achieved by tabulating as a function of N_s/N' (the number of circuit blocks between voters) the quantities

$$\frac{Q_R^{(L)}}{(3N'q^2)} \text{ and } \frac{Q_R^{(u)}}{(3N'q^2)} \quad .$$

These two quantities are a measure of the failure probability per voter group divided by q^2 (assuming $q \approx q_v$). These values are shown in Table II-A-3; it is not meaningful, in the context of the two networks, to show smooth plots connecting these points.

Table II-A-3

MEASURE OF LOWER AND UPPER BOUNDS ON FAILURE PROBABILITY

N_s/N'	$Q_R^{(L)}/3N'q^2$	$Q_R^{(u)}/3N'q^2$
1	4	10
2	9	55
3	16	243
4	25	963
5	36	3646

It should be noted that the upper bound is somewhat pessimistic since it is unlikely that circuit connections of the type shown in Fig. II-A-12 would occur in practical networks. However, these studies do indicate that care should be exercised in applying the reliability measures obtained from consideration of the simple cascade model, since these results tend to be optimistic.* If an accurate measure of the performance of a network is required, then it appears that a complete analysis must be performed. It is recommended that consideration be given to the development of computer-aided techniques for the rapid analysis of arbitrary restored networks; for systems with fairly high initial reliability the simple analysis technique based on linked elements and described in this report can be applied.

5) Techniques for the Realization of Multiple-Output Networks with Voter Redundancy for Fault Masking

In this section we briefly discuss a problem which apparently has not been previously considered. It was noted previously [Sec. II-A-1-b-1)] that the output functions in a multiple-output network must be realized independently in order to apply parity-check codes for the parallel checking of the outputs. This independent realization condition is required so that the number of outputs in error does not exceed the number of failed components in the network.

A related question arises in the consideration of multiple-output networks to which are applied the conventional restoration techniques, discussed in this report. The question is whether, if the outputs of a multiple network are each protected by voting-type redundancy (where the voters are placed only at the network inputs) the output functions should each be realized independently so as to minimize the failure probability.

---

* The results of the cascade can be applied if the density of voters is high--say one voter for every 2 or 3 circuit blocks--in which case the difference between the upper and lower bounds is probably less than the inaccuracies attendant to the determination of component reliabilities.

At present, we do not know the answer to this question for all function types, although we conjecture that the minimum failure probability is achieved if the function is realized minimally, regardless of the dependency of the outputs. An example illustrating this conjecture for triple replication is presented below, wherein the failure probabilities of two realizations of a serial decoding circuit are compared.

A serial decoder for 2^m signal lines can be realized as an m-level tree network, of the type shown in Fig. II-A-13, composed of simple, single-input, double-output sequential decision blocks. It is assumed that there are 3 replicas of the entire network, and a single perfect* voter is employed for each of the 2^m outputs.

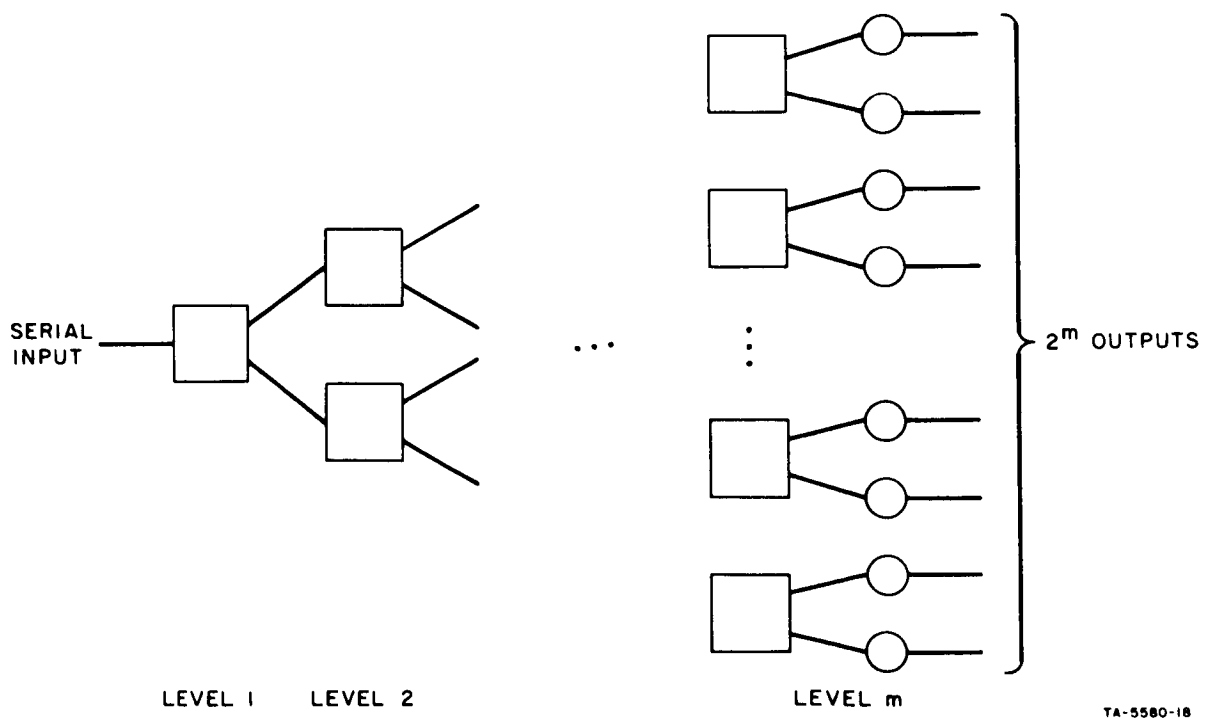


FIG. II-A-13 TREE REALIZATION OF SERIAL DECODER

* In order to simplify the analysis we have assumed, with no loss of generality, that the voters are perfectly reliable.

Another implementation of the decoder is shown in Fig. II-A-14, in which each of the 2^m outputs is realized independently by a cascade of single-input, single-output sequential decision blocks. We would expect that the sequential blocks of the tree realization and the cascade realization would be of comparable complexity, indicating that equal failure probabilities can be assigned to the blocks of both networks. Also, in the discussion to follow, we will employ the simplified analytical techniques based upon the assumption that the occurrence of more than two failures can be ignored.

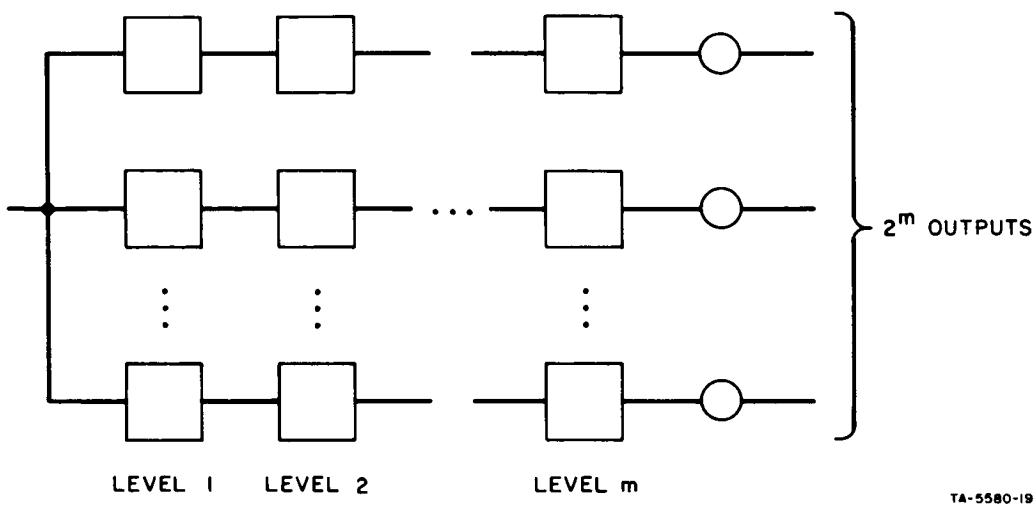


FIG. II-A-14 CASCADE REALIZATION OF SERIAL DECODER

We easily determine that the failure probability of $Q_R^{(c)}$ of the independent cascade connection is given by

$$Q_R^{(c)} = 3 \cdot 2^m (mq)^2 = 3 \cdot 2^m m^2 q^2 \quad . \qquad \text{(II-31)}$$

The analysis for the tree is as follows. The block in the first level is linked to all blocks in the network. Each block in level 2 is linked to one-half of the blocks in each of levels 2, 3, ..., m. Similarly each of the $2^{\alpha-1}$ blocks of level α is linked to $2^{\beta-\alpha}$ blocks in each of the levels $\beta = \alpha, \alpha + 1, \ldots, m$, and each is also linked to 1 block in each of the levels $1, 2, \ldots, \alpha - 1$.

Thus the expression for the failure probability of the tree, $Q_R^{(T)}$, can be reduced by application of Eq. (II-20) to

$$Q_R^{(T)} = 3q^2 \sum_{\alpha=1}^{m} 2^{\alpha-1}[(\alpha - 1) + \sum_{\beta=2}^{m} 2^{\beta-\alpha}] \quad . \qquad \text{(II-32)}$$

It can be verified that the above equation reduces to

$$Q_R^{(T)} = 3q^2[3 + 4m2^{m-1} - 3 \cdot 2^m] \quad . \qquad \text{(II-33)}$$

For the purpose of comparison the values of $Q_R^{(c)}$ and $Q_R^{(T)}$ for a few values of m are shown in Table II-A-4.

Table II-A-4

COMPARISON OF FAILURE PROBABILITIES
FOR CASCADE AND TREE REALIZATIONS

m	$Q_R^{(c)}$	$Q_R^{(T)}$
2	$48q^2$	$21q^2$
3	$216q^2$	$81q^2$
4	$768q^2$	$249q^2$
m >> 1	$6m^2 2^{m-1} q^2$	$6m2^m q^2$

It is seen that for large values of m the ratio $Q_R^{(c)}/Q_R^{(T)}$ is of the order of m/2; but for small values of m, (say ≤ 4) the difference between the reliability measures is probably negligible.

At first glance these results appear somewhat paradoxical since, in the application of parity-check codes for the masking of faults in multiple-output networks, a basic requirement is that all outputs must be realized independently. However, this basic independence tenet is not in fact violated. The code being applied here to each output is equivalent to the simple single-error correcting code which consists of one informa-

tion digit and two independently generated check digits for each output. This code as applied to the serial decoder in question (either realization) will mask single failures, and it is clear that the occurrence of any single fault cannot result in two errors on the three signal lines associated with any output.

6) Conclusions and Future Problems for Study

The ultimate aim of this study, as stated in Sec. II-A-2-a-1), was to indicate to a designer the expected improvement in reliability that could be anticipated from the application of the restoration technique, and also to provide algorithms for the optimum application of the technique. At this point in the research the aim has not been entirely achieved, but with the tools outlined in this report and in the many papers related to the subject, it is possible for a potential user of the restoration method to determine its application in most networks of interest by following systematic (though possibly lengthy) analytical procedures.

The major results of the study are as follows.

(a) Techniques are presented for the analysis of arbitrary restored networks, where the application of the analysis is subject to the constraint that a fairly simple failure model is pertinent. The analysis is facilitated if the original nonredundant network is reasonably reliable, although this is not a necessary requirement.

(b) A method is presented for determining upper and lower bounds on the failure probability of restored networks on the basis of the assumptions that the simple failure model is pertinent, that the network is composed of interconnections of a single primitive circuit block, and that the voters are placed optimally throughout the network.

(c) It is indicated that when the restoration technique is applied to multiple-output networks, a lower value of failure probability is probably achieved if each replica of the network is realized in a minimal manner--even though the outputs might be quite dependent--than if the network is realized with independent outputs.

A number of outstanding problems still remain, primarily related to the need for techniques by which restored networks which globally minimize the failure probability can be synthesized, subject to a maximum available redundancy. Some of the specific areas for recommended future research, related to this problem, are as follows:

(a) The development of techniques for determining for arbitrary networks the "form" of the restoration, so as to minimize failure probability. For example, if an overall redundancy of 5.5 is available, the question is whether the replication order should be 3 with a high density of voters, or 5 with a low density of voters.* In addition the expected reliability improvement realized by the application of several other forms of restoration should be quantitatively evaluated. These schemes include:

- Variable redundancy; i.e., replication order not constant throughout the network.
- Techniques wherein only a subset of the replicated circuit blocks are connected as inputs to each of the replicated voters--previous research in this area[232,243] has been concerned with qualitative descriptions.
- Generalized interwoven logic[243]--it is felt that the analysis will be extremely difficult, and at present the technique appears to be quite costly, especially when multiple failure correction is required.

(b) The development of easily applied techniques for determining the optimum placement of voters--initially for networks where the replication order is constant and each voter weighs all replicated signals equally. The placement techniques which have been described relate to the evaluation of failure probability for each set of voter positions either by simulation[75] (requiring approximately two hours of computer

---

* The results of Knox-Seith[164] which provided a solution to this problem for the cascade model, are discussed in Sec. II-A-2-a-2).

time for a network of 300 gates) or by systematic analysis.[141] It is felt that a much simpler technique can be developed, in particular when only double failures are considered in the analysis. We have conjectured that the following dynamic programming systematic procedure will always converge to the optimum placement.

- Consider an arbitrary initial placement of voters, and evaluate the failure probability. Move one voter in turn to each available position and for each position calculate the failure probability. If the failure probability with the voter in any new position is not lower than the initial failure probability, return the voter in question to the original position; otherwise place the voter in that position which provided the minimum failure probability. Repeat this operation, perturbing the position of each voter separately, until the failure probability is not further reduced. If this placement technique proves to be suboptimal, then it is recommended that consideration be given to determining for what network topologies a solution close to optimal is obtained. It is also of interest to determine, in particular for large networks, the probability that a random voter placement yields a solution close to optimal.

(c) The development of computer-aided techniques for the analysis of given restored networks, and also for the synthesis of optimum restored networks, possibly on the basis of the conjectured optimum dynamic programming approach described above. It appears that some LISP programming techniques designed for locating loops in a linear graph might be applicable for specifying the linked elements in a restored network.

(d) All of the analytical procedures developed for restored networks have been based upon a simple failure model* where it is assumed that failure of a component will not affect the state of any signals which appear as inputs to the failed

---

* See Sec. II-A-2-a-1) for a complete discussion of the assumptions attendant to the model.

component. This model is only consistent with networks in which there is some degree of isolation between gates. A more complex model based upon considering component failures as producing errors in both outputs and inputs, which has been considered in simulation studies,[75] can be examined by appropriately modifying the linking definition of Sec. II-A-2-a-3).

In describing the performance of complex networks, we have distinguished the network as either operating (correctly) or not operating, and then described means whereby the probability of the network not operating is minimized. However, the criterion of minimum failure probability is somewhat inconsistent with the tenet that a computer should function, although with possible loss in computation capability, as failures occur. (This point of view has been considered briefly before.[16]) Hence, for some multiple-output networks it is meaningful to assign a cost metric to the occurrence of each failure, and then allocate the redundancy so as to minimize the average "loss" in capability. In addition, it is meaningful to assign a probability measure to the occurrence of some members of the set of inputs. The inclusion of the cost function might alter the conclusion of Sec. II-A-2-a-5) relating to the optimum realization of multiple-output functions.

b. Techniques for the Combination of Fault Masking and Replacement

1) Introduction

In this section logical designs will be presented for three schemes of replacement-type redundancy. All of the schemes are autonomous in the diagnosis and repair of faults, and they also provide for a certain degree of fault masking during replacement. The autonomy and masking are provided by the employment of various forms of voting, so that the schemes might actually be considered as hybrids of voting and replacement redundancy.

An idealized model that encompasses all three schemes has been described and analyzed by Kruus.[171] It is hoped that the presentation of practical designs will enable designers of future complex systems to evaluate the proper system level for the application of these schemes.

The three schemes to be described may be characterized as follows:

(1) Adaptive voting (after Pierce[240])(Fig. II-A-15)--A basic functional network is replicated, and the outputs of the replicated units are combined in a variable-threshold network to provide a system output; if a unit dissents from the system output, it is disconnected, and the threshold is diminished so as to make the system output equal to the majority of the outputs of the remaining units.

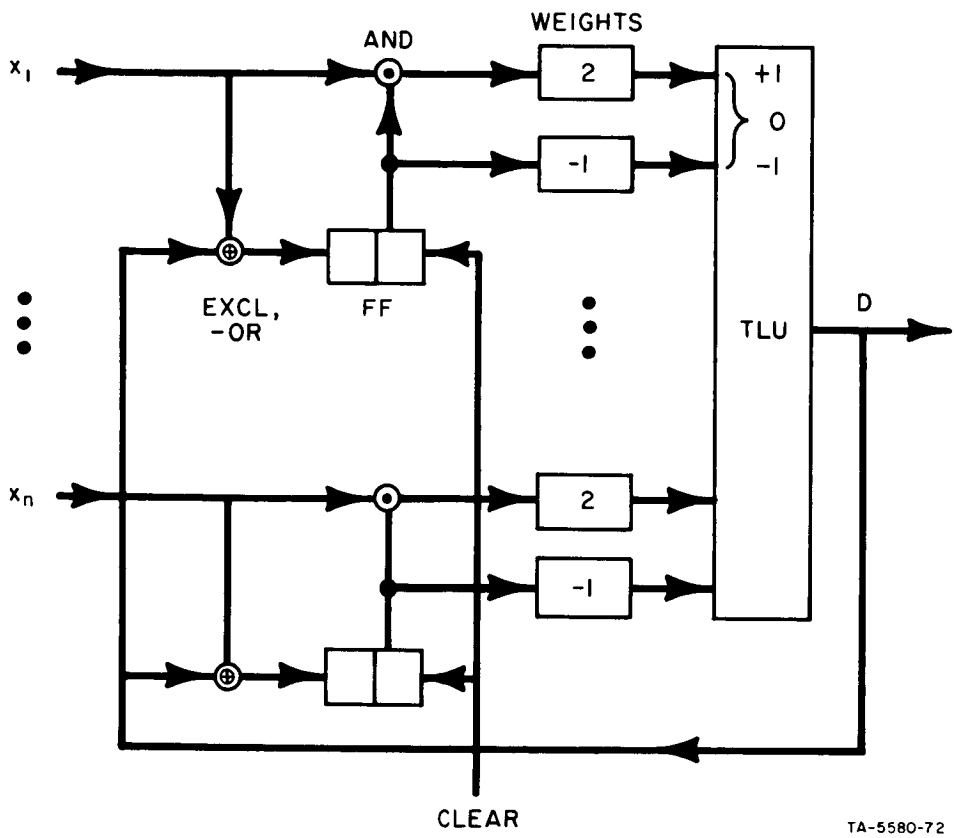


FIG. II-A-15 ADAPTIVE-VOTING SCHEME (using threshold logic)

(2) Switching-over-voting (Fig. II-A-16)--Replicated units are grouped in subsystems and are combined to provide single subsystem outputs; externally the outputs are selected by a stepping circuit that advances to a new subsystem when the connected subsystem fails or when it is undesirably close to failure; internally, subsystems employ

adaptive voting redundancy, as in scheme (1), for fault masking and for indication of the degree of closeness to failure.

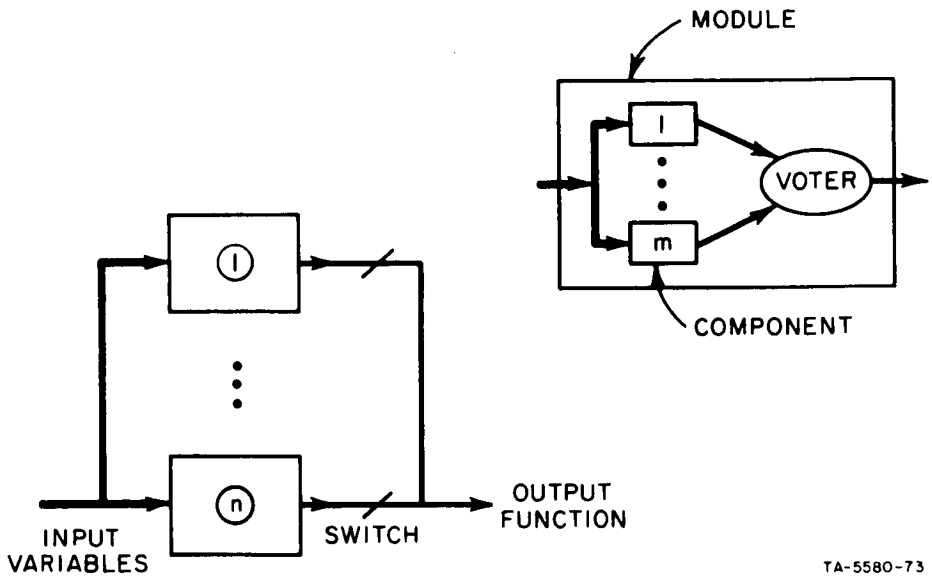


FIG. II-A-16 SWITCHING-OVER-VOTING SCHEME

(3) Voting-over-switching (Fig. II-A-17)--A fixed number of units is selected from a store of replicas, and their outputs are combined by majority voting; as a unit dissents from the system output, it is replaced by a unit in the store. Several replacement algorithms are feasible; e.g., the input whose unit was faulty may be distinguished, and a replacement unit directed to it, or all inputs may have fresh assignments, by selecting valid units in a predetermined order. The latter strategy appears to be better, because it is less susceptible to the propagation of faulty decisions among the subsystems.

The major advantage of these schemes over passive fault masking is the increased tolerable number of faulty subsystems--approximately N-2 instead of N/2, for N-order redundancy. Another advantage, of significance for spaceborne applications, is the economy of power consumption possible in schemes (2) and (3).

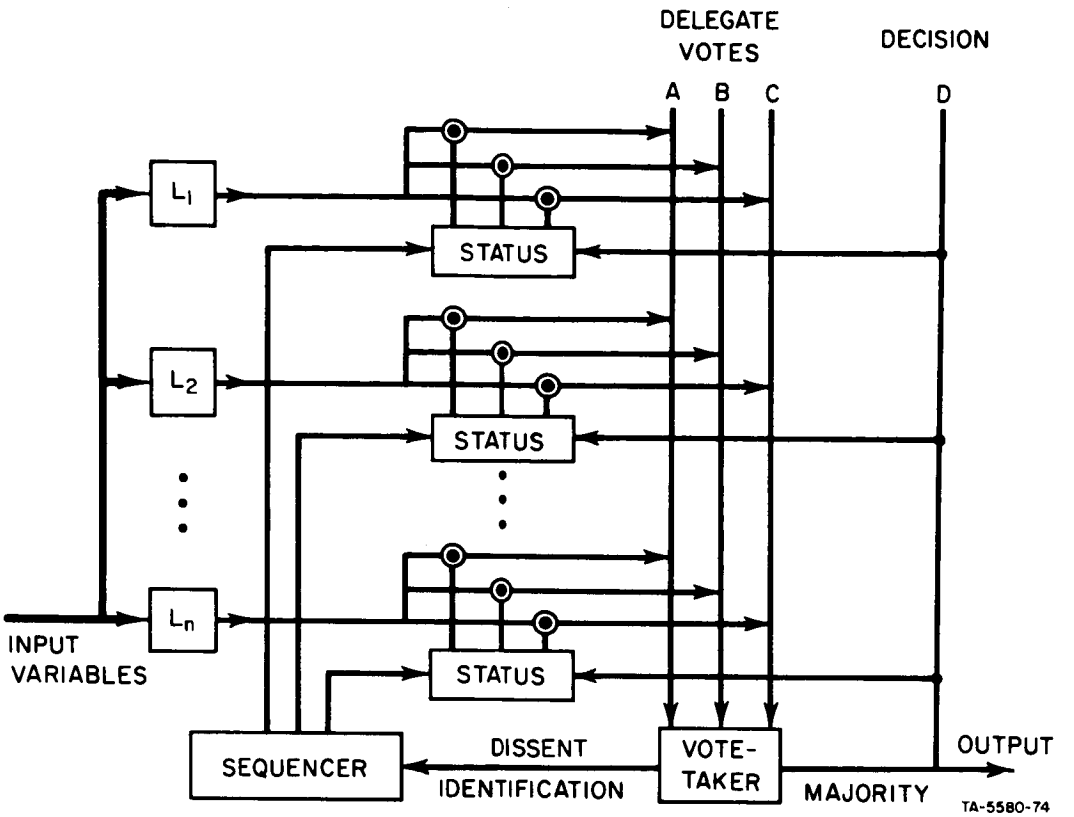


FIG. II-A-17 VOTING-OVER-SWITCHING SCHEME

There are several disadvantages. First, to varying degrees among the schemes, certain multiple failures occurring between replacements may cause the whole system to collapse. For example, in scheme (3), if a majority of the presently connected units are faulty, all of the units in store may be invalidated; in scheme (2), a subsystem might simply become stuck in a fixed, erroneous state if a majority of units in it become stuck. A system might be externally programmed so as to recover from such conditions, but such control adds to the system cost. A second disadvantage is that the number of components--hence the unreliability--of the control and switching is greater than for passive fault masking. Thus the minimum size of functional unit to which the schemes may be advantageously applied is greater. A third disadvantage is the difficulty of design for proper response to noise. Thus, if a unit has a transient fault, it may be disconnected. It would be desirable to reconnect these units, but it would be hazardous to do so without testing each one, since

if all were permanently faulty the faulty units might "outvote" the good units. The solution of this problem requires either a built-in sluggishness of disconnection, a capability for external diagnosis, or provision for autonomous verification of individual units prior to reconnection. The latter is feasible if at least two good units remain connected to serve for testing a candidate unit. The importance of these factors must be evaluated in the context of a particular system.

The idea of adaptive voting was suggested by Pierce.[241] He and others[195] have proposed implementing the scheme by the use of special elements, such as variable impedances with memory, or fuses. The technology for realizing the special impedance elements has not developed sufficiently to realize devices of adequate reliability for the missions of interest, and the use of fuses does not allow the reconnection of recovered logic units; hence in the discussion of the several schemes to be described, the use of conventional elements will be assumed.

2) Techniques for the Realization of the Adaptive-Voting Scheme

In this section, a number of techniques for realizing the Adaptive Voting scheme will be discussed. The first approach assumes the use of a 2N-input threshold logic unit, for order-N redundancy. The second approach assumes that only AND, OR and NOT elements are employed, and a number of alternative schemes employing such elements are described. Although the number of such elements are far in excess of the number of threshold logic weights employed in the first approach, the low cost and high reliability of such elements realized in microelectronic arrays make the approach worth consideration in future design.

Scheme using a linear-input threshold logic element:
A fixed-weight realization of Pierce's adaptive logic scheme is shown in Fig. II-A-15. Associated with each input signal, x_i, are a gate controlled by a "status" flip-flop, and two weights: +2 for the gated input and -1 for the ON state of the flip-flop. When the status flip-flop is ON, the net signal contribution to the threshold logic unit is +1 for $x_i = 1$, or -1 for $x_i = 0$. When the flip-flop is off, the net

contribution is 0. The threshold logic unit is set so that the output is 1 if the sum of all contributions is 1 or more, and 0 otherwise.

The output of the threshold logic unit is fed back to each channel, and upon any disagreement between the output and the input of a channel (indicated by a 1 at the output of an Exclusive-OR gate) the status flip-flop is reset, essentially disconnecting the channel.

The disadvantage of this realization is that the reliability of threshold logic circuits with a range of summed input variables of ten or more is not very high at present. In the following schemes, only binary switching elements will be employed.

Scheme using nonlinear elements only: The scheme illustrated in Fig. II-A-18 performs the same overall function as the previous one, but internally it uses binary signals only. The flip-flops perform the same functions of gating according to status, and their control is the same. Instead of combining the valid input signals so as to generate a single signal with a multivalued range about 0, a binary vector is generated. The elements of the vector, T_2, T_3, T_4, ... are the monotonic symmetric functions of the inputs; that is, the elements are 1 if at least 2, at least 3, at least 4, ..., respectively, of the N inputs are 1. The proper one of these functions is then selected to be the output, depending upon the number of input channels that are valid. Thus, T_2 is selected if the number of valid channels is two or three, T_3 if the number is four or five, and in general T_{j+1} is selected if the number of valid input channels is 2_j or 2_{j+1}, $j = 1, \ldots, m$; $N = 2m + 1$.

Thus the combining network may be decomposed into three parts, as shown in the figure:

(I) A network that realizes the monotonic symmetric functions

$$T_2, T_3, \ldots, T_{m+1}$$

(II) A network that realizes the symmetric functions

$$S_{2,3}, S_{4,5}, \ldots S_{2m,2m+1}$$

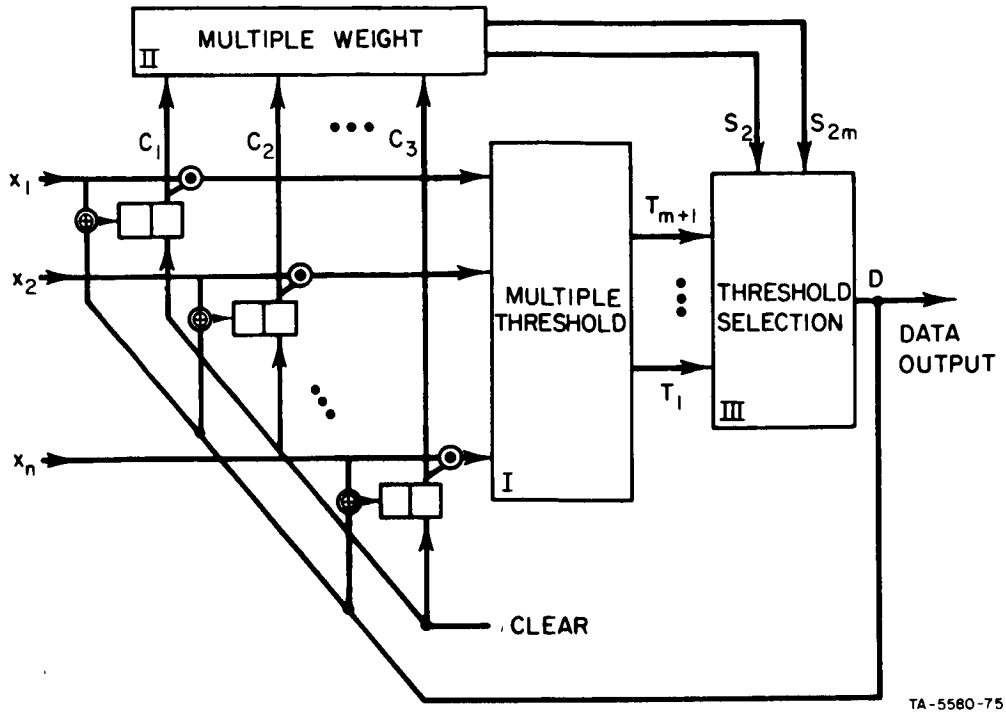


FIG. II-A-18 ADAPTIVE-VOTING SCHEME (using digital elements)

(III) A network that combines the S and T variables according to the function

$$D = S_{2,3}T_2 + S_{4,5}T_3 + S_{6,7}T_4 + \cdots S_{2m,2m+1}T_{m+1} \quad .$$

The symmetric functions $S_{2,3}$, etc., may be realized very inexpensively from monotonic symmetric (threshold) functions; hence the combining network may be obtained as in Fig. II-A-19, using two threshold-function networks, one slightly augmented, and a simple AND-OR network.

A number of approaches to the economical design of multiple-output threshold networks, employing simple nonlinear gating elements, are described in Sec. II-A-2-c.

3) Description of the Switching-over-Voting Scheme

This scheme is a simple extension of the adaptive-voting scheme; thus, each subsystem is adaptive, with the additional feature

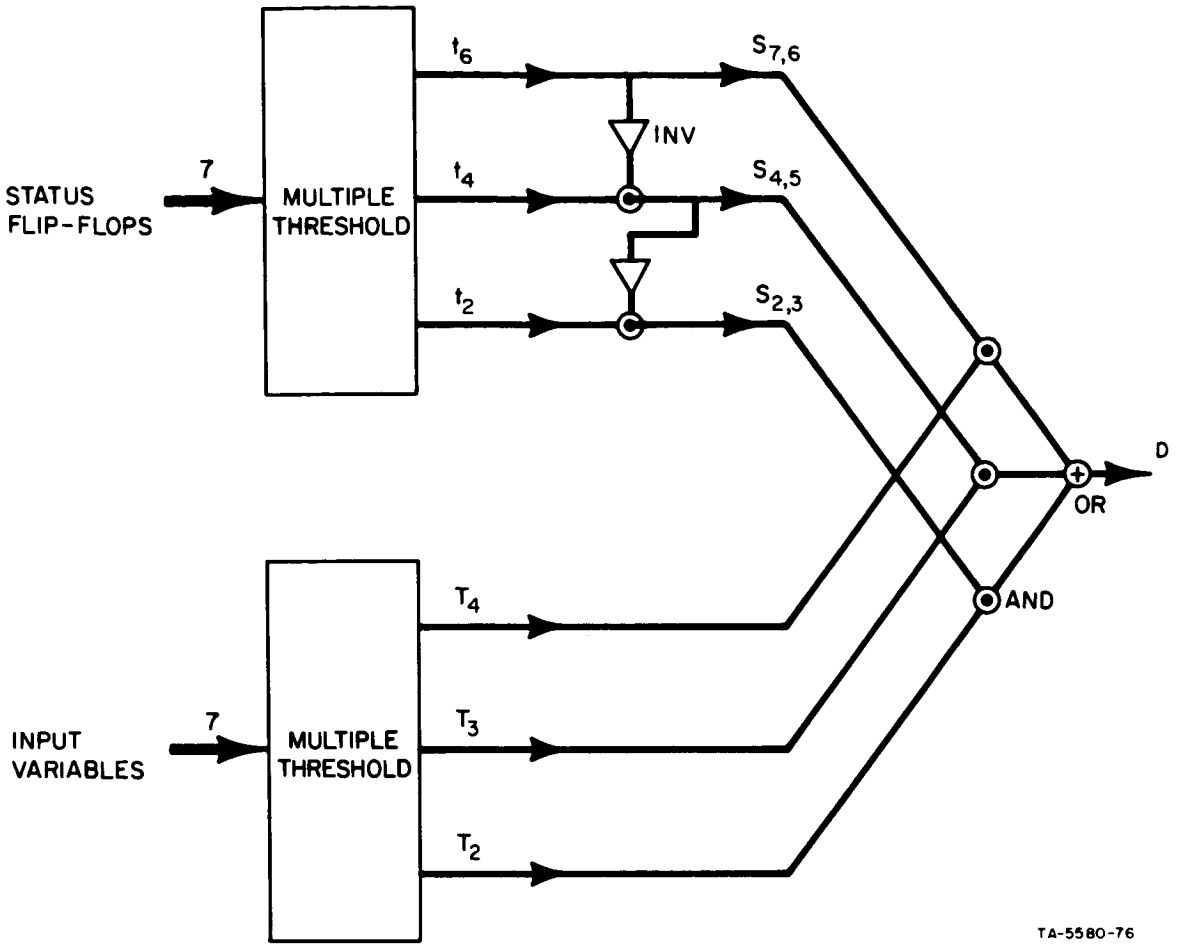


FIG. II-A-19 COMBINING NETWORK FOR DIGITAL-ADAPTIVE SCHEME

that a "warning" indication is given when the number of valid functional units is such that one more failure could not be masked.

The external control of subsystem replacement is quite simple: selection of a subsystem is determined by the state of a counter (shown in Fig. II-A-20) which steps upon receipt of a "warning" signal.

The designs of the component data and control subsystem are quite straightforward, and will not be described in further detail.

4) Description of the Voting-over-Switching Scheme

The voting-over-switching scheme is illustrated in Fig. II-A-17 for the case of three functional units taken at a given time. The switching and control functions are the costliest of the three

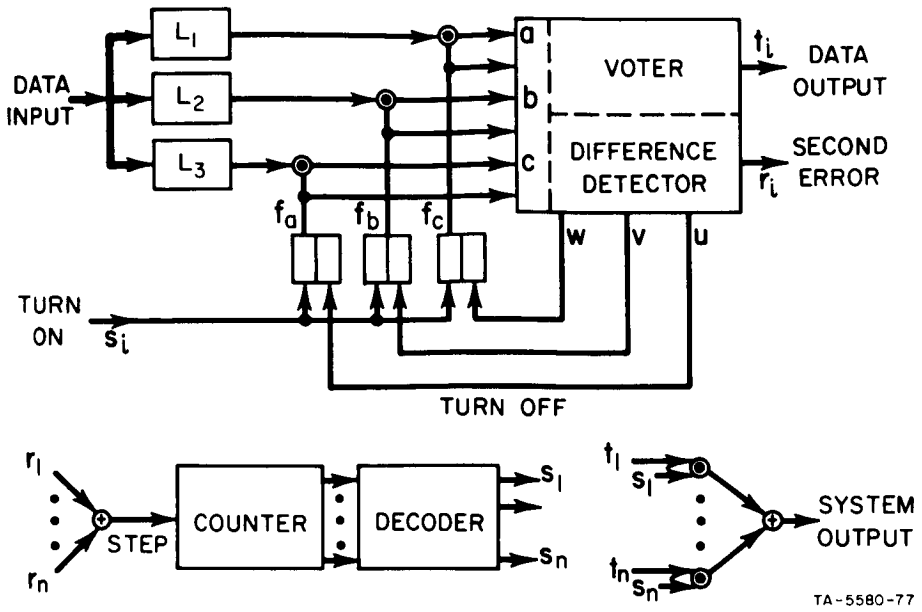


FIG. II-A-20 LOGICAL STRUCTURE FOR SWITCHING-OVER-VOTING SCHEME

schemes described, but the potential saving in power consumption is greatest.

In this system, the status information for each functional unit must have four values, indicating connection to one of the three channels (conveying data to the voter) or to none. The sequencer must implement one of the several possible strategies referred to in the introduction to this section.

The design of the signals that identify which channels dissent from the majority is straightforward, at least for low-order voting. For example, a dissent variable d_i may be defined, with the value 1 indicating that channel i dissents; then

$$d_i = x_i \oplus \text{Majority } (x_1, \ldots, x_N) \quad ,$$

where x_i is the i^{th} input to the voter. Greater economy is no doubt possible.

5) Comparisons and Conclusions

Some reasonable criteria for comparison of the three schemes described are

(1) The number of faulty functional units that may be tolerated

(2) The minimum power requirement

(3) The number of components required for the realization of decision and switching functions

(4) The number of simultaneous faults that cannot be tolerated.

The performance of the schemes with respect to these criteria is summarized in the following table, in which a is the number of units voting together in systems 2 and 3, b = N/a in system 2, and power cost is given assuming unit power in a functional unit.

System	Tolerable Number of Unit Failures	Minimum Power Cost	Cost of Voting	Cost of Switching and Control	Intolerable Number of Simultaneous Faults
Adaptive Voting	N - 1	N	high	low	(N + 1)/2
Switching-over-Voting	N - (2b + a + 1)	a	low	low	(a + 1)/2
Voting-over-Switching	N - 1	a	low	high	(a + 1)/2

Practical designs have been described for realization of several schemes combining voting and replacement redundancy methods. The decision and switching logic is more costly than the logic employed in simple fault masking, but technological developments may reduce these costs to an acceptable level.

The schemes are advantageous in the number of tolerable unit failures and in the possible reduction in power consumption, but they are somewhat more susceptible to collapse under simultaneous failures.

It is important to note that the scheme has merit even if it is not operated autonomously; that is, if the switching of a channel is controlled by an external computer rather than locally.

Further development of this approach should consider the application of redundancy to protect the decision and switching circuits, and the development of schemes for recovery from multiple simultaneous permanent or transient failures.

c. Voting Networks

1) Introduction

Although there are many published analyses of voting-type redundancy of arbitrary order of replication, there are surprisingly few published logical designs for voting networks, aside from the majority-of-three function. In the first part of this section, several designs for majorities of three, five and seven variables will be presented to aid in the assessment of circuit costs of high-order redundancy schemes and to serve as starting points for designs appropriate to particular technologies.

The second part of this section is concerned with the realization of voting-type circuits needed for the all-digital adaptive-voting schemes of Sec. II-A-2-b. These circuits produce a set of outputs that are the monotonic symmetric functions of their input variables, i.e., the functions T_j that are 1 when at least j input variables are 1. One of these functions is, of course, the majority function, and since the structures presented are in a canonical form, they provide a simple means of design for that function for any number of variables. Such a design, however, will generally be less efficient for that single function than the designs described in the first part.

Designs appropriate to several kinds of logic gates are presented, including threshold elements and AND-OR gate combinations. The AND-OR networks use more gates, but they are more easily produced by current technology, and considering advances in microminiaturization, their costs are not prohibitive.

2) Logical Designs for Simple Majority Networks

Use of single high-weight linear-input logic elements: The majority function is a linearly separable switching function; hence it may be realized by a single linear-input (or threshold) logic element, as illustrated in Fig. II-A-21. The notation M_k will be used for the majority function of k variables. For 2e + 1 variables, the appropriate input weight for each variable is +1, the appropriate bias is -e, and the range of the summed inputs is from -e to e + 1. The threshold circuit must thus be capable of resolving one unit in 2e + 1, and for increasing values of e the requirements on circuit precision become increasingly difficult to satisfy in practice (for example, see Coates and Lewis[43]).

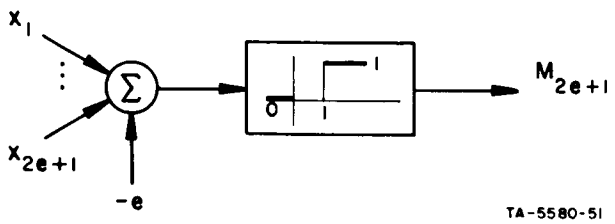


FIG. II-A-21 LINEAR-INPUT-LOGIC MAJORITY GATE

Furthermore, the circuit technology is not well suited to integration with non-linear circuits in a single monolithic structure; hence this approach is probably best used in a system in which all functions are realized by linear input logic.

Use of multiple low-weight linear-input logic elements: Amarel, Cooke, and Winder[7] have described designs for high-order majority-function networks composed of majority-of-three logic elements. Their designs for the majority functions of five and seven variables are given in Fig. II-A-22. These networks permit the use of low-weight, low-precision, linear-input logic circuits. Thus, they would be expected to have greater noise immunity and operating range than a single-element realization. The problem of compatibility in fabrication and operating levels with other system logic circuits remains.

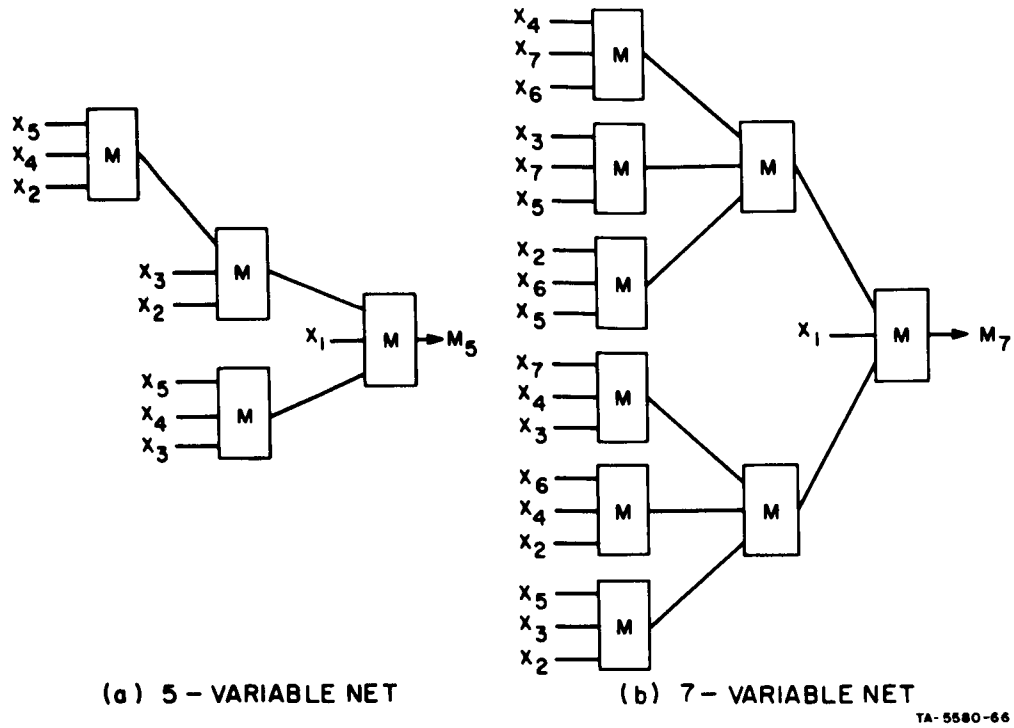


FIG. II-A-22 MAJORITY-ELEMENT MAJORITY NETWORKS (Amarel, Cooke & Winder)

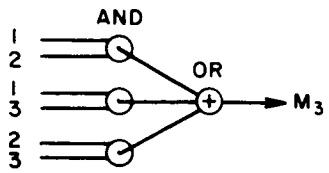
Use of AND-OR logic elements: Nonlinear logic elements are much more widely used than linear elements; hence it is of interest to investigate the design of majority-function networks composed of such elements, so that such networks may be physically integrated with the general logic networks of a system. We will consider here the use of AND and OR elements.

The majority-of-three function may be represented algebraically as

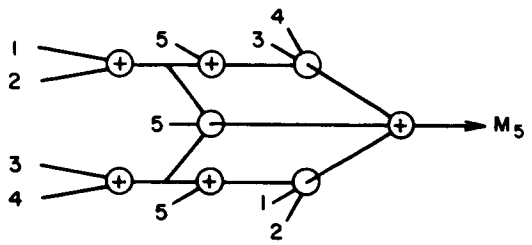
$$M_3 = x_1x_2 + x_1x_3 + x_2x_3 \quad .$$

The well-known network realization is given in Fig. II-A-23(a).* A realization based on $M_3 = x_1x_2 + x_3(x_1 + x_2)$ employs one less gate input, at the price of an additional stage of delay.

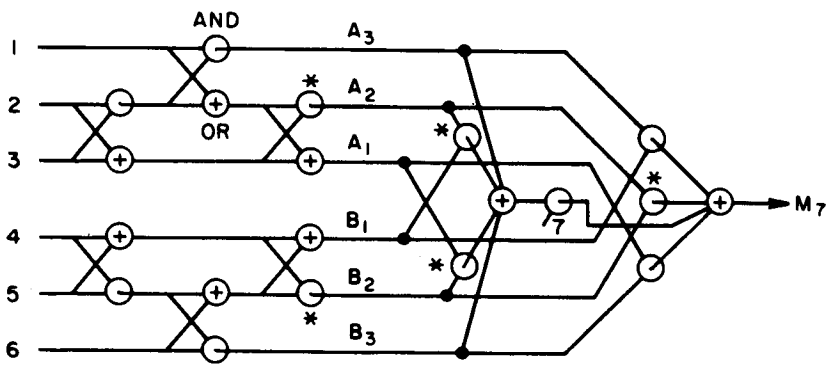
* In this figure the variables are represented by numbers.



(a) 3 VARIABLE NET



(b) 5 VARIABLE NET



(c) 7 VARIABLE NET

TA-5580-58

FIG. II-A-23 AND-OR MAJORITY NETWORKS

The majority-of-five function may be represented as

$$M_5 = x_5x_4(x_1 + x_2 + x_3) + x_3x_2(x_1 + x_4 + x_5) + (x_5 + x_4)(x_3 + x_2)x_1 \quad .$$

It is easily verified that the ten combinations of the variables, taken three at a time, are all represented. By introducing the intermediate variables

$$a = x_2 + x_3 \quad ,$$

$$b = x_4 + x_5 \quad ,$$

the function may be expressed as

$$M_5 = x_5x_4(x_1 + a) + x_3x_2(x_1 + b) + x_1ab \quad .$$

A network based on this expression is given in Fig. II-A-23(b). This realization has the lowest number of gates and the lowest number of gate inputs known to us.

In explaining the design of the M_7 function network, it is convenient to employ as an intermediate function the monotonic symmetric function T_k^m, which has the value 1 when k or more of its m input variables are 1. Then, grouping the seven variables, we employ the functions

$$A_k = T_k^3(x_1, x_2, x_3)$$

$$B_k = T_k^3(x_4, x_5, x_6), \; k = 1, 2, 3 \quad .$$

Then useful representations of M_7 are

$$\text{(Form 1)} \quad M_7 = A_3(B_1 + C_1) + A_2(B_2 + B_1C_1) + A_1(B_3 + B_2C_1) + B_3C_1 \quad ,$$

$$\text{(Form 2)} \quad M_7 = A_3B_1 + A_2B_2 + A_1B_3 + C_1[A_3 + A_2B_1 + A_1B_2 + B_3] \quad .$$

It may be noted that the sum of the subscripts of each product term is four, signifying that at least four variables must have the value 1 in order to make that product term true.

A useful set of intermediate variables is

$$y_1 = x_2x_3, \; y_2 = x_2 + x_3 \quad ,$$

$$w_1 = x_4x_5, \; w_2 = x_4 + x_5 \quad .$$

Then

$$A_3 = x_1y_1, \; A_2 = (x_1 + y_1)y_2, \; A_1 = (x_1 + y_1) + y_2 \text{ or } x_1 + y_2 \quad ,$$

and

$$B_3 = x_6 w_1, \quad B_2 = (x_6 + w_1)w_2, \quad B_1 = (x_6 + w_1) + w_2 \text{ or } x_6 + w_2 \quad .$$

A network based on Form (2) above, and employing these intermediate variables, is shown in Fig. II-A-23(c). The five gates marked by an asterisk may be combined into three gates in an obvious way; they are shown separated as an aid to following the realization of the terms of the functional equation.

It may be noted that this network costs half as much as a direct AND-OR realization of the majority-element network of Fig. II-A-23.

The cost in gates and number of inputs for M_3, M_5, and M_7 may be summarized as follows:

	Number of Gates	Number of Inputs
M_3	4	8, 9
M_5	8	20
M_7	18	44

At least for these cases, the costs approximately double with each increase in odd-order of redundancy.

3) Canonical Structures for Multiple-Output Voting Networks

Use of majority logic elements: The paper by Amarel, Cooke, and Winder referred to in the previous section also presents canonical network structures, composed of majority logic elements that realize the complete set of monotonic symmetric functions of the input variables. The structure is illustrated in Fig. II-A-24, where the notation (a/b) is equivalent to T_a^b as defined in the previous section.

Their realization is based on the identity

$$T_{a+1}^{b+1}(x_1, x_2, x_b, x_{b+1}) = x_{b+1} \cdot T_a^b(x_1, x_2, \ldots, x_b) + T_a^{b+1}(x_1, x_2, \ldots, x_b) \quad .$$

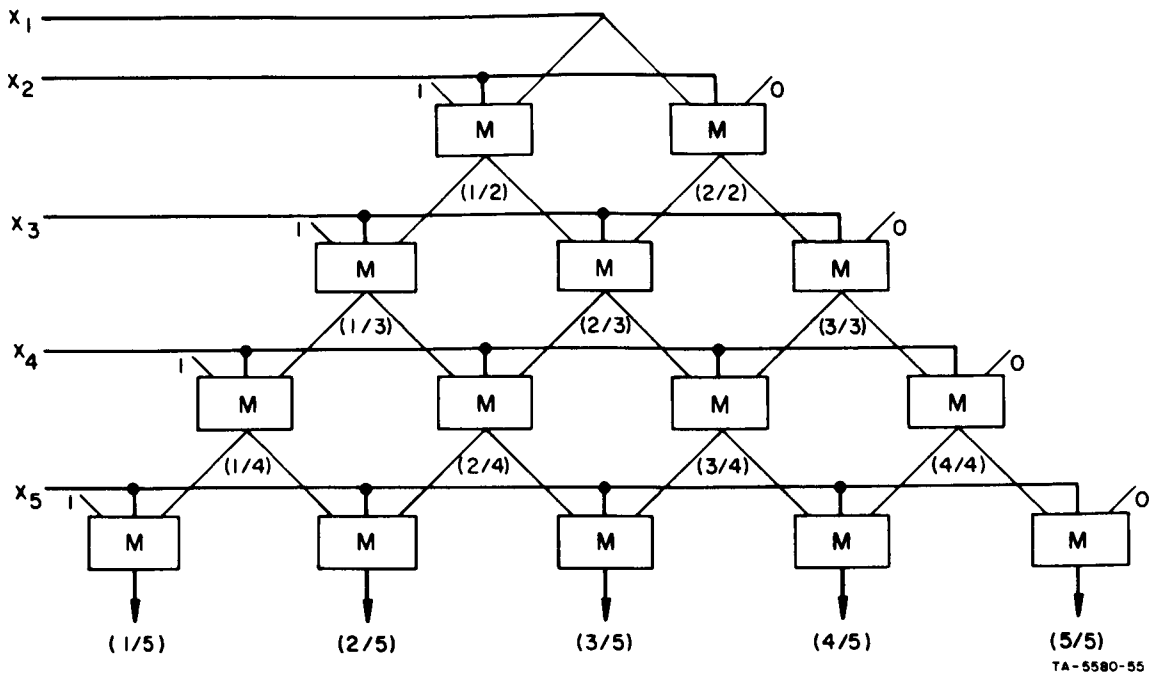


FIG. II-A-24 MAJORITY-ELEMENT MULTIPLE-OUTPUT VOTING NETWORK

A logic cell realizing this function could be built using one AND gate and one OR gate, but since the realization is restricted to majority gates, a highly redundant logical operation is actually employed, i.e.,

$$T_{a+1}^{b+1} = \text{Majority}\ (x_{b+1},\ T_a^b,\ T_a^{b+1}) \quad .$$

Use of AND and OR gates: One way of using AND-OR gates is to employ the structure of Fig. II-A-24, in which each cell contains one AND and one OR gate as described above. This construction requires that input variable x_j drives j gates. An alternate realization, in which each input variable drives exactly two gates, is shown for seven variables in Fig. II-A-25. Extension to more variables is obvious. The validity of this network is less obvious than that of the previous network, so a formal proof of its validity will be given.

We wish to prove that the network construction scheme shown realizes the desired functions for any number of variables. This will be done inductively by showing that a valid network for n variables

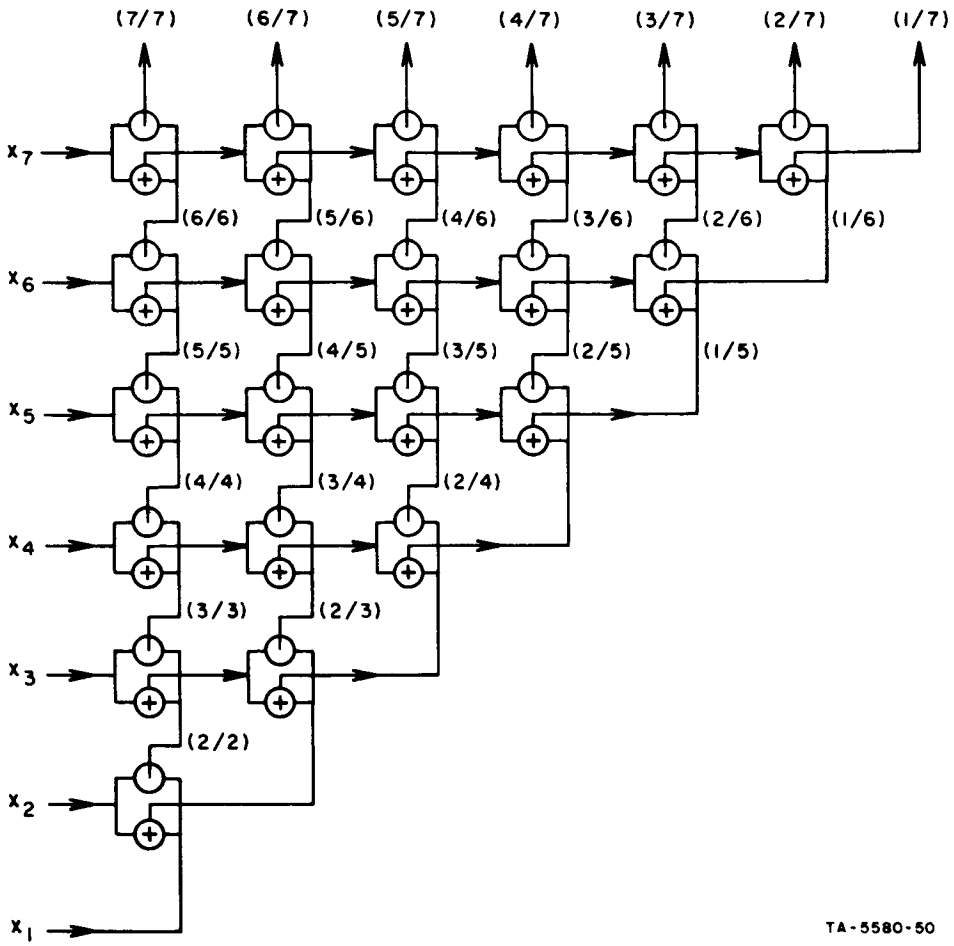


FIG. II-A-25 AND-OR MULTIPLE-OUTPUT VOTING NETWORK

may be augmented to form a valid network for n + 1 variables, and that the augmented network follows the stated construction scheme.

Let T_m^n be the monotonic symmetric function on n variables; that is, T_m^n is true if at least m of the variables are true. Let it be assumed that a network A produces the functions $T_n(x_1, \ldots, x_n)$, $T_{n-1} \cdot (x_1, \ldots, x_n)$, $\ldots$, $T_1(x_1, \ldots, x_n)$. We wish to augment the network A with a network B, with inputs $y_1, y_2, \ldots, y_{n+1}$ and outputs $U_{n+1}^{n+1} = y_1 \cdot y_2 \cdot \ldots \cdot y_{n+1}$ and $x_n, x_{n-1}, \ldots, x_1$. The x_i's also form the inputs for the n-variable network A, and are determined as follows:

$$
\begin{aligned}
x_n &= y_{n+1} + (y_n y_{n-1} \cdots y_1) \\
x_{n-1} &= y_n + (y_{n-1} y_{n-2} \cdots y_1) \\
&\vdots \\
x_2 &= y_3 + y_2 y_1 \\
x_1 &= y_2 + y_1 \quad . \qquad \text{(II-34)}
\end{aligned}
$$

Let the outputs of the combined $(n + 1)$-variable network be U_{n+1}^{n+1}, U_n^{n+1}, ..., U_1^{n+1}. We must show that

$$U_m^{n+1} = T_m^{n+1} \quad .$$

Proof:

(1) By construction of the network B, $U_{n+1}^{n+1} = y_{n+1} y_n \cdots y_1 = T_{n+1}^{n+1}$.

(2) Examine $U_m^{n+1} = T_m^n(x_1, \ldots, x_n)$, and suppose m or more of the y's are true. Then by Eq. (II-34) it follows that m or more of the x's are true; for, in particular, if $y_i = 1$, so does x_{i-1}. (In the special case where $y_1 = y_2 = 1$, we have m - 1 x's = 1 by the above argument, plus one more for the output $x_i = y_{i+1} + y_i \cdots y_1$, where $y_{i+1} = 0$ and all $y_j = 1$ for $j < i + 1$.) Hence U_m^{n+1} is 1 whenever m or more of the y's = 1.

(3) To show that $U_m^{n+1} = 0$ whenever m - 1 or fewer y's = 1, let at least n - m + 2 y_i's = 0 (corresponding to n - m + 1 x_i's); let them be

$$y_{i_1}, y_{i_2}, \ldots, y_{i_{n-m+2}}$$

where

$$i_1 > i_2 > \ldots > i_{n-m+2} \quad .$$

Then at least $n - m + 1$ x_i's $= 0$. From Eq. (II-34) they are

$$x_{i_1-1} = y_{i_1} + y_{i_1-1}y_{i_1-2}\cdots y_{i_2}\cdots y_1 \quad ,$$

since $y_{i_1} = y_{i_2} = 0$

$$x_{i_2-1} = y_{i_2} + y_{i_2-1}y_{i_2-2}\cdots y_{i_3}\cdots y_1 \quad ,$$

since $y_{i_2} = y_{i_3} = 0$

$$\vdots$$

$$x_{i_{n-m+1}-1} = y_{i_{n-m+1}} + y_{i_{n-m+1}-1}\cdots y_{i_{n-m+2}}\cdots y_1$$

since $y_{i_{n-m+1}} = y_{i_{n-m+2}} = 0$.

But if at least $n - m + 1$ x_i's $= 0$, then no more than $n - (n - m + 1) = m - 1$ of them are 1 and $U_m^{n+1} = 0$. Hence $U_m^{n+1} = 0$ whenever less than m of the y_i's $= 1$.

(4) By (2), $U_m^{n+1} = 1$ whenever $T_m^{n+1} = 1$, and by (3) $U_m^{n+1} = 0$ whenever $T_m^{n+1} = 0$; hence $U_m^{n+1} = T_m^{n+1}$ and the inductive step is demonstrated.

(5) Clearly the network construction results in the proper output functions for low values of n, say $n = 2$; hence the desired result is demonstrated for all n.

Use of sequential "diffusion" networks for multiple threshold detection: Given a set of n binary variables, one way to realize a threshold function is to sort the 0's and 1's into two strings

and to observe whether or not the string of 1's exceeds a given length. This behavior may be realized in a special kind of shift register, shown in Fig. II-A-26(a).

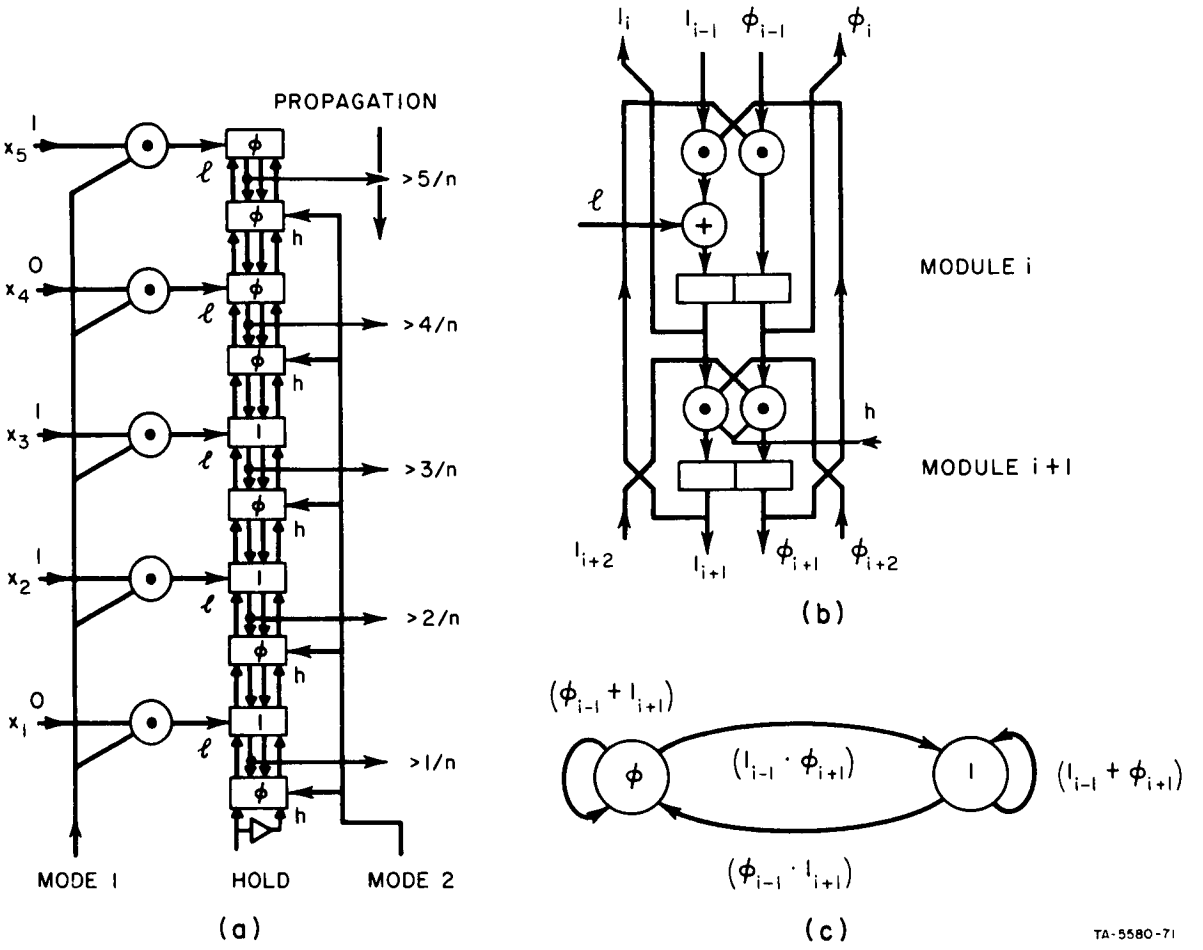


FIG. II-A-26 DIFFUSION-TYPE MULTIPLE-OUTPUT VOTING NETWORK

The register has three modes of operation. In the first mode it is loaded in parallel by the input vector. In the second mode the information is allowed to propagate through the stages asynchronously. After a time determined by the length of the register and by the switching times of the stages, the register will reach a stable state, and it may then be interrogated. In the third mode it is cleared by allowing all strings to shift out serially. Parallel clearing is also possible at greater expense.

The register is based upon Muller's speed-independent modules,[221] but instead of using a three-state code--0, 1, and Ø ("null")--only two states are needed, namely 1 and Ø. Cascades of such stages have the property that a data symbol (in this case, just the symbol 1) will propagate by being copied over a Ø symbol in a forward module unless a data symbol is resident in the module beyond--in which case, the first data symbol is held in place. In other words, the Ø symbol serves to separate 1 symbols of different origins. This rule results in a forward diffusion of 1 symbols, until a string of alternating 1 and Ø symbols is built up in successive modules. In the course of this diffusion, a given 1 symbol may be momentarily replicated, but all such replications will be in a contiguous string, bounded by Ø symbols. In the final resting configuration, all replicas deriving from a given symbol will coalesce into a single symbol. Thus, in the resting state, the presence of a 1 at the $2j^{th}$ module indicates that there were at least two 1's at the input.

Figure II-A-26(a) is a block diagram of the register, for the case N = 5. Each stage contains two modules, which are identical except that one may be loaded by external data. The output taps are self-explanatory. Clearing is accomplished by allowing the data to propagate out of the register.

Figure II-A-26(b) shows a pair of modules constituting a stage. Each module contains a single flip-flop and two or three gates. Feedback is provided to ensure safe asynchronous operation. The state diagram for a module in its propagation mode is given in Fig. II-A-26(c).

Some racing may occur in the transition between modes, but this is easily handled. It may be noted that the cost of this approach is linear with the number of variables.

Conclusions: A number of designs have been presented for single and multiple-threshold combinational logic networks, up to order seven, using AND and OR gates, majority gates, and flip-flops, and for sequential networks for multiple threshold detection. The latter scheme permits an exchange between time delay and amount of hardware that may be

advantageous for systems employing high orders of replication. These designs should facilitate estimates of cost and reliability for fault-masking systems.

3. Sequential Networks

a. Introduction

The networks discussed in Sec. II-A-2 above are combinational by virtue of the fact that their outputs are uniquely determined by their present inputs. That is, their proper operation does not depend in any essential way on memory, storage or delay. Of course, all physical devices involve some finite (though small) delay--the distinction between combinational and sequential networks rests on the point that such delays are merely incidental to the operation of combinational networks, while they are an essential part of the behavior of sequential networks. Loosely speaking, sequential networks--to be discussed in this section--remember some aspects of their past history (inputs and/or outputs) and make use of this memory to influence their present (and future) behavior.

Examples of sequential networks used in digital computing systems are counters, registers, accumulators, sequence generators, sequence sensors, classifiers, and decoders. Sequential networks are thus capable of much richer and more varied behavior than are purely combinational nets. By virtue of this richer behavior, the variety of misbehavior that may occur in the presence of faults is likewise much richer than with combinational nets.

In order to discuss the various possibilities that can arise, let us consider the standard (Mealy) model for (clocked*) sequential networks shown in Fig. II-A-27. Here the combinational portion of the network has been lumped together into one box, labeled "CL," while the storage functions are embodied in delay units. The vector $X = (X_1, \ldots, X_m)$ represents external inputs to the net; the vector $Z = (Z_1, \ldots, Z_p)$

---

* Clocked = synchronous; we shall not discuss asynchronous sequential nets.

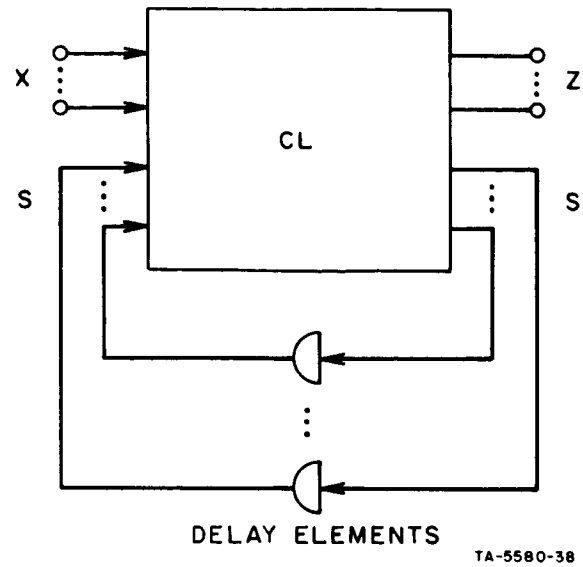


FIG. II-A-27 MODEL OF SEQUENTIAL NETWORK

represents its outputs to the external world. The state vector $S = (s_1, \ldots, s_n)$ of delay-element outputs embodies the network's memory of past behavior.

It is seen that the behavior of the net is governed by the following logical design equations:

$$S' = F(X,S) \qquad \text{(Next-State Equation)}$$

$$Z = G(X,S) \qquad \text{(Output Equation)}$$

where the variables X, Z, S, and S' are all vector Boolean variables. The variable S' represents the inputs to the delay elements (memory-excitation function) and it is thus the <u>next state</u> of the network, assuming that no faults occur. Thus, in terms of the discrete time variable t we have, as an additional relation,

$$S(t + 1) = S'(t) \qquad \text{(Delay-Element Equation)} \quad .$$

If our model had used, say, flip-flops instead of delays as storage elements, this last equation would of course have to be altered to describe flip-flop behavior. In other respects an entirely equivalent model would then result.

The three equations given above suggest that it would be convenient to partition the sequential network model further, as shown in Fig. II-A-28. Here, the memory-excitation function and the output function are separated, conceptually at least, into boxes marked F and G respectively. It must be recognized, however, that in a given realization of such a network, there may be a good deal of sharing of components between these two functions. Thus, in particular, a single component fault could conceivably result in incorrect outputs from both functions F and G.

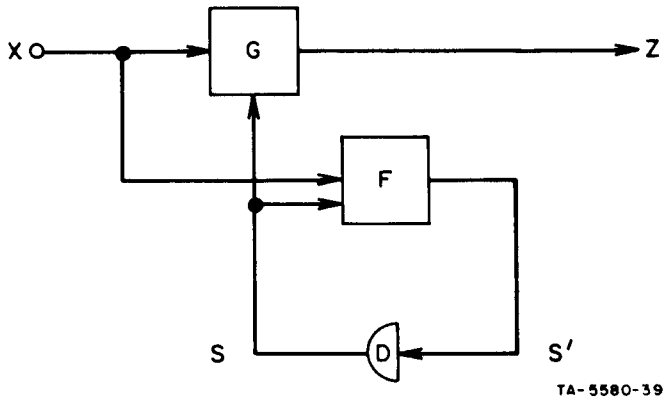


FIG. II-A-28 PARTITIONED MODEL OF SEQUENTIAL NETWORK

b. Classification of Faults

We propose now to classify and analyze the kinds of misbehavior that can result from faults in various portions of the network. We shall classify faults according to which of the logic functions they invalidate. Thus, we have the following types of faults:

(1) Faults affecting only the output function G(X,S) (output-only faults)

(2) Faults affecting only the storage elements (delay elements) of the network (delay-element faults)

(3) Faults affecting only the memory-excitation function F(X,S) (memory-excitation function faults)

(4) Faults affecting both functions G and F (output-plus-memory-excitation faults)

(5) Faults affecting both the combinational and storage portions of the net. (These may be combinations of faults of the other types, or they may be simple faults involving overall aspects of the network, e.g., faulty or missing clock pulses, out-of-tolerance supply voltages, etc.)

Of course, one may group these kinds of faults according to whether they result in errors in state-transition behavior or output-signal behavior (or both). Possibly a closer analysis would be worthwhile. We give next a brief description of some distinguishing features of the above types of faults, followed by a discussion of applicable fault-detection techniques.

1) Output-Only Faults

These faults are essentially combinational in nature. The state-transition behavior of the network is not affected--only the output signals which communicate this behavior to other portions of the computer system. Consequently, this problem can be treated entirely in combinational terms, and the methods of Sec. II-A-2 are applicable.

However, the availability of the memory-excitation logic and its delay-feedback loops can conceivably act as an aid to fault diagnosis, provided it is known a priori (or can reasonably be assumed) on the basis of other diagnostic information that there are no state-transition faults. Even without certainty as to the correct functioning of the state-transition system, that portion of the network can often be used as a test input generator for checking the output logic, provided that the state variables (and the external inputs) can be monitored.

2) Delay-Element Faults

In this case, the memory excitation function F is itself correctly generated, but the memory units fail to execute the proper transitions in response to F. It would seem that a highly probable variety of fault condition in this case is one in which one or more delays (or flip-flops) are "stuck" at 0 or at 1. Note, however, that in some technologies the occurrence of transient faults (e.g., occasional failure of

a flip-flop to trigger) is also quite possible. A paper by Rubio[273] considers error-correction techniques for such a situation, where "slow" flip-flops are used (for reasons of economy).

The distinguishing characteristics of delay-element faults are that the memory excitation is correctly related to the input X and present-state variables S, and that the output function is likewise correct for the stated conditions--yet the required state transitions are nevertheless improperly executed.

3) Memory-Excitation Faults

Though these are faults occurring in the combinational logic, they make themselves felt in terms of erroneous state transitions, much as do delay-element faults. Hence they may be much more difficult to locate (or even to detect) in some kinds of systems than are output-logic faults. Also, in all except the simplest kinds of sequential circuits, memory-excitation faults can lead to much more varied and complicated kinds of erroneous behavior than can delay-element faults.

However, in general terms, memory-excitation faults and delay-element faults are alike in that they both lead to state-transition errors. Much the same techniques are applicable to the detection of either kind of fault. We will usually be able to discuss these together in the remainder of this section.

4) Output-Plus-Memory-Excitation Faults

When a fault arises in the logical circuitry common to the production of the functions F and G (or when several simultaneous faults are present affecting both F and G), then the diagnostic situation is complicated by the fact that neither the output Z nor the memory-excitation function F(X,S) is correct. However, it seems reasonable to expect that such faults (though difficult to locate) are at least as easy to detect as faults in F or G alone. This will be the case when independent checking means are provided on the Z output and on S', for errors that remain undetected in one case may be detected in the other.

5) Overall-Network Faults

We have not yet discussed faults arising in portions of a sequential network which are common to (or affect) both the combinational and delay functions. Examples of such are power-supply or clock-pulse malfunctions, or shorts occurring between these two portions of the circuit, perhaps rendering two signals invalid. A good many such faults will be detectable by essentially nondigital tests: excessive circuit loading, for example, may result in poor rise time or out-of-tolerance voltage levels; or they may be detected by direct checking of the clock source at various points in the network.

It is not intended to minimize either the possibility or the variety of such fault conditions when we state that this section is not primarily concerned with them. We merely wish to point out that other means are available for their detection and diagnosis (see Appendices B and C).

c. Logical-Redundancy Techniques

We turn next to the principal subject of this section, a discussion of some logical-redundancy techniques for fault detection in sequential networks. Historically, the first treatment of this problem was undertaken by Moore.[213] His approach involved the design of a checking experiment to test a given (presumably nonredundant) sequential network, rather than the deliberate incorporation of redundancy to facilitate checking. Certain assumptions were used by Moore to guarantee the existence of a finite-length checking experiment. These assumptions, which involve both the original, unfaulted network and the admissible faults, are:

- The unfaulted network must be strongly connected--i.e., by using only signals applied to the external inputs it must be possible to put the network into an arbitrary state, regardless of its initial state. (The input sequence that accomplishes this will in general depend on both initial and final states.)
- There must be at most a finite number of admissible faults, and their effects on the state diagram (i.e., on the functions F and G) must be known or calculable.

Implicit in the second assumption is the requirement that no malfunction occurs during the course of the experiment.

Under the above assumptions it can be shown[213, 103] that in principle at least it is possible to design a checking experiment that will determine whether the circuit is behaving correctly, or whether some admissible fault condition has arisen. This experiment also determines which one of these malfunctions (if any) is present. Thus Moore's test is not only a fault-detecting experiment, it is also fault-locating.

Unfortunately, the amount of computation involved in designing a Moore experiment is far beyond the capabilities of even a large computer, except for quite simple sequential networks, and then only if the set of admissible faults is small. For example, if all malfunctions leading to a state diagram with no more than $N = 2^n$ states are to be treated, and the network has only a single binary input and a single binary output, then one must consider $(2N)^{2N}$ different state tables. Even if n is as small as 3, this leads to 2^{64} distinct state tables. One cannot even list all of these on any existing computer.

It is possible to make some progress, however, by admitting only a limited set of the most probable faults. The Moore method when applied in such cases is still cumbersome, but for reasonably simple machines it leads to checking experiments with a tractable amount of computation.

One of the difficulties with Moore's approach (and with that of Poage and McCluskey,[247] which is essentially a specialized simplification of Moore's method) is that it requires too much information. We are interested in many cases only in detecting a fault condition, without going into location on the spot. Replacement of the whole unit will lead to an operable system, and detailed off-line testing can be used later to locate the source of the trouble if desired. Hennie[127] has developed a completely different checking procedure which is simply fault-detecting but which is capable of protecting against large numbers of malfunctions. This procedure does not require the enumeration of the possible kinds of faults. Although Hennie's technique does not necessarily lead to the shortest possible experiments, it does seem to lead in most cases to

reasonably good ones. Thus, it is often possible to design a checking experiment whose length is proportional to N^2 or N^3 (where N is the number of states). However, in the worst cases, the experiment length may increase exponentially with N.

Hennie's methods are most practical when it can be assumed that (1) both the correct circuit and the admissible faulted circuit have no equivalent states (i.e., each has the same number N of states as a "reduced" machine), and (2) the reduced-state table has a distinguishing sequence such that the circuit produces N different responses according to which of the N states it occupied at the start of the experiment. However, the method is applicable also under slightly more general conditions (see Ref. 127). All of these conditions still require that the original machine be strongly connected, as in Moore's method.

The testing philosophy and methods introduced by Hennie have been carried somewhat farther and elaborated by Kime.[160, 161] Kime's principal contribution seems to have been in the direction of transforming a given circuit which does not possess a distinguishing sequence into a circuit which does, but also retains the desired overall sequential function. This is accomplished by (1) the addition of test points which make some (but not necessarily all) of the state variables S_i available as outputs, and (2) the addition of logic and a single input terminal.

It appears at this point in the development of the subject that the most promising avenues for future research on fault detection for sequential machines lie in the directions opened by Kime's work, i.e., in the incorporation of fault-detection features into the design of the original machines. The more basic work of Moore, Hennie and others is extremely valuable for its generality and for its indication of the severe difficulties which beset any really general approach to fault detection and still more to fault location. Practical solutions to these problems that are applicable to networks of realistic size and complexity seem to require a more pragmatic approach.

Indeed, there is a spectrum of regimes whereby fault detection (or location or masking) can be employed, depending on the extent to which

checking is integrated into "normal" circuit operation. At one extreme of this spectrum lies the philosophy implicit in Moore's and related studies, wherein checking is treated as an aspect of the device's operation entirely separate from its usual functioning. Here the network is subjected to a distinct sequential experiment whose purpose it is to determine whether the internal structure is functioning correctly. Since the length of such an experiment may run to hundreds or even thousands of input symbols, it is clear that the execution of the experiment may require appreciable idle periods. The experiment can be "time-shared" with normal use only in a gross sense. We might call this "off-line" or "intermittent" checking.

At the other extreme of this spectrum lie techniques--to be described below--which involve continuous checking (on-line) of the device on a bit-by-bit basis. Here the checking operations are completely integrated with normal circuit functioning, but at a cost in added circuitry. On the other hand, intermittent checking *per se* involves a cost in operating time (and power consumption) though it may be cheaper in terms of hardware.

Between these two extremes lie varieties of checking which interleave test operations with steps in the normal functioning of the circuit, but at different time scales, not at every clock pulse. For example, a counter can easily be subjected to checking at every *k*th input pulse by providing a recycling mod k counter to quiz the main counter's state so as to test for divisibility by k. Such intermittent checking, where applicable, involves a lower order of circuit redundancy than is the case with continuous on-line checking. It also possesses considerable flexibility of operation to suit a variety of conditions.

d. Schemes for Fault Detection

We next discuss in some detail several schemes for the incorporation of fault-detection capability (in the continuous mode) into the design of arbitrary synchronous sequential networks, such as are typically found in the central control portions of digital computers. All of the techniques to be described involve the introduction (in different ways)

of redundant states into the state diagram of the sequential network. The general principle used is that of requiring that only a subset of the network's states (the "valid" states) correspond to correct functioning, and any transition to an "invalid" state signals a faulty condition. Alternatively, it can be the state transitions that are checked. Two classes of redundant-state logic will be considered: state-parity checking and state-weight checking.

1) State-Parity Checking

This technique (closely related to Armstrong's error-correction scheme[10] is conceptually the simpler of the two. It draws heavily on the use of parity-check codes in communications for the purpose of error detection in message transmission. Here we employ sets of parity checks over the state variables of a sequential network to help insure the validity of a state, and hence of a state transition.

As a simple example, consider the introduction of a single extra state variable P_1 in addition to the existing state variables $S_1, \ldots, S_n$, where P_1 is defined to be the modulo-2 sum:

$$P_1 = S_1 \oplus S_2 \oplus \ldots \oplus S_n \quad .$$

Thus, we may form P_1' with combinational logic entirely independent of the logic which generates the memory-excitation signals S_i'. Then, any single (or any odd-order) error in the vector $(S_1', \ldots, S_n', P_1')$ will appear as a parity violation. If we provide P_1 with its own delay element, as in Fig. II-A-29, then on the next clock phase $t + 1$, the signals $S_1'(t), \ldots, S_n'(t), P_1'(t)$ will appear at the delay-element outputs. By checking parity over these $n + 1$ signals we will be able to detect any odd-order errors, not only in the combinational logic but also in the delay elements themselves. It is evident that by the addition of a single extra delay and its associated excitation logic we have made half of the possible erroneous state transitions detectable.

Since parity functions (especially of more than two or three variables) are awkward to generate in combinational logic, almost

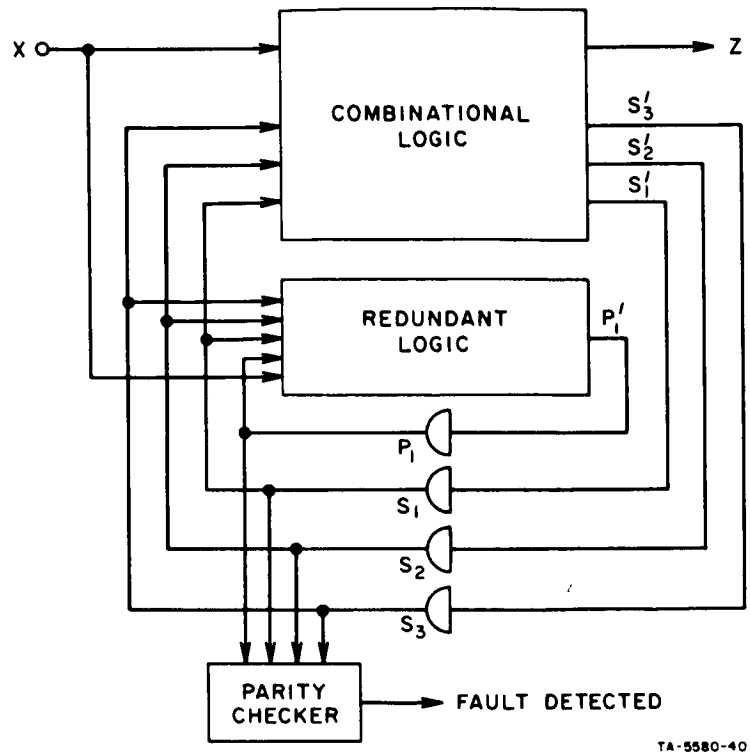


FIG. II-A-29 STATE-PARITY-CHECKED SEQUENTIAL NETWORK

regardless of the particular technology,* it might seem that the parity-checking technique would lead to considerable hardware costs. The situation is better than it appears at first sight since the memory-excitation function P_1' is not actually generated as a parity check over the other excitations, S_i'. (Indeed, if it were so generated the scheme would break down and check only the delay elements, not the combinational logic.) For example, if we happened to have

$$S_1' = S_2 S_3$$

$$S_2' = S_3 \bar{S}_1$$

$$S_3' = \bar{S}_1 S_2$$

* Relay-contact logic, cryotrons, etc., are exceptions.

then we see that

$$P_1' = \bar{S}_1 S_2 \oplus S_2 S_3 \oplus S_3 \bar{S}_1$$

$$= \bar{S}_1 S_2 \oplus S_2 S_3 \oplus S_3 \bar{S}_1$$

$$= M(\bar{S}_1, S_2, S_3)$$

where M denotes the three-input majority function, a function very readily realized with threshold logic.

Of course, the above example is a very special case. In general, the complexity of the combinational logic involved in generating a parity-check signal P_1' will be highly dependent on the particular forms that the other memory-excitation functions happen to have. In order to increase the detection capability over that provided by a simple parity check, we need to supply additional check signals, $P_2', \ldots, P_r'$ checking over independent subsets of the state variables S_i'. One scheme for doing this is provided by the Hamming codes. These are, however, not the most suitable for our purposes, since each Hamming parity check involves almost half of the nonredundant state variables, thus leading to a large number of exclusive-OR gates in the checking circuit. Better schemes are available in the form of low-density parity-check codes; see Gallager.[88] These codes can be designed to use no more than two or three state variables for each redundant check signal.

It is clear that if, say, r independent checks are used, then all but a fraction 2^{-r} of the states will be detected as invalid. The probability of detecting incorrect state transitions then increases to $1 - 2^{-r}$.* We come now to an objection which may already have occurred to the reader. It may be argued that the addition of redundant delay elements and combinational logic will increase the number of possible

* Strictly speaking this number $1 - 2^{-r}$ is only the fractional number of invalid state transitions from any given state; it becomes a probability only if we assume that all faulty transitions are equally likely.

incorrect transitions, and will also increase the probability of such transitions, so that we are no better off than before we added redundant states. It is indeed correct that both the number and probability of incorrect transitions will be increased by the addition of redundancy. However, adding a single delay element, while it doubles the number of possible erroneous transitions, does not double the probability of such an error. That would be the case only if we had to double the complexity of the whole network every time we added one redundant state variable. Of course, that is far from the case--typically, adding one delay element (plus logic) will add one unit of complexity to the network. Thus, it increases the complexity by about 5 percent if there were 20 state variables to begin with. We can therefore expect that the overall fault probability is increased by about 5 percent, but we have cut in half the probability of its going undetected.

To put the matter more succinctly: the inherent unreliability of the network increases roughly linearly with its complexity (as measured by the total number of state variables). The probability that once a fault has occurred it will go undetected, however, decreases exponentially with r, the number of redundant variables (i.e., as 2^{-r}) independently of the number of nonredundant state variables. We are therefore fighting a very favorable battle, rather than a losing one, when we increase r.

The major remaining questions in this area are:

(1) How best to arrange the parity-check signals (i.e., which Gallager codes, or others, to employ for typical sequential networks).

(2) What redundancy ratios, r/n, are most effective.

(3) How best to combine this technique (i.e., in what proportions) with other, purely combinational, fault-detection techniques or with intermittent-checking schemes.

It should be noted also that a number of authors have discussed the employment of parity checking in sequential networks for automatic error correction, i.e., for sequential fault masking, rather than for fault detection. Among these are Armstrong,[10] Rubio,[273] and Frank

and Yau.[85] While such possibilities should not be overlooked, it is our feeling that masking should not be applied except for relatively small networks. A particularly good discussion of this question can be found in Chapter VII of Pierce[243] (see also Sec. II-A-1).

2) State-Weight Checking

Another checking technique, in some respects analogous to state-parity checking but in others quite distinct from it, is implemented by restricting the valid states to lie among specified weight* classes. That is, we arbitrarily restrict our state assignments in the design of a sequential circuit to those state vectors containing specified numbers of 1's, say to the weights $\{w_1, w_2, \ldots, w_c\}$.† A variant of this technique is to restrict the weights of the state transition vectors $S \oplus S'$, rather than the states themselves.§ The simplest scheme of this sort uses a unit-distance code (Gray code) for transitions (i.e., only one flip-flop is allowed to change state in each transition); for a discussion of these codes, see Kautz.[150] This code seems especially suitable for counting circuits. More complicated versions are also possible and useful, wherein each transition may involve a small but fixed number of state variables changing.

A representative circuit for the implementation of a state-weight checking scheme is shown in Fig. II-A-30. For simplicity of description the network chosen is a scale-of-ten counter employing a distance-two, constant-weight code (two-out-of-five code) whereby five state variables are used in accordance with the following state assignment.

---

* The weight of a vector (in particular, of a state vector) is defined to be the number of 1's it contains.

† The reader will observe that the simplest case of parity checking (a simple overall parity check, such as P_1 in the preceding paragraphs) is equivalent to restricting state weights to even values, $\{0,2,4, \ldots\}$. The resemblance does not go much further, however.

§ This alternative technique was not discussed in relation to parity checking, since parity checking of the transitions $S \oplus S'$ is logically equivalent to checking the state vectors, S.

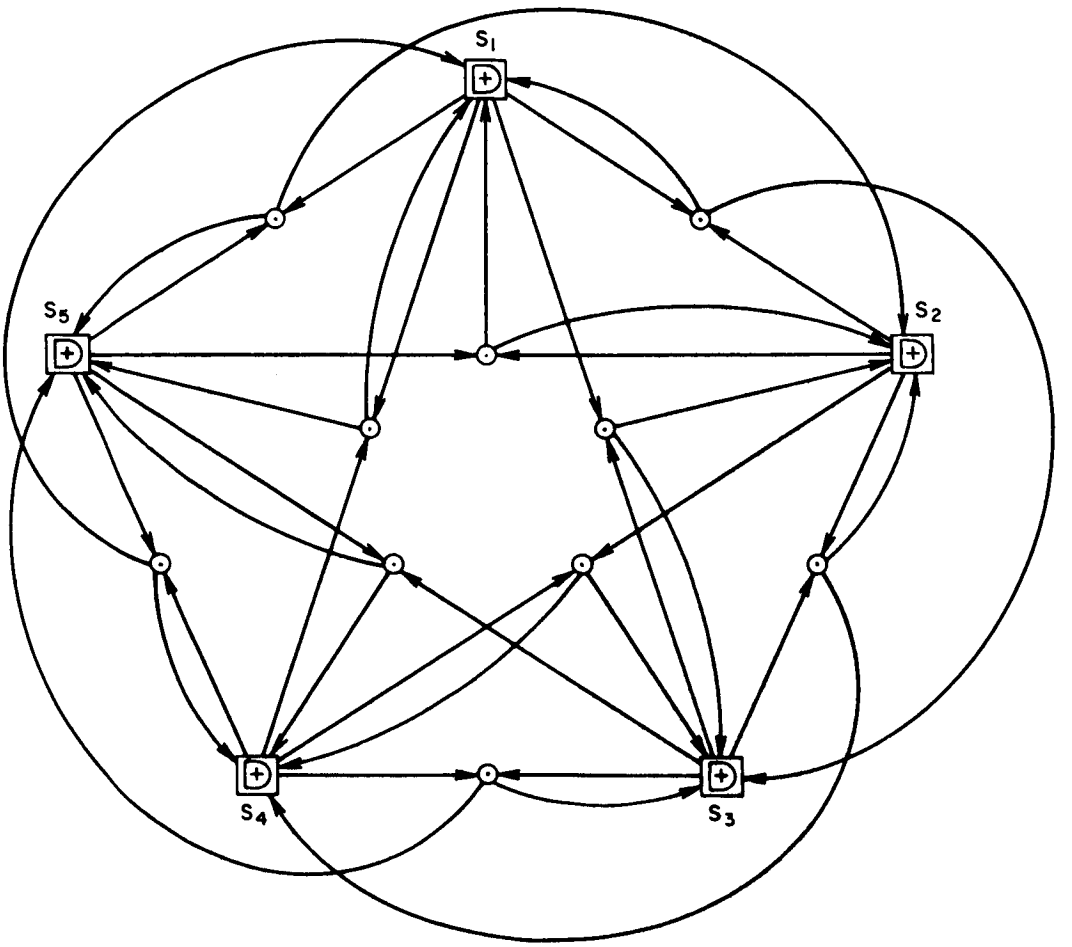


FIG. II-A-30 TWO-OUT-OF-FIVE COUNTER

State No.	S_1	S_2	S_3	S_4	S_5
1	1	1	0	0	0
2	1	0	1	0	0
3	0	1	1	0	0
4	0	1	0	1	0
5	0	0	1	1	0
6	0	0	1	0	1
7	0	0	0	1	1
8	1	0	0	1	0
9	1	0	0	0	1
10	0	1	0	0	1

It is natural to check the operation of this circuit with a two-out-of-five validity checker using the symmetric function $\phi_2(S_1, S_2, S_3, S_4, S_5)$. Zeros at the output of the validity checker then signal malfunction through the occurrence of an invalid state.

As indicated above, an alternative way to check the operation of this counter is by monitoring the weights of the "change" signals. If trigger (T) flip-flops are used for memory, then there will be exactly two active change signals at each step. If set-reset (R-S) flip-flops are used, there must be exactly one set signal and one reset signal at each transition. In all cases there are obvious ways to monitor these changes.

Circuits of the above types are attractive, perhaps, because of their logical simplicity. They also appear to be fairly competitive with parity-checked circuits in terms of efficacy in reducing the number of undetected faulty transitions. The two-out-of-five counter above uses 10 out of a possible 32 states. Thus, the fraction of undetected errors is 10/32 = 0.31 approximately. A binary counter would have required four nonredundant state variables; and with straight parity checking, one extra state variable would only reduce the undetected errors by a factor of 1/2. Thus the constant-weight counter is slightly superior in this case.

A more general scheme for state-weight checking is shown in Fig. II-A-31. Here the weights of the states are not restricted a priori. Full flexibility in state assignment to the nonredundant variables is available to facilitate economical (or otherwise desirable) synthesis. Instead, a set of check signals C_1, C_2, ..., C_r is generated (as in the parity-check scheme, though these C_i are not parity checks). The rule used in their generation is that the binary number represented by $C_r C_{r-1} \ldots C_2 C_1$, i.e., the quantity $\Sigma\, C_i 2^{i-1}$ is just equal to $w(S')$, the weight of the state vector S'. These check signals, C_i, are then applied (as memory excitation) to a redundant set of r delay elements. Thus, densely encoded information as to the weight of the next state vector is fed around the combinational logic and delay loop. At the delay element output terminals, the weight of the new state vector is checked against the indication provided by the binary number $C_r \ldots C_1$. Any failure of these numbers to agree results in a fault-detection signal.

The number r of redundant state variables required with this scheme is bounded above by

$$r \leq \langle \log_2 (n + 1) \rangle^*$$

where n is the number of nonredundant state variables. This relation follows from the fact that the weight of S can range in general from 0 to n, in the worst case. Of course, if the state assignments are suitably restricted to employ only a narrow range of weights, then r can be made smaller in particular cases.

The same principle can be applied to the checking of the weights of state transition vectors, rather than of the state vectors themselves. This alternative may be preferable in some cases, particularly when the state vectors can range over all weights from 0 to n but the transition vectors are limited to weights of, say, 1 or 2.

---

* $\langle x \rangle$ = smallest integer greater than or equal to x.

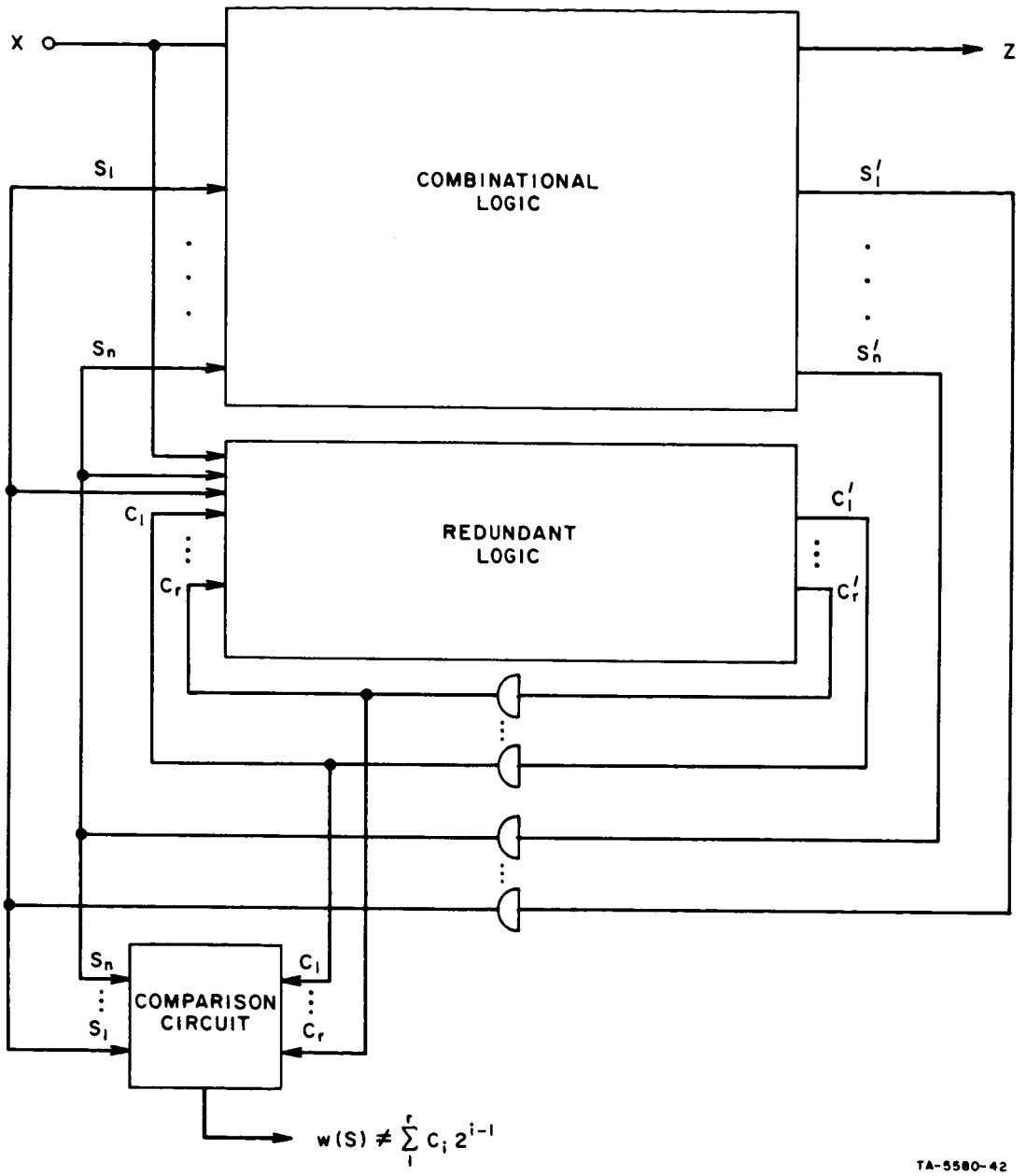


FIG. II-A-31 SEQUENTIAL NETWORK WITH STATE-WEIGHT CHECKING

We have suggested a good many schemes for the implementation of fault detection in sequential networks. Some of these are discussed only in relation to counting functions, yet they are applicable much more generally. For example, unit-distance coding can be used in the design of decoding trees.[73] What is still lacking is a clear understanding of how such schemes compare in cost, in reliability, and in general efficacy. The problem of efficient redundant state assignment for fault detection in sequential networks is intimately bound up with the general question of (nonredundant) state assignment; and the latter has long been known to be an extremely difficult problem on which progress has been made only recently.

Another large area ripe for future study is the design of sequential networks for easy diagnosability. The work of Kime[160] mentioned earlier may be relevant to this problem (though he was concerned with facilitating detection only).

B. Use of Codes for Storage and Arithmetic Operations

1. Introduction

Almost immediately after the use of error-correcting codes was proposed as a technique for achieving reliable communication over noisy channels, studies were conducted to determine the utility of codes for checking computer operations. Unfortunately, it appears that except for the checking of memory cells and for the checking of arithmetic operations, the use of codes is not feasible* for achieving reliable computation, because of the following factors.

(a) Serial checking, as is commonly employed for checking transient errors on communication channels, cannot be applied for checking component failures since generally permanent failures occur.

(b) Parallel checking is theoretically feasible, but in many cases single component failures result in errors on many data lines, dictating either the costly realization of networks wherein all outputs are independent or the use of codes with extended error-correction capability.

(c) Means must be provided for ensuring reliable operation of the encoding and decoding circuits, which are in many cases as complex as the protected networks.

Two cases where single component failures do not result in multiple data-line errors are in the memory section and the arithmetic section.

---

* Kautz[151] has described a class of codes which combines the properties of error checking, such as single-error detection, and unit distance--which means that successive code elements of the code sequence differ in only one bit. These codes are potentially useful for either an error-detecting code for an analog-to-digital converter (in the sense that the most likely binary errors are either detected or result in negligible analog errors) or as an error-detecting code for an asynchronous counter (in that a failure is detected which causes a flip-flop to fail to change state when it should or to change state when is should not). These codes have suffered from the unavailability of efficient encoding and decoding implementation, but Kautz has pointed out to the authors that by specifying a code which is somewhat sub-optimal it is possible to describe reasonable efficient encoding and decoding algorithms. With this advance these codes might prove to be useful for the indicated applications.

Avizienis[14] has proposed a system wherein the same arithmetic code is used for the checking of arithmetic operations and those storage operations which relate to the arithmetic data. The advantages of the scheme arise from the requirement of only a single encoder in the system, compared with the two encoders dictated by the use of two distinct codes. However, if the memory and arithmetic processor are checked separately then two arithmetic-type decoders must be included, each of which is more costly than a decoder for independent failures. The scheme also suffers in that it is difficult to specify different levels of protection for the memory and processor.

In the following two subsections we then discuss the use of different types of codes which are particularly appropriate for either arithmetic or storage. Although the decoders are still complex, some progress has been realized in the implementation of failure-tolerant decoders.

It should be noted that the following discussion is intended mainly to survey the prior coding research in order to distinguish cases where coding is pertinent to the realization of reliable computers, and also to distinguish future problems for research.

2. Codes for Checking Storage

It was indicated previously that most error-correcting coding techniques, as applied to computer circuits, are not attractive because of the need for costly extra circuits so as to provide failure-tolerant encoders and decoders. One possible exception to this tenet is for the error-control technique distinguished as threshold decoding,[5,352], wherein the estimated value of a particular information bit is determined by a majority vote of a function of the output bits from the memory. In this case the minimal implementations of the encoders and decoders are such that most patterns of failures are correctable, including failures in the decoders and encoders--not, of course, exceeding in weight the capability of the code. In this section we will briefly discuss the properties of codes which are amenable to threshold decoding, and also discuss the tradeoffs between error probability and additional memory locations required, attendant to the use of various error-correcting codes.

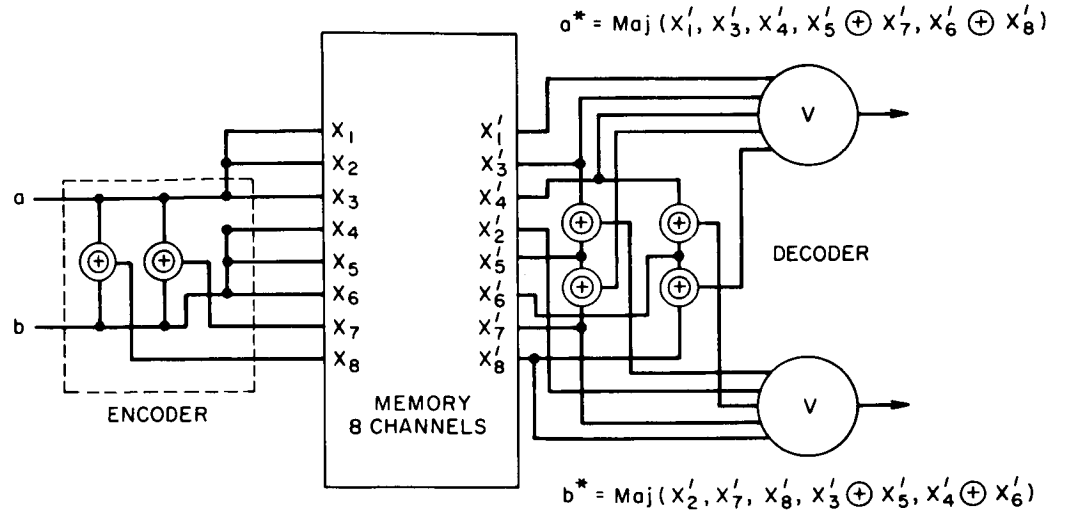
a. Threshold Decoding

As an example consider the implementation of a double error-correcting code for a memory-channel byte of two bits, as shown in Fig. II-B-1(a). In this case the optimum (least redundant) code requires 6 check digits. It is easily verified that all double failures occurring in either the encoder, the memory channel, or the exclusive-OR gates of the decoder are correctable. In addition, some of the failures occurring in the 5-input majority gates are also correctable if the triangular realization of Sec. II-A-2c is incorporated.

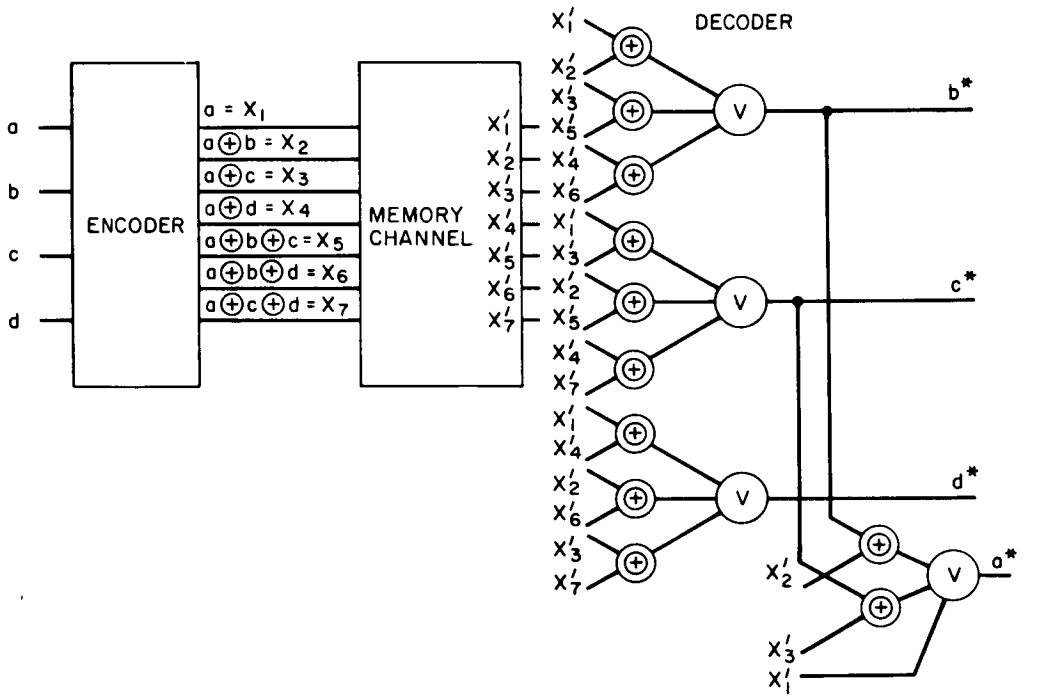
This feature of protection against encoder-decoder failures is not realized for implementation (such as those described by Kautz,[148] wherein a syndrome is first calculated and then utilized to uniquely distinguish the bit or bits in error. A single error in the determination of the syndrome will generally result in erroneous decoding.

It is of interest to observe the basis for the threshold decoding algorithm for this (8, 2) code. The code can be conveniently described in terms of the 6 × 8 parity check matrix (H matrix) shown below.

$$H = \begin{array}{c} \begin{matrix} x_1' & x_2' & x_3' & x_4' & x_5' & x_6' & x_7' & x_8' \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$



(a) (8,2) DOUBLE ERROR CORRECTING TB-5580-78



(b) (7,3) SINGLE ERROR CORRECTING TB-5580-29

FIG. II-B-1 THRESHOLD DECODING FOR MEMORY CHANNELS

The following set of 6 parity equations then must be satisfied by the code digits

$$x_1' \oplus x_3' = 0$$
$$x_1' \oplus x_4' = 0$$
$$x_1' \oplus x_2' \oplus x_5' = 0$$
$$x_1' \oplus x_2' \oplus x_6' = 0$$
$$x_2' \oplus x_7' = 0$$
$$x_2' \oplus x_8' = 0 \quad .$$

Transforming the above set of equations and appending the trivial identities $x_1' \oplus x_1' = 0$, $x_2' \oplus x_2' = 0$, we note that we can write a set of 5 equations for x_1' and a set for x_2', wherein no set variable appears more than once as an independent variable. These equations are:

$x_1' = x_1'$	$x_2' = x_2'$
$x_1' = x_3'$	$x_2' = x_7'$
$x_1' = x_4'$	$x_2' = x_8'$
$x_1' = x_5' \oplus x_7'$	$x_2' = x_3' \oplus x_6'$
$x_1' = x_6' \oplus x_8'$	$x_2' = x_5' \oplus x_8'$

If we let the estimates of x_1' and x_2' be the estimates of the information symbols a and b, then it is evident that any single or double errors will be masked by the majority decoding rule indicated in the figure.

For many codes, such simple encoder-decoder realizations are not possible, although it is possible to specify a "pseudothreshold" implementation in which the estimate of at least one bit is dependent upon the estimate of other bits. As an example, consider the realization shown in Fig. II-B-1(b)[5] of the Hamming single error correcting code, with 4 information bits and 3 check bits. As in the previous example most single failures occurring in the encoder and decoder are masked as well as, of course, those occurring in the memory channel.

At present it is not known what codes are implemented by the threshold-decoding technique, nor is it known what codes are implemented by the pseudothreshold technique as illustrated in Fig. II-B-1(b).* We have developed methods for specifying parity-check matrices based upon balanced incomplete block designs, for codes which can be guaranteed to be decodable by threshold decoding; but the research has not proceeded to the point where reporting is appropriate. It appears that these codes are somewhat less efficient than codes derived by other algebraic procedures--for example, Bose-Chaudhuri Codes.

b. Tradeoffs Between Memory Redundant Channels and Error Probability

Here we are concerned with the subdivision of the memory information channels into bytes to which we apply the error correction.

Although the reliability of a code group will increase with the number of errors that are correctable, the corresponding increase in complexity of the decoding equipment tends to decrease both the speed of operation and the reliability of the system. Hence, for the present examination, only single and double error-correcting codes will be considered.

Thus, for a given number of information channels, we wish to determine the reliability and the number of redundant channels resulting from the subdivision of the channels into code groups (bytes) of various sizes, for single and double error-correcting codes. It is assumed that channel failures are random and independent, and that all channels are equally reliable. Furthermore, we shall consider the reliability of a

---

* Massey[352] has shown that codes based upon maximal-length sequences can be threshold-decoded, and he has also shown that all Hamming codes are implemented by the pseudothreshold.

channel to be high, because only then is the use of redundancy beneficial for overall system reliability. In the analysis, the following definitions apply:

Q_s = probability that system fails
Q_b = probability that byte fails
q = probability that channel fails
p = 1-q
n = number of bits/byte
k = number of information bits/byte
$n-k$ = number of check bits/byte
b = number of bytes
$w = kb$ = number of information bits/system.

For single error-correcting codes,

$$Q_{b,1} = P(\text{exactly 2 errors}) + P(\text{exactly 3 errors}) + \ldots$$
$$= \binom{n}{2} q^2 p^{n-2} + \binom{n}{3} q^3 p^{n-3} + \ldots .$$

For $q \ll 1$, we approximate $Q_{b,1} \approx \binom{n}{2} q^2$.

Then $Q_{s,1} = 1 - (1 - Q_{b,1})^b \approx 1 - (1 - bQ_{b,1}) = b\binom{n}{2}q^2$.

For double error-correcting codes,

$$Q_{b,2} = p(\text{exactly 3 errors}) + p(\text{exactly 4 errors}) + \ldots$$
$$= \binom{n}{3}q^3 p^{n-3} + \binom{n}{4}q^4 p^{n-4} + \ldots .$$

For $q \ll 1$, we approximate $Q_{b,2} \approx \binom{n}{3}q^3$.

Then $Q_{s,2} = 1 - (1 - Q_{b,1})^b \approx 1 - (1 - bQ_{b2}) = b\binom{n}{3}q^3$.

Tabulations of $Q_{s,1}$ and $Q_{s,2}$ are given in Tables II-B-1 and II-B-2 respectively. The reliability Q_s, the total memory size nb, and A, the product of Q_s and memory size are given for a number of codes for bytes containing one to four information bits. The reciprocal of A is a useful measure (although it is clear that A is not universally applicable) of the effectiveness of a given redundancy scheme.

Table II-B-1

RELIABILITIES AND MEMORY SIZES
FOR SINGLE ERROR-CORRECTING CODES

n	k	b	nb	$Q_{s,1}$	$A_1 = Q_{s,1}^{nb/w^2q^2}$
3	1	w	3w	$3wq^2$	9
5	2	w/2	2.sw	$5wq^2$	12.5
6	3	w/3	2w	$5wq^2$	10
7	4	w/4	1.75w	$5.25wq^2$	9.2
8	4	w/4	2w	$7wq^2$	14

Table II-B-2

RELIABILITIES AND MEMORY SIZES
FOR DOUBLE ERROR-CORRECTING CODES

n	k	b	nb	$Q_{s,2}$	$A_2 = Q_{s,2}^{nb/w^2q^2}$
5	1	w	5w	$10wq^3$	50q
8	2	w/2	4w	$28wq^3$	112q
10	3	w/3	3.33w	$40wq^3$	133q
12	4	w/4	3w	$55wq^3$	165q

For equal memory size (3w each):

$$(3,1) \text{ single ECC,}^{*}\ Q_{s,1}(3,1) = 3wq^2$$

$$(12,4) \text{ double ECC, } Q_{s,2}(12,4) = 55wq^3 \ .$$

The ratio of probabilities of failure is

$$\frac{Q_{s,1}(3,1)}{Q_{s,2}(12,4)} = \frac{3}{55q} \ .$$

---

* It is recalled that an (n,k) error-correcting code (ECC) contains code words of length n with k information bits.

It is clear that for low probabilities of channel failures, i.e., $q < 10^{-2}$, double error-correcting codes are far superior to single error-correcting codes, ignoring the costs of encoding and decoding. For example, considering two schemes having equal total memory size, the (3, 1) single ECC and the (12, 4) double ECC, the ratio of system failure probabilities is 3/55q.

It was indicated that these studies ignored the complexity of the encoder and decoder circuitry in establishing a measure of the effectiveness of various error-correcting codes for memory channels. In future work measures which reflect these complexity factors should be established, at least for the codes discussed in this section. Since no complete theory* has been formulated concerning the encoder-decoder complexity as a function of the code, it will probably be necessary to establish the logical realizations of the circuits in the process of comparison. Some initial studies have indicated that the threshold-decoding scheme, besides providing for the masking of many encoder-decoder circuit failures, also appears to provide the least costly implementation (on the basis of a realization in terms of AND-OR gates).

3. Codes for Checking Arithmetic Operations

In this section we review the reliability techniques that apply specifically to the protection of arithmetic computations. Although this discussion does not mention the general reliability techniques that are described elsewhere in this report, it is tacitly understood that general techniques may be used in addition to or in place of the techniques discussed here.

Historically, reliable arithmetic computation techniques have closely paralleled reliable communication techniques. By redundant coding of operands it is possible to perform a consistency check on the results of arithmetic operations, much the same as the consistency

---

* For several codes Allen[5] has developed logical realizations of the encoders and decoders.

checks performed on code words that may have been corrupted in transmission through a noisy channel. The major efforts, therefore, have been to discover redundant number systems with good distance properties, to find easily implemented techniques for encoding, decoding, and performing consistency checks, and to find ways in which the redundant information can profitably be put to use.

Activity can be broadly classified into two areas: separable and nonseparable codes. Separable-code schemes are characterized by the use of check symbols that are treated as separate entities from the operands that they check. To detect computation errors that may occur during an arithmetic operation on a pair of operands, a special module operates on the corresponding check symbols of the operand pair and predicts the check symbol of the computation result. At the termination of the computation, a check symbol for the result is produced and compared to the predicted result, signalling a detected error if a disagreement is found. Figure II-B-2 shows a typical system based on a separable code. Error correction can be implemented, when code distance conditions permit, by calculating a correction term as a function of the predicted

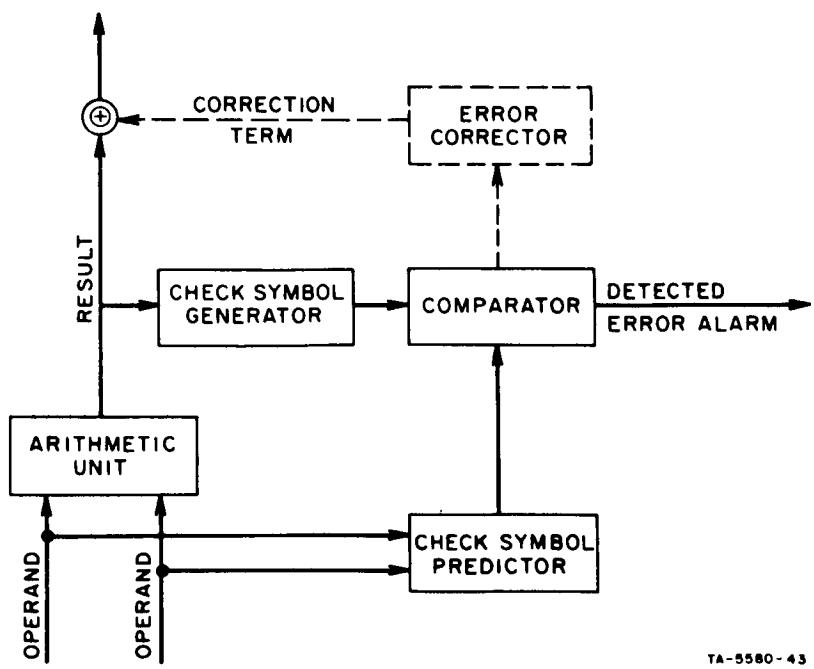


FIG. II-B-2 A SEPARABLE CODE SYSTEM

check symbol and the actual check symbol. This is indicated in Fig. II-B-2 by the dashed module labeled "error corrector."

Nonseparable codes are characterized by the coding of operands in a special form such that the results of correct arithmetic operations are again of the special form. Faulty arithmetic operations have a high probability of producing results that are not of the required form. To detect computational errors, the results of computation must be analyzed to determine whether they are properly coded. A typical nonseparable-code-based system is shown in Fig. II-B-3. In this system the analyzer signals a detected error when a result fails to satisfy the code requirements. If code parameters permit, the analyzer can also generate a correction term to correct the error, as indicated by the dashed line in the figure.

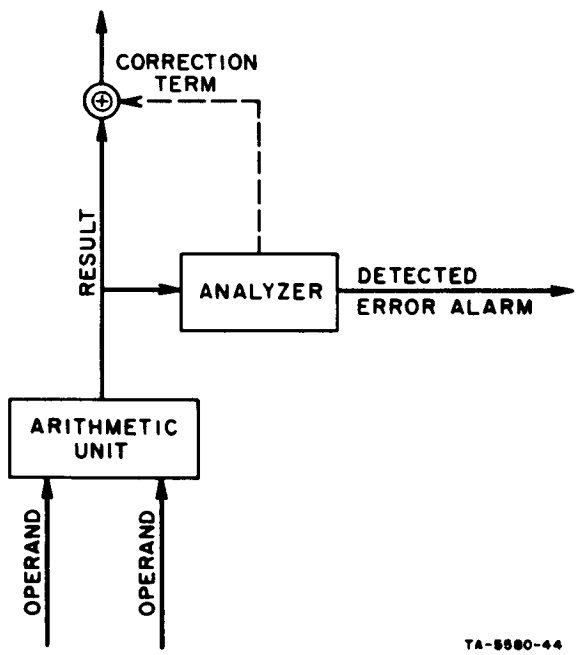


FIG. II-B-3 A NONSEPARABLE CODE SYSTEM

The remainder of this section describes each of the two areas in greater detail and concludes with a brief summary of the problems that remain to be solved to aid the implementation of these techniques.

a. Separable Codes

The theoretical foundation for separable code schemes has been formulated by Peterson[239] with the proof of the following theorem:

Theorem: Let $C(N)$ be the check symbol associated with the number N. For the addition operation $N_3 = N_1 + N_2$, let the check-symbol predictor perform the operation "*" such that the predicted check symbol $C(N_1 + N_2)$ satisfies

$$C(N_1 + N_2) = C(N_1) * C(N_2) \quad .$$

If there are fewer symbols in the check-symbol alphabet than there are integers in the permissible range of integers, then $C(N)$ must be the residue of N modulo b, where b is the number of symbols in the check-symbol alphabet, and "*" is addition modulo b.

The importance of this theorem is that it completely specifies the error-detection system given in Fig. II-B-2 when the operation is addition. The check-symbol generator in the figure computes the residue of the sum modulo b while the check-symbol predictor is a modulo b adder.

Since addition is the elementary arithmetic operation of a computer, all error codes must check addition. Checks of the other arithmetic operations can then be designed to make best use of the coding scheme which checks addition. Consequently, Peterson's theorem has far-reaching effects with respect to the design of systems to check all arithmetic operations.

Garner[89] has extended Peterson's work to show that a coding scheme that checks addition can be used directly to check multiplication. The predicted check symbol for multiplication is simply the product modulo b of the operand check symbols. Consequently, each of the components (except the error corrector) in Fig. II-B-2 is determined

when the operation is multiplication. The results of Peterson and Garner come directly as a result of the fact that addition and multiplication of integers in a computer are equivalent to the operations defined on a mathematical ring. Peterson's proof uses the fact that the check symbols must lie in a cyclic subgroup of the ring, while Garner notes that the subgroup must satisfy the stronger requirements of being an ideal of the ring. This completely determines the form of the operations on the check symbols that check multiplication and addition in the ring. Since subtraction in the ring is addition of the additive inverse of the minuend, subtraction is automatically checked if addition is checked.

Division is not, in general, a defined ring operation. Consequently, it is not possible to check division as directly as the other operations. The most thorough check is to check each addition and multiplication step of an iterative division. Since checking during the course of an iterative operation may substantially increase the time of the operation, an alternate approach is to perform a consistency check on the combination of divisor, dividend, quotient, and remainder at the completion of the operation. Since they satisfy the equation

$$n_1 = n_2 \cdot q + r$$

their check symbols must satisfy the same equation modulo b. Although this appears to be a valid check, it allows some errors to escape detection. Whenever one member of the pair n_2,q is 0 modulo b, the product $n_2 \cdot q$ is forced to be 0 modulo b, independent of the value of the nonzero member of the pair. Consequently, errors in n_2 or q cannot be detected in this scheme when one member of the pair takes on a correct value congruent to 0 modulo b. The choice of a method to check division is still an open question. Neither of the schemes described here is completely satisfactory. In some systems, the consistency check may suffice if the cost of undetected errors is negligible. In systems with more stringent reliability requirements, one is faced with the cost of step-by-step checking, unless a more satisfactory method can be found.

Thus far, we have discussed how arithmetic operations are checked with separable coding schemes. We have indicated that the check-symbol predictor is a modulo b adder and multiplier for checking addition and multiplication, respectively. Since these are commonly used devices, it is not necessary to consider how they might be implemented in this section. The check-symbol generator, on the other hand, is a device that lends itself to further study.

Since the check symbol for an integer n is its residue modulo b, the most direct way of computing the check symbol of n is to divide n by b, discard the quotient, and keep the remainder. Since division is usually much more complex than either addition or multiplication, the generation of a check symbol by division could be substantially more costly in terms of time or hardware than the operation to be checked. However, because of an observation by Rothstein[271] and extended by others,[100,14,215,216] it is possible to compute the residue without actually performing the division. In particular, if the modulus b is of the form 2^a-1 (more generally p^a-1 for base p computer representations) then

$$2^a \equiv 1 \bmod b$$

and therefore

$$(2^a)^r \equiv 1 \bmod b \quad .$$

Consequently,

$$k \cdot (2^a)^r \equiv k \bmod b$$

so that the extraction of a residue of a number reduces to the sum modulo b of the coefficients of the radix 2^a representation of the number. (This corresponds to the familiar process of casting out nines by summing digits in a decimal representation.) Hence, to compute a residue modulo $b = 2^a-1$, a binary representation is partitioned into a-bit bytes, each of which is treated as an integer modulo 2^a-1 and

summed modulo 2^a-1. Rothstein[271] and Germeroth[100] give other algorithms for the computation of the residue when b = 3. The JPL-STAR computer[14] actually uses this technique in its design for b = 15. The technique described above makes the calculation of check symbols practical without the expense associated with division hardware.

This completes the description of the devices associated with error-detection shown in the system in Fig. II-B-2. We now focus our attention on the correction of errors.

Before discussing the actual implementation of an error corrector, it is necessary to consider the error mechanism in some detail. Clearly no system can correct all errors, so it is desirable to be able to correct the most probable errors. It is possible to define an "arithmetic distance" such that the most probable errors have a small distance measure.

The following definitions are due to Peterson.[238]

Let the weight of a number n in base p be the least number of nonzero coefficients required to represent n as the polynomial

$$n = a_r p^r + \ldots + a_1 p^1 + a_0$$

where the coefficients are positive or negative integers with $|a_i| < p$.

Let the arithmetic distance between two integers n_1 and n_2 be the weight of $|n_1-n_2|$.

This definition of distance is somewhat different from the definition commonly used for transmission codes, but it is purposely so in order to account for the characteristics of arithmetic operations. During an addition operation, for example, the result of a single digit error can account for a burst of digit errors in the sum due to propagation of an error in the carry process. Such errors will show up as single errors (errors of weight 1) under the definition given above. In a parallel adder, the errors that correspond to a single component failure are single errors. Unfortunately, this is not true of serial

adders in which the multiple use of a single faulty component can cause a multiple error.

The necessary and sufficient conditions for single error correction have been derived by Brown[31] for nonseparable codes, but apply to separable codes also. In simple terms, a nonseparable code for the integers in the range $0 \leq i \leq 2^m-1$ can correct single errors if and only if each of the 2m possible single errors, $\pm 2^j$, $0 \leq j < m$, and the integer 0 have 2m + 1 distinct residues modulo b. To correct single errors, it is necessary to compute the difference between the predicted check symbol and the calculated check symbol. This gives the residue of the error. Since each of the correctable errors gives a unique residue, it is possible (usually by means of table look-up) to compute the correction term. Hence, the error corrector contains a modulo b adder and a table with 2m + 1 entries.

This completes the description of systems based on separable codes. Before proceeding to nonseparable code systems, it is worthwhile to mention some variations on the separable codes that have been described in the literature.

Although Peterson's theorem completely determines the nature of the separable-code system, it does not determine the nature of the number representation in the computer. Hence, the adders and multipliers need not be conventional. The most competitive choice of number representation next to radix representations is the residue representation.[91,297] This representation has certain advantages over radix representation with respect to elimination of carry propagations and the inherent modularity of the arithmetic-unit logic associated with the representation. Unfortunately, several operations such as sign detection, overflow detection, magnitude comparison, and division are much more costly in residue-based computers than in radix-based systems.[298,153]

Watson[320,321] and Moore[216] have investigated error-detecting and correcting redundant-residue representations which are similar to the separable codes described here. The redundant-residue systems are

characterized by modules like those shown in Fig. II-B-2 except that the arithmetic operations are performed by logic characteristic of the residue operations. Check symbols are residues, as in the radix representation, except that the check symbols themselves are represented by residues of moduli smaller than the check modulus b. Hence, the check-symbol predictor is a residue adder or multiplier rather than a conventional modulo b adder or multiplier. The process of generating check symbols from the residue representation is commonly termed "base extension." The simplest known method of base extension is much more complex than the residue-extraction method described above for radix representations (see Moore[216]). At best it requires several additions and a table look-up.

The use of redundant residues can simplify some of the problems associated with residue operations. The processes of sign detection and magnitude comparison, for example, are easier to implement with redundant residues than without.[216] Nevertheless, residue representation appears to be far less attractive than radix representation in general, while the high cost of base extension lends further support to the apparent unsuitability of residue representations for error-detection systems.

For historical reasons, it is pertinent to mention Garner's generalized parity scheme[90] in which the check symbols are parity checks on the operands. In this scheme the check-symbol predictor for an addition operation determines the parity of the sum from the parity of both addends and the generated carries. Because the carry-generation process itself is not checked by the code, it is not suitable for error-detection systems.

b. Nonseparable Codes

In view of material presented above, it is not surprising that the form of a nonseparable-code error-detecting system is almost completely determined by the properties of the addition and multiplication operations. To see this, let $F(n)$ be the coded form for the integer n, and note that $F(n_1) * F(n_2) = F(n_1 + n_2)$ where "*" is the operation on

code words that corresponds to addition. But this is precisely the condition that holds for check symbols of a separable code. Using this condition and others it follows that the structure required to check addition and multiplication is that the code words must lie in an ideal of a mathematical ring,[89] which is the same structure required for the check symbols of separable codes.

A suitable candidate, the AN or "linear residue" code, was first proposed by Diamond[60] and has subsequently been studied by Brown,[31] Peterson,[238] Henderson,[125,126] and Garner.[89] For these codes, the integer n is represented by the integer A · n where A is a selected constant. Since $An_1 + An_2 = A(n_1 + n_2)$, the arithmetic sum of two coded numbers is the code of the sum, so that ordinary addition can be used to sum two code words. Multiplication and division operations on code words are more complex, however, than the equivalent operations on uncoded integers. The product of two code words must be calculated by ordinary multiplication and reduced by a factor of A because $An_1 \cdot An_2 = A^2 n_1 n_2$. Division is checked by premultiplying the dividend by a factor of A and then performing ordinary division. Hence, to perform either multiplication or division, it is necessary to perform both operations on the code words. Avizienis[14] has used a clever technique for implementing the premultiplication by A required for division. When A has the form 2^a-1, the product n A · n can be computed by subtracting n from $n \cdot 2^a$, where the latter product is obtained by shifting n left a bit positions. This technique can also be used to encode numbers.

The consistency check for these codes is to determine the residue of the code word modulo A, and signal an error if the residue is nonzero. Note that the residue extraction can be performed by the technique described for separable codes if A is of the form 2^a-1 for binary computers. Similarly, a single error can be corrected by a table look-up or alternate calculation if and only if each possible single error and the integer 0 have distinct residues modulo A. In this case, as before, the definition of single error is an error of arithmetic weight one.

The implementation of a nonseparable coding scheme must include provision for coding and decoding numbers. Since these operations are multiplication and division, respectively, for the AN codes, the form of the operations is completely determined. There is special form of the AN code in which it is unnecessary to perform the division by A in order to decode a number. This case, the so-called "systematic code," is characterized by the fact that a subfield of the coded representation of a number is the uncoded representation of the number. Hence, decoding is accomplished by extracting the information subfield from the coded representation. Garner[89] shows that the format of a systematic code must be such that an integer n is represented by the left concatenation of a field c to the representation of n; i.e., such that n is represented by (c,n), so that decoding requires the extraction of the least significant bits. This quality also simplifies the coding somewhat in that only the most significant bits of the product A · n must be calculated in order to encode n. Several systematic codes have been constructed by Henderson,[125] and the constraints on code parameters for these codes have been described by Garner.[89]

Although we have briefly mentioned how to implement arithmetic operations with a nonseparable coding scheme, we have not discussed the problem of computing an additive inverse (negative). In conventional representations the subtraction of one number from another is usually implemented by computing the additive inverse and performing addition. Radix representations allow one to compute the inverse simply by complementing the stored form of the number, either with or without an end-around carry depending on the representation. The AN code does not, in general, allow this flexibility. However, it is always possible to find a constant B such that the coded form of a number is AN + B and the additive inverse of the number can be determined by complementing its representation. These codes were first reported by Diamond[60] with the description of the AN codes. Notice that arithmetic operations on code words coded in AN + B form usually require addition or subtraction of a contant B at

one or more points in the arithmetic processes that is otherwise not required for AN codes. Hence, it is not immediately clear that the advantage of simplified negation is worth the price of more complex operations.

Garner[89] has shown how to construct a systematic AN code with complement coding of the negative inverse by applying a special set of weights to the bit positions of the representations. This approach is particularly promising because it has the advantages of complement coding without requiring the inconvenient addition and subtraction of B. In order to implement the code, carries generated in the arithmetic units must be distributed according to rules derived from the weights of the bit positions. This does not appear to be as costly as the alternative of adding and subtracting B during operations. At least one code derived by Henderson[125] is a systematic AN code with complement coding, which uses conventional weighting of binary-bit positions. This kind of code is the most attractive of the many nonseparable codes described in this section.

It is apparently not feasible to use An and AN + B codes for the correction of two or more errors in arithmetic operations for several reasons, the most important of which is that the hardware required to correct more than one error is sufficiently complex to decrease the system reliability unless it can be protected by still more hardware. Peterson[238] gives a good table of single error-correcting AN and AN + B codes, while Massey's more recent paper[201] is a good up-to-date summary of AN codes. Garner[89] has the most complete collection of the many conditions on code parameters that determine separable, nonseparable, and (nonseparable) systematic codes.

c. Evaluation

To estimate the hardware costs of detecting errors with coded arithmetic logic, notice that the extra arithmetic units required for separable codes may be substantially smaller than the units they protect because operations are performed modulo b, where b is presumably smaller

than the system modules. Nonseparable codes do not require separate arithmetic units as such, but require that the principal arithmetic units be made to accept larger operands than otherwise required for the system. In both cases, the redundant arithmetic logic accounts for about a 30-percent hardware increase. Together with consistency checking comparison, and control logic, the amount of hardware required is about double that of an unprotected system. Imposed on the system is the additional cost of slower operating speed due to consistency checking operations.

The most competitive uncoded method of detecting errors is to use two identical arithmetic units and compare the outputs. The uncoded scheme requires at least twice the hardware of an unprotected system and results in a slight time penalty to make the comparison. It is difficult to compare the time penalties for the uncoded and coded protection schemes because part or all of the consistency-check and comparison operations may be concurrent with normal arithmetic-unit operations. The coded system may require less hardware, and could be used to pinpoint fault locations, provided that the arithmetic code has minimum distance three. A doubly redundant uncoded system cannot tell which of the two copies of an arithmetic unit contains a fault, although it can be used to indicate that a fault exists in a specific digit or carry line. On the basis of the estimates made here one cannot eliminate one scheme in favor of another. The point is that coding appears to be competitive with other reliability methods so that it should be considered with the other methods in system design.

With error detection, transient errors can be masked by repeating faulty operations. If repetition does not eliminate the error, then a faulty element is indicated. Hence, detection and repetition constitute a self-diagnosis system which, with replacement capability, constitutes an attractive approach to achieving system reliability. Since error detection alone cannot mask faults, it is necessary to include repetition, replication, or a suitable alternative in order to achieve high reliability.

Error correction is more difficult to evaluate. With error correction, it is possible to mask faults, except that the error-correction hardware itself must be checked. With the necessity to use redundant error-correction hardware, the attractiveness of the scheme is somewhat diminished. In order to decide whether or not error correction is desirable in a particular system, the designer should go through the exercise of comparing at least two competitive designs, one based on error correction and the other on replication or repetition with majority decision logic.

Of the coding schemes described here, the separable codes and the systematic AN codes are the most attractive. Both schemes offer the advantage of being able to identify the binary representation of a number in its coded form. This simplifies masking and shifting operations on numerical quantities. The principal difference between the two schemes is how the check symbol is formed. With separable codes, a predicted check symbol is calculated by a separate module. With systematic AN codes, the check symbol is integral to the coded form of the number, and is formed by the arithmetic unit during a normal operation. The two schemes require about the same amount of hardware to form the check symbol. Since the consistency check can be performed similarly for the two systems, they both require roughly the same amount of total hardware to implement. Separate code schemes offer more inherent modularity than the systematic AN codes, which may be an asset in systems that permit replacement of faulty modules. Again, the method that is best for a particular system must be selected on the basis of a detailed comparison of the two coding schemes against other methods of achieving reliability.

Among the outstanding questions that relate to the use of arithmetic coding are the following:

(1) Division checking is particularly unwieldy for separable codes. Are there techniques that can simplify this process?

(2) Are there ways to simplify the checking of multiplication and division for the AN codes?

(3) How can faults in serial operations be detected and/or masked?

(4) How can arithmetic codes be used to check non-arithmetic computer operations?

(5) Can arithmetic codes be profitably used to protect memory and input/output modules?

(6) The inherent modularity of residue number systems is still sufficiently attractive to keep interest in the idea alive. What techniques can be used to eliminate the problems of residue interacting operations?

(7) Several detailed systems designs should be developed using different methods of achieving reliability. This should shed light on the performance and cost of ultrareliable systems.

(8) It appears that the use of arithmetic coding is attractive for the location of failures in an arithmetic processor. Of course, error-correcting codes can be used which locate an error at a bit level, but the circuitry related to a bit of computation represents a circuit block which is not complex enough to conveniently reconfigure. A class of codes is required which can locate errors to within one of several consecutive bits.

III TECHNIQUES FOR DYNAMIC ERROR CONTROL

This chapter is concerned with techniques of logical analysis and design that are needed for the realization of computers in which the control of errors is dynamic, i.e., in which the logical interconnections among the components of the computer may be altered. In the case of autonomous error control, the error state of the computer is a subject of computation and control by high-level processes within the computer itself.

In this chapter we shall consider the various problems of analysis and design that arise in such systems. The first section deals with the overall design of the computer system, including the design of its structure, and the coordination of the various maintenance and computational processes. The second section deals with the design of tests for the detection and diagnosis of faults within the subsystem networks of the computer. The third section deals with the design of networks of the special kinds needed for the composition of the computer systems of interest.

In each section we attempt to characterize the problems of design in terms of their relevance to the overall objectives of system performance, in order to determine how well present engineering techniques satisfy the given design requirements, and to indicate what problems require further study. In some of the cases we present analytic solutions to several of the problems that were uncovered, and in others we present rough logical designs, in order to give concrete illustrations of general design approaches and to uncover problems of detailed design.

A. Problems of System Organization

This section considers the design problems relative to the structure and the operating modes of an advanced computer from the overall system point of view. In the first part, we examine the basic computational and maintenance processes that are desired and distinguish certain structural features that follow from these requirements. In the second part,

we examine the major maintenance processes in further detail and consider the problems of coordinating these processes. In the third part, we examine the major aspects of system structure and attempt to distinguish the problems of design of various components of the structure.

1. Basic Behavioral and Structural Characteristics of an Advanced Spaceborne Computer

In Sec. I-A-1 it was noted that computers for advanced, long-duration space missions will have to perform computations of great variety and complexity and with a range of priorities; that many of the computations will have to be performed at high speed and with large memory capacity; that the reliability of an electronic system with the required computational capability and mission time cannot be ensured without some degree of error control; and that the amount of human error control available will be very limited. We wish to determine how these characteristics affect the structural characteristics of a computer.

The requirements of complexity of computation and high performance clearly indicate the probable need for a high degree of parallelism of logical operations, although the degree of parallelism needed is not known at this time. It is appropriate to note the different kinds of parallelism that may be employed in future spaceborne computers. Some conventional and feasible domains of parallelism are: (1) the bits of a computer word, (2) the set of words in a vector, (3) the phases of an instruction cycle, (4) the set of instructions in a single program segment, and (5) several program segments belonging to one or more computations. The parallelism may have several forms: for example, the concurrency of operation may apply to all the elements of a single entity in the domain, e.g., all the bits of a single word; or it may apply, say, in an overlapping manner, to elements of several entities in a domain, e.g., the address calculation of one instruction and the arithmetic of a second instruction.

The need for a high degree of autonomous error control can have a substantial influence on computer structure. In Sec. II it was noted that logical fault masking, either fixed or adaptive, could be employed locally within a system to increase the reliability of a system without any need to substantially modify the system's basic logical structure.

However, there are two basic limitations inherent in the exclusive use of fault masking, both of which may be considered as inefficiencies in the use of redundancy. It will be seen that substantial modifications in system organization are needed to achieve error control that overcomes these limitations.

The first limitation of local fault masking is that it does not provide for the transfer of redundant equipment between functional locations; thus a functional location, such as a program-counter subsystem, may exhaust its fault-masking capability, while another location, such as a time-counter, may have a surplus of perfect parts. In general, there are many such functional locations in the central portion of a computer where failure is catastrophic for the system as a whole.

The second limitation of local fault masking is that it does not provide for soft failure, i.e., for the reallotment of computational resources among tasks according to their priority for the mission. It is well known[86] that the logic of general-purpose computation can be realized with a much smaller number of logic elements than is found in a modern computer. It may also be expected that a great range exists both in the value of the set of computational tasks and in the usable precision of their computations. It is thus seen that there exists a wide useful range for the exchange of equipment and performance in a complex spaceborne computer, and it is submitted that the design of an advanced computer should attempt to exploit this range to a high degree.

Translated into system terms, the overcoming of these limitations requires that the structure of the computer be reconfigurable. Thus, overcoming the first limitation calls for the capability of reassigning equipment among the functional locations within a given computer structure. Overcoming the second limitation calls for the capability of reorganizing the available equipment into a new general-purpose structure, and of modifying the programs so as to maximize the value of the computations performed. The key problems of design in achieving such capabilities are flexibility of structure, simplicity of diagnosis, and reliability of control.

Flexibility requires that the hardware should be capable of being logically interconnected in many useful ways in order to accommodate many fault patterns. This capability is enhanced in turn by a high level of modularity among functional units and by a high degree of uniformity in the structure of interconnections among the units.

Modularity, i.e., the use of a small number of different kinds of functional units or modules, increases the number of locations at which redundant equipment of a given type may be employed. Modularity is also consistent with technological considerations of reliable fabrication, as discussed in Sec. I-A-2. With the advent of complex monolithic arrays, it may be advantageous to employ a small number of complex module types that can be programmed by stored information or by external connection to perform one of a number of different functions in different functional locations. Uniformity of interconnection structure, e.g., as in cellular logic, would help to maximize the number of different possible configurations of functional units.

Simplicity of diagnosis requires that the fault status of the functional units of the computer should be accurately diagnosable in a short time and that the size of the program needed for such diagnosis should be small enough to be compatible with the combined resources of local memory capacity and the data link to a remote diagnosis facility. Reducing the number of module types also has the beneficial result of reducing the total size of the diagnostic program.

Reliability, in this instance, means that the control of such reconfiguration must be either fault-free or fail-safe, and that the reliability benefits of the reconfiguration scheme exceed the reliability losses produced by the added equipment. The reliability of switching and control is crucial to the whole approach of reconfigurability. It is well known that a reconfigurable system with perfect control and switching is superior to a fault-masking system, but the potential faults in the equipment needed for such functions may make the system less reliable than one in which the same amount of equipment is used in fault masking. High reliability of switching and control may be achieved by minimizing

the amount of equipment needed for a given complexity of function, and by the use of static or dynamic error-control techniques. It should also be noted that static error-control schemes may be useful in increasing the basic reliability of the modules and their intercommunication paths. The application of such means to particular control and communication structures is itself an important design problem.

In summary, it is suggested that in order to achieve the highest levels of reliable performance, an advanced spaceborne computer will need the following structural features to a high degree: parallelism of logical operation; modularity and programmability of functional modules; regularity and programmability of interconnection; and autonomous capability for fault diagnosis and control reconfiguration. It is also suggested that a number of error-control techniques, both static and dynamic, will need to be employed to enhance the reliability of basic functional units. It is not clear at this time that the use of redundant equipment in a reconfigurable structure will result in a more reliable system than the use of redundant equipment in a localized fault-masking system, nor is the optimum degree of reconfiguration for a given technology known. New schemes of system organization and network design for such reconfigurability are needed to permit a proper evaluation of this approach.

2. Organization of Basic Processes

In this section we wish to review the basic processes of general computation and maintenance computation that must be realized in spaceborne computer employing dynamic error control.

The basic computational processes in a general-purpose computer may be grouped as follows:

(G1) Executive: including the stepping of the major phases of an instruction cycle, control of "interrupt" action, communication with maintenance processes (i.e., for alarm) try-again, roll-back* and return.

---

* "Try-again," as suggested by the name, is an attempt to correct an error in a computation by repeating it, and "roll-back" is a return to a program step that preceded an error in order to regenerate (to the degree possible) information that was lost because of the error.

(G2) Instruction: including the determination of and access to an instruction, and the transformation of the address portion, e.g., by indexing or table look-up.

(G3) Operation: including the retrieval, computation, and distribution of operands.

(G4) Input-output: including selections or recognition of an active terminal, receipt of transmission of information, buffering, formatting, and preprocessing (e.g., integration).

The basic maintenance processes for a highly reconfigurable computer may be grouped as in the following list, which proceeds in order of increasing degree of system modification.

(M1) Passive error control: including localized fault masking and error correction.

(M2) Fault indication: including detection that an error has occurred, and the location of the general area of the fault that produced the error.

(M3) Transient discrimination: including attempts to correct an error by repetition of the general computational process.

(M4) Fault diagnosis: including the selection of and access to a subject logic network, presentation of test patterns, and retrieval and interpretation of responses, in order to detect, locate, and characterize faults.

(M5) Reconfiguration: including the generation of schemes for the re-allocation of hardware resources, the assignment of function for multi-functional modules, and the setting of interconnection paths.

(M6) Reorganization: including the generation of an alternative schemes system organization for realizing general-purpose computation, the relocation of data in storage assignment of functions and interconnections among modules, and the modification of program subroutines.

(M7) Alteration of tasks: including the determination of an appropriate allottment of the available hardware to the set of computational tasks.

In the order listed, the maintenance processes involve increasing losses in time, corresponding to the increasing seriousness of the fault conditions for which they are appropriate. It is therefore sensible to organize these processes in a hierarchy, so that the capability of accommodation of a given process may be fully utilized before employing a process of a higher order.

Process M1 is valuable in enhancing the basic reliability of the functional units of a computer, and it is especially needed for protecting those circuits that control the execution of the higher-order maintenance processes. Processes M2, M4, and M5 are essential to dynamic error control, and process M3, which must follow M2 if it is employed, is probably of value in space missions, in order to accommodate nonpermanent faults, such as transient errors in logic networks due to radiation bursts and power interruption, and data-sensitive errors in memory networks. Process M6 represents a higher-order capability that is not essential to dynamic error control, but which provides accommodation for more extreme error conditions. Its employment inevitably calls for some reduction in performance, hence process M7 must also be employed to some extent. Under some circumstances, process M7 may stand alone; for example, if a particular machine order is inoperable, it may be expedient simply to reduce performance for some task, rather than reconfigure the machine structure.

The addition of new processes brings new possibilities for error. Some policies that may help reduce errors due to failures in the maintenance processes are:

(1) Employ a maintenance process only when it is needed.

(2) Provide for remote human control of at least the initiation of a maintenance process.

(3) Provide many easy exists from a maintenance process to some stable (perhaps imperfect) operating configuration.

(4) Subdivide maintenance processes into small steps such that each one has only a limited effect on the system.

The design and organization of general computational processes is, of course, a highly developed art; but the design and organization of the maintenance computational processes is not well developed, especially for the present case, in which a high degree of autonomy is required. Further research is recommended to develop techniques for the design of these processes and the coordination of these processes with general computation.

3. Approaches to System Structure

a. Introduction

In this section we consider a number of possible approaches to the design of system structure, i.e., the assignment of functions to subsystems and the ordering of communication among subsystems.

The dominant qualities of the computers of interest, from a structural point of view, are the need for parallelism of computation within a reconfigurable structure and the distinctness of maintenance control. For the various functions of both general and maintenance computations, there is a choice as to the extent to which a given function is performed exclusively in a given network type. We shall consider how this choice appears in system structuring.

b. Approaches to Structural Parallelism and Functional Specialization for General Computation

In order to see how parallelism may be employed both for computation and for error control, it is instructive to examine the known schemes of parallelism for computation alone.

In the design of the conventional, serial computer (due to von Neumann), specialization of function is carried out to a high degree. Thus, as illustrated in Fig III-A-1, the functions of storage, processing, input-output, and control are realized in special networks (or subsystems). Figure III-A-2 illustrates three schemes for increasing the parallelism of some of these functions that have been realized in machines built within the past eight years. Figure III-A-2(a) illustrates parallelism in storage units. An early example of its use was in the Larc computer,[70] in which a number of units operated with overlapping access cycles. Figure III-A-2(b) illustrates parallelism in processors, and the distribution of control among processing units; an early example of this scheme is the Gamma 60 computer.[68] Figure III-A-2(c) illustrates parallelism in a combined storage and processing function. Such a system is often called an "associative" or "logic-in-memory" processor; Lee [179] conceived a machine in which the combined storage and processing elements connected essentially in a one-dimensional array, and Slotnick[288] conceived a machine

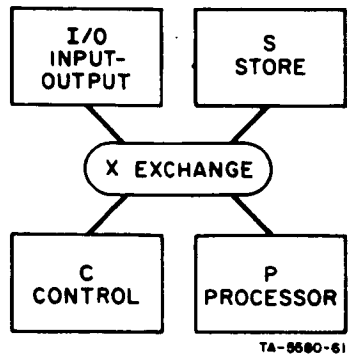


FIG. III-A-1 SERIAL COMPUTER
(von Neumann)

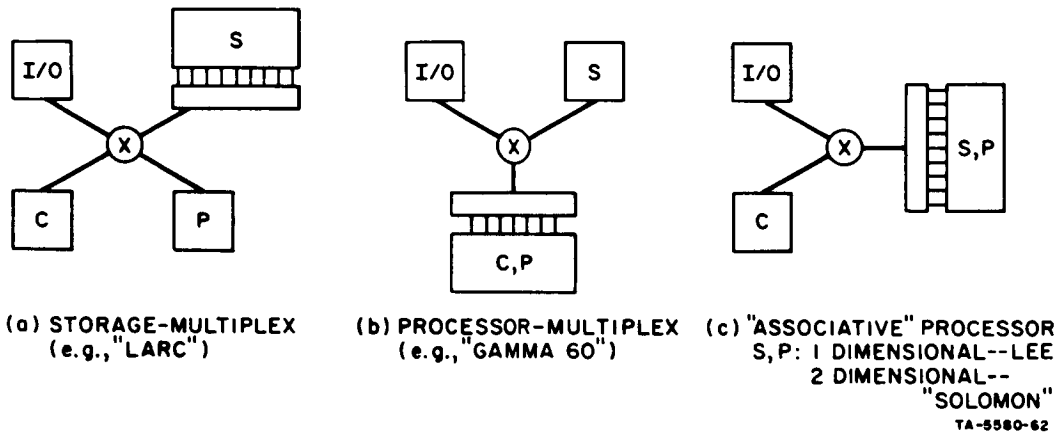


FIG. III-A-2 SCHEMES WITH LOCAL PARALLELISM

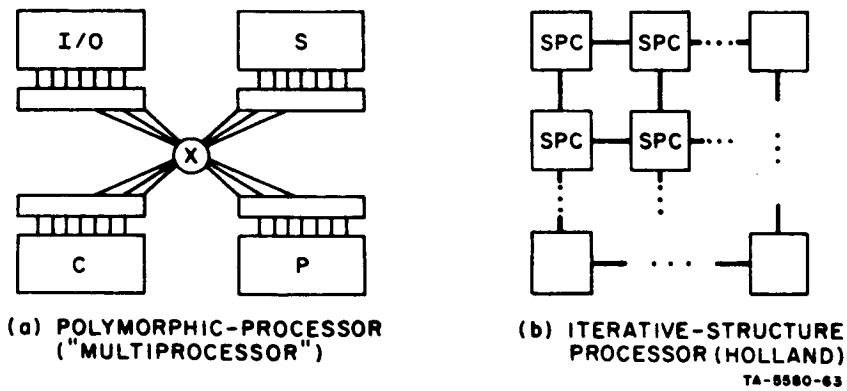


FIG. III-A-3 SCHEMES WITH GENERAL PARALLELISM

(SOLOMON) using a two-dimensional array. In all of these systems, either storage or control operates in an essentially serial-by-instruction mode. All of these schemes, of course, may be realized with different degrees of parallelism at the bit level.

Only two schemes of fully parallel processing have been discussed in the technical literature. One is the "polymorphic" scheme[349] (usually the structure connoted by the term "multiprocessor"), illustrated in Fig. III-A-3(a), in which the functional specialization of the von Neumann computer is preserved. The other, illustrated in Fig. III-A-3(b), is the iterative-structure processor (often called "the Holland machine"),[133] in which storage, processing, and control functions are combined in a cell and all cells are identical and regularly connected. In the polymorphic scheme, particular functional units that are to be combined may be chosen freely and the paths connecting them will usually be fixed for a whole computation, while in the iterative-structure machine the building of new paths among cells occurs at every instruction in order to retrieve the operands needed. The polymorphic scheme is superior with respect to efficient use of hardware, and the iterative-structure scheme is superior with respect to flexibility of reconfiguration. In both schemes, the parallelism in structure may be employed either for concurrency of independent or redundant computations, for reservation of spare parts, or for combinations of these functions.

c. Factors of Module Size and Specialization

A critical factor governing system design is the size of the basic unit of reconfiguration. It would seem to be an unnecessarily artificial constraint to assume that this unit should be identical to some traditional whole-function entity, such as a central processor or a register. Furthermore, there is no need to make such a unit identical to the contents of a single device package, since, in a multifunction monolithic device of the size to be expected within the next two years, discarding an entire module because of a single faulty output would be very wasteful of reserve logical capability.

d. A Suggested Model

Because of the high storage capacity required for the missions of interest, it is probably necessary to assume the continued use of specialized memory arrays, in order to achieve high density and low power consumption. These benefits apply both to magnetic memories and to monolithic semiconductor memory arrays. (The latter appear increasingly attractive for use in systems where volatility of information--i.e., loss of information with removal of power--is tolerable). A variety of memory types is likely to be needed, including variable destructive-read memories, variable nondestructive-read memories, and fixed-read memories. In addition, a special processing memory as shown in Fig. III-A-2(c) may be needed for some missions.

It is not clear to what extent a single structure can cover all computational functions, but it is clear that certain basic operations such as storage, counting, and addition occur both in processing and in control functions. Also, there is usually substantial freedom in the structuring of the networks that realize control functions, so that there could be a substantial similarity in the use of basic operations. There could therefore, be a substantial sharing in equipment; hence, an a priori separation of processing and control functions is not justified.

Since input-output functions have very special characteristics related to selection, formating, and buffering, a separation of input-output functions from other system functions appears justifiable.

The above considerations are embodied in a simple model of a parallel, reconfigurable computer illustrated in Fig. III-A-4. Sets of storage modules of various types S_1, S_2, ... and, optionally, a processing store SP are connected to X_c, a central exchange, by a multiple-channel switch or directory network. Similarly, sets of logic modules are connected to X_c by an interconnection network. The sizes of the storage and logic modules are not specified at this point; thus, for example, a given memory address range may cover a number of storage modules. Finally, a number of input-output interface modules are connected by a local interface exchange X_I to X_c, and by a terminal exchange X_T to the external terminals.

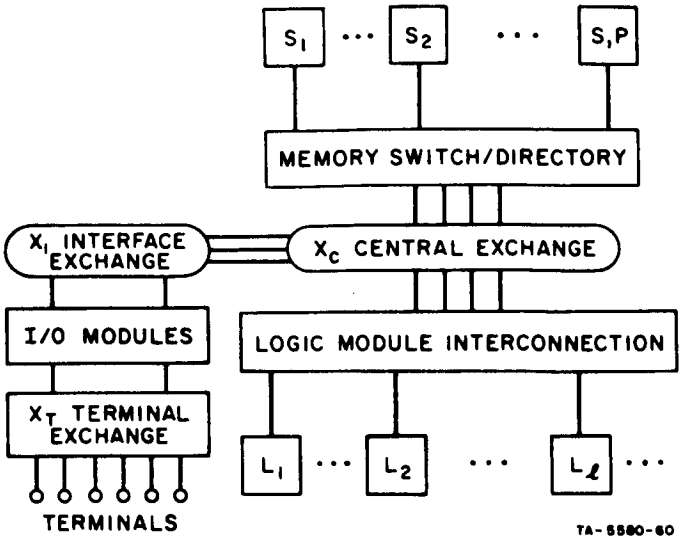


FIG. III-A-4 SCHEME WITH SMALL-MODULE PARALLELISM

This model is closer to a polymorphic structure than to an iterative structure. It may be expected that iterative structures will be advantageous for the realization of the interconnection networks and for other inherently iterative logical functions, due to their simplicity of testing and reconfiguration. The model does not yet reflect consideration of the maintenance computations. We consider these next.

e. Approaches to Structural Specialization for Maintenance Computation

Investigations have been made (e.g., by Manning)[197] of the extent to which a conventional computer is capable of diagnosing its internal faults. It has been found that the fraction of self-diagnosable faults is high, but that there are some faults that escape diagnosis either because the machine is blocked by the fault or because a given unit is logically essential to its own diagnosis. For autonomous operation it is therefore necessary to provide some equipment for the execution of the maintenance processes (i.e., diagnosis and reconfiguration) whose operation is not dependent upon the equipment being maintained.

A straightforward approach studied by Terris[303] is a system composed of a conventional serial-process computer, employed for general computation, in combination with a special primitive computer (called by Terris the "master machine"), employed for diagnosing faults and switching in spare parts within the first computer. Terris's design may be represented as in Fig. III-A-5, in which M is the master machine, P_G is the general computer processor and (S_G, S_M) is a single memory storing both general program variables and the maintenance program. The general computer accomplishes much of its own diagnosis, and the master machine serves to diagnose and remedy failures in a few critical operations in the general computer. The maintenance process may be considered a form of bootstrapping.

An interesting variation on the basic idea is described by Forbes, et al.,[84,1] in which a single bit-parallel computer can be partitioned into identical bit-parallel computers, each capable of serving as a diagnosing computer, and each with its own master machine.

As the complexity of the general computer grows, the size of the diagnostic and repair program will also grow. Furthermore, it is desirable to give special protection to the maintenance program, to avoid both accidental destruction of information and blockage of access to the information by failures in the general computer. Therefore, it would seem prudent to provide the maintenance section with its own program store, perhaps with most of it in the form of a nonvariable memory. Such a system is shown in Fig. III-A-6. In the figure, the various kinds of signals exchanged by maintenance logic and general computer logic--i.e., error alarm, test data, response, and configuration control--are distinguished as separate channels.

In extending the scheme to polymorphic (multiprocessing) computers, there is a choice as to whether the maintenance computer should exist as a distinct entity, or whether the assignment of maintenance functions to functional units should be subject to change even to the degree of flexibility that is provided for general computational functions

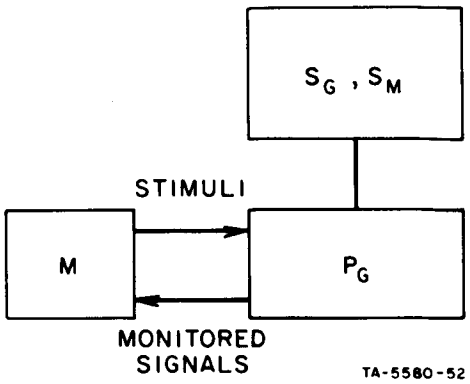


FIG. III-A-5 SYSTEM WITH SELF-DIAGNOSTIC COMPUTER AND MASTER MACHINE CONTROLLER

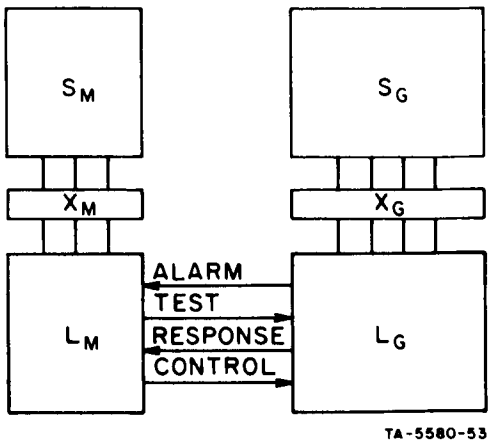


FIG. III-A-6 DISTINCT-MAINTENANCE-CENTER SYSTEM WITH SEPARATE WORKING AND MAINTENANCE COMPUTERS

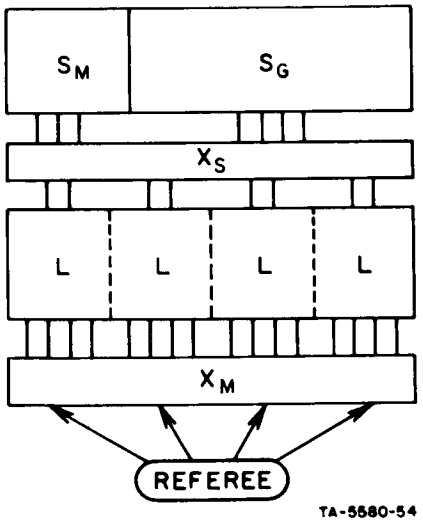


FIG. III-A-7 POLYMORPHIC SYSTEMS WITH FLOATING MAINTENANCE CONTROL

in modern polymorphic designs. A number of discussions of this idea have appeared (e.g., Joseph)[143] and a reliability model has been studied by Welch.[328] A sketch of such a system in terms of the polymorphic model previously developed is given in Fig. III-A-7. A special subsystem labeled "Referee" is provided, which has the task of assigning the maintenance role to a particular subcomputer. In this scheme the distinction between storage modules for maintenance and for general computation is preserved, but free access to all storage modules is provided to all logic subprocessors. Separate communication between modules for computation and for maintenance is provided by exchanges X_S and X_M respectively, although it is not clear that such separation is essential.

The choice between the two approaches is not an obvious one. Use of a special machine permits use of special measures to increase the reliability of the very critical maintenance function, i.e., reduction of its size (e.g., by making it highly serial) and application of high-order redundancy fault masking. Allowing the maintenance function to "float" among a set of identical processors has the advantages that maintenance computations may be performed with a higher logical power than in a primitive master machine, that a high order of redundancy is still available for protection of maintenance control, and that the computer has a homogeneous structure. Two design problems exist for which the costs of solution are not presently known; these are (1) the problem of protection against those faults within a processor that can block the transfer of maintenance authority, and (2) the provision of intercommunication among all processors for the special diagnostic and control information.

Future investigations of system organization should consider how the merits of the two approaches may be realized in an integrated structure.

f. Coordination of Information Types

It has been indicated that a self-diagnosing, reconfigurable computer should have a very uniform structure. At the same time, the number of information types that must be processed is very large. Some

of these types are: instructions, operands, memory, arithmetic, control, error indication, diagnostic tests, test responses, and configuration control.

Special care must be taken in order to avoid a proliferation of special codes, formats, and data paths within a computer. Avizienis[14] has indicated the benefits of uniformity of coding for transfer of operand information among the various functional sections of a diagnosable computer. Further work is needed to achieve uniformity among all the various types.

In addition to the problem of checking the correctness of information transfer, there is also the problem of ascertaining that a desired transfer did in fact occur. This may be facilitated by combining information messages into higher-order strings, perhaps containing a mixture of information types.

g. Problems of Subsystem Design

The novel structural features described in previous sections, together with new constraints and freedoms associated with developments in device technology, bring new problems of subsystem and network design. In the next sections of this chapter several of these problems will be explored. It may be expected that further investigations of these and other subsystem problems will place new requirements on the overall system structure that cannot be anticipated at this time.

Some of the important questions about subsystem structures are:

(1) In Module design: What shall be the sizes and the functions of the various modules? How shall fault-masking, error-detection, and fault-diagnosis aids be incorporated? How shall the reprogrammability and reconfigurability of modules be accomplished?

(2) Intercommunication-network design: How shall the network be designed so as to achieve high flexibility and programmability? How shall fault avoidance and fault masking be incorporated?

(3) Maintenance-control network design: How can the size of the network be reduced, while preserving capability for adequate bootstrapping of a complex computer? What is the best combination of static and dynamic fault masking to apply within the network?

B. Tests for Diagnosis of Fault Conditions

1. Introduction

A high degree of equipment reconfigurability implies the availability of accurate information as to the actual functional capability of the equipment. In ground-based computers, maintenance procedures may be conducted by intelligent technicians equipped with catalogs of fault syndromes, and capable of probing the structure of the computer at a great number of points. The spaceborne missions of interest to NASA may be manned or unmanned, and some useful communications with Earth may or may not exist. In all of these cases some autonomous on-board capability for the diagnosis of faults is either essential or extremely helpful, because of limited accessibility to test-points or because of a shortage of time. Limitations in storage capacity and accessibility in a spaceborne computer put high requirements on the completeness and efficiency of test schedules. Unfortunately, the test procedures that can be designed with present knowledge cannot be considered adequate.

In this section, the problems of designing test schedules for fault diagnosis will be considered in some detail. In parts III-B-2 and III-B-3, tests for combinational networks are considered, in which the choice of successive tests is unconditional or conditional on the responses to test inputs. A number of new results are included. In part III-B-3, the present state of the art of diagnosis is assessed and directions for further development are suggested.

2. Fault Diagnosis in Combinational Circuits Using Fixed Test Schedules

a. Introduction

To determine whether a network of digital-logic and storage elements is working properly, one may apply to the network all possible input combinations and sequences, and compare the resultant outputs with the corresponding correct outputs--using, for example, a faultless version of the same network. Any discrepancies indicate the presence of a fault.

Moreover, if the user is armed with a table showing which faults give rise to which patterns of discrepancies, he can readily distinguish any fault from the others--at least within a subset whose effects on the network output are identical. This procedure is perfectly valid for all types of digital networks--combinational and sequential, single and multiple output, gate-type and branch-type, binary and nonbinary, etc.--and all families of faults which have a more or less permanent effect on the behavior of the network.

Such exhaustive tests as these are usually much too long to be practical, however, and except for a few exceptional cases they are not at all necessary. It is normally possible to test a network for the presence or the presence and location of faults, by a schedule of tests which is shorter by one to several orders of magnitude than an exhaustive test.

In this part we consider the problem of devising economical test schedules for the diagnosis of fixed (i.e., nontransient) faults in an arbitrary combinational switching network. By "diagnosis" we mean to include the three separate cases when (a) any of a prescribed list of faults is to be merely detected, (b) the particular fault is to be located--that is, we are to determine which fault has occurred--and (c) the fault is located, but only to within the module (package on subnetwork) in which it occurred.

After defining these three minimization problems in mathematical terms, we proceed to a formal solution of each, for the case where it is assumed that the test schedule is fixed--that is, when the choice of the succession of test inputs which are applied to the network does not depend in any way upon the outcome of the tests. It is then shown (Part 3) that shorter test schedules can be expected for fault location when this assumption is not made--that is, when the choice of which test input to apply at each step in the testing is allowed to depend upon the outcomes of previous tests. A solution is offered for this case of serial testing. In describing these solutions, principal attention is given to single-output, binary networks; extension to the multioutput case is not difficult, and is described later. Nonbinary networks can also be handled easily.

The fixed-schedule solutions offered here may be considered to be satisfactory for derivation on a digital computer, for any combinational network having up to eight or ten inputs, several outputs, and about 100 faults. While some much larger networks can also be handled, procedures are presently lacking for generating even reasonably good test schedules for very large arbitrary networks. It is this remaining problem, as well as the problem of fault diagnosis in sequential networks, which may be considered to be the most important subjects for further research in this area. For a discussion of this problem, see Sec. II-A-3.

Most of the procedures to be described below are contained in the literature, but in a context having nothing to do with fault diagnosis. Consequently, the pertinent parts of them have been collected here, using a common notation and viewpoint with some original extensions and evaluations.

b. Formulation of the Problem

Given a single-output combinational network, there is no conceptual difficulty in imagining that an analysis of it has been conducted, in order to determine the effect on its output of each of various hypothetical faults. The results of such an analysis may be expressed in a multi-output table of combinations such as the one below.

x_n	...	x_2	x_1	f	f_1	f_2	...	f_j	...
0	...	0	0	0	1	0	...	0	...
0	...	0	1	1	1	0	...	0	...
0	...	1	0	0	1	0	...	1	...
	⋮								
1	...	1	1	0	0	0	...	1	...

The x_1, x_2, ... x_n are the input variables to the network; $f = f(x_1, x_2, \ldots x_n)$ is the fault-free (correct) output; and f_1, f_2, ...f_j . . . are the erroneous outputs, each corresponding to one of the possible faults which the desired diagnosis schedule is supposed to check. The left side of the table simply lists all 2^n possible combinations of the input variables. Note that no assumptions have (yet) been made about the nature of the faults--whether they are due to isolated or multiple component failures, to open or short-circuited devices or conductors, to short circuits between separate parts of the network, or to either sudden failure or slow degradation.

It will be convenient to reduce this table somewhat before stating the problem formally, as follows.

(1) Suppose some column f_j is identical to column f. This indicates that the jth fault has no effect on the network output, so that there is no way--and in fact, no need--to detect its occurrence. The column f_j may therefore be deleted from the table. This type of condition can occur either if the network is redundant or if certain of the faults cause local logical changes which leave the output the same.

(2) Suppose that two columns f_j and f_k are identical. This indicates that two different faults have the same effect on the network output, and for purposes of detection and location they must be treated together. One of the two columns may therefore be deleted from the table. It is easy to imagine how this condition could arise in practice.

After any such deletions, all of the columns $f = f_0$, f_1, f_2, ... f_m (say, for m distinguishable faults) will be different. We may collect these m + 1 columns into a 2^n-row binary fault matrix or fault table F:

$$F = \begin{bmatrix} 0 & 1 & 0 & \ldots \\ 1 & 1 & 0 & \ldots \\ 0 & 1 & 0 & \ldots \\ \vdots & & & \\ 0 & 0 & 0 & \ldots \end{bmatrix}$$

If a fixed schedule of input tests is to be employed to check a possibly faulty network, we are interested in economizing on the number of different test inputs--i.e., the length--of such a test. The problem is therefore one of selecting a minimal subset of rows of the matrix F that preserves a certain degree of distinguishability among the columns. More precisely, for the detection of the presence of any of the m faults, we want to delete from F as many rows as possible, so that:

The first column is different from all other columns.

For the location (as well as the detection) of the m faults, we want to delete from F as many rows as possible so that:

Every column is different from every other column.

Finally, a third minimization problem of interest arises from the common situation in which the faults are classed according to the module, package, or subnetwork in which they occur. Thus, if it is desired to locate a fault (column of F) only to within its preassigned module class, we want to delete from F as many rows as possible so that:

Every two columns which fall in different module classes are different.*

For this condition, the f column should be treated as belonging to a separate module class.

These three problems of fault detection, fault location, and fault location-to-within-modules correspond conceptually to the problems of error detection, error correction, and error location, respectively, in error-checking codes. Unfortunately, this appears to be about as far as this analogy can be carried.

One assumption must be made about the nature of the faults if a test schedule is to have meaning: we must assume that any fault which is to be detected or located has a duration in the network which is no less

---

* If columns f_j and f_k in simplification (2) above fall in separate module classes, neither should be deleted, but this requirement should then be relaxed to exclude this particular column pair.

than the interval of time over which the pertinent test inputs are applied. In practice, this means that the diagnostic methods described here are limited to fixed (i.e., permanent and semipermanent) faults. Some other means must be employed to protect the network against the effects of any transient or intermittent faults which are deemed likely to occur.

c. Formal Solution Using the G-matrix

We now show how the fault-detection problem and both fault location problems may be converted to familiar switching minimization problems.

Since it is only the distinguishability of certain columns of F which is at stake, we may conveniently express the distinctness condition in terms of a matrix G, each of whose columns is the modulo-2 sum of a different pair of columns of F that are supposed to remain different. That is, considering the same row of both F and G, a 1 is entered in the column of G labeled with the pair (i, j) if the digits in the two columns of F labeled f_i and f_j are different, and 0 otherwise. Under deletion of corresponding rows of F and G, two columns of F will then remain distinct if and only if the corresponding single column of G does not become a column of all 0's. Thus, the three conditions on F stated in the last section for fault detection, location, and location to within modules may be expressed as a single condition on the G-matrix; namely:

Delete from G as many rows as possible, so that every column is non-zero } Condition X

In the case of fault detection, the G-matrix (G_D, say) has just m columns, one for each column pair (f_0, f_j) in F (j = 1, 2, .. m). For the example used above, we have

$$G_D = \begin{array}{c} \begin{matrix} 01 & 02 & .. & 0m \end{matrix} \\ \begin{bmatrix} 1 & 0 & \ldots \\ 0 & 1 & \ldots \\ 1 & 0 & \ldots \\ \vdots & & \\ 0 & 0 & \ldots \end{bmatrix} \end{array} .$$

For fault location, the matrix G_L has $\binom{m+1}{2}$ columns, one for each column pair (f_i, f_j) in F (i, j = 0, 1, 2, .. m; i ≠ j):

$$G_L = \begin{array}{c} 01\ 02\ 12\ \ m-1,m \\ \left[\begin{array}{cccc} 1 & 0 & 1 & \ldots \\ 0 & 1 & 1 & \ldots \\ 1 & 0 & 1 & \ldots \\ \vdots & & & \\ 0 & 0 & 0 & \ldots \end{array}\right] \end{array}$$

In similar fashion, the matrix G_M for fault location to within modules is of intermediate width, having one column for each pair of columns in F which belong to different module classes.

Condition X expresses precisely the problem of finding a minimal prime-implicant cover of a given switching function from its prime-implicant table.[246] Good solutions to this problem are well known, and have been programmed for execution on computers for quite large tables.[254,105,44] We describe here a version of this procedure which is adequate for solving only simple problems by hand, but which nevertheless illustrates well the two main steps in all of the programmed algorithms:

(a) Simplification of the table to delete certain superfluous rows and columns

(b) Final selection of one or more minimal row subsets from the residual table.[206]

The justification of the following simplifications (a) is fairly obvious:

(1) Delete any row whose 1's all fall in the same columns as the 1's in some other row. That is, delete any row which is covered by, or is the same as, some other row.

(2) Delete any column which has 1's in all of the rows in which another column has 1's. That is, delete any column which covers, or is the same as, some other column.

These steps may be applied in any order until neither is applicable. The resultant matrix (G^*, say) has distinct rows and distinct columns; also, no row covers another row, and no column covers another column.

The selection (b) of a minimal row subset S is made by first labeling the rows of the simplified matrix G^* with binary variables a, b, c, ... each of which is to have the value 1 if its row is to be included in S, otherwise 0. We now form a Boolean function $L(G^*)$ as a product of sums, one sum per column of G^*, such that each sum contains just those row variables assigned to rows in which the corresponding column of G^* has 1's. The function $L(G^*)$ will therefore have the value 1 when and only when a sufficient subset of row variables a, b, c, ... have the value 1--namely, when every column of G^* is represented.

Expansion of this product of sums into a sum of products then expresses as individual products all of the alternative row subsets which satisfy the column condition. This allows one to select for S any one of the products which has the least number of variables.

As an example, consider the following matrix for n = 3, m = 7:

$$F = \begin{array}{c} \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{matrix} .$$

For fault detection we list the column sums (modulo 2) for the seven column pairs (0,1), (0,2), ... (0,7), to get

$$G_D = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{matrix}$$

The simplification step (1) first allows deletion of rows d and e, which are covered by rows f and a, respectively; then columns 3, 5, and 7, which cover column 2, may also be deleted by simplification step (2). This leaves:

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{matrix} a \\ b \\ c \\ f \\ g \\ h \end{matrix}$$

Rows c, g, and h (covered by row a), then the last two columns, may also be eliminated:

$$G_D^* = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{matrix} a \\ b \end{matrix} = \begin{matrix} f \end{matrix}$$

With the rows labeled as indicated, we obtain simply

$$L(G_D^*) = a\,(b \vee f)$$

$$= ab \vee af \quad .$$

One minimal set S therefore consists of rows a and b. The minimal F-matrix is just

$$F_{Dmin} = \begin{matrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{matrix} \quad .$$

and the list of input tests to be applied is simply

	x_3	x_2	x_1
a	0	0	0
b	0	0	1

.

For fault location, we sum all possible column pairs of F. The same example of an F-matrix yields the following G-matrix:

$$G_L = \begin{matrix} {}^{01}{}_{02}{}^{03} \ldots {}^{07}{}_{12}{}^{13} \ldots & & & & & & & & & & & & & & & & & & & & & & & & & & & & \ldots\ {}^{56}{}_{57}{}^{67} & \\ \left[\begin{matrix} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \end{matrix}\right] & \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{matrix} \end{matrix} \quad .$$

Using first the columns of lowest weight, most of the other columns may be deleted. Then rows d and e may be eliminated to give:

$$G_L^* = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \begin{matrix} a \\ b \\ c \\ f \\ g \\ h \end{matrix} .$$

Thus,

$$L(G_L^*) = (a \vee c)(b \vee f)(c \vee f \vee g)(a \vee g)(g \vee h)(c \vee f \vee h)(a \vee h)(b \vee c)$$

$$= (a \vee cgh)(b \vee fc)(c \vee f \vee gh)(g \vee h)$$

$$= abcg \vee abch \vee abgh \vee \ldots$$

A minimal subset S of input tests for fault location therefore consists of rows a, b, c, and g. The minimal F-matrix is therefore:

$$F_{Lmin} = \begin{vmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{vmatrix} ,$$

which corresponds to the schedule of test inputs:

	x_3	x_2	x_1
a	0	0	0
b	0	0	1
c	0	1	0
g	1	1	0

.

Both McCluskey[205] and Gill[104] have provided solutions to the fault-location problem, in the course of solving a seemingly unrelated problem in pattern recognition. The solution described above follows that of McCluskey. Gill's procedure is recursive. He shows how to

generate all possible solution subsets of rows of F for the first k columns only, from a listing of all such subsets for the first k-1 columns. (Nonminimal subsets are included, but any subset which contains another is not listed.) The list for k = 2 is easily formed by inspection, and the procedure is repeated successively for k = 3, 4, ... m + 1. The method involves a tremendous amount of bookkeeping, even for small problems, and cannot be considered to be very practical for present purposes.

For location of faults to within module classes, suppose that in the above F-matrix faults 1, 2, and 3 are associated with the same module, as are faults 4, 5, 6, and 7. The G-matrix for this case (G_M, say) is therefore the same as G_L, except that certain columns representing pairs of faults within the same module need not be included: (12), (13), (23), (45), .. (67). This leaves a narrower G-matrix:

$$G_M = \begin{array}{c} {\scriptstyle 01\ 02\ 03\ \ldots\ \ 36\ 37} \\ \left|\begin{array}{cccccccccccccccccc} 1&0&1&1&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0 \\ 0&1&1&1&1&0&1&1&1&0&1&0&0&1&0&0&0&1&0 \\ 1&0&1&0&1&0&1&1&0&1&0&0&1&0&1&1&0&1&0 \\ 0&0&1&0&1&1&1&0&1&1&1&0&1&1&1&1&0&0&0 \\ 1&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0 \\ 0&1&1&0&1&1&1&0&1&1&1&1&0&0&0&1&0&0&0 \\ 0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \\ 0&0&1&1&1&0&0&1&1&0&0&1&1&0&0&0&0&1&1 \end{array}\right| \end{array} .$$

After one cycle of column and row deletions, we obtain the matrix

$$\left|\begin{array}{ccccc} 1&0&0&1&0 \\ 1&0&1&0&0 \\ 0&1&1&0&0 \\ 0&0&1&1&1 \end{array}\right| \begin{array}{c} a \\ c \\ f \\ g \end{array} .$$

Subsequent simplifications yield

$$G_M^* = \left|\begin{array}{ccc} 1&0&0 \\ 0&1&0 \\ 0&0&1 \end{array}\right| \begin{array}{c} a = c \\ f \\ g \end{array} .$$

Hence

$$L(G_M^*) = (a \vee c)fg$$

$$= afg \vee cfg \quad .$$

Thus, a minimal F-matrix is

$$F_{Mmin} = \begin{vmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{vmatrix}$$

and a minimal test schedule is

	x_3	x_2	x_1
a	0	0	0
f	1	0	1
g	1	1	0.

It should also be pointed out that the procedure described above involving the G-matrix can also be applied to a fourth problem in fault diagnosis--namely, the problem of testing for the presence of one particular fault (with output f_k, say) which one might suspect to have occurred, it being already known that some fault has occurred.[41] In this case it is only necessary to form G from those column pairs (i,k), i = 1, 2, .. k - 1, k + 1, .. m. The matrix G will then have m - 1 columns. This situation is identical to fault detection, except that attention is focused on column k instead of column 0 of F.

d. Simplified Solution Using the $\hat{W}$-Matrix for Fault Location

The above example seems to be typical of most fault-location problems, in that considerable simplifications can be made in the matrix G. This is to say that the distinctness of most of the column pairs in F is taken care of automatically, if only a certain smaller subset of column pairs can be guaranteed to be distinct. One way to identify

these critical column pairs is to display the weights (number of 1's) of the columns of G in the (m + 1)-by-(m + 1) matrix

$$W = F^t F,$$

in which F^t indicates the transpose of F and matrix multiplication is carried out with exclusive-OR (modulo-2) addition used in place of element multiplication. Each off-diagonal entry w_{ij} of W is therefore the number of 1's in that column of G corresponding to column pair (i,j) in F. (The matrix is symmetric, so only the upper triangle of entries need be calculated.)

For the running example, this W-matrix takes the form

$$W = \begin{vmatrix} - & 3 & 2 & 6 & 4 & 6 & 4 & 5 \\ & - & 5 & 5 & 5 & 7 & 5 & 5 \\ & & - & 4 & 4 & 4 & 4 & 4 \\ & & & - & 4 & 2 & 4 & 2 \\ & & & & - & 4 & 4 & 4 \\ & & & & & - & 4 & 2 \\ & & & & & & - & 2 \\ & & & & & & & - \end{vmatrix} .$$

The column pairs corresponding to low-weight entries in W are certainly good candidates for the subset of critical column pairs for G*. Inspection of F now allows many of the noncritical column pairs to be excluded from further consideration; this exclusion may be conveniently indicated by simply deleting the corresponding entries in the W-matrix. In this manner, the formation of the G-matrix may be postponed until its width has been reduced well below the large value of $\binom{m+1}{2}$ columns.

As a computational convenience, it may be desired to enter not just the number w_{ij} of differing digit pairs between columns i and j, but the labels of the particular rows in which these columns differ. Selection of an entry as "critical" then allows one to delete immediately by inspection all other entries which contain the same row labels.

For the example, this labeled W-matrix ($\hat{W}$, let us say) becomes

$$\hat{W} = \begin{bmatrix} - & ace & bf & abcdfh & abgh & bcdfgh & adfg & abcdfg \\ & - & abcef & bdefh & bcegh & abdefgh & cdefg & bdefg \\ & & - & acdh & afgh & cdgh & abdg & acdg \\ & & & - & cdfg & ag & bcgh & gh \\ & & & & - & acdf & bdfh & cdfh \\ & & & & & - & abch & ah \\ & & & & & & - & bc \\ & & & & & & & - \end{bmatrix} .$$

(The compact listing of row labels in each entry is not meant to have any significance as a product.) Selection of the shortest entries bf, ag, gh, ah, and bc as critical now permits numerous deletions, leaving:

$$\hat{W} = \begin{bmatrix} - & ace & bf & - & - & - & - & - \\ & - & - & - & - & - & cdefg & - \\ & & - & - & - & - & - & - \\ & & & - & cdfg & ag & - & gh \\ & & & & - & acdf & - & cdfh \\ & & & & & - & - & ah \\ & & & & & & - & bc \\ & & & & & & & - \end{bmatrix} .$$

Similarly, entry cdefg may be deleted, since it includes entry cdfg. A deletable row of G (like $\underline{d}$ or $\underline{e}$) is readily identified in $\hat{W}$ as a letter which always occurs along with some other letter; such letters may be removed:

$$\hat{W} = \begin{bmatrix} - & ac & bf & - & - & - & - & - \\ & & - & - & - & - & - & - \\ & & - & - & - & - & - & - \\ & & & - & cfg & ag & - & gh \\ & & & & & acf & - & cfh \\ & & & & & - & - & ah \\ & & & & & & - & bc \\ & & & & & & & - \end{bmatrix} .$$

G_L^* may now be formed from the remaining entries in $\hat{W}$. Actually, however, it is easier to write down $L(G_L^*)$ directly from $\hat{W}$, bypassing the formation of G_L^*:

$$L(G_L^*) = (a \vee c)\ (b \vee f)\ (c \vee f \vee g)\ \ldots\ (b \vee c)$$

$$= abcg \vee abch \vee \ldots$$

as before.

If faults within the same module class need not be distinguished, then large blocks of entries in $\hat{W}$ may be deleted (or simply not calculated) right at the start:

$$\hat{W} = \left[\begin{array}{c:ccc:cccc} - & ace & bf & abcdfh & abgh & bcdfgh & adfg & abcdfg \\ \hdashline - & - & - & & bcegh & abdefgh & cdefg & bdefg \\ & - & - & & afgh & cdgh & abdg & acdg \\ & & - & & cdfg & ag & bcgh & gh \\ \hdashline & & & & - & - & - & - \\ & & & & & - & - & - \\ & & & & & & - & - \\ & & & & & & & - \end{array}\right] .$$

Selection of the shortest entries bf, ag, and gh as potentially critical now yields

$$\hat{W} = \left[\begin{array}{ccccccc} ace & bf & - & - & - & - & - \\ - & - & - & - & - & cdefg & - \\ & - & - & - & - & - & - \\ & & - & cdfg & ag & - & gh \\ & & & - & - & - & - \\ & & & & - & - & - \\ & & & & & - & - \\ & & & & & & - \end{array}\right] .$$

Entry cdefg still includes entry cdfg and may be deleted. Also row labels b, d, e, and h may be deleted, since these letters always occur with letters f, c, a, and g, respectively. This yields

$$\hat{W} = \begin{bmatrix} - & ac & f & - & - & - & - & - \\ & - & - & - & - & - & - & - \\ & & - & - & - & - & - & - \\ & & & - & cfg & ag & - & g \\ & & & & - & - & - & - \\ & & & & & - & - & - \\ & & & & & & - & - \\ & & & & & & & - \end{bmatrix} .$$

Hence

$$L(G_M^*) = (a \vee c)\, f\, (c \vee f \vee g)\, (a \vee g)\, g$$

$$= fg\, (a \vee c)$$

$$= afg \vee cfg.$$

Chang[40] recognized the importance of the problem of fault location to within modules problem, and offered a solution which he claims "tends to give a 'fairly good' set of test patterns." He does not prove this assertion, however. His method will be sketched in the next part of this section, since it appears to be more successful when adapted to serial test schedules than when used for fixed test schedules, as originally proposed.

e. Some Bounds on the Number of Tests Required

Let N_D, N_L, and N_M be the number of test inputs in the minimal solution subset S, for the cases of detection, location, and location-to-within-module-class, respectively. A tight upper bound[104] on all three of these quantities is provided by a particular case of the matrix F--namely, the case when all columns of weight zero and one are present. Thus, F is just an identity matrix of order m, bordered by a single column of 0's:

$$F = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & . & . & . & 0 \\ 0 & 0 & 1 & 0 & 0 & . & . & . & 0 \\ 0 & 0 & 0 & 1 & 0 & . & . & . & 0 \\ & \vdots & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & . & . & . & 1 \end{bmatrix} .$$

It is easily seen that no rows may be deleted without leaving a resultant matrix in which some column is the same as the first column. Moreover, the presence of even one additional new row in the matrix would allow at least one row to be deleted. Thus we have

$$N_D \geq m, \qquad N_L \geq m, \qquad N_M \geq m \quad .$$

Tight lower bounds may be obtained as follows. For the case of fault detection, the possible presence of a row such as $(0 \quad 1 \quad 1 \quad 1 \quad \ldots \quad 1 \quad 1)$ in F immediately yields

$$N_D \geq 1 \quad .$$

For fault location,[104] the most favorable case arises when F contains a subset of rows which constitute a binary coding of the columns. N_L such rows can generate as many as 2^{N_L} different columns, so

$$N_L \geq 1 + [\log_2 (m)]$$

where the brackets denote the integer part of the quantity within. Finally, location of a fault to within one of p module classes requires only that each module class have a distinct column coding; all columns in one class may be alike. Thus,

$$N_M \geq 1 + [\log_2 (p)] \quad .$$

Our experience with examples of random matrices tends to indicate that actual minimal test-schedule lengths fall closer to the lower than the upper bound. Whether or not these samples of matrices are truly representative of the patterns of faults in typical switching circuits is another, more difficult question.

f. Reductions in the Size of the Fault Table

In order to be able to handle networks of a practical size, it would be very desirable to reduce the width and height of the original fault table (F-matrix) below the values $m + 1$ and 2^n, respectively. We now show how some reductions can be made for the case of fault detection at a cost whose exact value is not presently known, though felt to be small.

If both the width and the height of the fault table are to be reduced appreciably, some sort of analysis of the internal structure of the network will be necessary. This is in contrast to the point of view taken so far, in which the network has been viewed only from its terminals. That is, the method used so far is one of testing the behavior of the network for every possible input and for every possible fault. It must be replaced by a method which asks: for which inputs will particular faults manifest themselves at the output terminal of the network, and (for fault detection) which other faults will have the same effect as these faults?

The literature contains a few pertinent contributions on this matter.[246,11, 41, 192, 87] Armstrong[11] (following a suggestion of Muller's) and Maling and Allen[192] propose approaches which, taken together, suggest that one may check for a fault in a single gate within the network by (a) applying those network inputs which will "sensitize' to signal changes the complete path from the gate in question to the network output (see Fig. III-B-1), and then (b) flex the remaining input variables through whatever sequence of combinations of values is necessary to check the proper operation of the gate. For most common types of gates the number of such combinations is just one more than

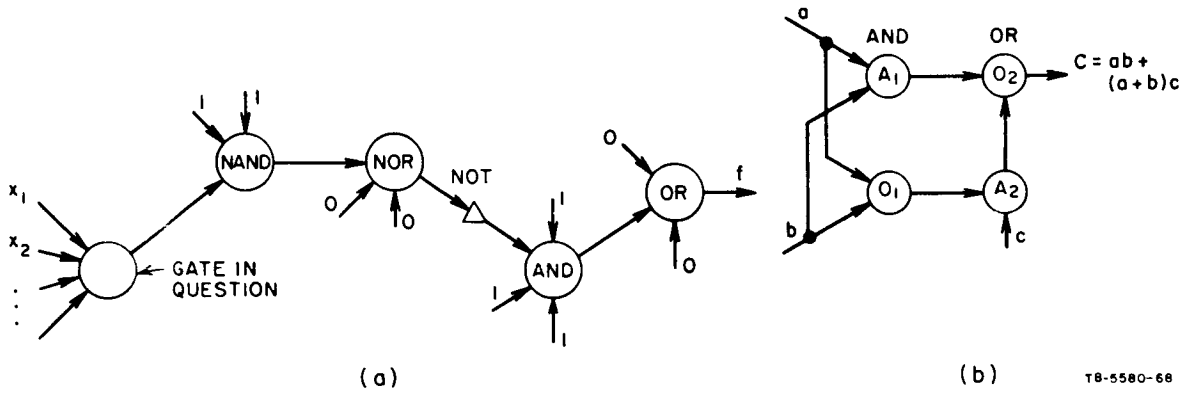


FIG. III-B-1 PATH-SENSITIZING TESTS IN GATE NETWORKS

the number n of free inputs at a particular gate; with more complex gate circuitry it may become as large as 2^{n_g}, the total number of input possibilities for the gate.[192] The complete test schedule for fault detection therefore consists of the union of these individual gate tests, taken over all gates which have one or more inputs driven by network input variables.* (Purely internal gates are checked automatically.) Any duplicate test inputs may be deleted, of course.

For example, for the size comparison cell in Figure III-B-1(b), the following test schedule is obtained:

	a	b	c
Gate A_1	1	1	0
	1	0	0
	0	1	0
Gate O_1	0	0	1
	0	1	1
	1	0	1
Gate A_2	0	1	0
	0	1	1

* Actually, this procedure must be modified somewhat for those gates whose fanout exceeds unity, in order to cover all of the multiple paths to the network output. See the Appendix to Armstrong's paper.[11]

For the first three tests, the path through gates A_1 and O_2 is sensitized by setting c = 0. Gate A_1 is tested by applying inputs a = b = 1, to give an output 1, then a = 1, b = 0, followed by a = 0, b = 1, which should give an output 0. (In a conventional AND-gate, there is no need to apply the input a = b = 0, since there is no reasonable way the circuit could fail so as to make it behave like an inequivalence gate.) For the second group of three tests, the path through gates O_1, A_2, and O_2 is sensitized, by setting c = 1 and holding ab = 0. Gate O_1 is then flexed with input combinations a = b = 0; a = 1, b = 0; and a = 0, b = 1. Finally, the last two tests (which happen to be duplicates of prior tests) arise from the flexing of gate A_2 by means of its free input c, having set ab = 0 and a + b = 1 to sensitize the path from A_2 to O_2. Thus, six different test inputs are required in the schedule. This may be shown to be the minimum number for this example, by the procedure described earlier in this memorandum.

Actually, Armstrong assumed an even simpler set of fault conditions in describing his proposed method--namely, that the only failures which can occur are those which result in a gate output being "stuck(at 0 or 1, for all possible gate inputs. For this case, each gate may be tested by applying only two tests, one of which tends to make the gate output 0, and the other of which tends to make it 1. All gates along the sensitized path are then checked automatically; therefore, the procedure need be applied only to those gates whose inputs are all network input variables. As a result, the derivation of a test schedule is simplified considerably.

For the above example, all single "stuck at" faults may be readily detected with just four test inputs, corresponding to the flexing of gates A_1 and O_1, each with its own path to the network output sensitized:

	a	b	c
A_1	1	1	0
	1	0	0
O_1	0	0	1
	0	1	1

.

Depending upon the network form, there may be a degree of arbitrariness in the selection of the input-variable combinations which are used to sensitize a particular path or to test a particular gate. The length of the overall test will in general depend upon these choices made for each path and each gate, since they will determine the extent of duplication of test inputs in the formation of the schedule. Armstrong[11] gives an algebraic procedure aimed at increasing the number of duplicates. At worst, however, one may proceed by listing the alternative tests which are valid for each path and each gate, and then making desirable selections from this list when the complete schedule (with multiple choices) has been compiled.

To summarize, the path sensitization method introduced by Armstrong provides a means for generating a test schedule for fault detection directly from the network, thereby bypassing the formation and manipulation of the fault table itself. The method appears to be an efficient one, although it is not guaranteed to lead to a minimal test schedule.

For fault location and fault location to within modules, no satisfactory procedure exists. Armstrong's procedure might be augmented to allow one to go back and include additional gate-input combinations that may be necessary to distinguish otherwise identical faulty outputs. However, what is most needed for fault location is a condition for the sufficiency of a trial test schedule, to be certain that all pairs of possible faults are indeed distinguishable on the basis of their output patterns. This problem remains unsolved.

Galey, Norby, and Roth[87] have also proposed a method for deriving a test schedule for the case of fault detection, again based upon analysis of the network structure. Their algorithm is similar to but longer than Armstrong's, but is probably better suited for execution as a computer program. Again, it generates the test schedule directly, without the necessity of deriving the fault table first, and may be extendable to fault location.

g. Implementation of the Test Schedule

After a test schedule has been derived by one of the methods discussed above, it is necessary to arrange for it to be applied on demand to a network under test. In addition, the results of the test must be condensed into a form suitable for evaluation and for transmittal either to a human or to an automatic switchover mechanism, in order that the appropriate repair action can be initiated. These tasks which precede the actual repair can be assumed to be performed by a device which we will call a diagnoser, as shown in Fig. III-B-2. The diagnoser can be realized in the form of either a special digital circuit or a computer program, and it may be located either physically adjacent to the network (i.e., in a spacecraft) or remote from the network (e.g., on the ground).

When the diagnoser is in the form of a circuit, it is composed of an autonomous sequence generator, which produces in sequence at its output each test input in the test schedule. For the case of detection, it also produces with each test input the corresponding correct network output. This output is applied to a comparator, that checks the actual network output against the correct output, noting any disageements. The sequence of comparator outputs is then merely "OR-ed" together, by applying it to a flip-flop (for example), the result of which is then available for repair purposes. For fault location, the sequence of network outputs must be decoded in accordance with the columns of F_{min}, in order to determine which fault has occurred. For human repair, and

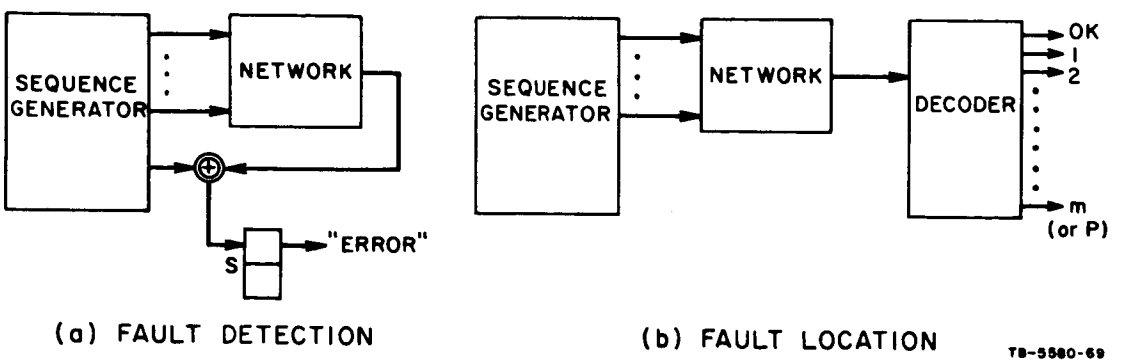


FIG. III-B-2 DIAGNOSER STRUCTURES

if time permits, a code book is probably the simplest way of decoding. Otherwise, a special decoder network must be constructed.

In any case, the specifications on the special circuitry needed can be precisely stated:

Sequence generator:

Given: A list of test inputs for the network under test, and (for fault detection) the corresponding outputs. This is just a truth table for the function f, with rows deleted in correspondence with the minimization of the matrix F, as described in the preceding sections. This list has N rows (where $N = N_D$, N_L, or N_M) and n or n + 1 columns.

Synthesize: An economical, autonomous sequential network which produces at its n or n + 1 outputs the rows of the given list, in any convenient order.*

Decoder:

Given: The minimum F-matrix, F_{min}, with the rows ordered in accordance with the row permutation used in the design of the sequence generator.

Synthesize: An economical single-input sequential network which produces a 1 at a unique one of its m + 1 outputs, for each N-digit input sequence which appears as a column of F_{min}. Other possible input sequences of the same length may be treated as "don't cares."

Both of these circuits are assumed to start from a unique starting state, which implies some means of reset to this state, and will probably require a synchronizing clock when incorporated into the rest of the system.

In view of the stringent reliability requirements on the diagnoser, one might seriously consider realizing it with multiaperture magnetic devices. These devices are also naturally well suited for sequence generation and decoding operations.

---

* Goldberg suggests generating first all the set of inputs which correspond to f = 0, followed by the set of inputs for which f = 1 (for fault detection).

When the diagnoser is to be realized as a computer program, the operations of sequence generation and decoding are described by flow charts, and the design and execution of the program present no difficulty. The process of sequence generation is no more than the successive retrieval from memory of N binary words of length n or n + 1. The flow chart for the decoding is a N-level decision tree, having 2, m + 1, or p + 1 outputs for fault detection, fault location, and fault location to within modules, respectively. The decision trees for these three cases are illustrated in Fig. III-B-3 for the running example.

While one could probably not justify a spaceborne computer solely for the purpose of fault diagnosis, the fact that a computer may be available anyway, and the flexibility it offers for diagnosing a large number of different networks, makes it an attractive solution.

h. Tests for Multiple-Output Networks

If the network for which a test schedule is being determined has q (>1) outputs instead of a single output, the problem-formulation and solution procedures described above remain essentially the same but are modified in detail, as follows.

(1) The entries in the f_j-columns of the fault table F, as well as in the entire F-matrix, are q-digit binary numbers instead of single binary digits.

(2) Two columns of the fault table should be considered to be identical when and only when all of the corresponding q-digit entries are exactly the same.

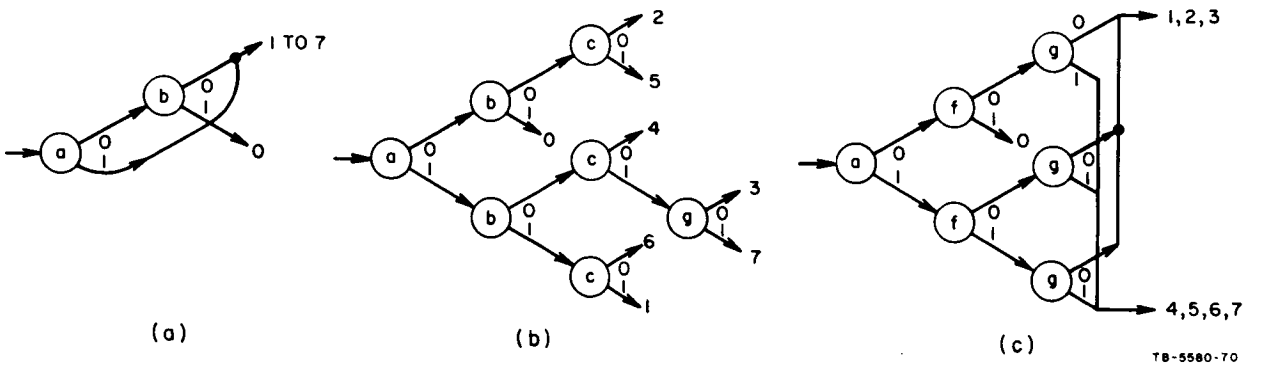


FIG. III-B-3 DECISION TREES FOR FAULT LOCATION

(3) The matrix G should have only single-binary-digit entries, according to the rule: if two multidigit entries in the same row of F differ in any of their digits, then the corresponding entry in G for these two columns is a 1; otherwise it is a 0. This rule is a direct reflection of the fact that a fault may be detected on any one or more of the q network outputs. The rest of the minimization procedure is carried out on G just as for the single-output case. Note that since the matrix G now usually has a greater number of 1's in it, the length N of the test schedule can be expected to be smaller, assuming that the other parameters remain the same. Similarly, in forming $W = F^tF$, element multiplication is defined to produce a 1 when and only when the q-digit elements are different in any digit. (Clearly, this reduces to exclusive-OR when the entries are single-digit numbers.) $\hat{W}$ is defined as previously.

(4) The Armstrong path-sensitizing procedure remains unchanged, so long as it is kept in mind that a path to any one or more of the q outputs is adequate to render a fault conspicuous.

(5) The sequence generator for fault detection now has n + q instead of n + 1 outputs, the decoder has q-inputs, and the number of exclusive-OR gates in the comparator is q instead of one. Similar increases apply to the realization of the diagnoser as a computer program.

It should also be pointed out that one possibility for reducing the length of a test schedule is the extraction of one or more selected nodes of the network as test points, which are then treated as separate outputs as far as the diagnosis is concerned. The key problem here is the selection of those circuit nodes which will result in the greatest reduction in the length of the test schedule for a given network. This and other problems worthy of further investigation are listed at the end of the next part of this section.

3. Fault Diagnosis in Combinational Circuits Using Serial Test Schedules

a. Introduction

The preceding part of this section presented a statement of the fault-diagnosis problem for combinational switching networks, and solutions for the most important cases when the test schedule is fixed.

We offer here corresponding solutions for the same cases when the test schedule is serial--that is, when the selection of successive tests depends upon the outcomes of previous tests in the schedule. In particular, we show that by using a serial test schedule there is nothing to be gained for fault detection, but for fault location and fault location to within modules the possible reductions in test-schedule length are substantial. A solution procedure is given which is easy to carry out and is reasonably effective, although it does not necessarily lead to a test whose length is absolutely minimal. Bounds on the length of the test schedule are derived. Finally, the principal problems remaining for further research are itemized.

It may be observed that the test schedules derived in the preceding part of this section are completely independent of the outcome of the individual tests in the sequence; moreover, the length of the schedule is independent of the order in which the tests are performed. It is quite conceivable, however, that after the first test input of a schedule has been applied and the output noted, the residual test schedule which is minimal with respect to a 0 output is not the same as that which is minimal with respect to a 1 output. Similarly, after two test inputs have been applied, the four partial test schedules which should follow may be all different in content and length; and so on for successive test inputs. We consider here the economies to be achieved by choosing each test input to be applied to the network on the basis of the outcomes of all previous tests in the schedule.

Solutions to this problem for the three cases of fault diagnosis, fault location, and fault location to within modules are best represented in the form of decision graphs, such as were used to

describe the fixed-schedule solutions in the preceding part; see for example Fig. III-B-3. For serial schedules, however, the row labels a, b, c, ... which are attached to the nodes of these graphs are no longer restricted to be identical over all of the nodes in the same level of the graph. The serial solutions which will be derived below for this same example are shown in Fig. III-B-4. It may be seen in Fig. III-B-4(b) that for fault location a shorter schedule results than was required in Fig. III-B-3(b). However, one now needs to know in which way the first test (g) turned out before the second test (f or h) can be applied.

The minimization of decision graphs of this type has been considered by Lee[178] and by Short.[283,284] Short showed that the problem is equivalent to the minimization of an important family of transfer-contact networks called disjunctive and exhaustive networks. The transfer-contact networks corresponding to the solutions shown in Fig. III-B-4 are given in Fig. III-B-5 in the same order. The direct graphical correspondence can be readily seen. While this analogy is useful theoretically and in a few specific problems, good general procedures are unfortunately lacking for obtaining absolutely minimal networks of this type.

Nevertheless, several methods are known for deriving economical networks which sometimes turn out to be minimal. All of these methods consist of a successive selection of the node labels of the decision graph (row labels of the fault table), working from left to right. After selection of the left node label on the basis of some criterion, the 0's and 1's in the corresponding row of the fault table effectively separate the table into two subtables, each of which corresponds to one of the two subgraphs (subtrees) to the right of the leftmost node in the decision graph. Each of these two subtables may now be attacked independently by exactly the same selection and reduction process to generate four smaller subtables (subtrees), and so on. The only difficult aspect of this method is the particular criterion employed to select at each step that row of F which ultimately causes this iterative procedure to terminate after a minimal (or reasonably small) number of steps.

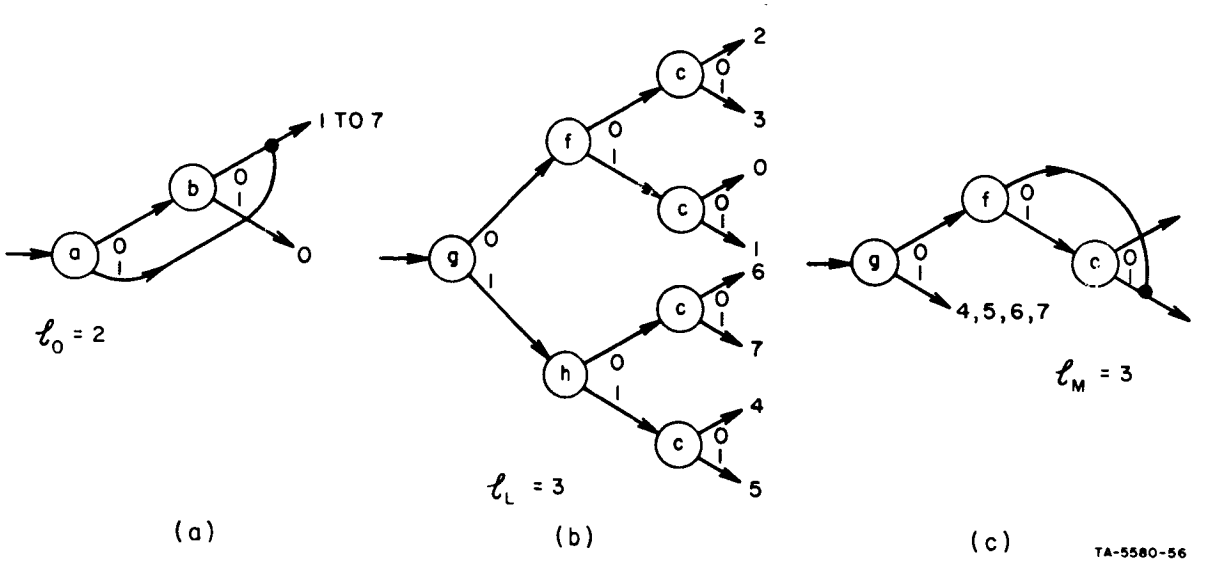


FIG. III-B-4 DECISION TREES FOR SEQUENTIAL TESTS

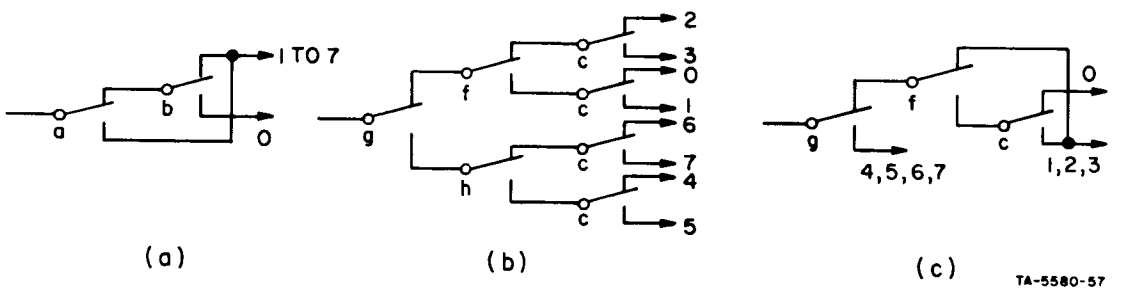


FIG. III-B-5 CONTACT-TREE ANALOGS OF DECISION TREES

Further discussion of this method, as well as examples, will be taken up separately for fault detection, fault location, and fault location-to-within-modules.

Serial fault location was first proposed by Brule et al.,[34] under some rather restrictive assumptions, and without giving any procedure for deriving a test schedule for a given network.

b. Fault Detection

The problem in fault detection is one of choosing a minimal subset of rows of F to distinguish the first column from all others. Thus, each step of the procedure results not in two but in only one residual subtable (subtree). Consequently, there is no advantage to be gained by performing the tests in any particular order, and the fixed-schedule solution having N_D tests is also optimal with respect to the minimal number ℓ_D of levels required. In fact, $\ell_D = N_D$.

Thus, serial testing offers no improvement over fixed-schedule testing for fault detection, and the same decision graph [Fig. III-B-3(a) and Fig. III-B-4(a)] solves both problems.

c. Fault Location

One way to select the appropriate row label at each step of the procedure is to try all possible remaining row labels. For even a small fault table, however, the total number of possible graph labelings which need to be tried to determine the minimal number of levels is astronomical. This approach is therefore impractical.

Sindeev[287] and Chang[40] employ two different criteria for selection of successive row labels of the decision graph for the fixed-schedule case. Their methods can also be applied to the individual steps in serial testing, however, and in this case yield exactly the same results. This extension of the Sindeev and Chang methods proceeds as follows.

Let the numbers of 0's and 1's in row i of a fault table or subtable be w_{i0} and w_{i1}, respectively. Sindeev proposes the selection of that row $\underline{i}$ which maximizes the amount of information (in an information-theoretic sense) which is gained by that row decision regarding the particular column which was "transmitted:"

$$J = -p_0 \log_2 (p_0) - p_1 \log_2 (p_1)$$

where $p_0 = w_{i0}/w$, $p_1 = w_{i1}/w$, and $w = w_{i0} + w_{i1}$. Manipulation of this expression for J reveals that its maximization is equivalent to minimizing the expression

$$(w_{i0})^{w_{i0}} \cdot (w_{i1})^{w_{i1}} \quad .$$

Chang, on the other hand, suggests the selection of that row i which maximizes the number of (0, 1) pairs between digits in that row--that is, which maximizes the expression $(w_{i0}w_{i1})$.

These two quantities in parentheses are simultaneously optimized by selecting that row which has the most nearly equal distribution of 0's and 1's; that is, that row (or one of the subset of rows) for which $|w_{i0} - w_{i1}|$ is minimal.

The use of this criterion appears to work very well for most problems. Applied to the F-matrix used as a running example in the first part of this section, namely

$$F = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{matrix}$$

any one of the rows c, d, or g should be chosen first, since these rows have $w_{i0} - w_{i1} = 0$. Selection of row g yields the two submatrices:

$$F_0 = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ h \end{matrix} \quad F_1 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

For the second step, one of the rows a, b, c or f should be selected from F_0, and one of the rows c or h from F_1, since only these rows have an equal number of 0's and 1's. Choosing the last alternative in each case gives the following partitions into four smaller submatrices:

$$F_{00} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{matrix} a \\ b \\ c \\ d \\ e \\ h \end{matrix} \quad F_{01} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \quad F_{10} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \end{matrix} \quad F_{11} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} .$$

For the third step, there are many possibilities, but choice of row c serves simultaneously for all four submatrices. The resulting decision graph is shown in Fig. III-B-4(b), and has $\ell_L = 3$ levels.

Sindeev gives an example in his paper of a fault matrix with 14 rows and m = 15. His own method (which he claims to yield a minimal number of levels in a fixed-schedule solution) applied to this example yields a decision graph having six levels. Application of Chang's method to to the same example yields the same fixed-schedule result, six levels. The G-matrix method described in the preceding part of this section yields five levels. Therefore Sindeev's method does not give a minimal result, as he claims. Moreover, both Chang's and Sindeev's methods can result in very long fixed schedules if the wrong choices happen to be

made at those points in the procedures when two or more rows are calculated to be equally desirable choices.

The risk of making such unwise choices appears to be much less for the serial-test procedure described above. Applied to Sindeev's example, this procedure gives a decision graph having only four levels.

d. Fault Location to Within Modules

The method described above applies with little change to the case of fault location to within modules. Following Chang, we now modify the criterion of row acceptance to count not all (0, 1) pairs, but only those in each row in which the 0 and the 1 fall in different module classes. This quantity is most easy calculated by subtracting from the total number of (0, 1) pairs the sum of the number of (0, 1) pairs which fall entirely within the individual classes, namely

$$R_i = w_{i0}w_{i1} - \sum_{j=1}^{p} w_{ij0}w_{ij1} \quad ,$$

where w_{ij0} and w_{ij1} are the numbers of 0's and 1's, respectively, in the j^{th} module class in row i. The row to be selected is the one with the largest row count R_i.

For the running example, with the previously used column partition (0) (123) (4567) into module classes, the row counts on the 8 rows of F are:

a: 15 - 2 - 3 = 10
b: 15 - 2 - 3 = 10
c: 16 - 2 - 2 = 12
d: 16 - 2 - 3 = 11
e: 7 - 2 = 5
f: 15 - 2 - 3 = 10
g: 16
h: 15 - 2 - 2 = 11 .

Clearly, row g should be selected first, and the same two submatrices F_0 and F_1 as arose in the previous section result. F_1 falls entirely within (and in fact covers exactly) module class (4567), so it need not be further decomposed. For the rows of F_0, the row-count values are:

a: 4 - 2 = 2
b: 4 - 2 = 2
c: 4 - 2 = 2
d: 3 - 2 = 1
e: 3 - 2 = 1
f: 4 - 2 = 2
h: 3 - 2 = 1 .

Any of a, b, c, or f should be chosen. Selecting f, only F_{00} need be further decomposed, and any of the nonconstant rows a, c, d, or h may be used. Figure III-B-4(c) displays the resulting decision graph.

Note that the graph of Fig. III-B-4(c) could have been obtained from the graph of Fig. III-B-4(b) by typing together outputs 1, 2, and 3, then 4, 5, 6, and 7, in accordance with the grouping of columns into module classes, and then reducing the resulting graph according to a well-known procedure for the simplification of transfer-contact networks.[284] While this method yields a minimum-level graph in the present example, it will not do so in general, and the procedure described above must be used to group together advantageously the outputs associated with each module class.

Similarly, a valid fault-detection graph could be obtained from the graph of Fig. III-B-4(c) by merging the outputs (1, 2, 3) and (4, 5, 6, 7). However, the resulting graph cannot be further simplified. The graph of Fig. III-B-4(a) has fewer levels, and is therefore preferable.

e. Bounds

The bounds derived in Sec. III-B-2-e on the minimal number N of tests in fixed test schedules apply without change of argument or

or result to the number ℓ of levels in serial test schedules:

$$1 \leq \ell_D = N_D \leq m$$

$$1 + [\log_2 (m)] \leq \ell_L \leq N_L \leq m$$

$$1 + [\log_2 (p)] \leq \ell_M \leq N_M \leq m \quad .$$

For most problems, of course, we can expect ℓ_L to be much smaller than N_L.

We have assumed throughout this section that the parameter to be minimized when optimizing a serial test schedule is the number ℓ of levels in the decision graph, since this number is proportional to the running time of the diagnostic test. If, instead, it is the total length of the diagnostic program (that is, the amount of memory space required to store the test schedule) which is of principal interest, then it is the total number d of decision nodes in the graph which should be minimized. However, a simple argument shows that for fault location, this number d_L is fixed for a given F-matrix, and is equal to m, the total number of faults. To see this, observe that each of the $2d_L$ output arrows from the d_L nodes of the graph terminates either on one of the m + 1 outputs, or on one of the $(d_L - 1)$ nodes to the right of the leftmost input node. Thus:

$$2d_L = m + 1 + (d_L - 1),$$

or $d_L = m$.

For fault detection, we have $d_D = N_D = \ell_D \leq m$. For fault location to within modules, all we can assert is: $\ell_M \leq N_M \leq d_M \leq m$.

f. Potential Economies of Serial Test Schedules for Fault Location

The following exemplary family of F-matrices demonstrates that there exist problems for which the number of levels in a minimal serial test schedule is approximately equal to the logarithm of the number of levels in the corresponding minimal fixed test schedule.

Consider first the particular F-matrix

$$F = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \end{matrix}$$

which has m = 7 and seven rows. For a serial test schedule, the decision tree has m + 1 = 8 outputs, so $\ell_L \geq \log_2 (8) = 3$. Three levels are clearly adequate, as shown in Fig. III-B-6(a). For a fixed test schedule, observe that all seven rows of F are necessary, since deletion of any row causes two columns to be identical. Thus, $\ell_L \geq 7$. This decision tree is shown in Fig. III-B-6(b).

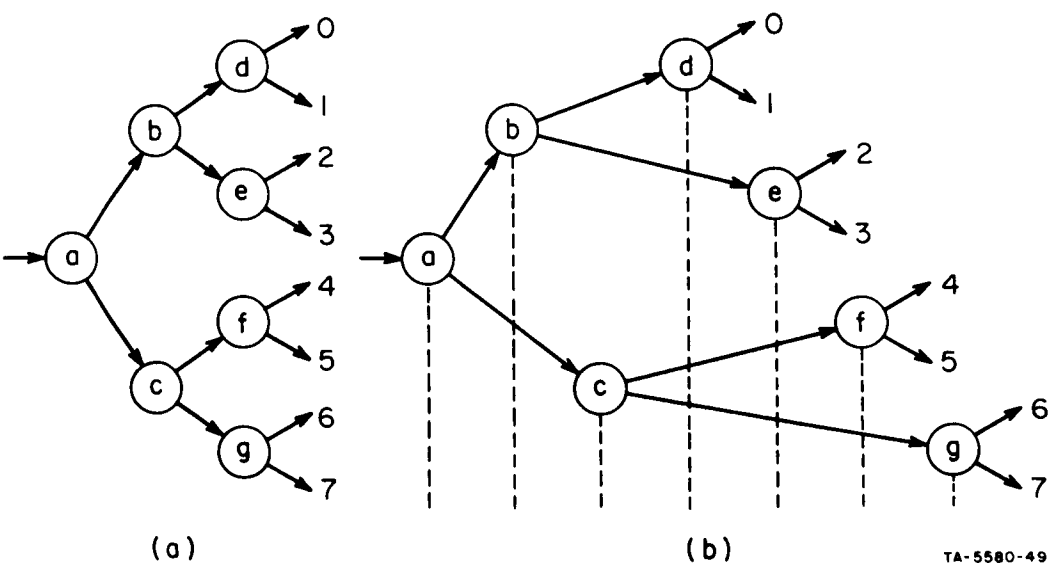


FIG. III-B-6 SEQUENTIAL DECISION TREE FOR A LIMITING CASE

More generally, let m + 1 be any power of two, and let F be formed according to the same pattern as was used above: the 0's and 1's in the first row separate the set of columns into two equal-sized groups; the second and third rows similarly divide each of these two groups in two, leaving four groups; the next four rows divide each of these four groups in half, leaving eight groups; etc. In each instance of group splitting by a row, 0's are entered in the columns corresponding to all other groups. Realization of the decision trees in the same pattern as shown in Fig. III-B-6 yields immediately:

$$\ell_L = \log_2 (m + 1) \text{ for serial tests}$$

$$\ell_L = m \text{ for fixed tests.}$$

However, regardless of F, the number of levels in an (m + 1) - output decision tree must fall somewhere in between these same values (see the Sec. III-B-2-e. Therefore, these are extreme values, and cannot be further separated.

This family of examples shows that there exist problems in fault diagnosis (location) for which serial test schedules are vastly superior to fixed test schedules.

4. Fault Diagnosis in Digital Computers: Present State of the Art

This part summarizes the present status of diagnostics as applied to digital circuitry and systems of the type anticipated for use in spaceborne computers. This status is evaluated with respect to the need for fault-diagnostic techniques, and recommendations are offered for future research work to satisfy the deficiencies that exist at present.

Diagnostic procedures for the detection and for the location of non-transient faults in completely arbitrary digital networks, subsystems, computers, and systems are presently either unavailable or inadequate. Satisfactory procedures exist for the derivation of diagnostic test schedules only for the class of combinational networks which (1) are not too large (8 to 10 inputs, and up to about 4 or 5 outputs) but otherwise arbitrary, and (2) are subject to a limited number of definitive, specified faults--not more than a few hundred for fault

detection, and not more than about 100 for fault location. In addition, procedures are known for the detection of the most common types of faults in large combinational nets, for the location of faults in some special cases of large combinational networks, and for a few special types of small and medium-sized sequential networks. For all other conditions, however, including many situations of practical interest, the only techniques which are available cannot be considered to be satisfactory, either because they lack generality, because they are too difficult to carry out, or because they lead to test schedules which are much too long. For some cases, no techniques have even been proposed.

For the known procedures, it is assumed that a general-purpose computer is available for carrying out the derivation of a test schedule for a given network. If this is not so, the maximum size of the network and the number of faults which can be handled are much reduced. It is also assumed that the resultant diagnostic test schedule, to be applied on demand to a network in actual service, can be expressed as a simple computer program which may be stored in the spacecraft computer or communicated to the spacecraft. If the testing cannot make use of an already existing digital memory or communications link for diagnostic purposes, then a special subsystem must be designed to serve this purpose. Presently known design techniques for such subsystems are available, but are not completely adequate, since they often lead to circuitry which is unnecessarily costly or which cannot itself be readily diagnosed or protected against its own faults.

It should also be noted that there is much to be gained in reducing the complexity and length of diagnostic test schedules, if one or more of the following additional techniques are employed:

(1) Extracting selected circuit nodes as test points or inserting additional control inputs, to facilitate testing

(2) Using "serial" test schedules--in which the selection of successive tests in the test schedule is made to depend upon the results of the tests applied earlier, in closed-loop fashion

(3) Designing the original network in such a manner as to make it more readily diagnosible.

Circuits are known for which considerable improvements can be demonstrated, using each of the above techniques. Unfortunately, however, practically nothing is known in general about how to actually achieve these improvements. In some cases, one or more of these techniques may be absolutely necessary in order to obtain acceptable test schedules. For example, it may be essential to employ a small number of test points when testing large sequential circuits.

In summary, there are unsolved problems, the solution of which would contribute substantially to the realization of advanced reliable spaceborne computers; these are*:

(1) Development of techniques for selecting an economically small number of test points or additional inputs for a given combinational network, in order to drastically reduce the length of the test schedule required for (a) fault detection and (b) fault location (to within a replaceable module).

(2) Evolution of a general approach to finding good diagnostic techniques for sequential circuits, and the derivation of some useful procedures for arbitrary moderate-sized circuits; also, the identification of those special classes of large sequential networks for which these procedures might still be used.

(3) Development of practical procedures for deriving economical test schedules for very large combinational networks.

(4) Investigation of the economies to be achieved and the methods which are appropriate when serial test schedules are used for combinational networks; also, the identification of those cases in which the use of serial test schedules offers the greatest advantage over fixed test schedules.

(5) Resolution of the principal questions regarding which system organizations should be employed to facilitate both diagnostic testing and subsequent repair (switchover to a spare or to a reconfigured system).

---

* It should also be noted that the first four of these topics are of considerable importance to the manufacture of integrated circuit packages for computer use.

(6) Study of the ways in which the original design of a digital network or subsystem may be modified in order to make it more susceptible to the diagnostic procedures which have been developed.

(7) Development of techniques for the design of special-purpose diagnostic circuitry, including estimates of the economies to be achieved through its use.

C. Design of Networks for a Reconfigurable Computer

1. Introduction

In Sec. III-A-1 it was pointed out that a reconfigurable computer should have high degrees of flexibility of structure, simplicity of diagnosis, and reliability of control; and furthermore, that both flexibility and inherent reliability of manufacture are improved by using a small number of different kinds of complex logic modules in the composition of the computer. It was also noted there that in order to realize the potential increase in reliability of reconfiguration, it is necessary to accomplish the switching of data and the control of reconfiguration efficiently, that is, to achieve a large number of configurations with a small number of switches. New network schemes and design techniques are needed to realize these unusual requirements.

In the next three sections we consider approaches to the design of networks that are especially well suited to the functions of interconnection, logical processing, and control, respectively. Other network types might be considered, e.g., some based upon combinations of these functions; but those chosen appear to offer good possibilities for efficiency of design.

The particular kind of interconnection network described in Sec. III-C-2 is called a commutation switch. It is intended to be complementary to the processing network described in Sec. III-C-3. Several approaches to control networks are discussed in Sec. III-C-4, one of which is based on the processing module of Sec. III-C-3.

All of the networks to be considered are designed to be multipurpose and programmable. The criteria of ease of diagnosis and incorporation of fault masking have not been incorporated, and constitute problems for further development.

2. Programmable Interconnection Networks

a. Introduction

The preceding discussion has served to describe the overall features of a system which can diagnose its internal faults and correct them by reconfiguring its subsystems. Of major importance in such a reconfigurable system are the circuits whose function is to provide interconnection between the operating modules and to disconnect faulty modules from the system. These circuits, which we call commutation networks, are discussed in this subsection. Our aim is to describe techniques which a designer can use to develop commutation networks which provide reliable flexibility of interconnection at low cost.

Initially we have selected a simple model for analysis, based upon the control of data flow between two parallel linear arrays of modules--as, for example, between two registers or between a register and memory. Figure III-C-1(a) shows the conventional interface between each of the M modules of two arrays, for the parallel transfer of data where there are no spare units. Figure III-C-2(b) shows the same two arrays, with the addition of N_1 - M spare modules on the input array and N_2 - M spare units on the output array. The function of the commutation network is to provide connection between a set of M input modules and a set of M output modules, and also, of course, to disconnect the paths from (and to) faulty modules. It is assumed that the modules of the two arrays are diagnosed by an external authority, which in turn furnishes status signals to the commutation network.

Usually in the parallel flow of data it is of primary importance to preserve the order of the signals; i.e., the digits contained in a set of M operating modules of the input array should be transferred in order to a set of M operating output modules. In some applications it might be possible to append an identifier to the signals traversing the arrays, indicating the source of the signal. In this case the commutation network is not constrained to preserve order. We ascribe primary importance to the order-preserving case, particularly in describing implementations, because in conventional computer functions the order

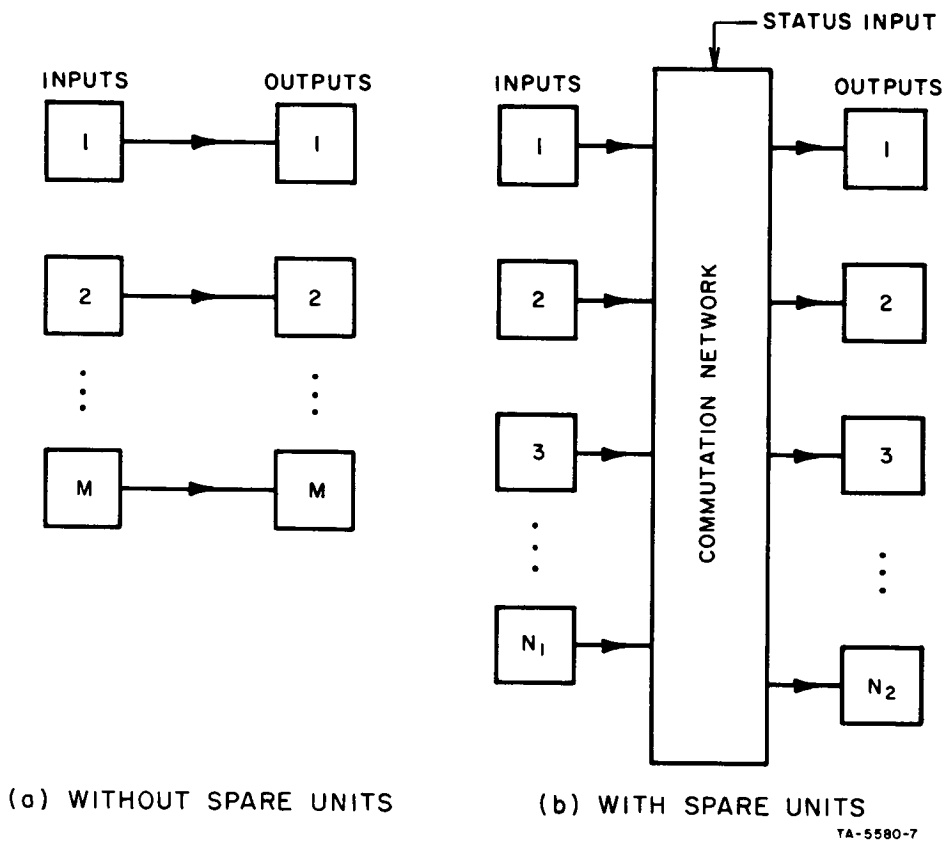


FIG. III-C-1 INTERFACE BETWEEN MODULE TYPES

of data is preserved; as, for example, in the transfer of data between registers and arithmetic units. However, we also briefly consider the non-order-preserving case because of the reduction in complexity afforded by such commutation networks.*

All of the networks considered here provide for total commutation, i.e., up to some number of faults at the input and some number of faults at the output, all arrangements of the faults are accommodated. Consideration of partial or nontotal commutation is an important problem for future study.

In Sec. III-C-2-b we consider an order-preserving asynchronous commutator in which the data "diffuses" along a register between the inputs and outputs. The operating status of the input and output modules is stored in the register.

A combinational approach, relating to both the order-preserving and non-order-preserving cases, is initially discussed in Sec. III-C-2-c. With this approach there are a set of possible connections between the inputs, visualized as single-pole single-throw switches, which are controlled by status signals. The design goal here is to minimize the number of switches required for complete flexibility.

In Sec. III-C-2-d we briefly discuss the conventional implementation of these commutation networks as a simple combinational net, and we also present a cellular implementation which is suggestive of a Holland-Machine structure. Techniques for achieving commutation switches which are failure tolerant are discussed in Sec. III-C-2-e.

---

* It is recognized that similarities exist between the design of efficient commutation circuits and the design of telephone central-office systems. (The reader is referred to Benes[24] for an analytic discussion of telephone interconnection networks.) However, there are several important distinguishing factors; e.g., in the commutation case all pairs of input and output modules are not required to be connectable, and also the network must function unattended. These factors are sufficient to require an extensive examination of both the interconnection system itself and the control modules by which the decisions are made.

b. A Sequential Commutation Network

This section describes a sequential buffer network that provides total order-preserving commutation with reasonable economy. Although the network contains logical delay elements, no clock signals are needed. If this scheme is used within a synchronous computer, higher network speed will result than if the system clock were used in an equivalent clocked network. The network is composed of a cascade of stages, with parallel input and output facilities, and with serial propagation with the cascade. In the propagating mode, the network acts essentially as a speed-independent cascade,* specially modified to accommodate invalid sources and loads. Another application of this mode of operation is described in Section II-A-2-C-3.

1) Overall Behavior of the Network

The overall behavior will be explained with reference to Fig. III-C-2. The network is seen to be a cascade of 2n stages, serving n data sources and n data sinks.

The use of a double-length register is one of several possible techniques for allowing a relative positive or negative displacement of the indexes of corresponding source and load channels. That is, depending upon the distribution of invalid source and data channels, the index of a source channel may be greater than, less than, or equal to the index of the corresponding load channel. Another approach might be to use a register of n cells, in which propagation may be forced in either direction.

Stages 1 through n receive information in parallel from source channels s_1 through s_n respectively, and stages n + 1 through 2n deliver information in parallel to load channels ℓ_1 through ℓ_n. It is assumed that some diagnostic process distinguishes the validity or invalidity of the data channels. Information as to which load channel shall receive data is stored within the corresponding cell of the

---

* Conceived by D. Muller.[221]

cascade. In the identification of the validity of data source channels, there are two design choices:

(a) The s signals may have a three-valued encoding, e.g., 0, 1, Ø, in which one symbol Ø indicates an invalid source.

(b) Information as to the validity of a channel may be stored within the network, at the corresponding cell.

Choice b will be assumed here, because it leads to the use of the same cells throughout the cascade.

Information as to the validity or invalidity of each data channel, then, is stored at the corresponding cell. This storage is assumed to be accomplished during a setup mode. The processing mode has three phases: write-in, propagation, and read-out. The first and last are parallel processes on the s and ℓ channels respectively. In the second, information propagates toward stages of higher index, and comes to rest in a stable configuration in such a way that symbols obtained from valid source channels are collected in a contiguous string, starting at the rightmost cell but skipping cells corresponding to invalid load channels.

In the event that the number of valid symbols exceeds the number of valid load channels, the leftmost symbols will be lost. If the opposite is true, the leftmost load channels will be unused. The design to be described provides that a special symbol indicating the absence of valid data is available for entry to such channels.

2) General Description of the Propagation Mode

Before describing the detailed design of the cells, the overall behavior of the propagation mode will be described, with the aid of Fig. III-C-3.

Each cell is composed of two stages, which may be considered to be identical for simplicity of description. The figure illustrates a case in which an incident pattern 1, 0, 1 appears on channels 1, 3, and 4 respectively and the pattern 1, 1, 0 is read out

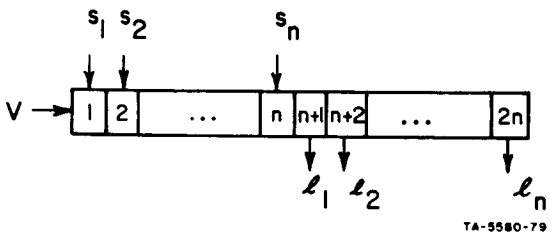


FIG. III-C-2 BLOCK DIAGRAM OF A SHIFT-REGISTER COMMUTATING NETWORK

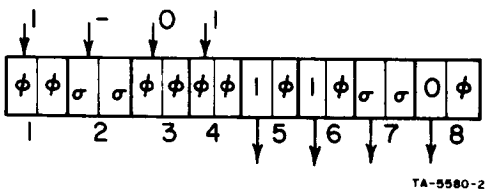


FIG. III-C-3 TYPICAL SYMBOL STATES IN AN ASYNCHRONOUS SHIFT-REGISTER COMMUTATOR

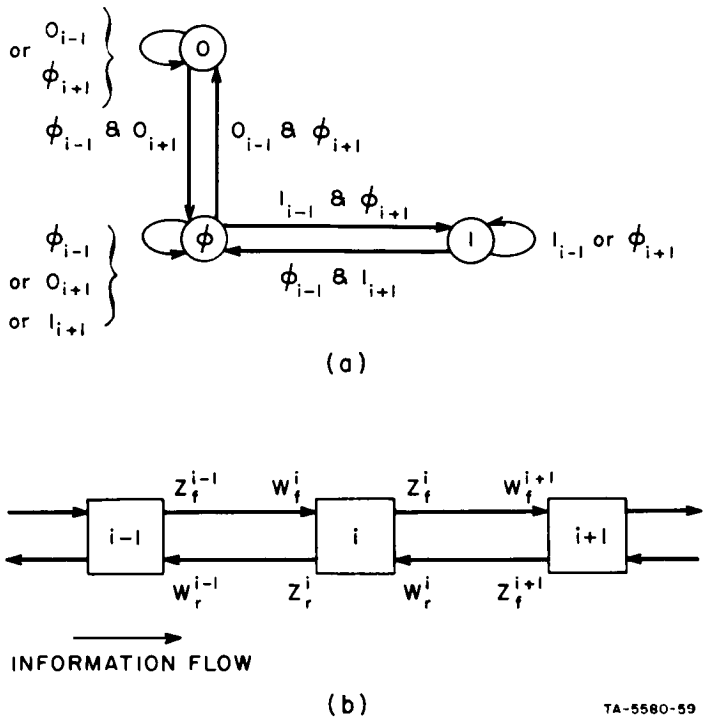


FIG. III-C-4 BLOCK AND STATE DIAGRAMS FOR A SPEED-INDEPENDENT MODULE

to channels 5, 6, and 8, respectively. The symbol σ in both stages of cells 2 and 7 indicates invalid channels. The $\emptyset$ symbols shown serve two functions: they serve to indicate the absence of valid data, and they serve to separate independent valid symbols; i.e., the network is designed so that two valid symbols may never come to rest in adjacent stages.*

In the propagation mode, then, each stage acts as a repeater of incident information, with temporary data storage. The effect of storing a σ symbol in a stage is to remove its temporary data storage capacity.

3) Description of the Cell Design

Since the network is derived from Muller's Speed-Independent module, a brief review of that module will be given. Following that, a state diagram will be derived, and then a logical realization.

Review of Muller's SI stage: Muller's cascade is illustrated in Fig. III-C-4(a). The stages are indexed i - 1, i, i + 1. The inputs to stage i are w_f^i and w_r^i, which are obtained from z_f^{i-1} and z_r^{i+1} respectively. All paths carry one of the three symbols 0, 1, $\emptyset$, and on the basic design $z_f^i = z_r^i$.

The state diagram for stage i is shown in Fig. III-C-4(b). On the branches, symbols with index i-1 appear at the w_f input, and symbols with index i + 1 appear at the w_r input. The symbols in the state circles are those presented on the z_r, z_f outputs.

The state behavior provides for the temporary storage and propagation of information, at a rate dependent on the time-response characteristics of the stages. The feedback of information in the

* In the transient mode, a given symbol may be temporarily replicated in two or more adjacent stages, but strings of replicas that are derived from different sources will be separated by one or more $\emptyset$ symbols. For example, the sequence ... 0 1 1 0 1 ... may appear momentarily in the cascade as the string ... $\emptyset$ 0 $\emptyset$ 1 1 1 $\emptyset$ 1 $\emptyset$ 0 0 $\emptyset$ $\emptyset$ 1 $\emptyset$ ..., and will come to rest in the form ... $\emptyset$ 0 $\emptyset$ 1 $\emptyset$ 1 $\emptyset$ 0 $\emptyset$ 1 ....

direction opposite to that of propagation serves to prevent accidental elimination of data. This feature may be exploited to permit the blocking of the flow of information by forcing the feedback path to some stage to a 0 or 1 status. The data will then accumulate in a contiguous string of alternating spacer (Ø) and data (0 or 1) symbols.

State behavior of an augmented cell: As described in the general description of the propagation mode of the buffer network, the effect of labeling a cell as invalid is to deprive it of its temporary data-storage capability. This is accomplished by adding a fourth state, σ, to a basic Muller stage, in which the forward-feeding output signal z_f^i repeats the forward-incident w_f^i signal and the feedback output signal z_r^i repeats the incident feedback w_r^i signal, each without delay. The output functions in the other three states are exactly as in the Muller stage.

There are several design possibilities for the transitions into and out of the σ state. Pernaps the simplest (at least for purposes of describing a basic design) is to assume one special input signal m_s, capable of forcing a transition into σ from any other state, and a second special input signal m_r, capable of setting the stage to some other state, most naturally the Ø state.*

The block diagram is shown in Fig. III-C-5(a) and the state diagram is shown in Fig. III-C-5(b). The conventions of Fig. III-C-4 apply to the inputs (with the addition of the m^i input, whose values are m_s, m_r, and $m_\emptyset$), but the outputs z_f and z_r are indicated explicitly at the state circles. The dotted lines indicate possible parallel data inputs and outputs.

For simplicity, two functions have not been shown in the state diagram; these are the writing of data into a cell from an external source, and the permitting or inhibiting of propagation within the cascade. The first is trivial, given the ability to accomplish the

---

* A more elegant means (requiring fewer special input lines) might be to provide that a string of 0's and 1's be entered serially (representing invalid and valid cells), and then "frozen" into σ and non-σ states upon special command.

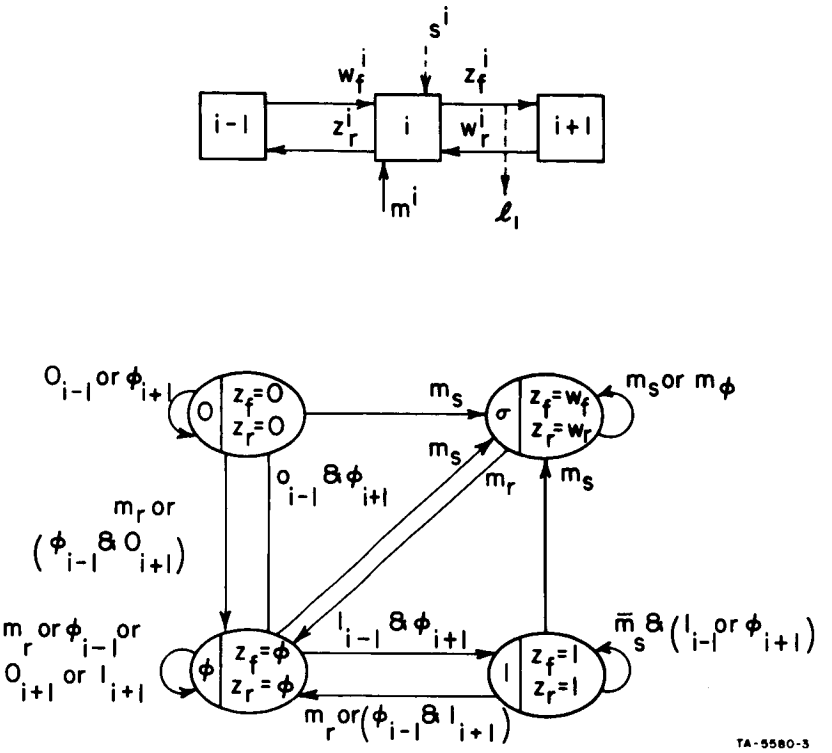


FIG. III-C-5 BLOCK AND STATE DIAGRAMS OF FULL SHIFT-REGISTER CELL

second. One satisfactory way to inhibit propagation is to provide a common signal to the second stage of all cells, that injects a 1 signal into the w_r input. This will serve to keep all such cells in the Ø state (at which state the cells naturally arrive following all complete propagations). To permit propagation, these 1 signals are removed. Even if the signals are not removed simultaneously, the presence of a Ø symbol within the driven cell prevents the loss of information.

By applying such a signal only at the last stage in the cascade, information may be caused to "pile up" in a stationary pattern at the end of the cascade. Removal of the signal will permit the discharge of the information from the cascade.

A logical realization: In the following illustrative realization, the w and z signal variables and the x state variable are encoded as pairs of binary variables, as follows: $w_f = (w_{f1}, w_{f2})$, $w_r = (w_{r1}, w_{r2})$, $z_f = (z_{f1}, z_{f2})$, $z_r = (z_{r1}, z_{r2})$, and $x = (x_1, x_2)$. The values corresponding to the symbols of the state diagram are

0 : 0 1

1 : 1 1

ø : 0 0

σ : 1 0 .

The m variable is likewise encoded as a pair, (m_s, m_r), and it is assumed that $m_s \cdot m_r = 0$.

Table III-C-1 lists the symbolic and logical functions governing entry to the four states, together with the output functions of the states. The conditions for external entry of data are also included, with $s_i = (0, 1)$ the external data, "write" = (0, 1), the sampling function, and p = (0, 1), the "propagate release" function.

Table III-C-1

STATE-TRANSITION AND OUTPUT LOGIC FOR REGISTER CELL

State x	Code x_1x_2	Entry Function--logical	Entry Function--Boolean	Outputs			
				z_{f1}	z_{f2}	z_{r1}	z_{r2}
σ	1 0	m_s	m_s	w_{f1}	w_{f2}	w_{r1}	w_{r2}
ø	0 0	m_r or $ø_{i-1}$ & $(0_{i+1}$ or $1_{i+1})$	$m_r + \bar{w}_{f2}w_{r2}p$	0	0	0	0
0	0 1	$(0_{i-1}$ & $ø_{i+1})$ or $(\bar{s}_i$ & write)	$w_{f2}\bar{w}_{f1}\bar{w}_{r2}p + \bar{s}_i$.write	0	1	0	1
1	1 1	$(1_{i-1}$ & $ø_{i+1})$ or $(s_i$ & write)	$w_{f2}w_{f1}\bar{w}_{r2}p + s_i$.write	1	1	1	1

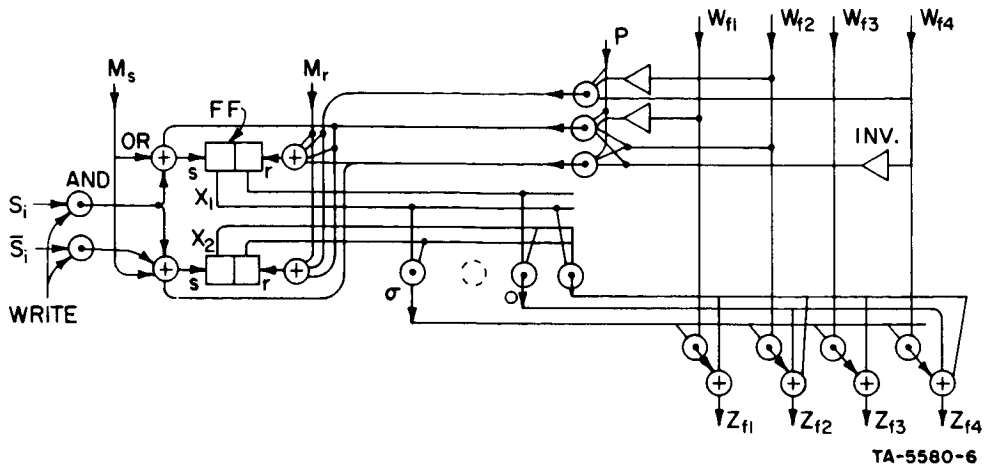


FIG. III-C-6 LOGICAL REALIZATION OF SHIFT-REGISTER CELL

Figure III-C-6 shows a logical realization using flip-flops, AND gates, OR gates, and inverters.

4) Summary

The scheme described is one of several possibilities. Other asynchronous schemes should be investigated, as well as the synchronous-sequential and combinational approaches. Some merits of the present scheme are (1) its complete flexibility within the limits of redundancy, (2) the linear growth in size with number of channels, and (3) the absence of the need for a clock. These merits are achieved at the cost of some control logic (not shown here) for the sequencing of the various phases of operation, and of time (not calculated here). An additional feature which may be useful in applications such as parallel arithmetic is that the order of information among the input channels is preserved at the output.

c. Combinational Commutation Networks--Minimization of Number of Switches

1) Introduction

In this section and the two succeeding sections we present a detailed examination of the combinational commutation networks. We distinguish these networks as combinational because in estimating the

expected complexity, for given values of N_1, N_2, M it is convenient to visualize the commutation network as a multiple-input multiple-output combinational network where the inputs are the data lines from the input arrays along with the status lines, and the outputs are the data lines to the output array. Indeed, these commutation networks can be implemented strictly as combinational circuits, working from the truth table; or for some applications it might be convenient to incorporate some sequential blocks.

At present we seek techniques for minimizing the complexity of the commutation networks. In order to facilitate the synthesis we propose to initially consider the commutation network as a switch network, i.e. a net of single-pole single-throw switches. We can then state the following problem relating to the synthesis of efficient commutation networks:

> For a given number M of operating modules in both the input and output arrays, and a set of N_1 - M spare input modules and N_2 - M spare output modules, find the switch network, denoted as the minimal switch network, with the least number of switches such that all sets of M inputs and M outputs are connectable both when the ordering of the input data is preserved at the output and when disordering is allowed.

It should be noted that the minimization procedure based upon this switch model is not necessarily an optimum approach for all applications, but it is felt that efficient designs will generally result from the consideration of the minimal switch networks as a point of embarkation.

We now distinguish types of switch networks according to the number of intermediate levels present between the input and output arrays. To be more specific, first consider the single-level switch network of Fig. III-C-7(a). In this case the lines between the N_1 inputs and the N_2 outputs represent switches that are either open or closed. A double-level switch network is shown in Fig. III-C-7. In this case there are a set of switches connecting the N_1 inputs with the N′ intermediate collection points, and also a set of switches connecting the N_1

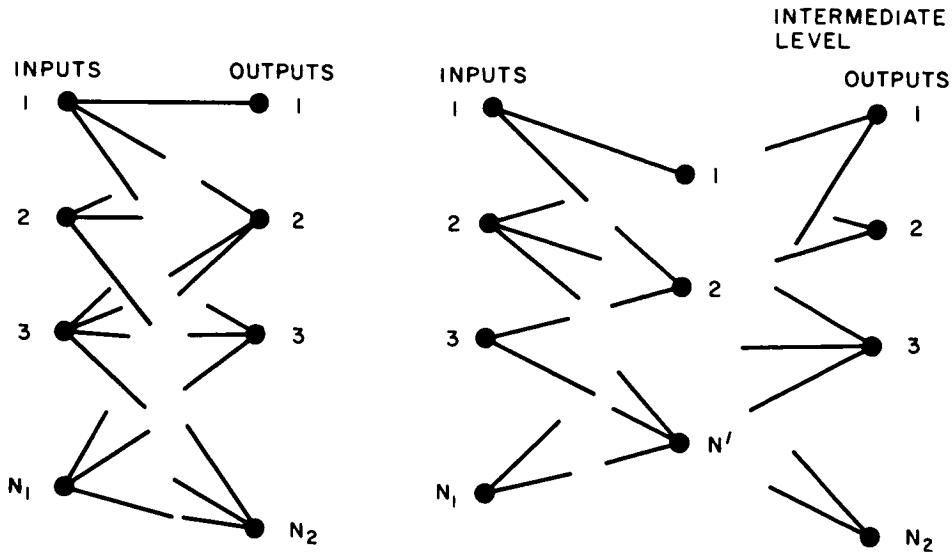


FIG. III-C-7 SINGLE AND DOUBLE-LEVEL INTERCONNECTION SCHEMES

outputs with the N' intermediate points. The extension to general multi-level switch networks is straightforward.

One immediately observes that the required commutation can be accomplished if the single-level switch network contains $N_1 N_2$ switches, but we will show that significantly fewer switches will suffice. In addition the commutation can be accomplished by a double-level switch containing $N' = M$ intermediate collection points and $M(N_1 + N_2)$ switches, but once again this connection is not minimal.

We will demonstrate the techniques for the realization of minimal single-level switch networks, for both the order-preserving and non-order-preserving cases. For most values of the parameters M, N_1, and N_2, double-level switch networks can be found which are less complex than the corresponding minimal single-level network. Unfortunately, we do not know the techniques for realizing precisely minimal double-level networks, nor do we indeed know the optimum number of intermediate levels, but a

technique is presented for realizing double-level networks which appear to be "near" minimal. Examples of networks which are less costly are discussed, so as to provide a basis for future research.

First we will consider the case where the ordering of the input signals is preserved at the output. Primarily intuitive methods will be employed to establish that particular switch networks can perform the required reconfiguration.

2) Single-Level Order-Preserving Network

The number of switches required for the single-level switch network may be counted with the aid of Fig. III-C-8, in which the horizontal lines represent input buses, the vertical lines represent output buses, and a heavy dot represents a switch connecting an input and output bus. For ease of visualization, the case $N_1 = 15$, $N_2 = 15$, $M = 5$ is illustrated. (Although the example indicates a case where $N_1 = N_b$, the theory is applicable for cases where there are an arbitrary number of spare modules for the input and output arrays; these general results may be useful when the input and output arrays are of different reliability.) Input 1 need be connected to only $N_2 - M + 1$ outputs to assure it of access to one of the M valid outputs. In order to preserve the order of input and output channels, input 2 must be able to cover the outputs of input 1 (since 1 may be inactive) plus one more (in case input 1 is active). Successive inputs must thus cover the outputs of the preceding inputs, plus one more, until the $(N_1 - M + 2)$th input. That input need not cover the first output, since if the first output is active it will be supplied a signal by one of the preceding $N_1 - M + 1$ inputs, one of which must surely carry a valid signal. Successive inputs, then, need cover one output terminal less.

The number of required switches $S_{SL}^{(op)}$, may be seen, by inspection of Fig. III-C-8 to be

$$S_{SL}^{(op)} = N_1 N_2 - 2\sum_{i=1}^{M} i = N_1 N_2 - M(M-1).$$

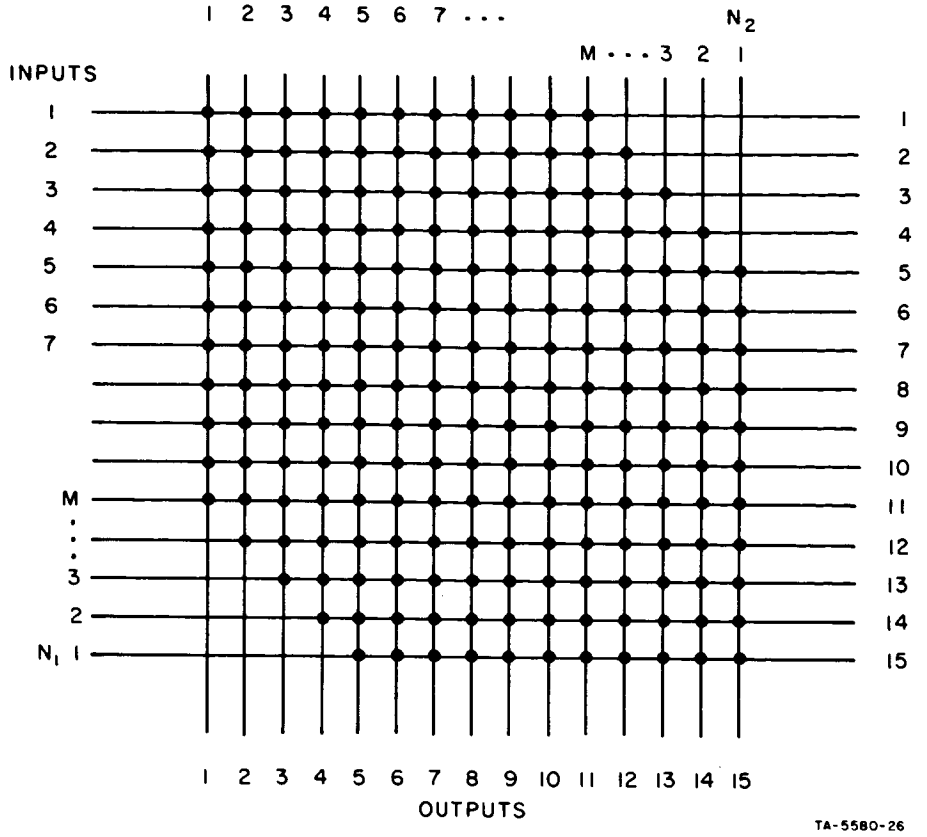


FIG. III-C-8 SINGLE-LEVEL ORDER-PRESERVING COMMUTATION NETWORK $N_1 = N_2 = 15$, $M = 5$

3) Double-Level Order-Preserving Network

Under some conditions, a double-level network may be advantageous. The structure of a possible network is shown in Fig. III-C-9, in which the horizontal lines represent input and output buses, and the vertical lines represent intermediate buses.

Considering the intermediate buses, labeled in numerical order, 1, 2, 3, ..., we may note that each bus must be connected to an active input. The first intermediate bus, 1, need by connectable only to the first $N_1 - M + 1$ inputs in order to serve the first valid input; the second intermediate bus, 2, need be connectable only to inputs 2 through $N_1 - M + 2$ in order to serve the second valid input, since, at worst, if the first valid input is at input $N_1 - M + 1$ (11, here), the second valid input must occur at input $N_1 - M + 2$, the third at input $N_1 - M + 3$, and so on, with the Mth at input $N_1 - M + M = N_1$.

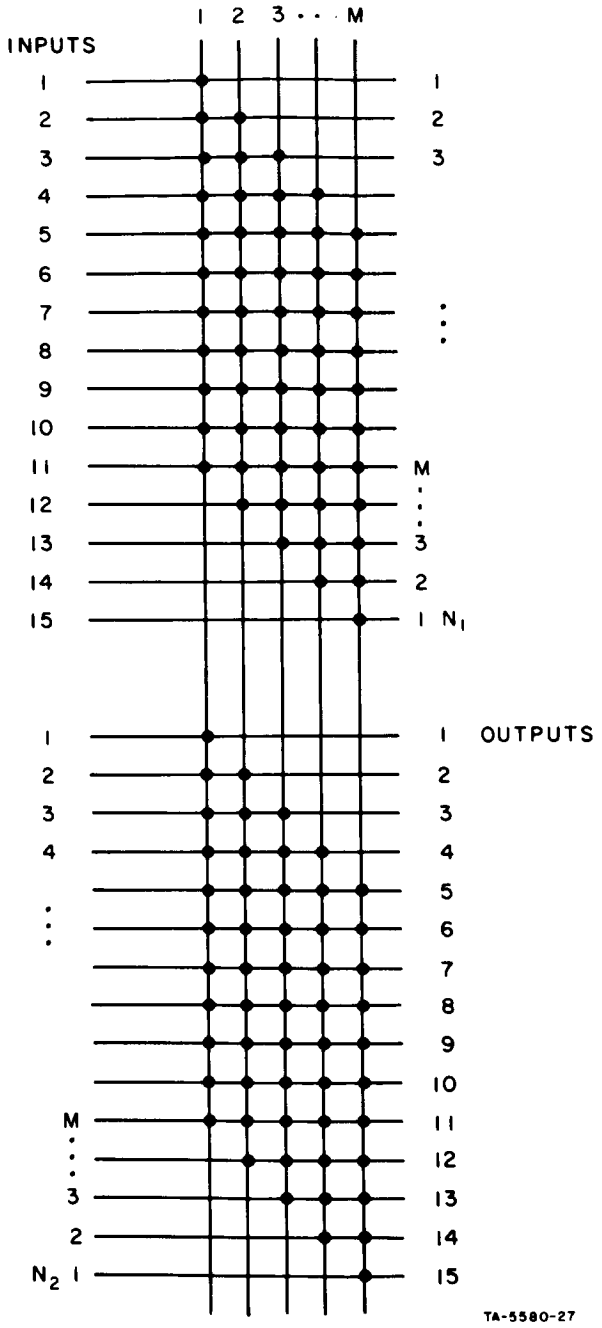


FIG. III-C-9 DOUBLE-LEVEL ORDER-PRESERVING COMMUTATION NETWORK
$N_1 = N_2 = 15$, $M = 5$

The second set of switches, connecting the intermediate buses to the outputs, follows the same construction. The number of switches $S_{DL}^{(op)}$ required for this two-level switch may be seen, by inspection of the figure to be

$$S_{DL}^{(op)} = M(N_1 - M + 1) + M(N_2 - M + 1) = M(N_1 + N_2) - 2M(M - 1).$$

It is instructive to compare the two networks with respect to the number of switches required. To be specific, we seek the relationship among the variables N_1, N_2, and M such that $S_{DL}^{(op)} > S_{SL}^{(op)}$. In order to simplify the analysis we will consider the case $N_1 = N_2 = N$. Then in order that $S_{DL}^{(op)} > S_{SL}^{(op)}$ it is necessary that

$$2MN - 2M^2 + 2M > N^2 - M^2 + M$$

which is equivalent to the condition that

$$N^2 - 2MN + M^2 > M$$

or

$$(N - M)^2 > M$$

or

$$N > M + \sqrt{M} \quad .$$

The "break-even" values of N for various M's are given in Table III-C-2; for values of N equal to and above those given, single-level switching is more costly than double-level switching, at least for the double-level switch described.

Table III-C-2

"BREAK-EVEN" VALUES OF N

M	1	2	3	4	5	10	20	50
N_{BE}	3	4	5	7	8	14	25	58

It is conjectured that this double-level switch network is the minimal double-level switch network (and in general the minimal multilevel switch network) when the order is preserved not only on the output buses, but also on all intermediate level buses. However, it is clear that the order need not be preserved at the intermediate level, and in fact we have found some examples which illustrate cases where less costly order-preserving networks result when disordering at the intermediate buses is allowed.* At present we have not been able to adequately generalize these examples to provide large classes of economical networks, but it appears that only a small reduction in cost is achieved when compared with the simple double-level realizations of Fig. III-C-8.

We now proceed to investigate non-order-preserving switch networks. Our motivation for studying such networks is based upon the fact that the realizations appear to be less costly than the order-preserving realizations,† and also that we envision future computers as being composed of large reconfigurable arrays of identical programmable modules in which case it might be feasible to consider an identifier appended to the data on each signal line.

4) Single-Level Non-Order-Preserving Network

The switch network schematic as illustrated in Fig. III-C-7(a) is suggestive of a graph called a bipartite graph.[233]§ The

* The realization of sample networks is discussed further on in this section when the techniques of non-order-preserving networks are described.

† It appears that the non-order-preserving networks offer significant economy only if N_1 and N_2 are quite unequal.

§ The method for establishing that particular non-order-preserving switch networks perform the required reconfiguration was formulated by B. Elspas on the basis of the properties of bipartite graphs. Briefly, a graph consists of certain points, called its vertices, and certain line segments connecting vertices, called the edges of the graph. A graph where the set of vertices is decomposed into two separate parts such that there are edges only between these parts is called a bipartite graph. This type of graph is commonly used to match a set of available jobs with a set of men, when each man is qualified for certain of the jobs.

set of input modules are represented by a set of N_1 vertices, the output modules by a set of N_2 vertices, and the switches by lines connecting the two sets of vertices.

It is indicated in the footnote below that the theory of bipartite graphs can be applied to the problem of matching men and jobs. A very powerful condition, known as the diversity condition[233] has been derived in order to establish whether a suitable job exists for each man:

> Suppose there are σ men applying for positions. Then each man can be assigned if and only if for each group of β men, for all $\beta = 1, 2, \ldots, \sigma$, there are at least σ jobs for which they are collectively qualified.

The diversity condition can be used to determine whether a particular single-level switch network can reconfigure M input and output modules despite the occurrence of $N_1 - M$ or fewer failures in the input array and $N_2 - M$ or fewer failures in the output array, as follows:

> The non-order-preserving switch network can perform the required reconfiguration if and only if each group of γ input modules, for all $\gamma = 1, 2, \ldots, M$, is connectable collectively to at least $\gamma + N_2 - M$ output modules, and if and only if each group of δ output modules, for all $\delta = 1, 2, \ldots, M$ is connectable collectively to at least $\delta + N_1 - M$ input modules.

It is convenient, in performing analyses utilizing the diversity condition, to describe the switch network in terms of an $N_1 \times N_2$ Boolean matrix B, where the entry b_{ij} is 1 if the i^{th} input module is connectable to the j^{th} output module, and 0 otherwise. Then the reconfigurability of a particular network can be stated in terms of the matrix B as follows:

> The switch network can perform the required reconfiguration if for the matrix B, a vector which is the Boolean sum of any γ rows, $\gamma = 1, 2, \ldots, M$, has weight (number of entries which are one) at least $\gamma + N_2 - M$, and if the vector which is the Boolean sum of any δ columns, $\delta = 1, 2, \ldots, M$, has weight at least $\delta + N_1 - M$.

The above condition will now be utilized in connection with a specific case. Consider the switch network shown in Fig. III-C-10 and its matrix B below.

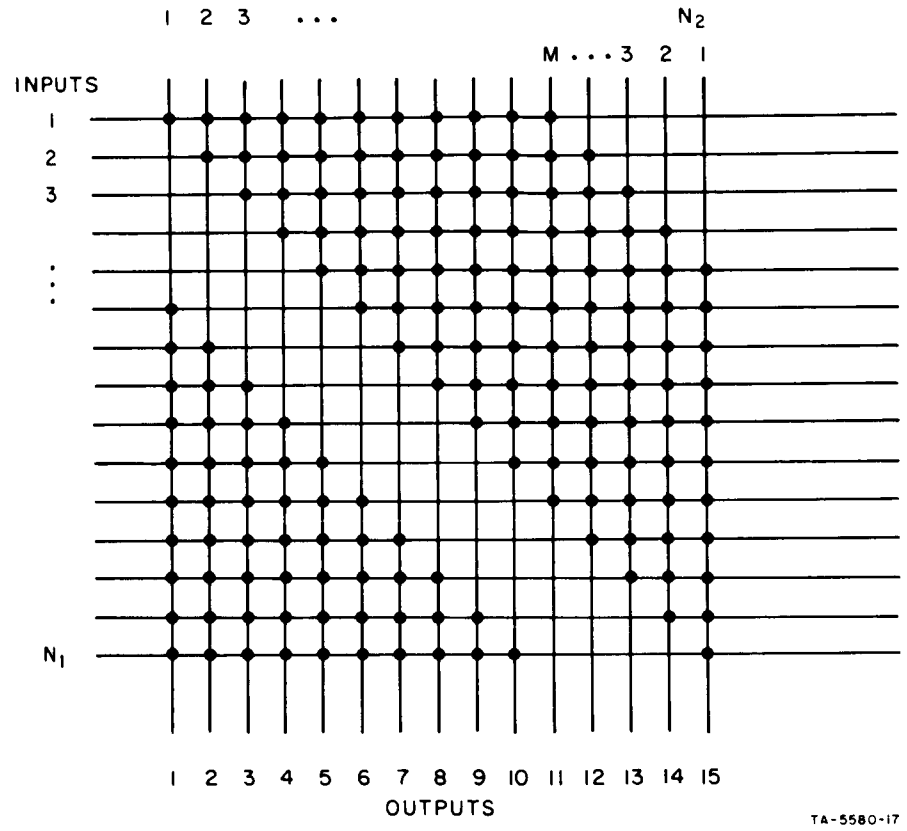


FIG. III-C-10 SINGLE-LEVEL NON-ORDER-PRESERVING COMMUTATION NETWORK $N_1 = N_2 = 15$, $M = 5$

$$B = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{array} \begin{array}{c} \begin{array}{ccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{array} \\ \left[\begin{array}{ccccccccccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \end{array}$$

Clearly no network is less costly since each row in the above matrix contains the minimum allotment of $N_2 - M + 1 = 11$ ones. It now remains to demonstrate that the Boolean sums of rows (and equivalently columns since the matrix is symmetric except for cyclic permutations) satisfies the weight conditions. Form a vector V given by

$$V = \bigcup_{\ell=1}^{m} (V_{k_\ell}), \qquad m \leq M = 5$$

where V_{k_ℓ} is the k_ℓ^{th} row vector of B and $\cup$ represents the Boolean sum of the vectors.

It is easy to verify that the minimum weight of v, $W_{min}(v)$ is given by $W_{min}(v) = N_2 - M + 1 + m$, which is the weight when k_1, k_2, ..., k_m are successive integers, where 16 is taken to be equivalent to 1. Thus the diversity condition is satisfied and the indicated switch network can perform the required reconfiguration. Although for the example shown it was specified that $N_1 = N_2$, a minimal single-level non-order-preserving switch network with, for example, $N_1 > N_2$ can be

synthesized from a Boolean matrix whose first column is a vector consisting of $N_1 - M + 1$ ones followed by $M - 1$ zeros, and whose succeeding columns are any set of distinct cyclic shifts of this column. For this network the number of switches $S_{SL}^{(nop)}$ is given by

$$S_{SL}^{(nop)} = [\text{Min}(N_1, N_2)] \cdot \left\{[\text{Max}(N_1, N_2) - M + 1]\right\} .$$

5. Double-Level Non-Order-Preserving Network

Here we are concerned with synthesizing double-level non-order-preserving switch networks which are approximately minimal. Techniques are presented from which double-level networks can be synthesized, although at present we do not know how close they are to minimal.

The general structure of the double-level network to be investigated is illustrated in Fig. III-C-11. It consists of a distributing network whose function is to distribute the information from any arbitrary set of M input lines onto any M of N' intermediate lines, followed by a collecting network whose function is to collect the signals appearing on any M intermediate lines for delivery to any M output lines.* In driving the number of switches required by the collecting network we note that it performs the same function as a single level non-order-preserving network with N' input channels. Hence the required number of collecting switches S_c is

$$S_c = [\text{Min}(N_2, N')] \cdot \left\{[\text{Max}\ (N_2, N') - M + 1]\right\} .$$

The number of switches required by the distributing network is considered below. It is not clear that this method of arbitrary distribution followed by collection is indeed the most efficient commutation technique, but since networks which are significantly more economical than single-level realizations did result in example cases, we will briefly pursue some procedures for synthesis.

* The double-level network as shown with the distributing network associated with the inputs is based upon the assumption that $N_1 \geq N_2$. If $N_2 \geq N_1$ then the distributing network should be associated with the outputs for optimum economy.

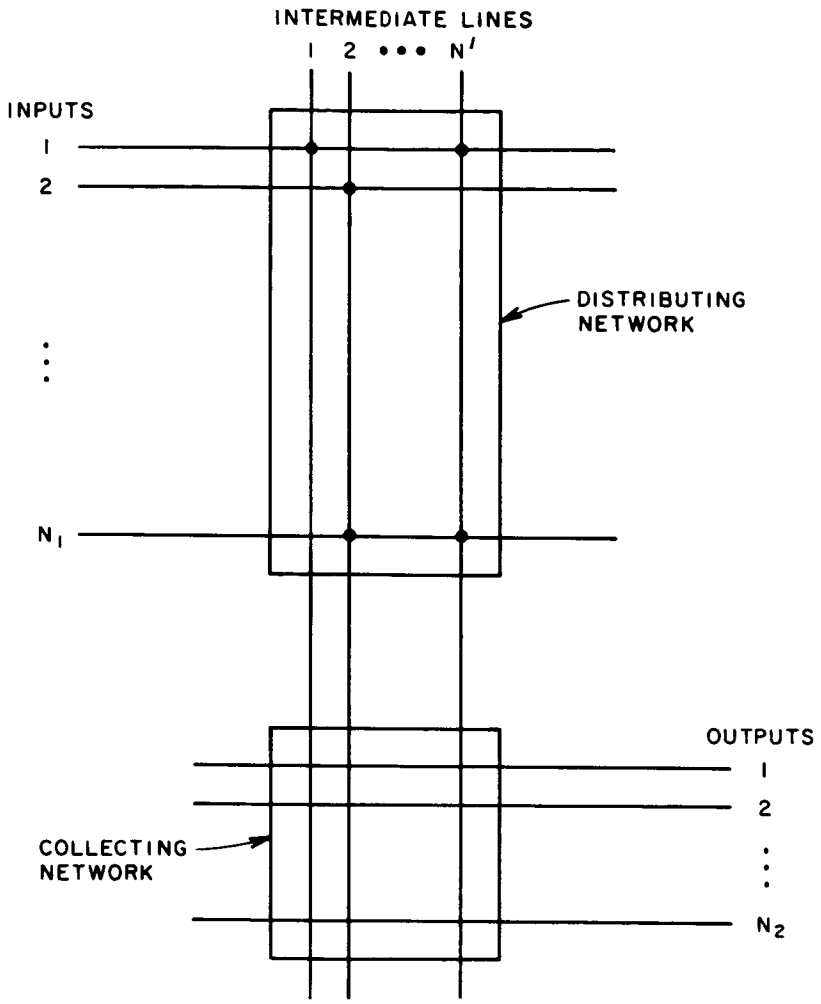


FIG. III-C-11 GENERAL DOUBLE-LEVEL NON-ORDER-PRESERVING NETWORK

Consider first the case where the number of intermediate levels assumes a minimal value, namely N' = M. Then the function of the distributing network is identical to that of the single-level network with N' = M output channels, and the total number of switches required, $S_{DL}^{(nop)}$, is

$$S_{DL}^{(nop)} = M[N_1 - M + 1] + M[N_2 - M + 1]$$

$$= M[N_1 + N_2 - 2M + 2] \quad .$$

It is noted that this result is identical to the number of switches required for the order-preserving case.

However, specifying N' = M does not generally result in the most economical double-level networks. For example consider a distributing network, with parameters $N_1 = 15$, N' = 6, M = 5 ($N_2 = 15$), described by the following matrix, B'.

$$B' = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} = N' \\ \begin{bmatrix} 1 & 1 & 1 & & & \\ 1 & 1 & & 1 & & \\ 1 & & 1 & 1 & & \\ & 1 & 1 & 1 & & \\ 1 & 1 & & & 1 & \\ 1 & & 1 & & 1 & \\ 1 & & & 1 & 1 & \\ & 1 & 1 & & 1 & \\ & 1 & & 1 & 1 & \\ & & 1 & 1 & 1 & \\ 1 & & & & 1 & 1 \\ & & 1 & & 1 & 1 \\ & & & 1 & 1 & 1 \\ 1 & & 1 & & & 1 \\ 1 & 1 & & & & 1 \end{bmatrix} \end{array}$$

We will show that the double-level switch network containing a distributing network related to the above matrix B', with the addition of the appropriate collecting matrix, is more economical than

the network illustrated in Fig. III-C-10. First it is necessary to prove that for the indicated distributing network each set of M input channels is connectable to M distinct intermediate lines.

This will be accomplished by using the diversity condition as related to the matrix B'; i.e., it is necessary to show that the weight of the vector formed as the Boolean sum of any j rows, $j = 1, \ldots, M$, is at least j. Clearly, the condition is satisfied for $j = 1, 2, 3$ since the weight of each row is 3, and the condition is also satisfied by $j = 4$ since all row pairs differ in at least 2 places (i.e. maximum overlap* is 2). However, we note that the number of vectors of weight 3 with overlap exactly 2 is limited to 4; hence if we consider 5 vectors there must be at least one row pair with overlap not exceeding 1. Then the Boolean sum of any 5 rows yields a vector of minimum weight 5, thus establishing the diversity condition.

It can be shown that the 4 underlined entries in the B' matrix can be discarded without affecting the diversity condition. Hence the distributing network contains $3(15) - 4 = 41$ switches, which when added to the $6(15 - 5 + 1) = 66$ switches required for the collecting array indicates a total of 107 switches as compared with the 110 switches for the double-level network with $N' = M$. Admittedly the relative improvement in economy is not startling, and indeed it appears that for $N_1 \approx N_2$, the double-level network with $N' = M$ is quite economical; but if N_1 and N_2 are somewhat unequal, significant economy is afforded by allowing $N' > M$.

Unfortunately we have not formulated deterministic techniques for deriving the optimum value of N', nor do we know techniques for synthesizing distributing networks for all values of the parameters N_1, N', M; but by a cut-and-try process it is generally possible to synthesize reasonably economical networks. The most efficient distributing networks we have found are based upon balanced incomplete block designs.

---

* The overlap of two vectors is defined as the weight of the dot product.

Before proceding to a discussion of the distributing networks based upon block designs it is convenient to consider the problem of determining a lower bound on the weight of the Boolean sum of vectors, subject to certain constraints on the structure of the vectors. Consider a set of vectors of weight W and maximum overlap λ_{max}. It can be shown from a result presented in connection with another problem[152] that the weight of the Boolean sum of j vectors, each of weight W with a maximum overlap λ_{max} is greater than j provided the inequality

$$j \geq \frac{\binom{j}{\lambda_{max}+1}}{\binom{w}{\lambda_{max}+1}}$$

is satisfied.

It is not difficult to show that if the inequality is satisfied for some integer j, then it is satisfied for all integers less than j. Hence if the matrix for a distributing network consists of rows of weight W with a maximum overlap, λ_{max}, then all sets of m inputs are connectable if the inequality

$$m \geq \frac{\binom{m}{\lambda_{max}+1}}{\binom{w}{\lambda_{max}+1}}$$

is satisfied. We will now demonstrate the technique of forming distributing networks from block designs.

Briefly, a block design* constitutes a multiparameter arrangement of objects, which for present purposes may conveniently be represented as a matrix of zeros and ones. The incidence matrix B″ of a so-called balanced incomplete block design (BIBD) with parameters

---

* For a complete discussion of block designs the reader is referred to Ref. 121.

(v, k, b, r, λ) has b rows, v columns, k ones per row, and r ones per column, and is such that the dot product of every pair of columns is just λ. The well-known identities

$$vr = bk$$

$$r(k - 1) = (v - 1)$$

must be satisfied.

Generally we will identify the rows of B'' with the switch connection from the input channels (i.e. $N_1 = b$, $M' = v$), since for a BIBD $v \leq b$. (Otherwise a distributing network with $N' > N_1$ would result, yielding an inefficient network.) However, in order to determine whether the diversity condition is satisfied for the case of interest it is necessary to know the maximum value of the dot product μ_{max} of any pair of rows of B".

For the case where the rows of B'' consist of all of the $\binom{v}{k}$ combinations of vectors of length v each with k ones, then $\mu_{max} = v - 1$. For example, the matrix B_1'' for the block design of this type with v = 6, r = 10, b = 20, k = 3, λ = 2, is shown below

B_1'' =

	1	2	3	4	5	6
1	1	1	1			
2	1	1		1		
3	1	1			1	
4	1	1				1
5	1		1	1		
6	1		1		1	
7	1		1			1
8	1			1	1	
9	1			1		1
10	1				1	1
11		1	1	1		
12		1	1		1	
13		1	1			1
14		1		1	1	
15		1		1		1
16		1			1	1
17			1	1	1	
18			1	1		1
19			1		1	1
20				1	1	1

Applying the inequality concerning the weight of the Boolean sum of vectors, we note that the matrix B_1'' describes a valid distributing network for all m satisfying $m \leq 5$.

Another case where μ_{max} is known for a BIBD is for $\lambda = 1$, in which case $\mu_{max} = 1$. Many examples of BIBD's with $\lambda_{max} = 1$ have been tabulated.[121,45]

It is of interest to compare the number of switches in the various types of networks (minimal or approximately minimal) distinguished as single-level order-preserving, double-level order-preserving, single-level non-order-preserving, double-level non-order-preserving ($N' = M$), and double-level non-order-preserving ($N' \neq M$). The results for two examples are tabulated in Table III-C-3 for commutation networks with parameters $N_1 = 20$, $N_2 = 6$, $M = 5$ and $N_1 = 63$, $N_2 = 28$, $M = 13$. The appropriate distributing network for the former example (illustrated with the appropriate collecting network in Fig. III-C-12 is based upon the matrix B_1'' with $N' = 6$, and the distributing network for the latter case is based upon the block design with parameters $b = 63$, $v = 28$, $k = 4$, $r = 9$, $\lambda = 1$, in which case $N' = 28$. (It is easily verified that the "pertinent inequality" is satisfied for $m \leq 13$ with these parameters.)

Table III-C-3

COMPARISON OF SWITCH NETWORKS

	$N_1 = 20$, $N_2 = 6$, $M = 5$	$N_1 = 63$, $N_2 = 28$, $M = 13$
Single-level order-preserving	100	1608
Double-level order-preserving	90	871
Single-level non-order-preserving	96	1428
Double-level non-order-preserving ($N' = M$)	90	871
Double-level non-order-preserving ($N' \neq M$)	72	644

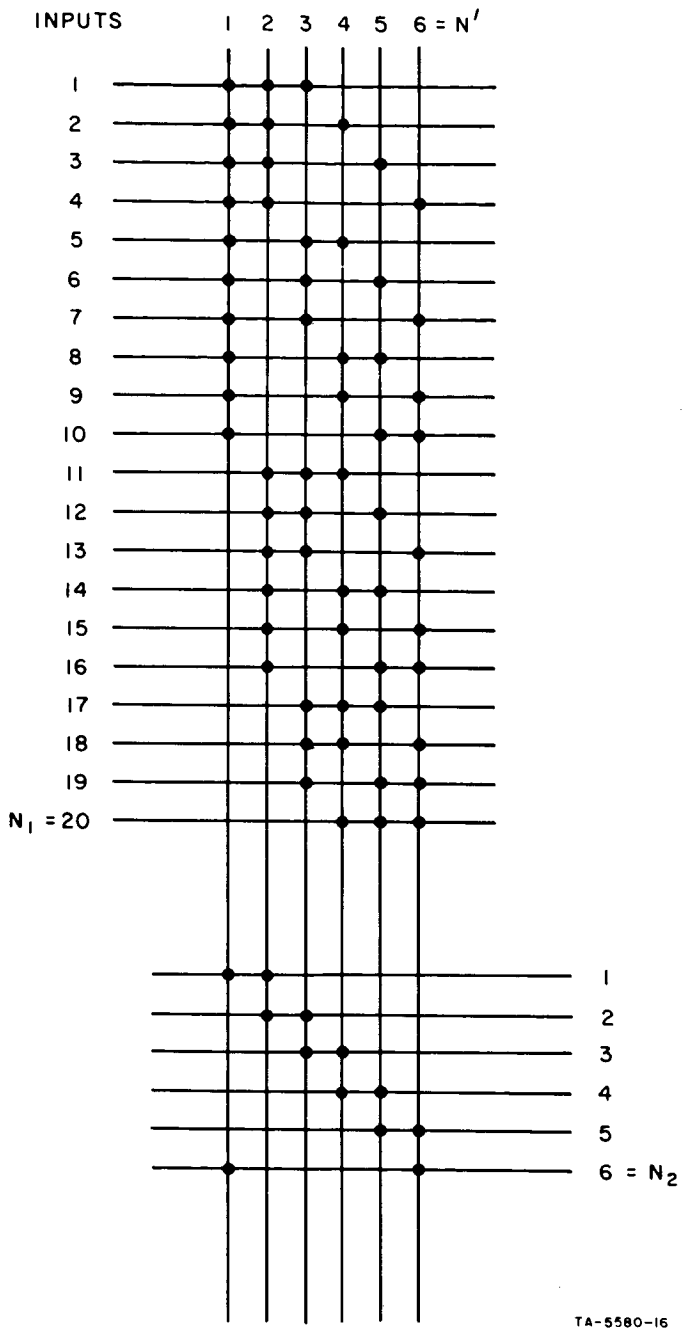


FIG. III-C-12 DOUBLE-LEVEL NON-ORDER-PRESERVING COMMUTATION NETWORK $N_1 = 20$, $N_2 = 6$, $N' = 6$, $M = 5$, 72 SWITCHES

It is seen that the double-level non-order-preserving switch networks with $N' \neq M$ are clearly the most economical. Admittedly these examples are somewhat artificial since the parameters were chosen so as to be consistent with the parameters of known balanced incomplete block designs. We cannot state a formal procedure for synthesizing economical double-level networks for all values of parameters N_1, N_2, M, but the following method appears to result in "fairly good" networks.

Choose a BIBD with parameter $\lambda = 1$; parameter k a minimum such that the inequality for the number of active channels M is satisfied; parameter $b = N_1$*; and parameter $v > M$. If several designs satisfying these properties exist, then a cut-and-try procedure is employed in order to select the design which results in the most economical network.

Before concluding this section we can present an order-preserving network which is more economical than a double-level network of the type shown in Fig. III-C-9. The network for parameters $N_1 = 45$, $N_2 = 4$, $M = 3$ consists of a distributing network based upon the BIBD with parameters $b = 45$, $k = 2$, $v = 10$, $r = 9$, $\lambda = 1$, and a special collecting network consisting of $vN_2 = N'N_2$ switches.† Hence the total number of switches required in the double level network is $2(45) + 10(4) = 130$. The double-level switch network based upon the structure of Fig. III-C-9 requires $3(45 + 4) - 2(3)(2) = 135$ switches.

Our primary purpose in presenting this example (which is somewhat pathological) is to indicate that many theoretical questions remain unanswered concerning commutation networks which are minimal in the context of the switch-network models. We have presented methods for synthesizing minimal single-level networks, both order-preserving and non-order-preserving, for all values of the parameters N_1, N_2, M.

---

* If no design with $b = N_1$ exists then a design with $b \approx N_1$ can be altered by adding or deleting a few rows, so as to form an incidence matrix containing N_1 rows.

† Clearly this network is order-preserving since the special collecting network can transfer any disordered pattern on the intermediate lines, onto the output lines so that the order at the input is preserved.

It was shown that significant economy is realized if double-level realizations are considered. Techniques were presented for synthesizing double-level switches which, although shown to be nonminimal, represent an adequate engineering solution. For the practical case of an order-preserving network with $N_1 \approx N_2$, the indicated network appears to be close to minimal. For future work in this area the following studies are recommended.

(1) Consider the derivation of lower bounds on the number of required switches, with an arbitrary number of intermediate levels, in order to indicate how close to minimal are the networks presented in this report.

(2) For the order-preserving case with multiple levels, consider additional examples where disordering of the signals on the intermediate levels is allowed.

(3) Consider other tools besides balanced incomplete block designs for the specification of distributing networks. Partially balanced incomplete block designs have been studied extensively and many more examples are tabulated of these than of BIBD's. In addition the structures of matrices used to specify error-correcting codes appear to be applicable here.

(4) Consider the design of partial commutation switches (defined in Sec. III-C-2-a).

d. Setup and Control Circuits

In the previous section we considered a model which visualized the commutation circuits as a network of single-pole single-throw switches. Our motivation for studying that model was based upon our belief that logic designers in implementing commutation circuits might prefer to employ realizations in which the commutation function is executed by combinational means. In this case the switch network is programmed by an external executive when reconfiguration is required.

The switch-network models which were presented should provide an adequate basis for design.

In addition it is possible to synthesize the commutation network by combinational techniques exclusively. The circuit, as relating to the single-level switch network, can consist of a net containing $N_1 + N_2$ inputs indicating the status of each of the modules of the input and output arrays, N_1 signal inputs, and N_2 signal outputs.

Indeed, the resultant net as synthesized from a truth table will be quite large, although some simplification is possible by the consideration of "don't cares." The circuit as relating to the double-level network will consist of two combinational parts--the first part containing the $N_1 + N_2$ status inputs and the N_1 signal inputs but only N' outputs, and the second part containing N' signal inputs, $N_1 + N_2$ status inputs, and N_2 outputs. We have not yet considered the estimation of the complexity of these realizations.

It is not immediately clear that the most economical implementation is afforded by incorporating exclusively combinational logic for the control. Moreover, some simplification could be realized by noting that all connections need not be established anew after each failure; rather, the circuit could "translate" to different connections with only slight modification in organization. Comprehensive study will be required to specify synthesis techniques which will minimize the overall cost of the combinational commutation switch, and indeed also to ascertain the specific advantages of the combinational approach as compared with, for example, the diffusion circuit discussed in Sec. III-C-2-b.

Of major importance in the consideration of the implementations are methods for ensuring that the resultant circuits are failure-tolerant. This question is considered briefly in Sec. III-C-2-e.

As in our other studies, our aim with respect to the design of commutation circuits is to present realizations which achieve reliability at low cost and which are compatible with the anticipated future technology. In accordance with this goal we chose to consider cellular

implementations of the commutation networks. Although the circuits based upon this approach will be costly compared with the combinational types, considering component count exclusively, they do offer the following advantages:

(1) The network is composed of a rectangular array of identical cells.

(2) The control operation is straightforward in that the paths between the input and output channels are formed automatically.

(3) The networks can be diagnosed for faults in a straightforward manner.

(4) It appears that low-cost redundancy is easily incorporated into the design, permitting the by-passing of faulty cells.

One type of cellular order-preserving commutation switch which has been conceived shown in Fig. III-C-13, is based upon the structure of the Holland Machine.[133] The purpose here is to first delineate distinct nonintersecting paths between each of the M operating input and output channels. (It is assumed that the paths are formed essentially anew upon the detection of each failed input or output module.) Once the appropriate paths have been defined, the network returns to the operating state in which data is transferred between input and output modules, essentially through combinational logic attendant to the cells.

The path-defining operation can be described as follows [Fig. III-C-13(a)]. The network consists in general of a rectangular array of Max (N_1, N_2) x M cells. The input and output modules which are operating are distinguished, and a path commencing at the first operating input seeks the first operating output by the following strategies, executed in preferred order by the cells.

(1) An attempt is first made to establish a path between the cell in question and the cell immediately above.

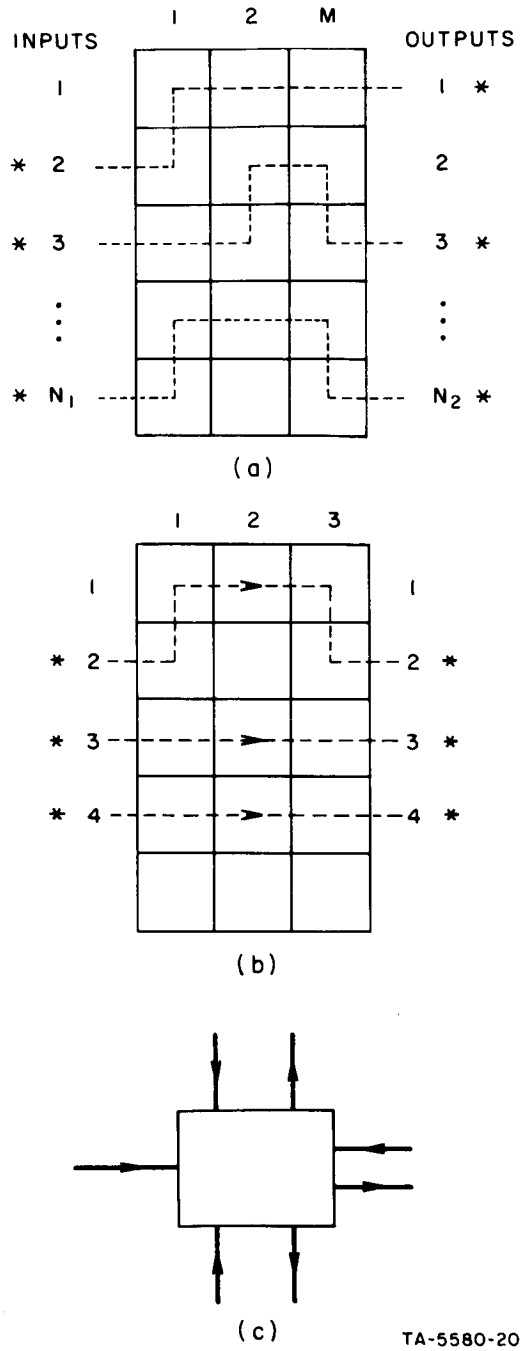


FIG. III-C-13 HOLLAND-TYPE CELLULAR COMMUTATION NETWORK

(2) If a path has already been defined including the cell above, or if the cell in question is on the upper boundary of the array, or if the cell "northeast" has been included in a path*, then an attempt is made to establish a path to the adjacent cell on the right.

(3) If connection with the adjacent right cell is not possible, then an attempt is made to establish a path to the cell immediately below.

The path is complete when it reaches the cell adjacent to an operating output module. The inputs and outputs required of such cells for the path formation are shown in Fig. III-C-13(c). Since each cell is to deliver a signal to one of three adjacent cells, two flip-flops are required for each cell.

Other cellular realizations have been conceived, relating to the minimal switch-network designs of the previous section. At present we have not formulated well-defined measures for the comparison of the various commutation-circuit realizations, but it is felt that a major factor will be the cost of ensuring failure-tolerant operation. In the following section we discuss techniques for synthesizing redundant commutation switches.

e. Failure-Tolerant Interconnection Networks

Here we are concerned with establishing techniques whereby the reliability of commutation circuits can be improved. Before proceding to synthesis studies it is appropriate to question the need for reliable commutation circuits. Most studies concerned with the estimation of the reliability of reconfigurable computers have either neglected the reliability of the interconnection networks, or have considered the commutation circuits as part of the hard core. In either case it has been assumed that the amount of equipment allocated for the commutation networks is a negligible portion of the overall computer. Our studies, as well as others,[302] have indicated that the reconfiguration method is

* This last restriction is necessary to avoid the creation of a dead-end path as would result if in defining the path between input 3 and output 3 [of Fig. III-C-13(b)] connection was permitted to the cell in row 2, column 2.

feasible only if the blocks which are replaced are sufficiently small. Admittedly this is imprecise, and indeed it is recommended that consideration be given to quantitative determinations* of the optimum-size replaceable network block, but at present it will be sufficient to note that the self-repair method should probably be applied at present to such regularly structured units as a block of memory, or a portion of a register or adder. In addition, such blocks as the programmable control units to be described in Sec. III-C-4 appear to be of approximately the proper complexity for replacement.

We have made a brief qualitative assessment of the properties of the fault-masking techniques discussed in Sec. II-A-1 as applied to both the diffusion register and the combinational and cellular commutation networks. We note that a commutation network is essentially a block with many outputs which requires, on the average, little equipment per output, although it requires a large overall amount of equipment. The voting type of redundancy is of little value here since the voters will form a significant portion of the network, and unlike the cases discussed in Sec. II-A-2-a, the voters cannot be replicated conveniently. Thus the reliability of a large set of nonredundant voters will essentially specify the reliability of the entire net. Particular timing problems are encountered in applying the voting technique to the diffusion register of Sec. III-C-2-b. in that each copy of a replicated register operates asynchronously. It is possible to consider additional redundant channels associated with the commutation circuits, and then employ the error-correction techniques discussed in Sec. II-C-2. However, it is felt that the additional equipment required for the decoding might result in too costly a system.

---

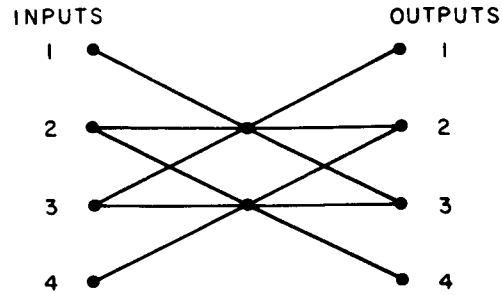
* On a qualitative basis we note that if complex blocks are replaced then a large number of spare blocks must be available, and if small blocks are replaced the attendant commutation circuits tend to be more complex than the replaceable blocks. Of course various policies are possible whereby large blocks are replaced, but repaired off-line.

Since the commutation networks can be modelled (as in Sec. III-C-2-c) as a net of switches, it appears feasible in the context of this model to specify additional switches beyond the minimal number required, so that modules are still connectable in the presence of switch failures. We have considered the problem of synthesizing such failure-tolerant switch networks considering two types of switch failures--permanent shorts and permanent opens--and the resultant increase in cost is not severe.

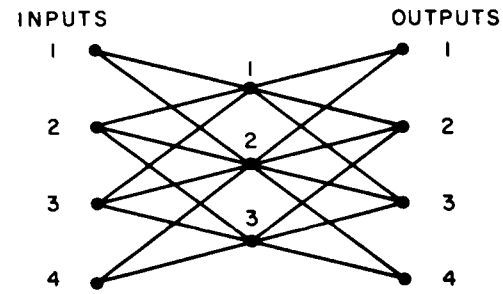
For example, consider the nonredundant double-level order-preserving switch network of Fig. III-C-14(a) with parameters $N_1 = 4$, $N_2 = 4$, $M = 2$. A network containing only 8 additional switches which is single switch-failure-tolerant is shown in Fig. III-C-14(b). It is clear that an additional channel is required on the intermediate level since the shorting of a switch connecting to a faulty input or output module will cause the pertinent intermediate line to be inoperative. On an intuitive basis we note that there exists a Moore-Shannon[214] type two-level single fault-masking hammock net, between each pair of input and output modules which are connectable.

A double switch-failure-tolerant network, requiring 32 redundant switches, is shown in Fig. III-C-14. The synthesis techniques illustrated here can be extended to specify switch networks with arbitrary failure tolerance, by noting that for each additional failure an additional level, and an additional channel for each intermediate level are required.

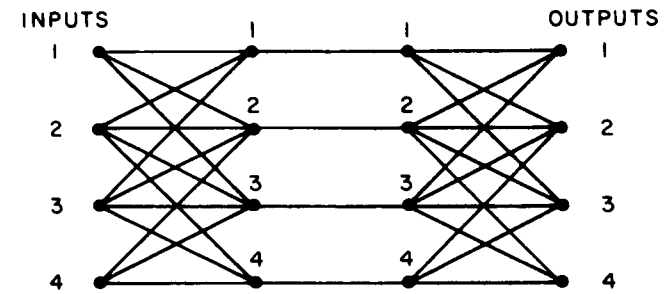
There are several unanswered practical questions concerning the applicability of the spare-bath technique. Means must be provided for the diagnosis of the commutation network in order to locate suspected faulty switches. It is not clear at present what is the optimum diagnosis policy, but it appears that the exhaustive checking of all connection combinations in a large redundant commutation network will result in lengthy tests. In this instance a good policy might involve waiting for a switch failure, and then using the information thus derived, concerning the location of the faulty output channel, to form a shorter test.



(a) DOUBLE-LEVEL, NONREDUNDANT



(b) SINGLE FAILURE TOLERANT



(c) DOUBLE FAILURE TOLERANT

TA-5580-9

FIG. III-C-14 NONREDUNDANT AND REDUNDANT COMMUTATION NETWORKS

Moreover in the implementation of a redundant switch network, possibly as a combinational net, care must be taken so that single component failures do not result in multiple equivalent switch failures. The problem here, however, is not as formidable as in the case of most multiple-output networks, in that there is little dependence between the outputs of the commutation network. An important question is, for example, "In a network which can tolerate at least single switch failures, what multiple failures can be tolerated?" The answer to this question would lead to designs whereby selected portions of the network could operate dependently.

We have not yet considered ways to incorporate redundancy into the cellular commutation networks considered previously. However, it appears that provision can be specified for bypassing faulty cells in the path-delineation phase. It is not presently clear how the bypassing instruction can be reliably implemented.

f. Conclusions and Problems for Future Study

In this Sec. III-C-2 we have presented the results of an initial study concerned with the synthesis of the commutation circuits required to effect the reconfiguration of modules upon the location of faults. This exploratory examination was primarily concerned with a simple model where each module of a given array (called the input array) is required to transfer data to a module of an output array, and where spare modules are present on both arrays.

One approach to the commutation-circuit design problem is based upon a diffusion register of speed-independent cells. This network, although it is attractive because of an economical design, exhibits several drawbacks. The operation is slow because the data is transferred through many sequential cells, and also it appears to be difficult to incorporate low-cost redundancy into the design.

A somewhat different approach to the problem was considered, based upon a multiple-input, multiple-output combinational net, which could be modelled as a network of single-pole single-throw switches, somewhat suggestive of a telephone exchange. The problem of deriving

networks of this kind which contain a minimal number of switches was considered; although many networks which appear to be close to minimal were synthesized, the global minimum has not been achieved. The solutions obtained appear to be adequate in the engineering sense, however. Techniques were discussed for the practical implementation of such networks and some attention was directed towards the realization of failure-tolerant implementations. In addition cellular implementations, which afford simple control operation, were briefly discussed.

Many questions have been uncovered in this brief examination. The problem still remains, for the simple model of the two reconfigurable arrays, of determining the globally minimal switch networks for both total and partial commutation networks. There remains also the problem of synthesizing improved economical and reliable implementations.

3. Programmable Processing Modules

In this section we describe the detailed design features of a modular arithmetic processor where the functions of computation, storage and primitive control are all combined in an iterated set of replaceable modules.

a. General Structure of a Modular Processor

The module to be described was designed to be suitable for the composition of a reconfigurable parallel processor for arithmetic and Boolean operations. A number of the design features of the module make it attractive for other system uses. These are described following the explanation of its functioning within a parallel processing unit.

Many of the logical processes in the central processor of a bit-parallel computer are naturally realized in iterative structures such as registers, counters, and accumulators. In collecting the logic elements that process the various words into modules, there is a choice to be made as to whether the collection is bit-oriented or word-oriented; i.e., whether a module will process corresponding bits of a number of words, or whether it will process all the bits of a single word.

Thus, if a processor serves w words of b bits each, if up to t words are active at a given time, and if each bit has a total of s input and output signals associated with it, the following dimensions result from the two approaches:

Bit-oriented:

b modules

st leads per module for signals

t $\log_2 w$ leads maximum* per module for selection of active elements.

Word Oriented:

w modules

sb leads per module for signals

1 lead per module for selection of the module.

For the word lengths and numbers of active words common in general-purpose computers, sb is substantially greater than $t(s + \log_2 w)$; hence the bit-oriented approach would result in modules with substantially fewer leads. As mentioned previously, this tends to increase the reliability of the module; it also reduces the number of switch points in a reconfiguration network.

These arguments are the motivation for the design example of a reconfiguration parallel processing unit, illustrated in Fig. III-C-15. The processor is composed of n identical functional modules, where n may be greater than the number of digits of the words to be processed. Adjacent modules exchange data bidirectionally, with outputs B_{OL} and C_{OL} taken to the left and B_{OR} taken to the right. The set of outputs $(B_{OL}^{(1)}, B_{OL}^{(2)}, \ldots, B_{OL}^{(n)})$, and the set of inputs $(B_E^{(1)}, B_E^{(2)}, \ldots, B_E^{(n)})$ are taken in parallel, and are joined to the external system by a commutation switch of the kind described in Sec. III-C-2. The set of modules also communicates with an external control as follows: (1) the B_{OL} outputs of the first and last modules provide overflow and underflow

---

* This result is seen as follows. In order to specify sets of w words each containing 0, 1, 2..., t words, then $\log_2 \left[\binom{w}{0} 1 \binom{w}{1} + \ldots + \binom{w}{t}\right] \leq t \log_2 w$ bits are required.

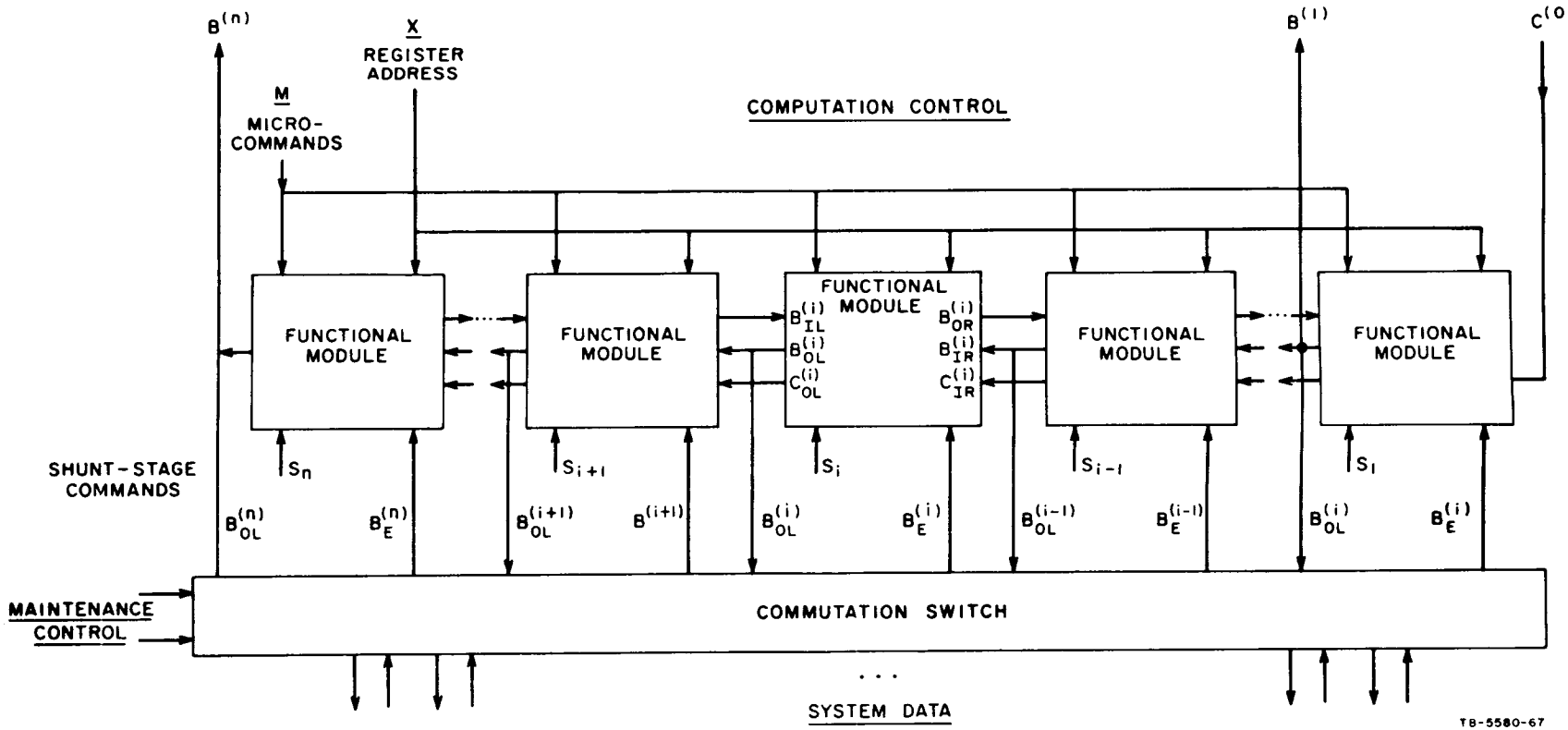


FIG. III-C-15 A RECONFIGURABLE PARALLEL PROCESSING UNIT

status information to the control in arithmetic operations, and (2) the control issues coded microcommands on a set of buses M, and coded register address information on a set of buses X. Both sets go to all modules. Within each module the X information specifies the selection of a bit of stored information, and the M information specifies the logical operation to be performed.

A key feature of the system is the method of reconfiguration. If a given stage fails, means are provided for shifting its function and the functions of all stages of higher order to the corresponding next stages of higher order. For the data that flows between adjacent stages, this shift is accomplished simply by logically short-circuiting the faulty stage, under the individual command of the signals S_1, S_2, ... S_n. These signals are assumed to be derived from an external maintenance control source. For parallel-access data, the commutation switch must be designed so as to displace all bits of order higher than a given index, for any index, preserving order in the displacement. The design of such switches is discussed in Sec. III-C-2. This mode of reconfiguration makes possible the use of all spare units, if needed, no matter what the configuration of faulty stages, without the necessity of providing that a given spare be capable of being switched into any faulty position.

In the next part of this section, the interior design of the module will be described; and in the following part, illustrative microprograms will be given for several familiar logical operations.

b. Module Description

The module will be explained with reference to Fig. III-C-16. The main data operation is accomplished by a full-adder network, labeled Σ, with inputs a, b, c and outputs s (sum) and c_0 (carry). The accumulator flip-flop A may record either s or c. The b data is obtained from one of a set of storage elements β_1, ... , β_k, belonging to system registers 1 to k respectively, or from one of three external inputs B_{IL}, B_{IR}, and B_{IE}, corresponding to the b outputs of the left and right cells and the external system input, respectively. The k internal sources of b data are selected by a decoder, with external inputs X and outputs X_1, X_2,

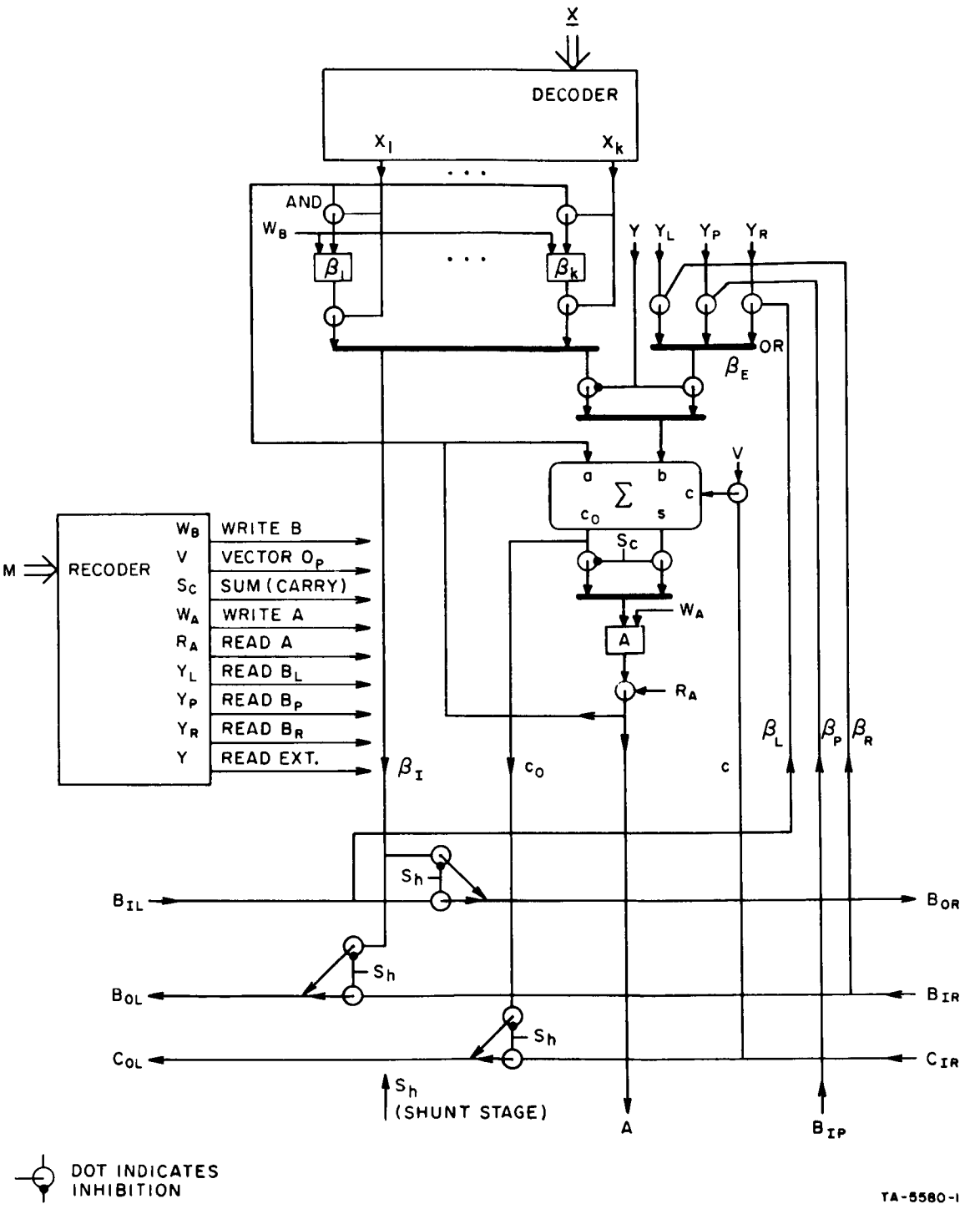


FIG. III-C-16 MODULE FOR A RECONFIGURABLE PROCESSOR

..., X_k.* The external inputs are selected under the commands Y_L, Y_P, Y_R. Shifting operations may be accomplished by a combination of commands; thus if X_j and Y_L are energized, all the bits of the j^{th} word will be presented to the b inputs of the adder networks of the modules one place to the right.

The value of the A element may be read out as an input to the adder, a, and as the data input to the set of β storage elements. The c input is equal to the external C_{IR} input in arithmetic operations, and zero in vector operations. The various commands W_B, W_A, R_A, V, S_c, Y, Y_L, Y_p and Y_R are derived from external command signals M. Finally if the stage is allowed to operate normally (indicated by the signal S_h FALSE) the outputs B_{OR},B_{OL}, and C_{OL} are equal to β, β, and c_O respectively, while if the state is to be shunted out these outputs are equal to B_{IL}, B_{IR}, and C_{IR} respectively. The details of the logic are given in Table III-C-4.

Using the data and control logic described, it is possible to construct a number of useful operations that can be programmed to accomplish a variety of useful functions. Table III-C-5 contains a list of such basic operations, conventionally called microoperations. The following notation is used:

(1) A refers to the set of A-elements.

(2) B_y (x) refers to a set of B-type data elements selected according to the address index x, where x = 1, 2, ..., k for the k internal sources, x = λ for the null input, and for shift-index y = I, L, P, R, for Internal, Left-, Parallel- and Right-external sources. If no y is specified, y = I is understood. Thus, for example B_L(3) refers to the presence at each adder b-input of the stored B bit of word 3 of the module to the left.

(3) B_O refers to the set of module outputs; in active modules $B_O = B_{OL} = B_{OR}$.

* Providing an identical decoder to all modules causes the total parts count to be high, but the separate decoding increases reliability by decreasing the number of module terminals, and by reducing the damage due to a fault in a decoder. Also, it may now be noted that another means is available for accommodating faults within single storage elements, aside from shunting out the entire stage; that is, to assign the function of the entire register to which the fault element belongs. This may be done by changing the address code externally, at the central control unit.

Table III-C-4

LOGIC EQUATIONS FOR A PROCESSING MODULE

Storage elements: $\beta = (\beta_1, \beta_2, \ldots, \beta_k)$; A

Data Inputs: B_{IL}, B_{IR}, B_{IP}, C_{IR}

Control Inputs: $X = (x_1, x_2, \ldots, x_s)$; $M = (m_1, m_2, \ldots, m_c)$

Maintenance Input (for Shunting Stage): s_h

Storage Selection Variables Decoded from X: $X_1, X_2, \ldots, X_k$

Control Variables Derived from M:

W_B (Write B)	Y Read External B
W_A (Write A)	Y_L Read B_L
R_A (Read A)	Y_P Read B_P
V (Vector Operation)	Y_R Read B_R
S_c (Sum-Carry Choice)	

Intermediate Variables:

Selected Internal β $\beta_I = X_1\beta_1 + X_2\beta_2 + \ldots + X_k\beta_k$

Selected External β $\beta_E = Y_L\beta_L + Y_P\beta_P + Y_R\beta_R$

Composed Inputs to Adder:

$$a = A\ R_A$$

$$b = Y'\beta_I + Y\ \beta_E$$

$$c = V\ C_{IR}$$

Adder Outputs: $s = \text{Parity}(a, b, c)$

$c_0 = \text{Majority}(a, b, c)$.

Data Inputs to Storage Elements

Element	Change Condition	New State
A	W_A	$s_c\ S + S'_c c$
β_i, $1 \leq i \leq k$	W_B	$a\ X_i$

Data Outputs:

$$B_{OR} = S'_h\ \beta_I + S_h\ B_{OL}$$

$$B_{OL} = S'_h\ \beta_I + S_h\ B_{OR}$$

$$C_{OL} = S'_h\ c_0 + S_h\ C_{IR}$$

$$A = a$$

Table III-C-5

BASIC MICROOPERATIONS FOR A MODULAR PROCESSING UNIT

Micro-operation Code	Symbolic Operation	Description
M_1	$\underline{A} \leftarrow \underline{0}$	Clear A
M_2	$\underline{A} \leftarrow \underline{B}_y\ (x)$	Load A from a specified B
M_3	$\underline{A} \leftarrow \underline{A} \cdot \underline{B}_y\ (x)$	Accumulate logical product of A and a specified B
M_4	$\underline{A} \leftarrow \underline{A} \Sigma \underline{B}_y\ (x) \Sigma C_0$	Accumulate sum of A and a specified B, with initial carry
M_5	$\underline{A} \leftarrow \underline{A} \oplus \underline{B}_y\ (x)$	Accumulate mod-2 sum of A and a specified B
M_6	$\underline{B}\ (x) \leftarrow \underline{0}$	Clear a specified internal B
M_7	$\underline{B}\ (x) \leftarrow \underline{A}$	Copy A into a specified internal B
M_8	$\underline{B}_0 = \underline{B}\ (x)$	Read a specified B at the B_0 output

Table III-C-6

MICROOPERATION CODES

	B Spec		B Control	A Control		Logic	
	Shift Y	Index X	Write B	Read A	Write A	Vector Operation	Sum/Carry 1/0
M_1	0	λ	0	0	1	1	1
M_2	y	x	0	0	1	1	1
M_3	y	x	0	1	1	0	0
M_4	y	x	0	1	1	1	1
M_5			0	1	1	0	1
M_6	y	x	1	0	0	-	-
M_7	0	x	1	1	0	-	-
M_8	0	x	1	0	0	-	-

Table III-C-6 gives the appropriate excitations of the internal control signals required to realize the given microoperations.

In the next part, microprograms are presented for several familiar processes.

c. Microprograms for Common Functions

The basic microoperations given for the processing module may be applied to a cascade of modules in obvious ways to realize the behavior of familiar computer functional units, such as a bidirectional shift register, a counter, and an adder. Subtraction may be accomplished by adding the modulo-2 sum of the subtrahend (obtained by applying microcode M_5) to the minuend, with an injected carry, C_0.

The structure is also obviously well suited to serve as a set of index registers, with built-in adder for index arithmetic.

A program for arithmetic multiplication is given in Table III-C-7. All the components of this process are held in the B registers of a single processor, including Multiplier, Multiplicand, Product, and Cycle Counter; hence these components must be processed serially. The program loop has ten significant time steps. In conventional practice, independent structures are used at least for the multiplier and the cycle counter, with the advantage of greater parallelism, and at the cost of many more data paths.

A program for decoding a binary vector is given in Table III-C-8. The object of the program is to yield a 1 at the output of the j^{th} stage of the processor, where j is the binary-number equivalent of the binary vector. The calculation employs a set of vector constants c(1), c(2), etc., stored in the B registers, as shown in the table. The program is the sequential equivalent of 2^m-1 combinational functions; for example, the function at stage i is

$$p_i = c_i(1)^{(x_1)} \cdot c_i(2)^{(x_2)} \cdot \ldots \cdot c_i(m)^{(x_m)}, \quad 1 \le i \le 2^m-1,$$

where a given c is taken directly if its exponent is true, or complemented if its exponent is false. For the vectors indicated, this process results in $p_i = 1$ when $(x_1, x_2, \ldots, x_m)$ is the binary equivalent of i, as desired.

Table III-C-7

MICROPPROGRAM FOR MULTIPLICATION

Register Use

B (1) Multiplier*

B (2) Multiplicand*

B (3) Product

B (4) Cycle Counter

LSD = least significant digit

MSD = most significant digit

Program

Step	Condition†	Operation	Description
1		$\underline{A} \leftarrow \underline{0}$	Clear Accumulator
2		$\underline{A} \leftarrow \underline{A} \Sigma \underline{B}_I (3) \Sigma C_0$	Start Ring Counter
3		$\underline{B} (4) \leftarrow \underline{A}$	Store Count
4		$\underline{A} \leftarrow \underline{B}_R (4)$	Load and shift ring counter left
5		$\underline{B} (4) \leftarrow \underline{A}$	Store count
6		$\underline{B}_0 = \underline{B}_I (4)$	Read Count
7	$B_n = 1$	Exit	Count ended, multiplication ended
8		$\underline{B}_0 = \underline{B}_L (3)$	Read product, looking right, test
9_0	$B_0 = 0$	$\underline{A} \leftarrow \underline{B}_L (1), B_n \leftarrow 0$	Load multiplier, shifting right, set MSD = 0
9_1	$B_0 = 1$	$\underline{A} \leftarrow \underline{B}_L (1), B_n \leftarrow 1$	Load multiplier, shifting right, set MSD = 1
10		$\underline{B}_I (1) \leftarrow \underline{A}$	Store multiplier
11		$\underline{A} \leftarrow \underline{B}_L (3)$	Load product, shifting right
12		$\underline{B}_0 = \underline{B}_I (1)$	Read multiplier, test LSD
13_1	$B_n = 1$	$\underline{A} \leftarrow \underline{A} \Sigma \underline{B}_I (2)$	Accumulate multiplicand into product
14		$\underline{B}_I (3) \leftarrow \underline{A}$	Store product
		Return to Step 4	

* Assume factors in place at start

† If condition is not satisfied, skip step.

Table III-C-8

MICROPROGRAM FOR DECODING A BINARY VECTOR

Register Use

Register	Function	Initial Value
B (1)	constant c(1)	(1 0 1 0 1 0 1 0)
B (2)	" c(2)	(1 1 0 0 1 1 0 0)
B (3)	" c(3)	(1 1 1 1 0 0 0 0)
B (4)	" c(4)	(1 1 1 1 1 1 1 1)
B (5)	Input vector	(0 0 0 0 0 $x_3x_2x_1$)
B (6)	Temporary product	(0 0 0 0 0 0 0 0)

Program

Step	Condition	Operation	Description
1		$\underline{A} \leftarrow \underline{B}_I$ (4)	Load accumulator to $\underline{1}$
2		$\underline{B}$ (6) $\leftarrow \underline{A}$	Store temporary product
3		$\underline{A} \leftarrow \underline{B}_L$ (5)	Load accumulator to input vector shifting left
4		$\underline{B}$ (5) $\leftarrow \underline{A}$	Store input vector
5		$\underline{A} \leftarrow \underline{B}_I$ (1)	Load first constant
6		$\underline{B}_0 = \underline{B}_I$ (5)	Read next input vector bit
7	$B_0 = 0$	$\underline{A} \leftarrow \underline{A} \oplus \underline{B}_I$ (4)	Complement first constant
8		$\underline{A} \leftarrow \underline{A} \cdot \underline{B}_I$ (6)	Accumulate first logical product
9		$\underline{B}$ (6) $\leftarrow \underline{A}$	Store temporary product

Repeat steps 3 - 10 twice more, replacing B (1) first by B (2), then by B (3)

For example, for m = 3, i = 6, the binary vector for i = 6 is 011. Then $p_6 = (0)^0 \cdot (1)^1 \cdot (1)^1 = 1 \cdot 1 \cdot 1 = 1$.

An alternative way of realizing this function is simply to subtract the number 1 from the binary vector, treated as a number, until the value zero is reached, and at each step shift a fiducial 1 digit one stage to the left. This process would take an average of 2^{m-1} major steps, compared with the m major steps of the given process.

d. Other Uses of the Module

The module described has a number of features that make it useful for realizing general logical functions. For example, suppose it is desired to realize an arbitrary switching function of d variables where $d \leq 2^k$. If the truth table (with column elements $t_1, t_2, \ldots t_k$) for the function, is stored in the like-index β elements of the module and the variables are applied to the X decoder inputs, the resulting β_I line realizes the function

$$\beta_I = t_1 X_1 + t_2 X_2 + \ldots + t_k X_k = \beta (x_1, x_2, \ldots, x_d)$$

where X_i is the 1-indexed i^{th} min-term of the set of d variables; β_i may thus be set to be any switching function of the d variables by proper choice of the t's.

As a further enhancement, the s output of the adder may be set to provide the function

$$s = \beta (x_1, x_2, \ldots, x_d) \oplus V \cdot c,$$

where V and c are single, independent variables. This form makes the module well suited to the realization of so-called "ring-sum" canonical compositions of arbitrary switching functions.

e. Problems for Further Study

The following problems for further study are evident:

(a) Study augmentations of the given design that enlarge the range of applications.

(b) Develop efficient means for testing the module.

(c) Study means for ensuring that the stage-shunting action may be accomplished reliably, e.g. by fault masking.

(d) Consider designs which incorporate register bits of more than one index.

(e) Consider module designs that incorporate more control functions.

4. Programmable Control Units

a. Uses of Programmability in a Control Unit

With few exceptions the control units of modern computers have a fixed logical structure. In self-repairing reconfigurable computers, several reasons may be distinguished for making the control unit a variable structure, subject to external programming. The major uses for such variability are the following:

(1) To allow for failures in functional blocks by changing the hardware address of the block employed for a given function

(2) to allow for modification in the microsequence for a given order, if hardware capability for that order is lost

(3) in the control unit of a given processor in a multiprocessor, to allow for specialized operation of the processor by assumption of a special order code

(4) to accommodate faults within the control unit itself.

In the first three uses, the variability in operation could be achieved at the program level; but providing it in the control unit permits greater compactness in the order code, or higher normal speed of operation, or both. The benefits of the fourth use depend upon how

the reliability of the unit is affected by the added equipment needed for the programmability.

In the next part, several methods for achieving such programmability will be discussed.

b. Approaches to the Structuring of Modular Programmable Control Units

Fixed-function control units are usually quite complex in structure. The criteria of feasibility and modularity suggest the use of a high degree of regularity in structure. In this part, three approaches will be considered that emphasize such regularity. The modularization of control units is currently a subject of widespread investigation, and the schemes described should be taken only as examples of possible approaches.

1) Control Based Upon a Microprogram Memory Store

The well-known microprogram structure[330] for control is well suited for realizing a programmable control unit. It employs an addressable memory, the contents of which are called a microprogram,* and the state of the central unit is defined by the memory word currently selected. Each such word carries information specifying (1) an output excitation, and (2) the address of the next word (state) or words which may be its successor in a program sequence. The output excitation includes both the specification of functionsl units--e.g., registers--and of function--e.g., shift operation. In operation, the code for a given machine order is used to address and retrieve a stored microorder; thereafter the sequence of accesses is self-sustaining. The result is the production of an arbitrary sequence of control signals that implement the machine order.

---

* Early advocates of this scheme proposed that the stored microprogram be alterable, but almost all realizations have employed permanent-storage memories. The present discussion, of course, assumes variable storage. An interesting scheme combining fixed and variable stores has been proposed by Grasselli.[112]

Several design approaches may be followed to provide for branching within the sequence. One known scheme is to add special logic to the access switch, so that when the address specified by a branching instruction is applied to the access switch, the memory line selected depends upon the state of some external logical variables.

The following scheme (which is believed to be original) does not require any augmentation of the access switch.

Let the address be the base-2 number specified by the m-bit vector $(X, Y) = (X_0, X_1 \cdots X_{a-1}, Y_0, \cdots Y_{b-1})$, with x digits having lower significance; and in each word, in addition to the X and Y segments, let there be a bit B, which, if true, signifies that the word is a branch point; then,

(1) in a nonbranch word, the X digits are interpreted as the least significant digits of the address of the successor word, while

(2) in a branch word, the X field is interpreted as a mask upon the external control variables, such that if $x_i = 0$, the i^{th} address digit is 0, while if $x_i = 1$, the i^{th} address digit is the i^{th} control variable, say z_i.

For example,

let (X, Y) = (0101, 10110),
for B = 0, next address = (0101, 10110), while
for B = 1, next address = ($0z_2$ $0z_4$, 10110).

By this scheme, up to 2^a-way branching is possible at a given step; however, if two branches refer to the same successor, at least one will have to pass by way of an intermediate nonbranching step, which will have a full range of addressing. A limitation of the scheme is that as the number a of external variables increases, the number 2^{m-a} of branch points decreases. One way of extending the number of usable external variables would be to decode the X variables so as to select one of 2^a external variables as a single binary condition. This method,

with some refinements, is described by Kampe.[144] A general structure covering these variations is illustrated in Figure III-C-17. Data paths for modifying storage are not shown.

It is clear that various schemes may be devised that do not require use of a special access switch; this means that it is possible to use main memory as a backup for a microprogram store, in the event that part of the microprogram store is lost because of a permanent failure.

Another attractive way of accommodating faults in the store is to use an associative memory. Such a memory provides for relocation of words within memory without change of address code; but of course the given fault must be localized to a few words in its effect in order for relocation to be useful.

Finally, it may be observed that all the methods of error control for memories, such as error-correcting codes, may be employed to increase the reliability of the control unit.

One of the main limitations of this approach is that large numbers of branch conditions, and complex branch conditions, are not handled with great flexibility. Further development of the approach should seek to increase this flexibility.

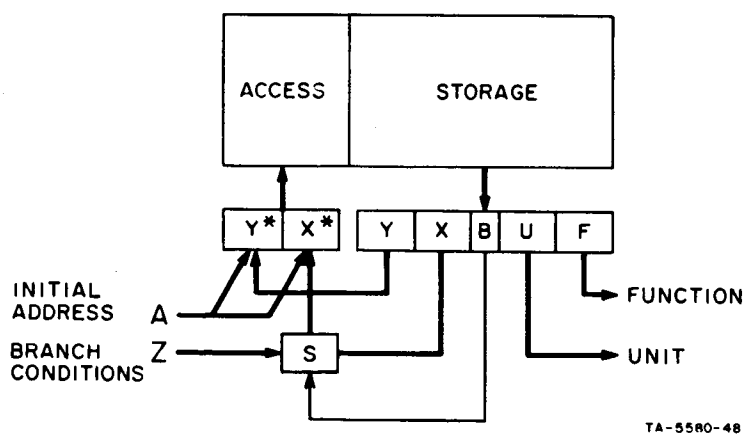


FIG. III-C-17 A MICROPROGRAM CONTROL UNIT

2) Control Using a Programmable Cellular Network

The use of a network of logic elements clearly provides more complexity of logical operation than does a similar number of elements in a memory structure. Recent investigations (by S. Wahlstrom and others) at Stanford Research Institute have indicated the practical feasibility of building logic networks with substantial variability in function. These investigations are part of a general study of cellular networks, i.e., networks of logic modules, having a uniform, primarily neighborly interconnection structure[210,211,212,285] the approach considered provides for storage of information within the cells of such a logic array. This information would specify both the logical function performed by the cells, and the choice of particular connections of a cell to its neighbors and to signal buses, from among the available connections.

An example of such an array is illustrated in Fig. III-C-18. The light lines indicate the signal paths available at each cell, the heavy lines indicate the particular paths that are active in the illustrated design, and the dashed lines indicate the paths used to program the array; the x variables are the inputs, the F variables are the outputs, and the letters a, b, ..., i represent the logic functions realized by the

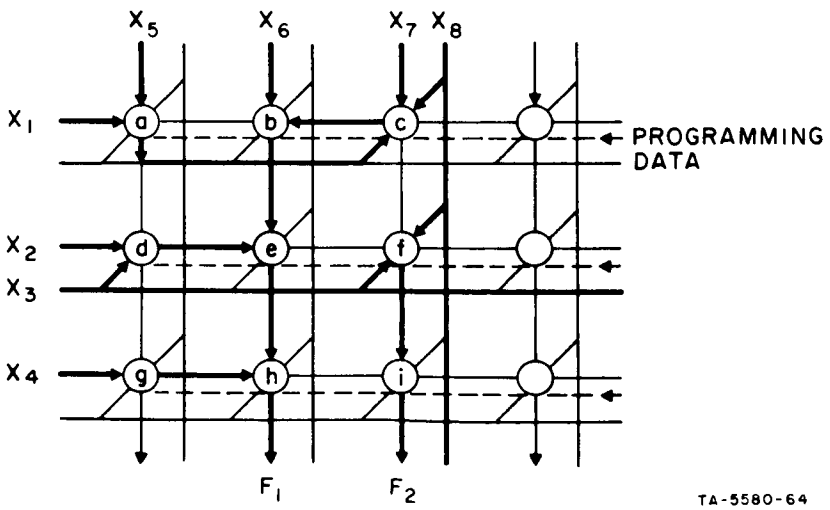


FIG. III-C-18 A PROGRAMMABLE CELLULAR LOGIC NETWORK

individual cells. In the arrays studied thus far, typical functions are: a set of combinational functions of two or three variables; a single universal function (e.g. NOR) of six or seven variables; or a single flip-flop.

A more detailed view of a cell is given in Fig. III-C-19. The information stored in box f controls the functions performed by a multifunctional logic network N, and the information stored in boxes t, u, v and w controls the connections of the cell terminals to the inputs and outputs of the network N. Signal paths for the introduction of program data are indicated by dashed lines. Clearly, program data can be designed to compensate for faulty cells.

The number of storage elements needed for control of a cell is substantial, but with the advent of microelectronic arrays of high component density, the cost of programmability may not be prohibitive. The most natural form of storage would be flip-flops, which would allow the use of the same technology as the controlled circuits. This has the possible disadvantage of volatility of information with loss of power, but it would seem to be quite a straightforward matter to record the program state of a network in the main nonvolatile system memory.

This example is meant only to illustrate the basic ideas, since there are many possible variations in cell functions, in interconnection structures, and in the means of introducing program data. The general design problem, of course, is to develop arrays that have

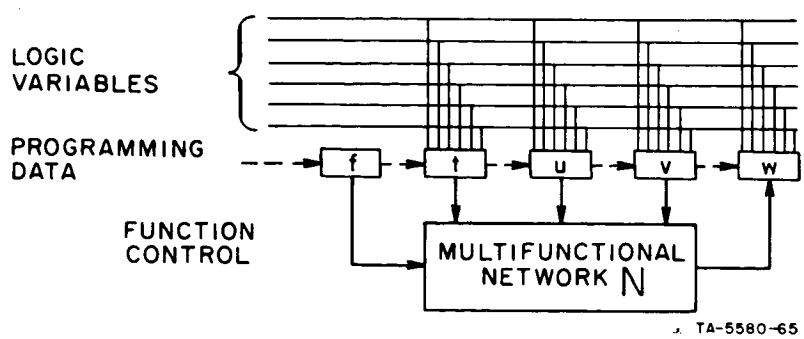


FIG. III-C-19 DETAILS OF A PROGRAMMABLE CELL

a good combination of flexibility and economy. Additional design problems of special relevance to reliability are

(1) The design of arrays that are easy to test and diagnose

(2) The design of arrays that provide for avoidance of faulty cells and connections with minimum sacrifice of nonfaulty elements.

It is to be expected that cellular networks are, in general, easier to diagnose and reconfigure than noncellular networks.

3) Control Based upon a Network of "Universal" Logic Modules

Advances in microelectronics have resulted in an increase in the potential complexity of prefabricated networks. Since the number of possible switching functions grows exponentially with the number of variables, a serious problem of standardization of fabrication arises because of the large number of different networks that are needed to realize arbitrary functions.

It is well known that a given m-input combinational network may be used to realize a number of functions of the m variables, by permuting and complementing the input variables at the terminals. If it is permitted to tie terminals together or to apply constants arbitrarily, the number of independent input variables, say n, is of course less than m; but the fraction of the total number of possible switching functions of the n variables that may be so realized is potentially greater than the number that may be realized by permutation and complementation at an n-input network. It is not at all obvious which networks offer the greatest flexibility for such realizations. In particular, it would be very useful to have a network which provides all such functions for some appreciable number of variables. Kautz has suggested the problem of finding universal logic modules (ULM's). These are defined as follows:

> Consider a (combinational) logic net with m input terminals and two (complementary) output terminals. The m input terminals may be connected freely to any of 2 (n + 1) source wires, carrying

the variables, $x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n$ and the constant signals zero (0) and one (1), respectively. If under arbitrary connections of this sort the output terminals produce all n-variable Boolean functions (and their complements), f, $\bar{f}$, then we refer to the net as an (m,n) ULM. The principal questions associated with such nets are:

(1) Find ULM's with m = minimum for n up to say, 4 or 5.

(2) Determine the dependence on n of the minimum m = M(n).

(3) Alternatively, find good estimates (upper and lower bounds) on M(n).

This problem is currently under investigation by Kautz, Elspas, and Stone at Stanford Research Institute under Institute sponsorship. The function $F_3(a, b, c) = a \oplus bc$ is readily seen to be universal for two variables. Investigations have disclosed that all functions of three variables may be obtained from the function of five variables

$$F_5(a, b, c, d, e) = e' F_4(a, b, c, d,) + e(abc),$$

where F_4 is the Harvard function of index 87. F_4 may be represented by the set of min-term indexes (0, 1, 2, 5, 11, 12), or by the expressions

$$F_4 = b'a'(d' + c') + dca' = a(dcb' + d'c'b).$$

Minimal functions for more than three independent variables are not known, but upper and lower bounds have been investigated. The following limits are known for M(n), the minimum m for an (m,n) ULM for the next few values of n:

$$6 \leq M(4) \leq 8,$$
$$10 \leq M(5) \leq 18, \text{ and}$$
$$17 \leq M(6) \leq 37.$$

If the value of M(n) proved to be excessive for the n of interest in a particular design, it would also be useful to have a small set of modules that together would provide coverage of all switching functions of n variables, or even of a large subset of all the functions.

For the realization of a reconfigurable control system for a computer, it would be necessary to provide controllable means for permuting, complementing, and busing the inputs to a ULM. A programmable universal logic module (PULM) could then be composed, as illustrated in Fig. III-C-20 of an (m, n) U.L.M. (labeled U), fed by an m-input, n-output connection network (C), which can be programmed by an external input. The connection network itself must be rather complex, and it is not inconceivable that using a larger value of m than M(n) may result in the more economical overall design.

For the large number of variables found in a modern control unit, it is clear that a number of PULM's would have to be combined in some larger network. For maximum reconfigurability, the interconnections within that network should have some degree of programmability, both for modification of the control functions and for the replacement of faulty PULM's. Such a control system is sketched in Fig. III-C-21.

Several completely open questions pertaining to the design of such a system, in addition to the problem of ULM minimization already mentioned, are as follows.

(1) The design of the internal connection network C

(2) The design of the external connection network N to achieve a useful degree of flexibility

(3) Suitable means for incorporating memory within the overall structure.

The present discussion has been concerned with combinational logic networks. The notion of universality may also be applied to sequential networks, and, practically, it would also be desirable to have modules with flexible, even if less than universal, state behavior. This topic is currently being studied by a number of investigators.[229,295] The design of simple multipurpose combinational-sequential modules has also been discussed by Ledley.[177]

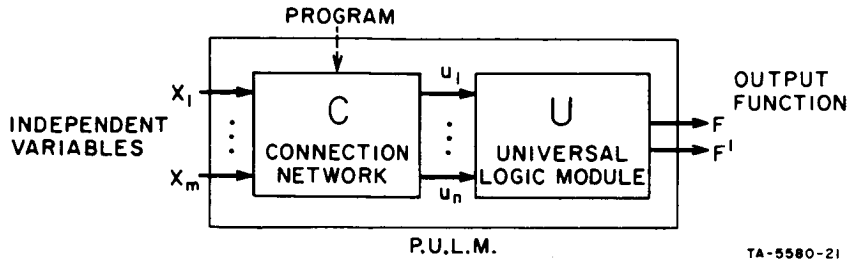


FIG. III-C-20 LOGIC MODULE BASED ON A PROGRAMMABLE UNIVERSAL LOGIC MODULE

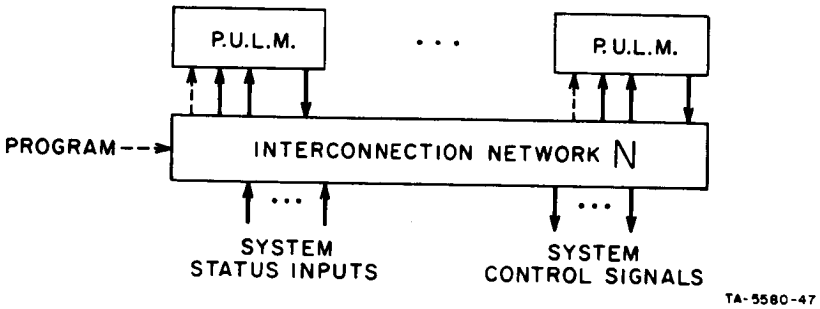


FIG. III-C-21 RECONFIGURABLE CONTROL UNIT BASED UPON PROGRAMMABLE UNIVERSAL LOGIC MODULES

IV CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE STUDY

A. Conclusions

In discussing the conclusions of the report it is convenient to assess our results on the basis of the goals of the study set forth in Sec. I-A, which can be summarized as follows:

(1) To examine the known techniques for reliability improvement to determine their adequacy for the achievement of long mission life.

(2) To conceive and evaluate new schemes of system design and operation that offer promise of advancing the state of the art.

(3) To recommend future directions of research which will aid in the improvement of present techniques and also in the development and realization of new schemes.

Each of these three areas received significant attention during the course of the study as reviewed below:

(1) In the examination of known techniques, the approach taken was to distinguish those sections of a hypothetical spaceborne computer to which distinct problems of reliability apply, and to assess the utility of existing reliability enhancement techniques in the light of spaceborne requirements and of advances in device technology. Most of the logical design concepts previously examined have applied to static error control--in particular, to fault-masking techniques. Several significant analytical problems have not been satisfactorily solved; these problems involve the optimum application of the techniques, and the estimation of the reliability improvement realized by the techniques. However, the results which have been obtained indicate that the exclusive application of present static error-control techniques cannot lead to designs which achieve the required mission life under the severe constraints of the spaceborne environment. We conclude, however, that fault masking techniques are very useful for the protection of limited crucial functions, and that existing known error-detection schemes are useful for diagnosis, e.g., as illustrated by the use of error-checking codes for arithmetic processors or for sequential circuits.

(2) In the study, a number of error-control processes were distinguished that, if implemented reliably, are capable of realizing substantially higher reliability than can be achieved by fault masking alone. These schemes relate to dynamic error control, in which a computer is subject to reconfiguration in structure and program. This mode has been discussed in recent literature, but the study has found a substantial lack of design knowledge appropriate to the practical realization of a computer exhibiting a capability of automatic diagnosis and reconfiguration. Indeed of primary concern here is a new viewpoint on the overall design of a computer system, including the design of its structure, and the coordination of the various maintenance and computational processes.

In summary, it is suggested that in order to achieve the highest levels of reliable performance, an advanced spaceborne computer will need to have the following structural features to a high degree: parallelism of logical operation, modularity and programmability of functional modules, regularity and programmability of interconnection, and autonomous capability for fault diagnosis and the control of reconfiguration. A number of error-control techniques, both fixed-structure and variable-structure, will be needed to enhance the reliability of basic functional units. These needs are summarized in the next section.

In addition to the general conclusions stated above, many conclusions have been derived concerning particular aspects of the analysis and design of various reliability schemes. The reader is referred to the individual sections for detailed discussions of these conclusions.

B. Summary of Needs for Technique Development

We have distinguished the following major needs for further development of reliability techniques:

(1) Fault-masking techniques represent the area of technical interest that has received the most attention previously, ranging from the protection of simple contact networks to the construction of very complex adaptive networks that can tolerate a wide variety of internal failures. However, it still remains difficult to actually calculate the probability of failure of any but the simplest network structures. Thus new techniques are needed to facilitate the analysis of complex fault-masked networks. An example of the more general analytic techniques that are needed is the consideration in Sec. II-A-2

of comparative advantages between tree-like compositions of multiple-output switching functions and compositions that minimize the number of outputs that are affected by a given element failure. There is a great need for further investigations of this type for different network structures and for different probabilities of failure amongst the elements. Most failure analyses assume independence among different faults. This assumption is made in order to yield an analytically tractable model, but it is usually unrealistic from a practical point of view.

(2) An immediate consequence of this last point concerning the independence of failures is the need to understand the design of systems wherein it is true that faults are correlated only over a relatively small and easily definable range of elements. For example, in a modular system, it is desirable to constrain fault propagation to within the module that first suffers a fault. If this can be done, then the assumption of fault independence between modules is tenable. One implication of this desire to minimize fault propagation between modules is that such systems will probably minimize communication between modules; i.e., the modules will be sufficiently complex that much useful computation can take place entirely within each one, with communication between them limited as much as possible to summary-type information that is highly protected.

(3) This requirement on intermodule independence must also involve the environmental facilities such as power supplies, radiation-protective devices, and the like. Thus a power-supply failure that results from a fault in a given module cannot be allowed to disrupt all the other modules as well. Much work is thus needed in extending redundancy techniques to the protection of these peripheral factors.

(4) Another consequence of the desire for modularity and reconfigurability is the need for the development of the family (or families) of modules themselves that simultaneously meet the various system requirements imposed upon them, including:

   (a) Compatibility with the dimensions and constraints in logical and topological structure characteristics of modern semiconductor technology, especially that of large-scale integrated networks.

   (b) Sufficient complexity to allow a considerable amount of calculation to take place entirely within the confines of a given module.

   (c) Sufficient flexibility for the same module to be usable for several different computational tasks depending on its particular assignment or reassignment within the system, and for the

family of modules to be complete--i.e., so that all tasks can be accomplished by the set.

(d) Sufficient accessibility for the modules to be easily diagnosed on the advent of trouble, and easily switched from one point in the system to another.

(e) Suitable scaling of complexity so that each replaceable module is a small enough fraction of the overall system for the redundancy ratio required to be minimized.

As seen in Sec. III-C-3, for the case of regularly structured functions (e.g., arithmetic or memory) such modularity is not hard to achieve. For more complex, irregular functions (e.g., control) several new approaches may be seen, but more work is needed to determine the best approach and to develop practical design.

(5) The problem of diagnosing modules only emphasizes the more general fault-diagnosis situation. The formal fault-diagnosis requirements are understood (and have been reported in this document), but we are still a long way from understanding approaches that are practical, particularly for systems as large as the modules of a reconfigurable system will probably be. As well as more efficient techniques for large combinational networks, we need more work in serial testing and in the design of efficient diagnosers themselves. Finally, the problem of the design of networks that are intended to be easily diagnosable has only recently been introduced, either in this report or in the general literature. Much further work in the utilization of auxiliary terminals for monitoring purposes as well as for test inputs is needed. Also the diagnosis of sequential machines, in general, remains almost completely an "art form" for any but the simplest machines.

(6) A great deal of work is needed in the programming, whether by software or hardware means, of systems that can autonomously (and efficiently) change their program mix in response to external problem demands, as well as in response to gradual failures within the modules. Such approaches should include the accommodation of problems of highest priority first--resulting in some tasks simply not being handled--as well as arrangements that involve a selective degradation of the problems that are handled--whether involving a decrease in accuracy, or an increase in solution time.

(7) The role of the new generation of integrated circuits must be more fully evaluated, both in terms of the realization of specific redundancy techniques that are appropriate for them, and in the opening up of new possibilities they imply because of the changing cost factors involved--e.g., the diminution of the total number of components as a prime contributor to expense. Again, with regard to modularity, it is important that these new components be designed hand-in-hand with the design of modules that meet the previously summarized requirements.

(8) The need for better means of calculating the reliability of static redundant systems has been mentioned, and the need also exists, and is probably compounded in the case of dynamic systems. Work is thus required on analyzing the reliability of systems under different rules of reconfiguration, and under different reliability assumptions concerning the switching system itself by means of which the dynamic reconfiguration is actually determined and carried out.

(9) The design of reliable and efficient interconnection switching systems for the reconfigurable spaceborne computer remains an unsolved problem area. Some specific designs were discussed in Sec. III-C, but much work remains in the achievement of the goal of flexibility of interconnection in a design which is itself failure-tolerant, and also in the design of control modules by which the decisions are made.

(10) All the various techniques that have been mentioned for module design and error-control procedures, both in equipment and in programming realizations, must be coordinated in practical system designs. A number of approaches have been suggested in the literature which differ in the extent of the control of diagnosis and reconfiguration that is invested in special equipment and in the balance of external and self diagnosis for subsystems. New approaches need to be considered and evaluated.

(11) A number of special problems are found in the design of memory systems. Several well-established schemes exist for data-channel protection, and a number of potentially useful schemes have been suggested for access-switch protection, but because of the close interaction of physical and logical design in memory systems, there is a need to test various schemes by carrying out complete designs.

(12) A number of possible applications have been noted for the use of magnetic-logic networks, in which the high reliability and the lack of volatility of information with loss of power of this technique may be exploited without intolerable reduction in system speed. There is a need for detailed analytical and experimental work to verify the validity of this preliminary view.

(13) Finally, the spaceborne system will in general be serviced by some sort of radio link with either the ground station, or with a "mother ship" control station. However, the role that such a link can play is widely variable, depending upon the distances involved, the time available, and the specific problem mix as a function of time into the mission. Much further analytical work needs to be done in determining first the range of mission characteristics, and then the relative roles to be played by the ground and spaceborne stations with regard to diagnosis, backup computation, control, reconfiguration specification, idle-time discourse, and storage of programs.

C. Summary of Suggested Problems for Future Research

Detailed suggestions for future research are presented in the various sections of this report. In this section we give brief summaries of these suggestions, listed by sections and appendices.

(1) Consider variations on coding and adaptive logic schemes to include redundant outputs and integrated restoration (Sec. II-A-1).

(2) Develop improved computer-aided techniques for analysis of complex restored nets and methods for globally optimizing placement of restorers; extension of model to larger classes of fault types (Sec. II-A-2-a).

(3) Develop more economical hybrid fault-masking switchover realizations, and provide for noise insensitivity; extend to to multiple-output networks; and incorporate fault masking in the switching networks (Sec. II-A-2-b).

(4) Develop efficient realizations for high-order threshold-function networks using NOR elements (Sec. II-A-2-c).

(5) Develop techniques for applying the parity-check and state-weight types of redundant-state encoding for error detection in a range of useful sequential networks (Sec. II-A-3).

(6) Develop codes and encoders that allow efficient, fault-masked instrumentation of transmission-type error-control codes (Sec. II-B-1).

(7) Develop more easily instrumented arithmetic codes for error detection and location, investigate possible improvements in residue coding, and compare alternative arithmetic checking schemes by detailed designs (Sec. II-B-2).

(8) Develop a framework for the design of maintenance programs that are well coordinated with hardware-maintenance processes (Sec. III-A-2).

(9) Develop and evaluate new schemes of system organization for maintenance and general computations, especially those suited to polymorphic (multi-processor) structures; develop techniques for coordinating the flow of various data and control information; specify subsystems so as to achieve high modularity (Sec. III-A-3).

(10) Develop improved techniques for fault diagnosis of large multiple-output combinational networks, and of important types of sequential networks; determine good means for utilizing test points; develop techniques for including ease of diagnosis in the original design of a network (Sec. III-B).

(11) Develop efficient means for control of commutation networks and for avoidance or masking of faults within the network; develop and evaluate practical path-seeking cellular interconnection arrays; extend present investigations to multiposition switching (Sec. III-C-2).

(12) Develop and evaluate more powerful logic modules; incorporate aids for fault diagnosis; incorporate fault masking for crucial functions (Sec. III-C-3).

(13) Develop and evaluate new schemes for realizing programmable control units, especially to incorporate complex functions of a large number of variables; develop schemes for microprogram control to incorporate memory backup and branching (Sec. III-C-4).

(14) Develop and evaluate schemes for logical error control of memory access-switch failures; investigate error-control needs of special types of memory systems (e.g., associative, fixed); study the interaction of logical error-control techniques and physical design techniques (Appendix A).

(15) Develop practical designs for distributed power-supply system; investigate feasibility of magnetic switching (Appendix B).

(16) Carry out detailed analytical and experimental evaluation of proposed all-magnetic logic network schemes (Appendix C).

Appendix A

ERROR-CONTROL TECHNIQUES FOR MEMORY SYSTEMS

Appendix A

ERROR-CONTROL TECHNIQUES FOR MEMORY SYSTEMS

1. Introduction

The primary effort of this task is to assess the manner of adding redundancy to the main memory subsystem of a spaceborne digital computer and the potential gain in doing so, and to distinguish those areas where more work is needed. Special memory types, such as permanent memories and associative memories are not included in this review. The techniques described are generally applicable to these memory systems, but special techniques may be advantageous for those types.

Part 2 of this Appendix presents the model and the assumptions made in the several analyses. Part 3 examines a redundancy scheme based upon replication of whole memory modules. Part 4 examines several schemes for error control applied to the bit-channel, word-select, and supply and control sections of a memory module. Part 5 describes the logical design of a parallel encoder-decoder and Part 6 presents the conclusions and recommendations of this study.

This study has emphasized the present state of the art of memory-error control; thus the quantitative estimates for the benefits and costs of the particular schemes described are based upon the use of off-the-shelf components. The main effect of expected future reductions in the size of logic components will be to increase the feasibility of schemes involving complex operations on data and address information.

2. General Discussion of the Problem

The primary function of the main memory subsystem is to accept data, address and control information, store that data in a specified location (word register) for an indefinite time, and return it error free upon demand to the parent system. This basically simple function establishes

a requirement for four separate functional units within the system. The data section receives, stores and delivers information, one word at a time. The access section controls the selection of the word location being processed. The cycle control section encompasses the sequential control of the signal sources that accomplish the reading and writing processes. The support section supplies operating power and thermal control.

The information and control signals for these functions appear on a number of busses. The data bus carries the data bits (of which it is assumed there are b) in a data word to (and from) the main memory subsystem (MMS). The address bus carries the address information during either the store or the fetch cycle to specify the address or location of the word register desired. The cycle control leads provide the timing and control signals for the individual steps of the store and fetch cycles. During the store cycle a valid address must be on the address bus and valid data on the data bus. The specified word register is then cleared to all zeros and the data word is copied into that register. The fetch cycle requires only an address. The contents of the specified word register are copied out onto the data bus and then rewritten into the word register without change.

The power supply is required to produce a minimum of three forms of power: one for write drivers, another for read drivers, and at least one other form for logic circuits. The term environmental control is used to refer to any necessary sensing and compensation for temperature-sensitive elements.

The overwhelming majority of the physical storage techniques for the data sections of present-day main working memories involve some form of magnetic storage elements. All of these techniques have in common a strong, complex intermix of circuits with both logic-level binary signals and analog signals. Furthermore, the analog signals exist in close proximity both at the driver power level and at low (near-noise) levels. Much of the art of memory design is in where and how these circuits are mixed. The wiring itself is an art and must take into account the

electrical and mechanical properties of the wire, as well as the winding patterns. The circuit problems are difficult and many. The success of the memory depends on attention to circuit and mechanical design details and the behavior of the parameters of magnetic material. In this Appendix, we consider the data section of the main memory subsystem as a single, asynchronous entity with error-control capability independent of the rest of the computer system. Integration of internal and external error-control schemes is an important problem for future study.

We wish to discuss here the techniques of redundancy for error protection separately from techniques of good design. To do this we need a simple functional model of the MMS. The model we will use is based on the connections to the parent system as shown in Fig. A-1.

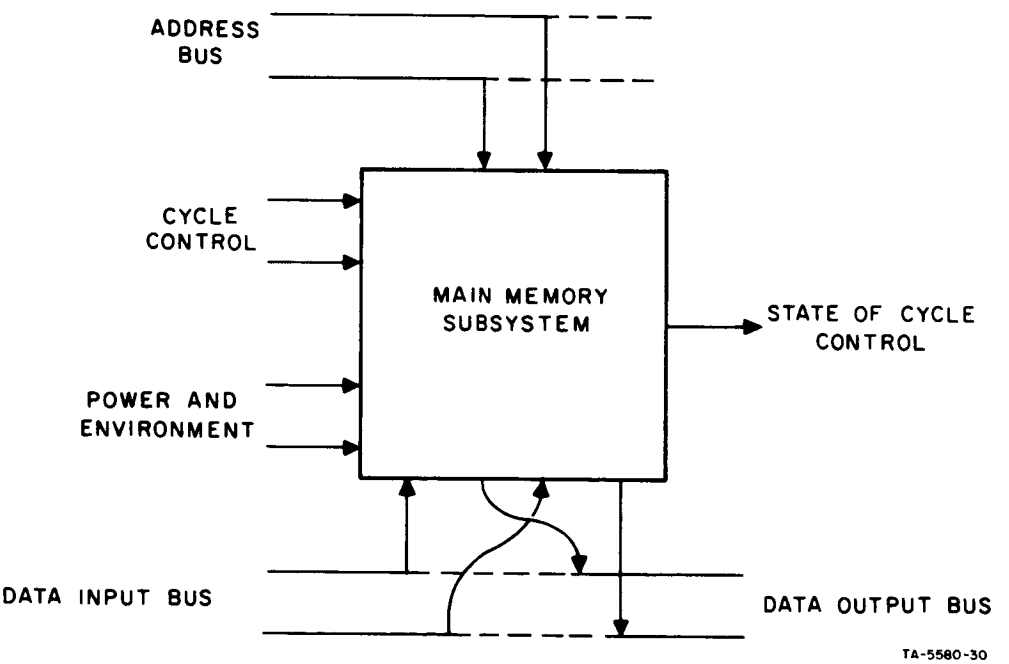


FIG. A-1 FUNCTIONAL CONNECTIONS BETWEEN MAIN MEMORY SUBSYSTEM AND COMPUTER SYSTEM

The data bus will be gated to a single data register; thus each bit of any one data word going to and from the main memory system will pass through this data register. All of the equipment for a single bit

is called a bit channel. Hence a bit channel includes the gating for a single bit from bus to register, the gating from register to digit driver, the digit driver itself, the array of storage cells for that bit, the digit sense amplifier, the gating back to the register, and the gating from register to output bus.

The model includes an address register, gated to the address bus, word drivers for both reading and writing, and address decoders for the selection of individual word registers. All of the necessary parts of this portion of the MMS which are required to excite or select a single word register and make it receptive to digit drive or make it excite a digit sense amplifier are called an access circuit. The number of leads threading an excited or selected word register from the access circuitry is generally what determines the so-called dimensionality of a memory. The model is receptive to only two types of cycle control--store (clear/write) and fetch (read/restore). It is assumed to operate completely asynchronously, i.e., once either cycle is started it will go to completion in a time referred to as the cycle time, and return the subsystem to a state of readiness to initiate either type of cycle.

The intersection of a bit channel and an access circuit is referred to as the bit cell or storage element. The present model is sufficiently general to represent the wide variety of storage elements available for spaceborne computers. These include single- and multiple-aperture ferrite cores, orthogonal-aperture ferrite cores (biax), thin planar magnetic films (bicore) and thick or thin circumferential films (plated wire). Monolithic-semiconductor memories do not require the conversion from logic level to drive level nor the conversion from sense level to logic level, since they do not exhibit the very large attenuation between drive and sense analog levels in the bit cell.

There are three primary classes of redundancy techniques which will be considered when comparing models. They are (1) circuit redundancy, achieved by using series and parallel connections of components, (2) logical fault masking, and (3) dynamic error control, achieved by sequential fault detection and active switchover to a nonfaulty unit.

It should be noted that circuit redundancy may be applied, practically, in only a limited number of places due to the reduction in circuit margins which usually results. Hence, the emphasis in this section is on logical fault masking and dynamic error control.

There are a number of criteria which could be used for the comparison among various proposed memory organizations. For spaceborne computers, the most important of these are (1) power consumption, (2) weight, and (3) the system success probability as a function of failure of a functional part within the subsystem.

To develop these criteria for comparisons of different organizational schemes, we will assume four memory sizes: (1) 256 words $\times$ 8 bits/word; (2) 256 words $\times$ 24 bits/word; (3) 4096 words $\times$ 8 bits/word and (4) 4096 words $\times$ 24 bits/word. The storage elements are assumed to be 0.03 $\times$ 0.018 inch single-aperture ferrite cores. The access is assumed to be of the coincident-current type, with both ends of the drive lines available for the extensively used, so-called "switch-sink" type of word-register selection. This selection scheme requires 4 wires per core. Data is transferred in and out in parallel, and a cycle time of 1 microsecond is assumed.

Under these assumptions for w words in the memory, with b bits per word, there will be (b + w) register cells for data and address registers, (4b + 2w) transfer gates to move information into and out of the registers (in both directions in the case of data registers), b digit drivers, b sense amplifiers, b $\times$ w cores, and finally not less than $4(w)^{1/4}$ switch-sink line drivers. These assumptions establish numbers of primary components.

On the basis of an examination of currently used components, we have assumed a 30 $\times$ 18 mil ferrite core and 1 inch of #30 copper wire/core--12 milligrams, and 40 milliwatts per μs/cycle. For these values, the weight of the copper overshadows that of the ferrite. The weight of the frames and other packaging components is assumed to rise

approximately in proportion to the number of cores. If each core in the word switches every cycle and the store is cycled at the maximum rate, b × 40 mW will be dissipated in the store. (This, of course, does not account for the power dissipated in the half-selected cores).

For register cells (RC) and transfer gates(TG) we have assumed the Fairchild Semiconductor DTμ line of integrated circuits and a popular 5-flatpack "mother board." Sense amplifiers (SA) and driver/switches (DS) are assumed to be gated, and operate with a duty factor of 0.5. Furthermore, no more than b + 4 drivers can be active simultaneously, i.e., b digit (inhibit drivers) and 1 switch and 1 sink each for X and Y word selection.

These assumptions yield:

Register cell	2 grams	80 milliwatts
Transfer gate	1 gram	4 milliwatts
Sense amplifier (with threshold R)	4 grams	360 milliwatts × 0.5 = 180 mW
Driver/switch (with terminating R)	4 grams	800 milliwatts × 0.5 = 400 mW

These assumptions yield the following relative power and weight figures for the basic irredundant memory sizes which are to be used as standards for comparison.

Words X Bit/Word	Part	Weight (gr)	Power (mW)	Cores	Weight (gr)	Power (mW)
256 x 8	16 RC	32	1,280	2,048	24.5	320
	48 TG	48	172			
	8 SA	32	1,440			
	24 DS	96	4,800			
Total		240	7,690		24.5	320
256 x 24	32 RC	64	2,560	6,144	74	960
	112 TG	112	448			
	24 SA	96	4,320			
	40 DS	160	11,200			
Total		432	18,448		74	960
4,096 x 8	20 RC	40	1,600	32,768	393	320
	56 TG	56	224			
	8 SA	32	1,440			
	72 DS	228	4,800			
Total		416	8,064		393	320
4,096 x 24	36 RC	72	2,880	93,304	1,180	960
	120 TG	120	480			
	24 SA	96	4,320			
	88 DS	342	11,200			
Total		630	18,880		1,180	960

3. Error Protection by Replication of Whole Memories

a. Triplication with Voting

One of the conceptually simplest means of protecting against any type of error in the MMS is defamiliar scheme of replication and voting. In this scheme input data to the MMS from the control leads and the address and data busses is sent simultaneously to three independent, irredundant copies of the MMS. During the fetch cycle, the selection of a particular word register is identical in all three modules. The output from the individual bit channels is combined with a three-input voter (assuming triplication) before being placed on the output bus. Thus, an individual corresponding bit cell in each data register would be gated to the input of a single voter whose output would drive one bit position on the output data bus.

The advantages of this scheme are numerous. First, no new design, redesign, or alteration to the design of an existing irredundant MMS is required. The only equipment needed in addition to the two full replicas of the MMS is a three-input voter for each bit position in a data word. This redundancy scheme, called "TV" for triplication with voting, would protect against failures in any one module regardless of whether that failure occurred in the bit-channel hardware, the access circuits, the cycle control, or the power supply. The cost of such potential gain is quite high; that is, the redundancy ratio is 3, plus a small fraction for the voting circuits. Alternatively, for a given size and weight of memory, the capacity is cut down by approximately a factor of three. It may be noted that the same redundancy ratio is applied to all components. Since the components differ widely in their reliability, the relative reliability, improvement for the various components is quite uneven in this scheme.

In order to be able to compare different schemes, we assume a single-aperture, coincident-current ferrite-core design, with a destructive-readout fetch cycle. The bits of the data words (or characters) are entered and retrieved in parallel. In accordance with common practice,

it is assumed that both ends of the word selection drive lines are available; hence a switch-sink (or driver-switch) scheme is applicable. It is also assumed that address decoding is done in a diode decoding tree before the conversion from logic level to power level is made.

In order to assess the potential gain in performance, we let X be the probability that one bit is in error at the output of a set of bit channels for a single MMS module.

The probability, Q, that the system is faulty is:

$$Q_{TV} = 1 - [(1 - X)^3 + 3(1 - X)^2 X]^b, \text{ for the redundant system}$$

$$Q_{NR} = 1 - (1 - X)^b, \text{ for the non-redundant system.}$$

For the 8-bit or character-access case, neglecting higher orders of X, this reduces to

$$Q_{TV} = 24X^2, \text{ versus } Q_{NR} = 8X.$$

For the word-access case, we have

$$Q_{TV} = 72X^2 \text{ versus } Q_{NR} = 24X.$$

In each case, $Q_{TV}/Q_{NR} = 3X$. Thus, the effectiveness of the redundancy increases with channel reliability.

These probabilities hold regardless of the word capacity of the memory, although X will increase with increased memory size.

If we assume that a "voter" is approximately the equivalent of a register cell in power and weight, we can compute the relative costs as follows.

Size	Part	Weight (gr)	Power (mW)	Cores	Weight (gr)	Power (mW)
3(256 x 8)	56 RC	112	4,480	6,144	74	960
	144 TG	144	576			
	24 SA	96	4,320			
	72 DS	288	14,400			
Total		640	23,776		74	960
3(256 x 24)	128 RC	256	10,240	18,432	221	2,880
	336 TG	336	1,344			
	72 SA	288	12,960			
	168 DS	672	34,600			
Total		1,552	59,144		222	2,880
3(4K x 8)	68 RC	136	5,440	93,304	1,180	960
	144 TG	144	576			
	24 SA	96	4,320			
	120 DS	480	14,400			
Total		904	24,736		1,180	960
3(4K x 24)	132 RC	264	10,560	294,912	3,539	2,880
	360 TG	360	1,440			
	72 SA	288	12,960			
	264 DS	1,056	34,600			
Total		1,968	59,560		3,539	2,880

It is noted that for a large memory (say 4000 words and 24 bits/word) the cores account for approximately 64 percent of the total system weight, and the inhibit drivers account for a significant portion of the total power. Since both the core weight and the inhibit drive power increase proportionately with the replication order, it is of interest to consider redundancy schemes which do not require constant replication of all components. One example of such a technique involves the use of error-correcting codes, as described in Sec. 4-a of this appendix. Various other techniques can be conceived which involve augmenting the storage section with additional data channels (comprising less than 100-percent redundancy) and also with additional bits/word for which simple parity checks are applied; the access circuitry can be protected by triplication and voting. Such a scheme is particularly attractive for memories in that the cores probably comprise the most reliable section of the memory and hence require the least protection.

b. Duplication with Parity Checking

The word capacity of the main memory is frequently the limiting feature in the capabilities of a spaceborne computer system. This makes it highly desirable to seek techniques which provide suitable measures of protection and permit the realization of the maximum storage capacity within given constraints of power and weight.

A considerable measure of protection can be afforded by merely duplicating (rather than triplicating) the MMS, provided some means is available for identifying and selecting the valid output. Such a "duplex" scheme was proposed by Kemp[157] and has been designed and used on the Saturn V computer.[61] In order to provide for a validity check, an additional bit channel is added to each block of the MMS to permit parity checking on the output. Inputs are fed to both modules of MMS simultaneously. During the fetch cycle, the output word is checked for parity and in normal operation only one output is connected to the output data bus. This data register is used in conjunction with the one module so long as parity is correct. With each restore half-cycle, the data

from this correct data register is used to restore the information read from the selected module into both duplex modules. On detection of a parity error, operation is transferred entirely to the other module.

This scheme, called DP for duplex-parity, offers good protection against all single-bit failures that are detectable by a parity check. This may be an error in a bit channel or an error in the access circuits which partially stimulates multiple locations in turn yielding a number of improper bit channel outputs. By using odd parity, even the failure of the cycle control circuits to give any output would be protected against. Errors which cannot be detected by a single parity checker, cannot, of course, be protected against.

Another important feature of the scheme is that good protection is afforded against failures in access circuitry that affect a small number of words. In an extreme case, up to w single-word faults could be tolerated between the two sections.

The probability of system success, P, for the duplex-parity checker (DP) arrangement can also be found as a function of the probability of failure in a single bit channel, S.

$$P = 1 - \Pr(\text{system failed})$$
$$= 1 - \Pr^2 (\text{1 module failed})$$

The probability of one module failing is 1 - P (all bit channels in a module are good) or $1 - (1-X)^b$. Hence the probability of system success is

$$P = 1 - [1 - (1 - X)^b]^2$$
$$P = 1 - [1 - 2(1 - X)^b + (1 - X)^{2b}]$$
$$P \approx 2(1 - X)^b - (1 - X)^{2b} \quad .$$

For the character-access case this is

$$P_{DP} = 1 - 84X^2$$

and for the word access case

$$P_{DP} = 1 - 576X^2 \quad .$$

We can extend this analysis to include errors in access and/or control by considering the following:

Let

X = probability that a bit channel is in error

S = probability that an access and control group (A-C) is in error;

then the probability of system success is

$$P = \Pr(\text{all A-C good}) + \Pr(\text{at least 2 out of 3 bit channels good}) + \Pr(\text{exactly 2 A-C good}) + \Pr(\text{all bit channels good})$$

$$P = (1 - S)^3 \left[3(1 - X)^2 - 2(1 - 3)^3\right]^b + 3(1 - S)^2 (1 - X)^{2b}$$

$$P \approx 1 - 3bX^2 - 6bXS - 6S^2$$

which reduces to

$$P_{DP} = 1 - 24X^2 - 48SX - 6S^2 \text{ for character access}$$

$$P_{DP} = 1 - 72X^2 - 144XS - 6S^2 \text{ for word access.}$$

While the memory capacity, w, does not appear in these results, it must be understood that the probability of error in access or control (S) certainly increases with increasing w. As would be expected, as S approaches zero, the probability of success, P, approaches the value indicated in the previous paragraph where S was neglected.

The relative power and weight for the duplex modes can be found by assuming the exclusive-OR gate for the parity check requires approximately the same power and weight as the register cell. The 9-input parallel checker for the character requires 11 gates and the 25-input word requires 28 gates. These will, of course, be required on both modules within the duplex (DP) organization. These assumptions yield the following (the notation 2[256(8 + 1)] indicates a system composed of two 256-word memories, with eight data bits and one check bit per word).

Size	Part	Weight (gr)	Power (mW)	Cores	Weight (gr)	Power* (mW)
2[256(8 + 1)]:	54 RC	108	4,320	4,608	55	720
	96 TG	96	384			
	18 SA	72	3,240			
	50 DS†	200	10,400			
Total		476	18,344		55	720
2[256(24 + 1)]:	122 RC	244	9,760	12,800	154	2,000
	232 TG	232	928			
	50 SA	200	9,000			
	114 DS§	456	23,200			
Total		1,132	42,888		154	2,000
2[4,096(8 + 1)]:	64 RC	128	5,120	73,728	885	720
	112 TG	112	448			
	18 SA	72	3,240			
	82 DS	328	10,400			
Total		640	19,208		885	720
2[4,096(25 + 1)]:	130 RC	260	10,400	204,800	2,457	2,000
	248 TG	248	992			
	50 SA	200	9,000			
	178 DS	712	23,200			
Total		1,420	43,592		2,457	2,000

* Although the duplex scheme is not as costly as the triplication method (and of course not as powerful from the standpoint of error correction) it still requires strict duplication of all memory channels--producing a costly increase in power and weight. The techniques of the following part are more economical, although providing comparable protection.

† A maximum of 2[9 + 4] driver/switches (DS) can be "on" at any one time.

§ A maximum of 29 driver/switches can be "on".

4. Error Protection by Redundancy Within a Memory

There are many ways of controlling transient or permanent errors which do not involve replicating the entire MMS. These fall generally into two rather natural groups: (1) those that increase the number of bits per word--i.e., redundant bit channels--and (2) those that increase the number of address locations--i.e., redundant storage registers. Both approaches will be discussed in this section. In applying redundancy, precautions should be taken to avoid overloading heavily stressed circuits. For example, if a redundancy scheme increases the load on the current drivers, which are usually stressed more heavily than other parts, they may be expected to be more susceptible to failures, unless separate access circuits are added to handle the increased load.

a. Redundant Bit Channels

Many faults may occur independently among the bit channels; hence the use of error-correcting codes may provide an efficient method of fault masking.

One manner in which this might be accomplished is illustrated in Fig. A-2. Since the incoming data is assumed here to contain no redundant bits, the generation of the necessary bits to store in the redundant bit channels must be done within the MMS during the store cycle. During the fetch cycle these bits are used to correct erroneous bits from the MMS before they are placed on the output bus. The number of errors per word that are detectable or correctable by this technique depends on the design of the particular error-correcting code. We wish to illustrate here the practical implementation of such codes for several word sizes.

A familiar and very effective family of codes is the family of single-error-correcting Hamming codes. Circuits to implement this scheme for the 8-bit character, the 24-bit word and the 24-bit word broken into 3 8-bit bytes are discussed in Sec. 5.

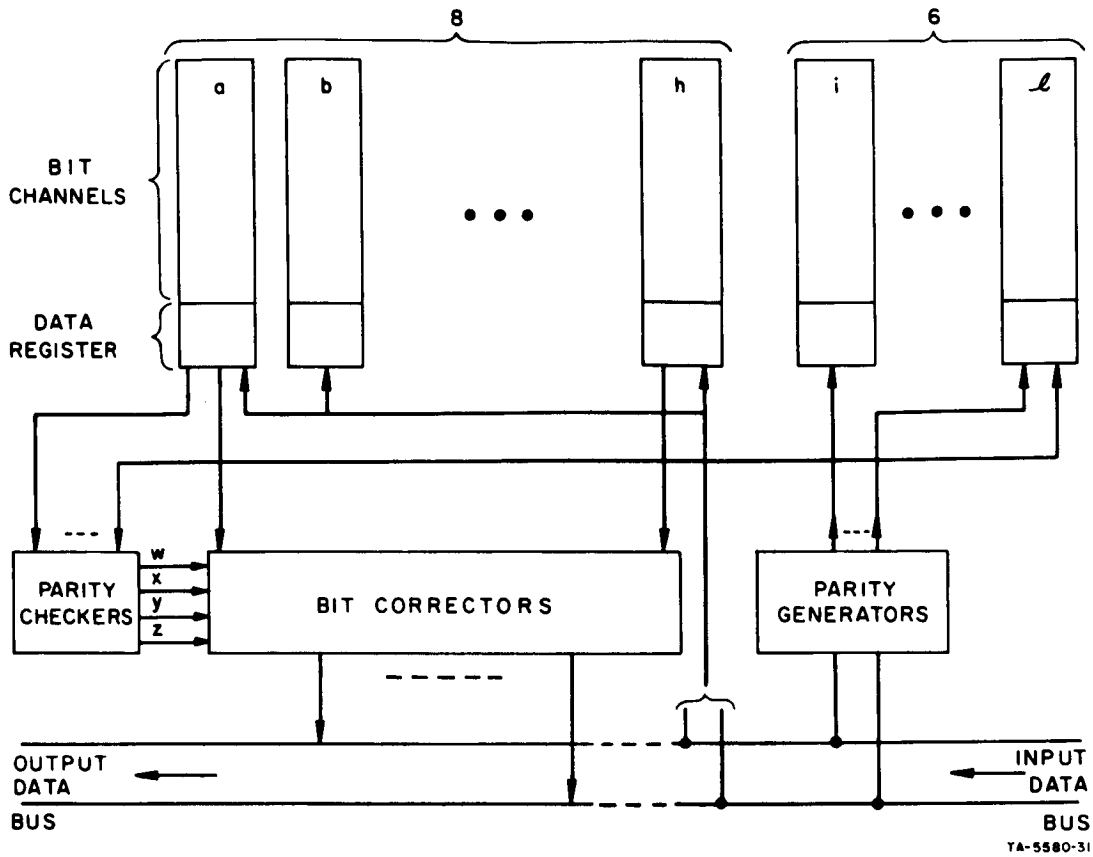


FIG. A-2 REDUNDANT BIT-CHANNEL CONNECTION

For an 8-bit byte or character, four redundant bit channels are required. Besides adding to weight and power dissipation, the individual drive lines must now drive more cells or cores; hence they must either be redesigned or suffer some loss in expected reliability. Protection is afforded against failures in gating, the data register, digit drivers, storage cells, and sense amplifiers. No protection is afforded against failure to select the correct word, the encode-decode equipment, or the cycle-control hardware.

The probability of system success for the single-byte (or single-character) case can be computed as follows. There must now be 12 bit channels--8 data and 4 parity. Again X is the probability that a single

bit (or bit channel) is in error, P the probability that the system is good, and Q the probability that the system is not good. Then

$$P = (1 - X)^{12} + 12X(1 - X)^{11} \quad .$$

Neglecting high order terms,

$$Q = 1 - P = 66X^2, \text{ for the character, versus } Q_{NR} = 8X.$$

Since this does not take into account failures in the access subsystem this holds for both word capacities considered. If we now assume 3 bytes, similarly protected to make up a word, we have

$$P = [(1 - X)^{12} + 12X(1 - X)^{11}]^3$$

and

$$Q = 1 - P = 198X^2, \text{ versus } Q_{NR} = 24X.$$

For both cases,

$$Q/Q_{NR} = 8.25X \quad .$$

The minimum number of redundant bits to protect a 24-bit word against single-bit errors is 5. Hence, to find the system success probability for 29 bit channels using single error correction, we have the following.

$$P = (1 - X)^{29} + 29X(1 - X)^{28}$$

$$Q = 1 - P = 406X^2$$

Each of these cases neglects higher powers of X.

The relative power and weight are only slightly increased over the irredundant case and are as follows.

For the character case:

Size	Part	Weight (gr)	Power (mW)	Cores	Weight (gr)	Power (mW)
256 (8 + 4):	20 RC	40	1,600	3,072	37	480
	60 TG	60	240			
	12 SA	48	2,160			
	28 DS	112	6,400			
Total		260	10,400		37	480
4,096 (8 + 4):	24 RC	48	1,920	49,152	590	480
	68 TG	68	272			
	12 SA	48	2,160			
	76 DS	304	6,400			
Total		468	10,752		590	480
256 (24 + 5):	37 RC	74	2,960	7,424	90	1,160
	127 TG	127	508			
	29 SA	116	5,220			
	61 DS	244	13,200			
Total		561	21,888		90	1,160
4,096 (24 + 5):	41 RC	82	3,280	118,784	1,426	7,160
	135 TG	135	540			
	29 SA	116	5,220			
	93 DS	372	13,200			
Total		705	22,140		1,426	7,160

Thus we see that the cost of the protection here is significantly less than the cost of either the triplication scheme or the duplex scheme. Although the protection afforded by the triplication scheme is somewhat

greater than the error-correcting scheme discussed here, if the probability of an element failure is low then the protection levels are approximately the same.

For the case where a word is treated as 3 bytes of 8 bits each, the following costs apply:

Size	Parts	Weight (gr)	Power (mW)	Cores	Weight (gr)	Power (mW)
256 x 36:	44 RC	88	3,520	9,218	111	1,440
	148 TG	148	592			
	36 SA	144	6,480			
	68 DS	272	16,000			
Total		652	26,592		111	1,440
4,096 x 36:	48 RC	96	3,840	147,456	1,770	1,440
	156 TG	156	624			
	36 SA	144	6,480			
	100 DS	400	16,000			
Total		796	26,944		1,770	1,440

b. Redundant Words

The use of error-correcting codes on the bit channels is of no help in masking faults in the access section. Faults in an access switch usually invalidate at least one word, and usually a block of words. Certain faults in the access switch or in the decoding circuits may invalidate all words.

One straight-forward way of preventing such catastrophic failure is to subdivide the access equipment into independent parts, each giving different sets of words, so as to limit the extent of propagation of a given fault. In use, this requires that the system have the addressing flexibility to permit transfer of data to different locations in memory. Such flexibility is well developed in modern computers designed for

multiprogramming or for multiprocessing. Its implementation is aided by use of a special directory table, which specifies the physical location of memory addresses, in groups of addresses known as blocks, or pages. This table may be located in an ordinary memory, or perhaps replicated in several memories, for safety. No discussion of this technique has been found in the literature, but it would seem to have sufficient merit to justify further development.

c. Accomodation to Access Faults

A fault in an access system may invalidate one or more words. It would be desirable to shift the contents of such words to new locations. One way to accomplish this relocation is to change the address code of all instructions that call upon that word.

For those words that are directly addressed, the change may be accomplished by changing the contents of the address field within the instruction word stored in memory. Determination of those instructions that address a given word may be accomplished in one of two ways: either by exhaustively searching all instructions within memory or by waiting until such references actually occur (at which time the location of the calling instruction is revealed).

For those words that are accessed by the stepping of a counter or by some other arithmetic operation, or by a composition of several address subfields, such changes are impractical. One solution for these kinds of access is to use an associative memory in which such exceptional addresses are stored together with the address of the substituted words, and to drive the memory with address data in parallel with the main access switch. The output of the associative memory may then be used to substitute for the nominal, unusable address.

The merit of such schemes is that faults in access equipment may be accommodated with equipment low redundancy compared to other known schemes.

d. Addition of Access Redundancy

A very interesting scheme of error control for memories has been suggested by C. A. Allen.* This scheme uses redundant bit channels plus the encoder--decoder for error-correcting codes exactly as in the previous section. The essential addition is that each bit channel is provided with its own word-access circuits for address decoding, and with read/write drivers. In this organization, failures in any one set of access circuits affect only a single bit channel. The error is then corrected as the word is transferred to the output data bus during the fetch cycle.

This scheme has obvious disadvantages for magnetic memories. The access circuits must be replicated b + k times. Each of these circuits must not only decode the address input but amplify the decoded signal from logic level to drive level. The replication of amplifiers (drivers) is not likely to be practical. However, future memories may well be built from techniques which require only decoding--that is, in which the access information remains at logic levels. Integrated circuits with a flip-flop for a bit-storage cell are an example.

Much of cycle control is associated with the access circuits. The rest is associated with transfers, such as from register to bus. If circuit redundancy is applied to this portion of cycle control it is possible to design an MMS which is completely protected against single part failures throughout the subsystem, and which costs considerably less in power and weight than its TV counterpart.

e. Redundant Access Circuits

Probably more design effort has gone into the development of reliable access switches than any other part of the MMS. In the area of design for fault prevention, some knowledge of the mechanism of failures is required. A very interesting study has been done by Minnick,[110,111] based on the assumption that the ferrite and wire portions of memories are far more reliable than the semiconductor portions. Techniques are

* Described in a graduate seminar at Stanford University, Fall 1965.

then developed which employ magnetic switching elements for address decoding, and transistors are assumed to be restricted to the driving of the magnetic switches. Thus, a number of drivers are turned on to accomplish the selection of a single storage resistor, and the energy from these drivers is combined in a magnetic access switch to do both the selection and driving of the specified storage register. The switch is wired according to certain codes, such as those based on block designs.* The design is such that the failure of a single driver changes only the amplitude of the current supplied to the selected memory line. A further development in the design consists of placing a load core on the ground-return side of the memory lines so that its switching resistance assists in regulating the amplitude of current which passes through the storage module, in order to tolerate the variation in current due to a faulty driver. A further requirement of these selection switches is that the access lines to memory be single-ended; thus it does not permit a switch-sink type of decoding. The utility of this technique is greatly increased if the address information is generated in a redundant code at the computer, since the drivers require a redundant code for their excitation. If irredundant addresses are transmitted from the computer to the MMS, an encoder from the address register to the drivers or from the address bus to the address register must be provided. Minnick also considered design techniques for a number of recoders using magnetic elements.[111]

It would be desirable to evaluate the total weight and power costs of several of the schemes described by Minnick. These costs are probably very high compared to all-semiconductor realizations, but recent advances in the miniaturization of magnetic elements may increase the feasibility of the approach. A disadvantage of the scheme is the small number of faults that may be accommodated with reasonable cost.

Very little has been published on detection schemes for telling whether or not the proper storage register has been selected. An

---

* See Section III-C-2-c of this report for an illustration of this method.

interesting detection scheme has been published by the General Electric Company[157] as to whether or not any word has been selected or whether more than one word has been selected. This scheme involves the addition of one core plane or bit channel, which lacks an inhibit driver. Each storage register then has an additional core, and since there are no inhibit circuits, the core switches on read and switches back on restore, giving a sensible output on every cycle. Two separate sense amplifiers are used on reading--one set with a threshold level for a single core switching and the other with a threshold level for two cores switching. If neither amplifier senses a switched core during a cycle, no storage register has been selected and there has been improper operation in the word control. If the single-threshold amplifier switches but not the double-threshold amplifier, proper operation is assumed. This scheme gives no indication as to which storage register was selected, but consideration of several access-switch schemes indicates that an exchange of single register selections due to a fault within an access switch is extremely unlikely. If the double-threshold amplifier switches, it indicates that two or more storage registers have been simultaneously selected during that cycle. This information could be used in conjunction with parity checks on the data to switch over to an entirely different block of core memory. This scheme has been given the term "memory-cycle validation check."

f. Redundant Material in the Storage Module

Single-aperture ferrite cores have proven to be exceedingly reliable elements and little has been published on redundancy on the element level within the storage module. The fact that readout requires destruction of the stored information and relies upon the electronics external to the storage module to restore the information has caused considerable concern about the possible loss of information due to transient electronic failures. Considerable effort has gone into the development of techniques to provide nondestructive readout (NDRO).

It is difficult to design an NDRO memory to the same tolerances as a DRO memory of the same size, speed, and power consumption. The principal technique of construction is based on the use of some multiaperture (usually two-aperture) magnetic element. Problems of drive-current tolerance have led to the use of large structures, with consequent heavy use of power. Use of more complex (three-aperture) elements can be helpful, but this requires extra drivers and wire. These costs of weight, power, and reduced reliability should be evaluated with respect to the extra costs of protecting DRO memories by special control circuits.

These weaknesses apply to an NDRO memory which is required to record new information during its mission. It is useful in spaceborne missions to have NDRO memories in which writing occurs only prior to launch, perhaps using externally fed writing currents. In this case, the main disadvantage is the extra weight due to the use of (generally) heavier memory elements.

g. Redundant Cycle Control

The function of the circuits within the cycle control is to accept the two commands from the computer, distinguish between them, and then generate the detailed sequence of control pulses which will turn on gates for data transfer, initiate drive pulses, and gate sense amplifiers at appropriate times. The two primary types of circuits that are used for this function are tapped delay lines and special counters. Delay lines may of course be replicated and their outputs combined in majority voters. For lines in which the expected failure mode is dropout, only duplication is needed. Fault-tolerant counter circuits have been described in the literature and they are generally smaller and require less power than delay lines.

h. Redundant Power and Environment Control

Current must move through the access lines, and hence through the storage cores, in opposite directions during the write and read portions of a cycle. This is generally accomplished by having separate current drivers which operate from power supplies having opposite polarities.

As temperature increases in a ferrite core, the drive current required to switch it decreases. This change in drive current as a result of change in temperature is usually built into the power supply with a temperature-sensing device to control the voltage of the driver supplies. While many designs are available to permit this variation in drive voltage with temperature, no literature has been found on the use of redundant techniques in the power supplies. A third supply voltage is generally required, separate from the two just mentioned, to supply all of the logic circuits within the main memory. Since historically power supplies and turn-on transients have occasioned a great deal of lost data, it is surprising that so little attention has been paid to this part of the memory design. This problem is discussed further in Appendix B of this report.

5. Design of a Parallel Encoder/Decoder

Several techniques based on the use of error-correcting codes have been described which permit the addition of redundant bit channels to achieve an improvement in reliability, with redundancy ratios less than 2. In order to conserve operating speed, it is desirable that the code-processing functions be performed on all data channels in parallel. The design of parallel encoders has been studied in the past, but there is little information as to the practical costs of such networks. The use of error-correcting codes with additional bit channels to store the redundant digits would provide a method of masking any single-error fault within the bit-control module. In this section we study the design of an encoder placed between the input data bus and the data register, with the mating decoder placed between the data register and the output data bus. Such placement of the encoder/decoder will mask faults within the data register, the digit drivers, the storage module, the sense amplifiers, and the transfer gates, but of course not in the encoder/decoder itself.

The design of the encoder/decoder was carried out assuming a commerical line of integrated circuits in order to get some reasonable estimate of size, speed, and power consumption. The design is based upon the following parity matrix.*

$$P = \begin{array}{c} \begin{array}{cccccccccccc} A & B & C & D & E & F & G & H & I & J & K & L \end{array} \\ \left|\begin{array}{cccccccccccc} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right| \end{array}$$

This matrix is based on the 12,8 single-error-correcting Hamming code. The data bits are represented by bit positions A through H and the four redundant check bits are represented by positions I, J, K, and L. Hence twelve bit channels are required for storing a given data word. The task of the encoder is to generate the information which will be stored in the redundant bit channels. As the operation of the encoder/decoder is described, it will be helpful to follow Fig. A-3.

The encoding is accomplished by checking parity over those bit positions in the data word for which ones exist in a given row of the parity matrix. Thus, for example, bit I is obtained as the mod-2 sum of the information in channels A, D, E, G, and H.

Decoding is done in the following manner. Four parity checkers are built, whose outputs are designated W, X, Y, and Z, one corresponding to each row of the parity matrix over all bits containing one, now for all 12 columns. The four bits from these checker outputs are termed the "syndrome character." If there have been no errors, these 4 parity-checked outputs will all be zeros. An error in any one bit channel will cause one or more of the parity checkers to give a one output, generating a syndrome character that is not all zeros. For single-bit error, the possible syndrome characters will correspond to one of the columns A

---

* The reader is referred to Sec. II-B-2 for a general discussion of error-correcting codes for storage.

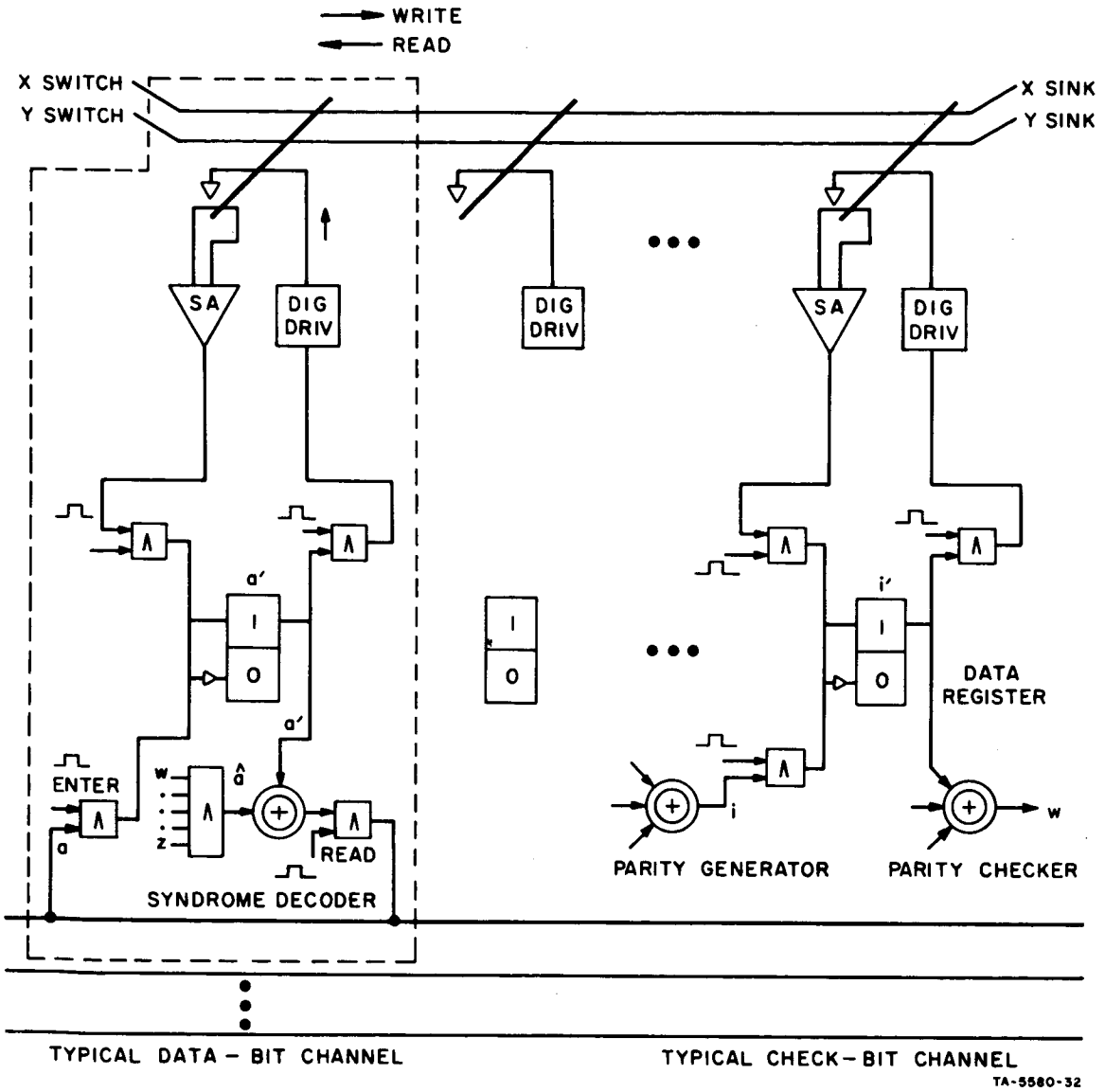


FIG. A-3 ERROR-CORRECTING CODE IN BIT CHANNELS

through H. Occurrence of a single error will result in one channel being indicated, and the error in that channel may be corrected by inverting its data before transferring it to the output bus. That column, then, must have its bit output from the data register in error. This is done by including an exclusive-OR gate in the output transfer, so that any line containing a one bit from the syndrome decoder will invert the output of that bit channel as it is placed on the data bus.

The parity generators for generating the Ith and Kth bits during encoding are assumed to be built from Fairchild integrated circuits, model DT L 930, and are shown in Fig. A-4. Similar parity generators are required for the generation of the J and L bits and also in the decoder for the generation of the W and X, Y and Z. Fig. A-5 shows the necessary integrated circuits to accomplish the syndrome decoding, the bit correction, and the data register to data bus transfer gates for bit channels A and B.

It should be noted that a single four-input gate can be used to detect the fact that an error has been generated somewhere even though the error is masked in the transfer. That is, if the syndrome character has a 1, i.e., if it is not all 0's, then an error has been detected and masked. This fact can be transmitted to the computer for possible status-of-equipment analysis.

The Hamming-code equations for a parity matrix to permit single error detection and correction in a 24-bit word require 5 redundant bit channels, or a parity matrix with 29 columns and 5 rows. Such a parity matrix can be obtained from a binary counting sequence by eliminating the all-0 columns, the all-1's column, and the column with four 1's. The parity generator then requires the design of parity checkers over 15 bits.

The parity matrix for 24 information bits plus 5 redundant bits is:

$$P = \begin{array}{c} \begin{matrix} C_1 & C_2 & A & C_3 & B & C & D & C_4 & E & F & G & H & I & J & K & C_5 & L & M & N & O & P & Q & R & S & T & U & V & W & X \end{matrix} \\ \left[\begin{array}{ccccc|ccccc|ccccc|ccccc|ccccc|ccccc|ccccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right] \end{array}$$

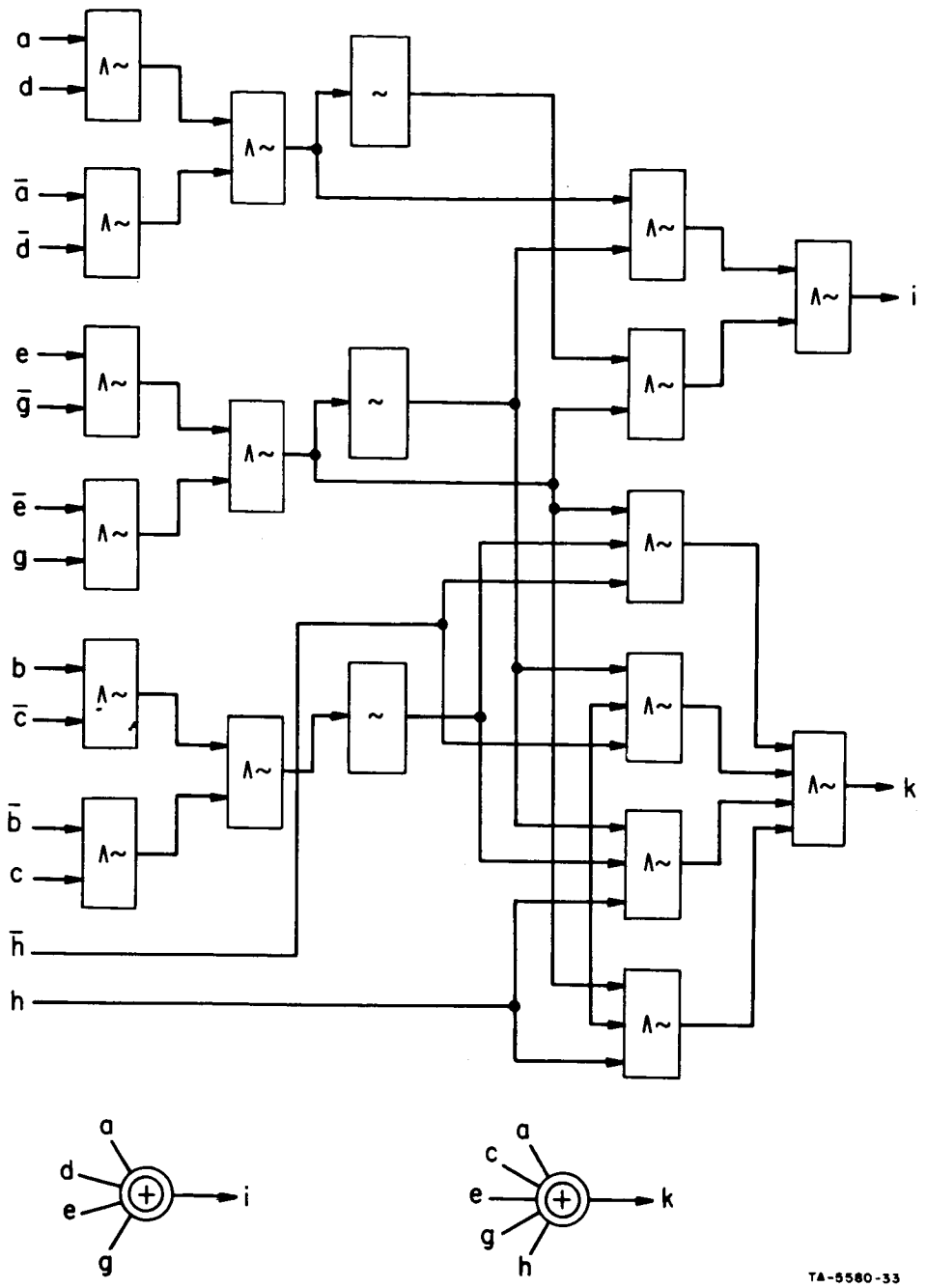


FIG. A-4 PARITY GENERATORS FOR i AND k

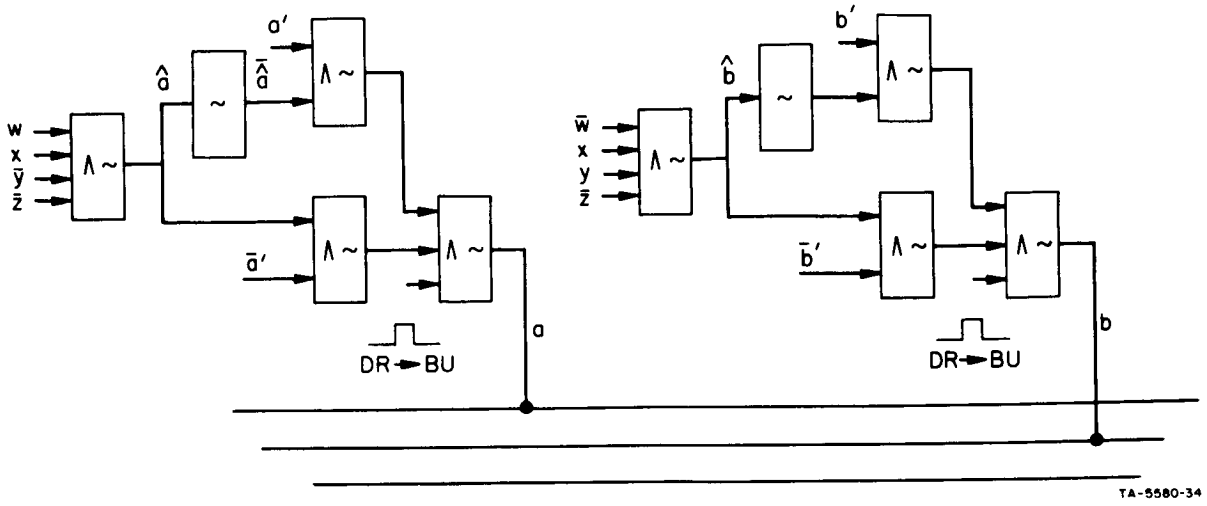


FIG. A-5 SYNDROME DECODER, BIT CORRECTOR AND DATA REGISTER TRANSFER GATES

The columns with single 1's are chosen to be the check bit, since they can be generated with a single parity generator over the 1's in that row.

6. Conclusions

Several schemes have been described for the control of errors within the various sections of a random-access main-memory system. The relative costs in power and weight and the relative improvement in reliability have been compared for a number of major approaches, for several parameters of word size and count. The evaluation of costs is made on the basis of cost parameters for present off-the-shelf components. Future developments in technology promise to reduce the proportional weight costs of the electronic subsystems.

The use of error-correcting codes for masking independent faults in bit channels is practical and very beneficial.

Several schemes have been discussed for masking, avoiding, or otherwise accommodating faults in access equipment, but their complexity is such that a good comparison would require very extensive analysis, and the analysis of some schemes would have to be tied to particular circuit schemes. Further investigation of these schemes is recommended. Of the various schemes, paging appears to be the easiest to employ.

The need for new techniques for the protections of power supplies has been noted.

It is recommended that techniques of error control be investigated for various special kinds of memory systems not considered here; e.g., associative memories, fixed memories, and buffer registers, assuming the various appropriate device technologies. Also, although the scheme described here are generally applicable to NDRO memories, special redundancy techniques may be useful to particular memory structures. These possibilities should be investigated further.

Because of the close physical interaction among the several functional sections of a memory, it is necessary to evaluate a set of schemes with respect to the overall reliability of an integrated memory system. Such evaluation requires the conducting of design exercises based upon the choice of particular sets of operational requirements and particular device technologies. It recommended that such design studies be undertaken.

Appendix B

DISTRIBUTED POWER-SUPPLY SYSTEMS

Appendix B

DISTRIBUTED POWER-SUPPLY SYSTEMS

1. Introduction

The design of power supplies has been too often considered as a separate (and usually secondary) system problem. It is very clear that as high-performance systems grow more complex, the power-supply and power-conditioning equipment must be carefully designed as an integral part of a system, rather than being added on as a separate system component. There are many possible power-supply methods for ultra-reliable computers, ranging from single (nonredundant) designs to methods of using distributed power supplies, or at least distributed power conditioning.

Early in the present project we felt that the distributed power supply (having a separate power unit for each logic group) had merit; we now feel that some form of distributed, noncentralized power supply is essential to the success of complex, long-life computer systems for space use.

In the following pages the advantages and disadvantages of distributed power conditioning systems and the interdependence between supply and logic circuits will be discussed. Three examples of possible ways to design power-supply systems are given, along with comments on these examples, and some notes on areas where additional work is needed.

2. Advantages of Distributed Power-Supply Systems

(1) There is nearly complete independence between a logic circuit group and other subsystem parts.

(2) Various logic circuit types of different origins could easily be merged to fulfill new system needs, without being required to power the units from available supply voltages. This is particularly true if ac coupling is used at data interfaces between modules.

(3) A power cutoff switch could be included in each module, so that unused modules would not consume power.

(4) Current limiting is much easier, making it easy to protect the raw power source from damage.

(5) There is very little electrical interaction between modules through the power supply.

(6) Semiconductor device reliability is better, since a large number of small-junction transistors and diodes are used, rather than a few large-junction units.[278]

(7) Switching a single power-input line to a multiple-output logic module, in combination with the use of decoupling diodes at each output, provides an economical means for switching the whole module in and out of the system.

3. Disadvantages of Distributed-Power-Supply Systems

(1) A DPS system is heavier than a single nonredundant system, possibly heavier than two single systems, unless higher frequency converters are used in the DPS.

(2) Circuits are more complex, since each DPS module supply is nearly as complex as a single large supply.

(3) Electrical efficiency is lower, particularly if some conditioning is required before the raw dc power is distributed.

(4) The load requirements of each logic group should be similar, to avoid too many different types of DPS in a given system.

Further comparisons are not particularly useful unless specific systems are compared. The entire computer system should be considered without attempting to treat the power supply separately.

4. The Interdependence Between Power Supply and Logic Circuits

a. Noise Problems

Many digital system problems can be traced to the power supply, or are blamed on the power supply. Usually the troubles are due to "noise" caused by logic-circuit loads being switched on and off abruptly in the course of normal computation tasks. Unless the power-supply output impedance is extremely low, the voltage on the power supply varies rapidly as the loads are switched, causing noise on the output bus. If the noise voltage is of sufficient magnitude, false triggering of circuits can occur.

The usual remedy is to use bypass capacitors distributed among the groups of circuits, so that the short-duration currents are drawn from a local capacitor. The most widely used form of digital logic uses dc level shifts to define a 1 or 0, rather than ac coupling of pulses between circuits or groups of circuits.

If several different power supplies are used in a system employing dc coupled logic, the power supplies must have nearly equal voltages, or the noise margins of the circuits will suffer.

The ratio between the 1-to-0 voltage swing and the noise-voltage magnitude is a good measure of the susceptibility of a digital system to noise problem. Some circuits require only a few tenths of a volt, and are affected by very small noise voltages on the power supply (or on data leads), while others require several volts to trigger, and are therefore relatively unaffected by power-supply noise. High-speed circuits are more susceptible to supply problems, unless great care is taken to minimize the length of leads.

If distributed supplies are used with dc coupled logic systems having poor noise immunity, ac coupling at the data interfaces would be very desirable, since a few tenths of a volt difference between the individual module supplies would not degrade the noise immunity of the logic circuits.

If high-level devices such as field-effect transistors were used in logic circuits, ac coupling at the data interface would probably not be required, and the design of a workable application of DPS system would be much easier.

Note that the ac noise problems in a computer using DPS are not as serious as in single-power-supply systems, since individual module regulators are in effect cascaded (in series), greatly reducing mutual coupling between modules.

b. Fault Location, Isolation, and Corrective Action

When a malfunction occurs in a computing system, it is desirable to determine where the fault is, to isolate the defective elements, and to take the best available course of action to restore as much computational ability as possible.

When a power-supply fault occurs, the first order of business is to protect as much of the system from consequential damage as possible. If the fault results in a transient overload, it may be desirable to simply limit the fault current to prevent damage to the raw supply, and restore voltage to the module after a brief interval. If excessive current demands persist, then the module should be disconnected from the raw supply to prevent energy loss to a useless module. If voltage can be restored after a momentary fault, then logical checks should be made to determine whether the computational performance of the module has been impaired.

Note that a logic check will always determine whether significant damage has occurred, so that the main function of the module supply itself is to prevent consequential damage to other system elements. Permanent disconnect of the module is thus a supervisory function, while protection from damage is a local, self-contained function of the module supply itself.

Future computers will operate on very low power, as evidenced by the work being done by Fairchild.[343]

5. Examples of Three Possible Designs for Power-Supply Systems

Of the many designs that could be used for supplying power to a spacecraft computer, three methods have been selected as examples of power conditioning. These examples are shown in Fig. B-1.

Method 1 is the most conventional of the three, and is used in some systems already designed. Raw dc power is "chopped" in an dc-to-ac converter, rectified and filtered, and regulated by a series regulator.

Method 2 is similar to Method 1, except that the regulators are distributed and located at each logic group. This kind of system has been tried, but not used to any extent.

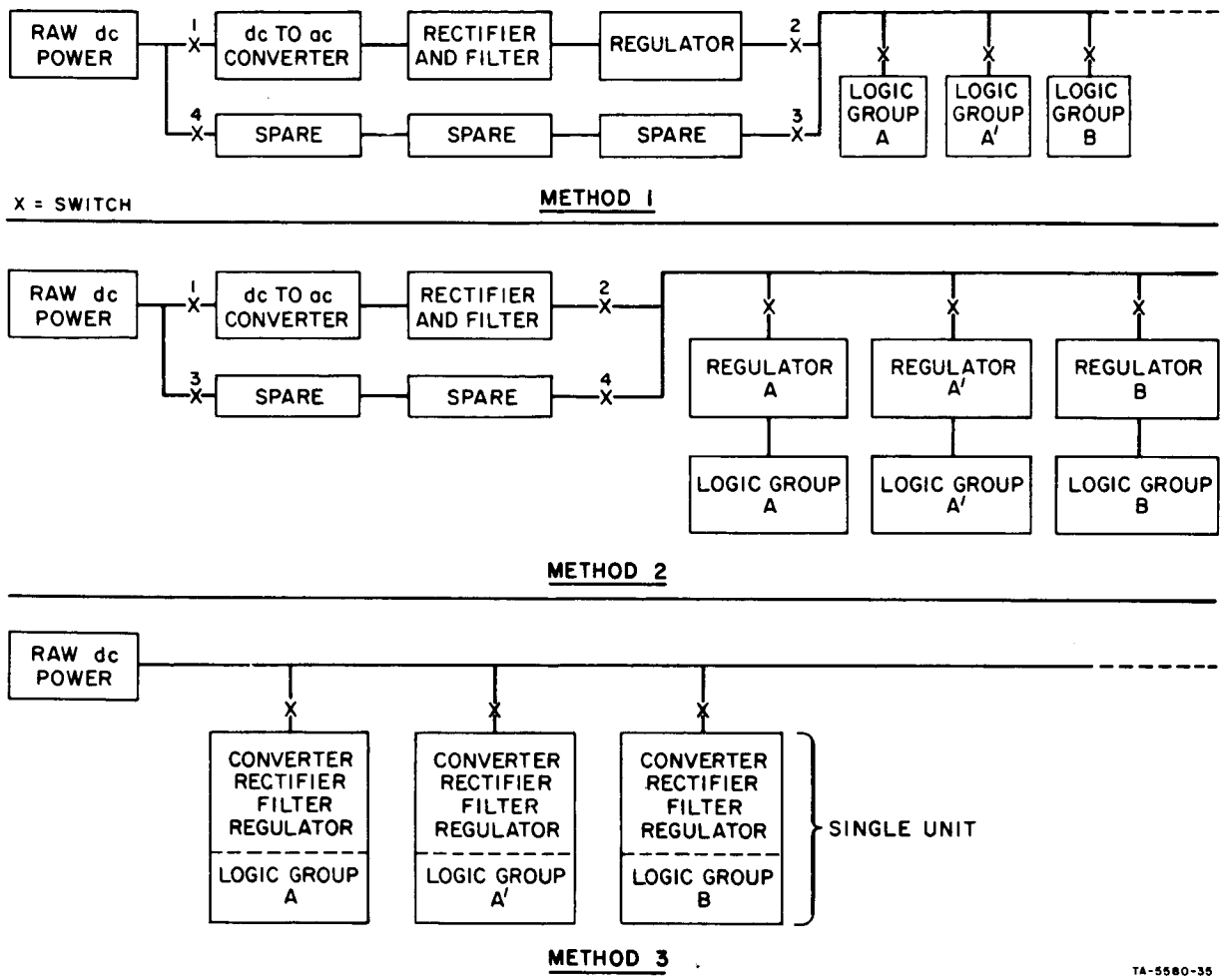


FIG. B-1 POWER CONDITIONING SYSTEMS

Method 3 represents a true distributed power-conditioning approach, since raw dc is wired to each power-supply converter/regulator, and no elements are common to all logic groups.

The scheme used for switchover to a spare supply (Methods 1 and 2) is very elementary: open 1 and 2 and then close 3 and 4. More complex schemes using more switches and crossovers have been proposed. These schemes allow a spare regulator, for instance, to be switched into the existing supply. If the switch reliability is not considered, the complex schemes could have much higher reliability than the simple spare-supply concept. Switch reliability is important, and the control problems of the complex rerouting schemes are serious. Weight penalties of two to three times the normal supply weights are also involved. The complex schemes have therefore not been widely used.

Although individual regulators are not presently employed at each logic group, series filter inductors or resistors and bypass capacitors are frequently used.

Method 3 certainly involves more parts than the other examples, but it should be easier to manage from a supervisory standpoint, and the weight penalties are not severe.

6. A Possible Configuration for a Power-Control System

The converter/regulator associated with each logic group should also include a disconnect function so that (1) the logic group can be removed from the source of energy in the event of a fault within the group, and (2) unused groups can be disconnected to conserve power.

It is believed that such disconnect circuits should have a toggle action, so that if directed "off" or "on," they will remain in the desired position, even though a general power failure has occurred. A possible scheme for such a system is shown in Fig. B-2. Each power-control element has a square-loop magnetic core associated with it, so that when the core is "set," voltage is applied to the logic group. Power would be applied to the computer by applying coincident pulses to crosspoint J1 (Logic Group A energized), J-2 (B energized) etc.;

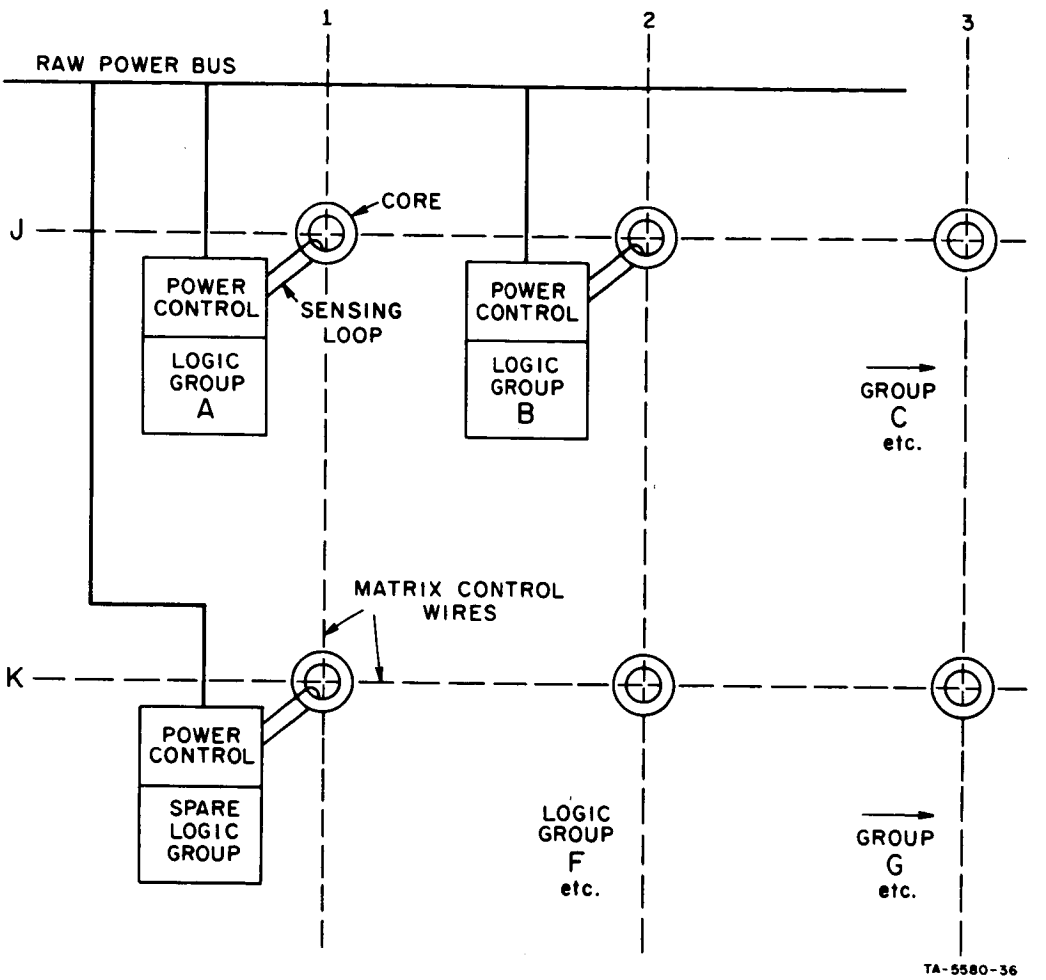


FIG. B-2 POSSIBLE POWER-CONTROL SYSTEMS

if logic group B failed, or was suspected to have malfunctioned, cross-point J-2 would be pulsed off and the spare group turned on by pulsing K-1. Magnetic control of this type would be easy to address, and has the additional advantage of being electrically isolated from both the data and power-supply components.

A carefully designed system for controlling power would probably simplify data-switching problems since a failed logic unit would not deliver any erroneous signals to adjacent units. As pointed out earlier power can be conserved by turning off unused units. A simple method for doing this by means of a series transistor is described in a very recent article by Clift.[42] Note that this could be the same transistor used in the regulator, or a separate unit. At least one commercial computer already uses power switching to conserve power within an integrated circuit memory.[253]

7. Weight and Power Required for a Distributed Power Supply

As an example of how much one would have to pay in terms of weight and power loss for a distributed supply system, we have considered a single-voltage 20-watt supply. A well-regulated conventional supply would weigh 350-500 grams and consume about 22 watts from the raw supply under nominal input-voltage conditions. In this power range, the chop frequency would probably be approximately 1000 Hz. Weight could be reduced by increasing the frequency, but efficiency would suffer.

Small supplies with oscillating converters can be made light and efficient. As an example, consider the power supplies built at SRI for the NASA PIONEER experiments. These units have the following specifications:

Input voltage: 28 ± 9 volts, 80 mA nom. (3.1 watts)
Output voltages: -3, +5, +12, +2.5
Regulation factor: over 10,000
Chopper frequency: 30-40 kHz
Efficiency: 85% under nominal conditions, input may vary from 19 to 37 volts
Power output: 2.6 watts, 90% of the power being in the 12-volt circuit
Weight: 55 grams, including transformer and filter capacitors.

We have estimated that a 1-watt, single-voltage unit would weigh about 25 grams.

Four degrees of split-up of the power supply have been considered; a 1-section, a 25-section, a 36-section, and a 64-section system. The individual supply sections would have to deliver slightly over 1 watt for the 16-section unit, and about 1/3 watt for the 64-section.

Figure B-3 is a plot of the weight of the distributed system and the amount of power needed for operation. Note that even for a 64-element system, the weight is less than triple that for a single supply. In making this estimate, we have assumed the transformer and capacitor weights per unit would decrease from 10 grams for a 16-section system to about 5 grams for a 64-section system, that the semiconductor weight would decrease from 12 grams to about 4 grams, and that the core-switch weight would be constant at about 4 grams. Efficiency was assumed constant at 85 percent, except that each unit consumes about 10 mW for voltage-reference circuits.

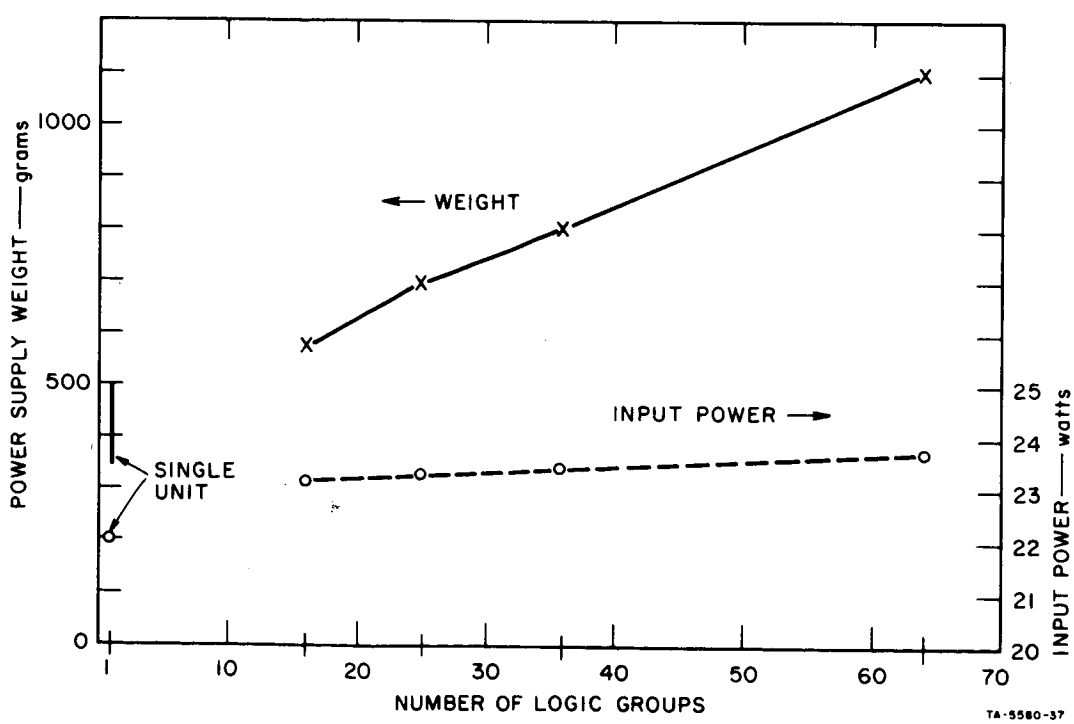


FIG. B-3 WEIGHT AND POWER REQUIREMENTS FOR 20-WATT DISTRIBUTED SUPPLY

Note that the small supplies can operate efficiently with a high oscillation frequency (50 kHz or more), allowing the use of very small, light transformers and smaller filter capacitors. With the possible exception of the series regulator transistor, power dissipation is low enough so that integrated circuits could be used for control and reference amplifiers. Some integrated circuits are already available for this purpose.[339]

8. Conclusions

The distributed power-supply notion can be applied to the design of a system of "self-powered" logic groups using individual dc to dc converters with very slightly more power loss than conventional single-supply systems. The weight, including on/off control circuitry, should be only about twice the weight of a single-unit supply, since the chop frequency of the individual units is higher than for the large (single) unit.

The design of distributed power-supply systems is closely allied with the design of the logic groups themselves. Unless the data-interface circuits are ac coupled, slight differences in the relative voltages of the supplies will cause loss of operating margins for data signals between logic groups.

The average power consumption of a distributed power-supply computing system can be reduced by turning off unused units. On/off cycling should not present a serious reliability problem.

Although the work we have done deals mainly with dc to dc converters, and indicates that a workable system could be made with such units, other possibilities such as ac distribution systems should be investigated. Either sine-wave or pulse-waveform ac systems could be used, but a careful comparison needs to be made between dc, sine-wave, and pulse-waveform systems. The desirability of using rough preregulation of the raw power source also needs attention.

The use of a magnetic core for memory in the on/off switches for power supply is feasible, but perhaps the possibility of using a thin-film memory element should be investigated, since thin-film elements would allow construction of small, reliable switches for power-control purposes.

Appendix C

APPLICATION OF MAGNETIC LOGIC

Appendix C

APPLICATION OF MAGNETIC LOGIC

1. Introduction

In this section of the report we give the results of a brief survey and evaluation effort that we have undertaken to ascertain the role, if any, that magnetic logic should have in future spaceborne computers employing integrated semiconductor circuits. We think it is important to conduct such a survey in a program aimed at ultrareliability because digital magnetic elements have proven to be highly reliable in their application to memories and logic. To anticipate our conclusions, we find that it is feasible to use magnetic circuits in conjunction with integrated semiconductor circuits in a high-speed (MHz region) ultrareliable computer to effect a substantial increase in the overall system reliability. We reach this conclusion chiefly on the basis (1) that magnetic elements per se are extremely reliable, and (2) the speed of operation of magnetic logic circuits is adequate for performing certain functions. By way of contrast to this general conclusion, an example is described wherein an attempt to apply magnetic logic circuits leads to a questionable increase in reliability.

In addition to the reliability of magnetics, there are other characteristics that make their use attractive. Magnetic circuits provide nonvolatile information processing with no standby power; they are essentially immune to many types of noise; and they are highly resistant to all nuclear-radiation components, both steady-state and transient.

One approach to the use of magnetics in future space programs is to follow the past pattern of decentralization of functions rather than using a complex centralized computer. In such an approach magnetics are used in systems (sub-systems) that are essentially separate from the systems that use integrated semiconductors. A magnetic programmer,

a time-sequencer, and an A/D converter* are examples of separate magnetic systems. While this is a concept that bears serious consideration, it is not one that we shall deal with here.

In what follows, there is a brief discussion of the reliability of magnetics, and then possible approaches to applying magnetics are discussed. We have categorized these approaches according to the kinds of functions magnetics can perform in conjunction with high-speed integrated semiconductor circuits.

The first category is that of monitoring the performance of the semiconductor circuits to determine when (and perhaps what) corrective measures are required. Two types of monitors are discussed: one is a sophisticated current meter and one performs digital operations that are the same as certain portions of the semiconductor unit.

The second category we discuss is that of magnetic switches. Several different types and functions of switches are included.

The third and last category is that of a "hard-core" backup control. In certain of these backup-control schemes the nonvolatile characteristics of magnetics is essential. All of the schemes rely on the long life of magnetics.

2. Reliability of Magnetics

The basic premise that motivated this survey is that magnetic elements are ultrareliable. As a part of a recent SRI project for NASA an attempt was made to determine a reliability figure for magnetic ferrite elements.[122] The conclusion was that ferrite cores do not fail in service, so long as they are operated within their physical limits, even when operating at temperatures up to at least 250°C.† The reason

---

* For one published example of a magnetic control unit, see Ref. 277.

† More specifically, this conclusion is for manganese-magnesium ferrite cores. Most square-loop ferrite cores are of this type and hence failure information (really the lack of it) could be gathered only on this type of ferrite. There is no known failure mechanism for these cores.

that no quantitative reliability figure has been established is that there is no failure data. One method that has been used is to estimate the number of cores used in memories that a particular company has produced, and then estimate the number of failure-free operating hours for these cores. This number of core-hours is then taken to be the mean-time-between-failures. For ferrite cores the MTBF has been variously estimated to be between 10^{10} and 10^{12} hours (i.e., 10^{-10} to 10^{-12} failures/hour).[340,294]

A reliability analysis of a system that uses magnetics must include the reliability of windings, connections, and associated nonmagnetic components. It appears that it is really these parts that determine the reliability of a magnetic system. As a part of the NASA work mentioned above, a reliability analysis was made of a magnetically implemented digital system.[124] For this particular purpose the following failure rates were supplied by Langley Research Center:

Part	Failures per 10^8 Hours*
transistor (discrete part)	2
ferrite core (wound)	0.01 (assumed value for worst case)
solder joints (inspectable)	0.01

Note that the failure rate for the ferrite core was supplied as a worst-case value, and is consistent with the core-failure data above. We see from this table that inspectable solder joints are highly reliable--this is important in a core-wire system. Furthermore, redundant joints could be used to further increase reliability if necessary.[166,346]

It is instructive to compare the failure rate of ferrite cores with that of integrated semiconductor circuits; the latter may be as low as 2 failures per 10^8 hours. This number comes from a recent survey conducted by TRW Systems Inc., in which many users of integrated circuits were

* Rates are based on high component reliability employing 100-percent screening for known weaknesses, approved derating policies (stress level assumed to be 50 percent), and approved fabrication techniques. Failure rates correspond to 65°C maximum ambient and 15°C temperature rise for part.

contacted. The average failure rate assigned to integrated-circuit devices that are 100 percent screened is 7 failures per 10^8 hours, and the lowest rate reported is 2 failures per 10^8 hours. We estimate that a typical integrated circuit to which these rates are applicable comprises about 30 components. (In this context it is important to note that a magnetic implementation of a function often requires fewer components than does a semiconductor version.)

A comparison of these failure rates indicates that a single ferrite toroid may have from 2 to 5 orders of magnitude lower failure rate than an integrated circuit. We recognize that a comparison such as we are making here is open to question from several points of view. Nevertheless, the large difference in the failure rates ascribed to the ferrite core and an integrated circuit are significant and support the contention that magnetics should be considered for ultrareliable applications.

3. A Magnetic Monitor Concept

a. A Metering Monitor

One method of using reliable magnetic circuits to increase system reliability is to use magnetics to monitor critical electrical variables within the main computer, such as power supplies and high-current pulse drivers.

The nature of the monitor and the functions it performs can conceptually take several forms. One possibility is to use a magnetic instrumentation system--a metering monitor--that would measure voltages and currents and give an electrical signal indication when an out-of-specification condition is detected. The voltages and currents could be dc, pulse, or both. That such a magnetic system is feasible has recently been demonstrated by the development of a magnetic telemetry system[122,123] wherein currents from sensors are digitized and, in effect, commutated.* Currents as low as 10 microamperes and as high as one-quarter

* The sensor currents per se are not commutated in this system and no analog amplification is required. This magnetic implementation of telemetry does not follow the standard organization for such a system.

ampere can be measured with 1-percent accuracy. Voltages are measured by determining the value of the current flowing through a known resistance.

b. An Information-Sampling Monitor

Another possible way to use magnetic circuitry to monitor performance is to detect errors by processing selected data in parallel with the main computer. In the event that an error is present in the sampled block of information, this error will propagate through the high-speed system until the magnetic unit has completed its testing of the block. A means of stepping backward in the program and starting the processing again after the equipment fault has been corrected is implied in this concept. Alternatively, the system must tolerate the incorrect processing that occurs during the error-detection interval.

A variation of the sampling technique is to generate error-detecting codes in the fast processor and use the magnetic processor as reliable error detector, e.g., a parity or arithmetic-code checker. If the speed of such checking is too slow, the magnetic circuits may be used for a delayed-output verification of a high-speed checker.

4. Implementation of Magnetic Monitor

There are three parts into which the implementation of an information-sampling magnetic monitor can be logically divided: input and output buffers, a comparing circuit, and the processor. The processor is the most complex portion of the magnetic unit. There are several possible magnetic approaches to implementing the processor; these can be categorized into semiconductor-magnetic and all-magnetic logic schemes. Typically, semiconductors are used with both types of logic schemes, but in the case of the all-magnetic logic the semiconductors are used only to generage clock pulses. It is possible, however, to generate clock pulses from an ac source (e.g., a sine or square wave) without using semiconductors.[123,17] Because the intent here is to achieve ultrareliable operation, the use of a clock-pulse source that does not require semiconductors should be considered. The semiconductor-magnetic logic schemes are usually faster than the all-magnetic logic schemes, and within the all-magnetic logic

category the nonresistance schemes are usually faster than the resistance schemes. Resistance schemes generally have greater tolerance to temperature and drive variations than do the nonresistance schemes.[26]

In this survey we cannot go in depth into the characteristics and attributes of the various technologies that are applicable to the magnetic processor. However, the following magnetic digital systems represent technologies that may be applicable to the processor: Sperry Gyroscope Magloc Computer;[338,267] Burroughs D210 Magnetic Computer;[319,340] Univac Ferractor-type Magnetic Computer[30] (principally of historical interest); Di/An Controls Core-Transistor-Logic systems;[166,342] Stanford Research Institute MAD Feasibility Machine;[51,52] IBM Flux Logic Evaluation Assembly (FLEA);[347,348] Stanford Research Institute Magnetic Versatile Information Corrector (MAVERIC);[72] Stanford Research Institute Atomic Reactor Control Module;[48] and Bell Telephone Laboratories Magnetic Stored Program Computer.[208] Because of the speed limitations of magnetic circuits, it is pertinent to note that the Sperry Magloc Computer has a 300 kHz bit rate, an addition time of 86 μs, and a multiplication time of 3.87 ms for 24 bits. The Burroughs D210 has a 100 kHz bit rate, an addition time of 30 μs and a multiply time of 570 μs for 24 bits. For the benefit of the reader who is interested in the details of magnetic techniques, we have listed bibliographies on magnetics as Refs. 27, 120, 230, and 19.

In addition to the low-speed processor, the information-sampling magnetic monitor comprises a comparing circuit, input and output buffers, sampling gates, and synchronizing circuitry. These do not appear to be difficult to implement in a manner that will be compatible with the technology used in the magnetic processor. However, in the joining of the integrated semiconductor units and magnetic units, the interface problem must be carefully considered. It is probable that signal amplification will be necessary in order to drive the magnetics from the semiconductor units. If the number of transistor amplifiers becomes a significant number when compared to the number of components in the processor, then the reliability increase achieved by a magnetic implementation over an integrated semiconductor version can become small or nonexistant. Another way of

saying this is that the magnetic-monitor approach can add significantly to the system reliability if the functions performed by the processors are moderately complex. (This aspect is also discussed in the section of interconnection switches.)

5. Magnetic Switches

The function that we envision for a magnetic switch is to direct power or information (data) to or from a module on command of an electrical signal, e.g., a signal from a magnetic monitor. Magnetic switches are attractive because they offer high reliability, dc isolation (for noise immunity and for coupling between subsystems that have different signal levels), low susceptibility to noise from external sources, and nonvolatile storage of information. In the case of an information switch the power level will be low and the rate of change of the information could be in the low MHz region. For the power switch the typical power level will be higher than for the information switch, and the frequency will be in the low or tens of kHz range. For both types of switches, low speed in the switchover from one module to another one is acceptable, and switching will be required only when a malfunction has developed in the computer.

The switches that are described below are of both the information and power types. With one exception, the investigation to date indicates that these switches merit further investigation.

a. Converging Switch

The converging switch is the name given to a unit that gives access to one of N information sources from any one of K data processors.[32] The processors supply the switch with the address of an information source. The converging switch contains address-decoding circuits and magnetic-gating structures. The magnetic-gating structure is a multipath ferrite device that operates on the balanced-circuit principle. An information signal upsets the balanced condition to store or transfer a logic 1. The particular device reported in Ref. 32 uses a low coercive ferrite material (between 0.15 and 0.2 oersteds) and requires 30 mA to upset the balance. An early production model produced by Western Electric connects to 128

sources, each with a 26-bit capacity; it has an access time of 1 μs and a cycle time of 2 μs. The number of semiconductors required for the switch is not reported. The converging switch was originally developed for a missile application and is now being used in a modified form within the Bell System.

b. Interconnection Switch

This switch exists as a paper design and is the result of work undertaken by SRI for NASA.[314] The switch is part of a redundant computer that is implemented by integrated semiconductor circuits. The interconnection switch is used in two different places. In the first application, one of a set of three arithmetic units is connected to a processor, and in the second application one of a set of three memories is connected to the processor. Information flows through the switch in both directions at a 1 MHz rate; voting also takes place in the switch. Toroids and multiaperture devices are used in the switch implementation. The investigation of this switch revealed that it was possible to perform the necessary functions magnetically, but there are undesirable features. Perhaps the major problem lies in the fact that a transistor driver is required at each input terminal to the magnetic unit in order to increase the signal power level sufficiently. This results in a large number of added semiconductors. Another major problem arises from operating at 1 MHz--the power dissipation is high. (By way of reference, it takes about 0.1 watt to switch a 30/50 memory core at 1 MHz.) The attributes of magnetic implementation of this switch as compared to a semiconductor version of the same switch are summarized in the table below.

Advantages of Magnetic Version	Liabilities of Magnetic Version
Reliability is increased somewhat	Number of semiconductors used is between 1/2 and equal to number required in an all-semiconductor version
Has nonvolatile characteristic	Power is greater (10 times at 1 MHz, equal at 100 kHz)
	Weight and volume are greater

This switch is an illustration of an application where the use of a magnetic implementation is questionable. The functions required of the magnetics are quite simple, and the interface between the integrated circuits and the magnetics therefore becomes a major problem.

It is apparent from this example that if a block of magnetic circuits is interposed between blocks of integrated semiconductor circuitry, the functions required of the magnetics should be at least as complex as those that could be performed by the equipment in the interface circuits. An exception to this generalization arises if the power level of the integrated circuits is adequate to drive the magnetics directly without added amplifiers. Balanced magnetic circuits have reportedly operated with an input current of only 30 mA (but probably at less than 1 MHz).[32] It is also worth noting that 30 mA is about the amount of current that is required to "tip" the flux in coherent-rotation switching in a metallic thin film.[113] Since it is possible to get 30 mA (and more) from integrated circuits that include a line driver as one of the available units, these integrated-circuit units could be used to drive magnetic logic circuits. (These units have an amplifying junction that is on the same silicon chip as other junctions and components.) The reliability of an integrated-circuit system is more a function of the number of chips than of the number of junctions, so it may be possible in some systems to incorporate a line driver on the chip without reducing the reliability of the integrated-circuit portion of the system. This would mean that the semiconductor-magnetic interface would be effected without a reliability penalty at this point in the system.

It should be pointed out that the balanced magnetic scheme cited above is a bipolar scheme that equates a logic 1 with a particular polarity of output signal and a logic 0 with the opposite polarity signal. This bipolar characteristic gives rise to some problems; for example, it cannot be used as a switch to connect and disconnect a pulse train. Additionally, the "balanced" feature of the scheme means that a clock source switches flux repeatedly in a magnetic structure, irrespective of the logic state of the circuit. This can mean high power consumption.

The problem of the semiconductor-magnetic interface should be further investigated.

c. Data-Path Switch

The switch to be described has certain interesting features of dc transmissions, with dc isolation. Figure C-1 is a sketch of the switch.

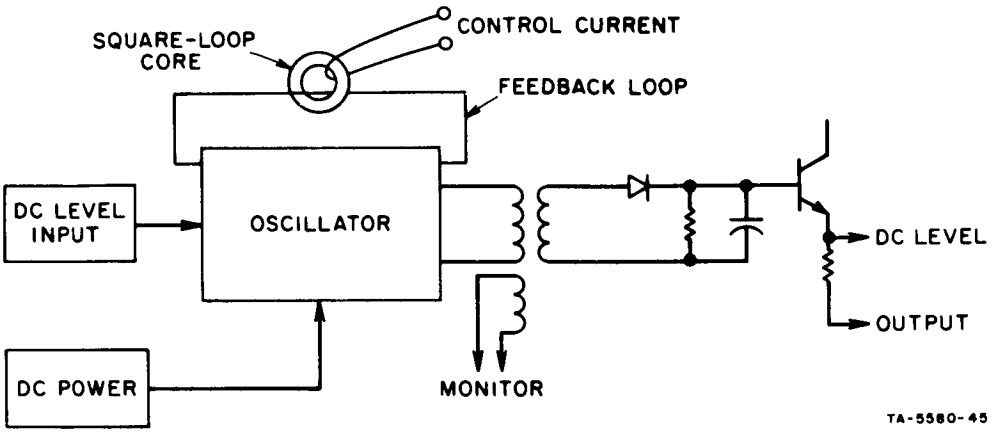


FIG. C-1 DATA-PATH SWITCH

In this switch the oscillator is powered from the same module that supplies the input data to the switch, and the output transistor is powered by the module that receives the data. When an input signal is received the circuit will oscillate if the feedback conditions are correct. The oscillator output is coupled through a transformer and, upon rectification, it becomes the input to the following stage. Units operating in a manner similar to that just described are available from Dynamics Instrumentation Company (floating digital drivers) and from Radiation Incorporated (modular solid-stage telegraph relay). In the data-path switch shown above we have added to the circuit a magnetic core that controls the feedback of the oscillator. The state of the magnetic core determines whether or not the circuit will oscillate upon receipt of an input signal; therefore, the switch can be activated under logic control. This means the switch could be used in redundant systems to switch modules in and out of the system upon receipt of an electrical

signal. The details of this switch have not been worked out, but we think that it is feasible and potentially useful for reliable computer applications and that it bears further consideration. The advantages we see for such a switch are the following.

(1) Dc coupling is achieved so that logic levels, rather than pulses, are transmitted.

(2) Isolation is complete--there are no grounding problems.

(3) There are no power-supply interaction problems.

(4) Low-power, coincident-current setting of the magnetic element is possible because only feedback power is controlled.

(5) The state of the switch (on or off) is nonvolatile (with power failure) because of the magnetic element.

(6) It is easy to monitor data flow by adding a winding to the output transformer, or by coupling through a small capacitor.

(7) The geometry of the magnetic control element is simple; that is, a toroid is adequate and a multipath device is not required.

Another switching method that has merit for data-path applications makes use of a multiaperture magnetic element (a MAD). Only a single highly reliable oscillator is required for the entire system for this switch implementation; however, an amplifier is required between the input to the MAD and the integrated semiconductor circuitry. The required dc output is obtained by rectification as in Fig. C-1.

d. Power Switching

In a redundant system we believe it is beneficial to switch power on and off to individual modules for several reasons: (1) it simplifies information switching; (2) it simplifies testing; and (3) it permits a reduction in operating power. It may be possible to replace information-path switching with power-supply switching. Magnetic technology is an attractive candidate here because of its reliability and nonvolatility. If all the power goes off and is subsequently restored, memory of the information states at the subsystem level immediately preceeding the failure will be valuable information.

Since a magnetic core responds to alternating current, magnetic implementation of a power switch means switching ac power. For this reason the only real differences between a magnetic switch for power and a switch for data pulses is the power level and frequency. A module in a computer typically requires dc power, so using a magnetic switch requires that rectification be carried out in each module. This rectification requirement is consistent with the concept of individual power supplies for modules rather than one big supply for the entire system. This concept is discussed in Appendix B of this report.

Because of the similarity between an information-flow switch and a power switch when magnetic implementation is employed, the data-path switch described above is a candidate for an ac power switch. Likewise, the MADs that are used in the interconnection switch and in one version of the data-path switch can be used for controlling the flow of ac power. The suitability of these approaches has not been evaluated in detail in this survey.

A switching method that was not discussed as an information-flow switch is that of a ferroresonant circuit.[292] This type of nonlinear, bistable circuit relies upon the change in inductance of a reactor as it is driven into and out of the saturation region. There are two characteristics of these circuits that lead us to consider them for power switching rather than information switching: (1) they are capable of handling reasonably large amounts of power, and (2) they control a continuous ac wave rather than pulses.

A particular ferroresonant circuit is shown in Fig. C-2.[135] When T_1 is saturated the inductance of the winding L_1 has just the right value to resonate with capacitor C_1 at the applied carrier frequency. The same is true for T_2, L_2, and C_2; however, only one of the series circuits can be resonant at a given point in time. If both circuits were to be resonant simultaneously then the voltage drop across C_c would increase beyond the value it has when a single branch is resonant. With this increased voltage drop, the current magnitude possible in the two branches is not sufficient to maintain both T_1 and T_2 in saturation. Therefore

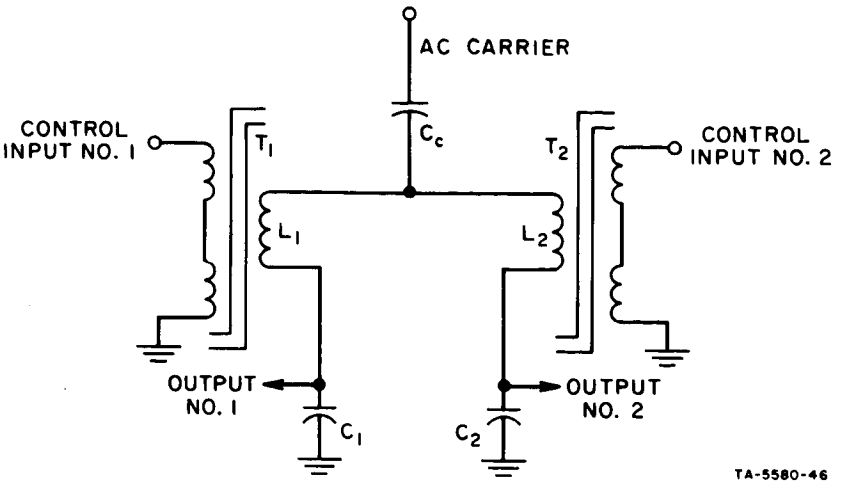


FIG. C-2 FERRORESONANT SWITCH CIRCUIT

only one branch can be resonant and drawing a large current. The other branch draws only a small current. Output power is obtained across C_1 or C_2.

Ferroresonant circuits have been the basis for digital circuits such as flip-flops and shift registers, and have been used in steady-state ac circuits as voltage regulators.

Up to this point we have considered using the ac power switch as a means for controlling the dc power supplied to a module. An additional possibility is to use such a switch to control the distribution of clock power to the various modules. The feasibility and utility of a magnetic clock switch needs to be further investigated, as does the possibility of applying the principle of ferroresonance to ac power switching.

6. Backup Control

There are two types of backup control systems that we point out here: one that is used to re-initiate operation of the integrated-semiconductor computer after it has become inoperative, and one where certain portions of the semiconductor computer are replaced by magnetic

units in the event of semiconductor failure. The purpose of the first type of backup is to protect against a power failure that shuts down the entire computer. When such a failure occurs but is not permanent, it then becomes important to reestablish the operation of the computer when system power is restored. It is not sufficient to simply allow the power to be reapplied to the semiconductor logic circuits, because the logic state that the flip-flops would assume is indeterminate. Even if the flip-flops did assume predetermined states it would be necessary to execute certain functions to reach the desired position in a program and initiate the proper mode of operation. An ultrareliable magnetic backup unit could be beneficially used to reinitiate operation of the semiconductor circuits when power is restored. The backup unit would execute certain primitive functions--e.g., control processor flip-flops would be cleared to a reference state (that may be dependent upon the point in real time at which failure occurred), and a program sequence would be started. The return of power to the system would in itself provide the input signal required to bring the magnetic unit out of its passive state and give it control of the entire computer. After proper operation conditions were established, the magnetic backup unit would be "locked out" of operation except for periodic updating. While the magnetic unit was locked out, the operating speed of the computer would not be impaired. In this type of backup the nonvolatile characteristic of magnetics is an essential ingredient.

A backup control unit of this type could also be used to protect against transient failures other than those of the system power supply. If a transient caused malfunction of large segments of the computer so that the utility of the computer as a whole were impaired, then the backup could reestablish proper operation. In this mode of operation the backup unit would be given control of the computer upon receipt of a signal other than power supply turn-on.

In the second type of backup, upon command, a magnetic unit permanently replaces a faulty semiconductor unit, such as a control sequencer or program counter, at a sacrifice in operating speed and possibly with a reduction in the type of functions that can be performed. This type of control unit can be updated periodically, like the one described above, to prepare it for operation on demand.

An interesting variation of this second type is to duplicate, with a magnetic computer, all computations of the semiconductor computer that are critical for the mission. This redundant computer could operate in parallel, or it could be started when failure occurs in the main computer. Alternatively, the concept of a redundant magnetic computer could be reserved for missions of very long duration; use of the magnetic unit instead of the semiconductor computer could save an appreciable amount of power in the phase of the mission where the vehicle is at a great distance from earth and high-speed computation is not essential.

The two types of backup controls described above are applicable at the system level. The principle of magnetic backup can also be applied at the circuit level. At this level magnetic circuitry could be switched in and out as it is at the system level, but this would probably result in too complex a switching system to actually increase system reliability. A better method of supplying a backup at the circuit level is to use the magnetic elements in an intimate mix with the semiconductors. We propose, for example, in conjunction with certain critical semiconductor flip-flop circuits, that toroids be set and reset according to the state of the flip-flop. The primary purpose of the toroids is to remember the state of the flip-flop and maintain this information in the event of power failure. The toroids are not essential to the circuit operation except for this nonvolatile characteristic. A circuit that does operate in this manner at modest speeds has been reported by Harry Diamond Laboratories.[199] An alternate method of providing the same nonvolatile feature is to use a portion of the main memory to store the state of the flip-flops. There are advantages and disadvantages to both approaches that need to be further evaluated.

7. Conclusions and Recommendations

The reliability of magnetic-logic circuits is superior to that of integrated-semiconductor circuits by several orders of magnitude, and therefore magnetic logic should be applied to future ultrareliable spaceborne computers. The high reliability of magnetic-logic systems derives from the fact that ferrite-core failures are unknown and the fact that inspectable solder joints are very reliable. In addition to long life, magnetics have other characteristics that are important in ultra-reliable systems: they require zero standby power and are therefore nonvolatile, they are immune to most kinds of noise, and they are radiation-tolerant.

The operating speed of magnetic circuits is a restriction upon their general application in the megahertz region. However, this brief survey indicates that there are substantial areas where magnetics can and should be used in conjunction with high-speed integrated-semiconductor circuits. We have discussed several different methods of applying magnetics in concept, and have given an indication of the kinds of schemes and circuits that can be used.

We recommend that the following steps be taken:

(1) Further effort should be devoted to strengthening the concepts presented here, and analytical and experimental work should be carried out to support or refute the ideas and circuits discussed.

(2) Because this survey is admittedly incomplete in scope and depth, a continuing effort of this nature is recommended.

(3) In support of (1), we recommend that one or two specific concepts be selected for a detailed design feasibility study. Such a study should include circuit-level problems of speed, synchronization, power required, and interface compatibility with integrated-semiconductor circuits.

Appendix D

A SURVEY OF THE PUBLISHED LITERATURE ON THE ATTAINMENT OF RELIABLE SYSTEMS THROUGH THE USE OF REDUNDANCY

Appendix D

A SURVEY OF THE PUBLISHED LITERATURE ON THE ATTAINMENT OF RELIABLE SYSTEMS THROUGH THE USE OF REDUNDANCY

1. Introduction

The primary thesis of this program for developing means for achieving ultrareliable spaceborne computers has been that such systems can only be achieved through the judicious use of redundancy. While it is of course essential that component reliability be as high as possible, and that all elements be operated well within the physical tolerances that guarantee their continued operation, it is also necessary to provide backup facilities that allow systems to tolerate internal failures, whether transient or permanent, so that the computational mission can be successfully carried out. For as the complexity of computer systems increases--as represented most obviously by the enormous increases in the number of components required--almost any level of guaranteed reliability of individual elements becomes insufficient to provide a satisfactory probability of successful mission completion. These observations are particularly pertinent in the case of extended spaceborne missions where the possibility of unprogrammed maintenance and inspection routines is severely limited, and where successful use of the radio link for such activities cannot be successfully carried out unless careful anticipatory provisions have been made concerning the types of spares to be installed and the interconnecting links for installing and removing them on the detection of a fault.

These arguments are not new with this program, of course, and their general validity has been recognized for at least a decade. As a result, a great deal of effort has been expended in understanding just how redundancy can be "judiciously" applied to systems, that is in such a fashion that the overall reliability is actually increased. A not insignificant factor that has compounded the problem of evaluating such systems has been the paucity of good analytical tools for actually calculating the reliability of complex configurations, or even of providing good lower

bounds so that overall mission probabilities can be estimated. Finally, a new technological factor has appeared that is effectively changing the rules of the game--we refer to the imminent widespread availability of extremely reliable, very small, batch-fabricated elements having extremely low power dissipation. The availability of such elements implies that the number of components involved is not the critical factor in measuring system cost--whether from energy, volume, or weight points of view--and makes it possible to seriously consider large ratios of redundancy, if indeed the resultant increase in reliability of the overall system can be demonstrated.

As a result of the unquestioned relevance of redundancy techniques in the construction of extremely reliable systems, the technical literature is replete with reported activities that attempt to cover one aspect or another of the large area. An important aspect of this program, then, has been to instigate a general and continuing appraisal and review of these activities that have been reported and are available to us. Much of the literature in this field has not been of a high technical quality, and much of it is no longer relevant to today's technology. Hence, in making a survey of activities in reliability theory it is necessary to prune away much material in order to highlight the efforts that do seem to be important.

Our goal in this section, then, is to present a critical and selective survey of the literature that is relevant to the attainment of reliable networks and systems through the judicious use of redundant structures. In the following section we attempt to restate briefly the point of view which leads to the particular categorization of topics that we have chosen. Our concluding section contains a brief discussion of the various activities that have been reported in the different categories of reliability technology.

2. Summary of Subject Areas

a. Overview

The intention of this survey is to provide the interested reader with an introduction to the literature on redundancy techniques by several means. First of all, we provide an outline, or categorization of topics, which serves to partition the field into the various technical areas against which the reader may sharpen his own perception and conclusions regarding the valid lines of technical inquiry. Admittedly, the outline we have chosen is one relevant to the concepts of modularity and reconfigurability that we have regarded as essential to the program addressed in the main body of this report.

Secondly, within this outline of topics we make reference to specific, selected articles that are in general easily available to the researcher;* in addition, we briefly summarize their contribution to the technology.

The conclusions of the report (Sec. IV) essentially reflect the conclusions of this literature compilation. Although valuable technical contributions have been made in the development of systems that are more reliable through the application of redundancy techniques, and although these contributions are increasing in number and quality, it is clear from a survey of the published literature that much remains to be done before efficient and demonstrably reliable systems can be realized. Many unsolved problems were uncovered as a result of this survey; a sampling of these is contained in the listing of recommendations for future research--Sec. IV.B.

---

* Included in the referenced articles are results from an earlier survey of Soviet activities in the field of reliability theory.[286] This survey, in the form of a preliminary draft, was presented at the Workshop on the Organization of Reliable Automata earlier this year. It was supported partially by this program, and partially by the Air Force Cambridge Research Laboratories. Since the Soviets have also been quite productive in addressing themselves to these technical areas; and since most of the work is available in translation to researchers in the United States; and finally, since work in this country and in the Soviet Union represents the overwhelming majority of work done anywhere in these subjects, the Soviet references have been freely included in this survey whenever appropriate from the technical point of view.

It will be noted that almost all of the referenced articles have appeared within the last ten years, and the great preponderance of them within the past four years. This results partly from the selection process, of course, but mainly occurs because that is simply the way the density of publication has taken place--and it is still on an upward slope.

Finally, it should be noted that several rather comprehensive bibliographies on reliability topics have appeared, and these are appropriately referenced in their place below. It should be emphasized here, though, that no attempt has been made here to supplant these bibliographies in terms of comprehensiveness, although, of course, we shall make note of some entries that appeared subsequent to their publication. Our goal has been strictly to provide a selective reference to the literature, from which the reader can proceed to his own ends.

b. Categorization of Subject Areas

In this program, redundant, modular, highly reconfigurable systems have been identified as the basic organizational structure that holds the most promise for the successful attainment of the mission objectives of spaceborne computers. Accordingly, this guide to the literature is structured to partition the referenced papers in a way that best serves this point of view. Given a module that is a subsystem within a larger complex, we have pointed out that there are two fundamental ways in which redundancy can play an integral part in its functioning within the system. These two ways are differentiated by the role played by the terminals of the module. In static redundancy techniques, faults are accommodated within the module itself (e.g., by fault masking) and the terminal activity is unaffected. In dynamic techniques, terminal activity plays an essential role (involving fault detection, diagnosis, and the resultant reconfiguration). These two basic categories are reflected in Secs. 4 and 5 in the discussion below, and all the other categories are, in a sense, supplementary to them.

Thus the introductory Sec. 3, below, is concerned with papers of a general nature. They involve either arguments supporting the need for

redundancy, or tutorial or survey papers on the subject, or developments in general reliability theory, including the calculation of the probability of failure, or pertinent papers on the characteristics of specific components. Also included are references to the several rather extensive bibliographies that have appeared and are easily available to the reader.

The application of redundancy to other than computing networks and subsystems (e.g., power supplies), as well as papers concerned with environmental aspects (e.g., additional weight requirements of redundant systems), is briefly reported on in Sec. 6.*

3. Discussions of General Background

a. On the Need for Reliable Systems

The problem of achieving reliable systems in the face of today's severe mission requirements has been well recognized in the published literature. We mention several of these articles in order to set the stage for the survey of the technical contributions that have been made.

In the first place, it is clear that the use of redundant structures is but the last step in a hierarchy of measures that can be taken to attain fault-free operation. It has been pointed out that increased measures to achieve reliability are necessary in all stages of system life, from the original design to the final installation and subsequent maintenance of the system.[130] Certainly it is necessary that the component reliability be as high as possible, that adequate attention be given to the details of fabrication and assembly of equipments, and that circuit designers carefully recognize the existence of tolerances and the need for conservative designs.[37] Nonetheless, in systems consisting of thousands of elements, it is necessary to provide also for alternative

---

* All of the reference numbers in this Appendix are keyed into the common reference list that serves the entire report. This reference list, in turn, has been alphabetically ordered so that it may serve a separate function as a selected bibliography to the subject (although it has not been supplemented as a bibliography--nonreferenced articles do not appear in it).

system responses to accommodate to the almost certain event that an unknowable number of components will fail before the end of the desired life of the equipment--indeed, the likelihood is high that some failures will occur before initial turn-on of power.

Furthermore the point has been made that the failure mechanisms in modern solid-state components are such that periodic replacement and testing simply do not make sense any more.[8] The reason for this is that it is asserted that the state of the art has advanced to the point where the few failures that do occur are purely random in nature. Present component lifetimes are on the order of one failure per 10^{10} component hours, and all identifiable causes of trouble have been eliminated to the point where it is impossible to obtain adequate life-test data on what the characteristics of the components truly are.[8] From these facts it is reasoned that only redundant structures can be effective in markedly increasing mission time, and even in deriving bounds that are useful in estimating what the effectiveness time actually is.

Of course, the mission time is increased if it can be guaranteed that all of the elements in a redundant system are initially working properly, and the initial testing of such systems that are specifically designed to ignore the occurrence of faults poses some problems. This problem has been considered by Masters[202] who asserts that it may not be necessary to know that all equipment is faultless initially, but only that enough of it is functioning to be able to assert something about the mission probability of success. In his discussion he develops some analytical formulations relevant to such estimates.

Redundant structures may be effective, of course, in either of the two fundamental ways that have been emphasized in this report. One way is through the provision of auxiliary networks and schemes for the detection and diagnosis of faults, with the implication that subsequent manual or automatic action will be taken to replace the failed part. The other way is to design the original networks so that a certain class of faults can occur without affecting the input-output relations, i.e., by utilizing what we have called static redundancy techniques. Both of these techniques may play a role in the same system.

In the first method, involving the use of dynamic redundancy techniques, it is not at all clear just to what degree the human operator can effectively play his role in the closed system. In unmanned space vehicles his role is necessarily no more than performing analysis on the basis of telemetered data (over noisy channels), followed by appropriate action signalled over the same channel in order to initiate corrective procedures that must be implemented within the craft itself. Even in the case of manned vehicles, the need for automated repair on spaceborne vehicles seems substantiated. In a recent interview with Roger Chaffee, one of the astronauts in the Apollo program, it was claimed that all of the stabilization, control and communications electronics utilize redundancy, usually a switchable redundancy at a rather high level.[116] When asked what role he thought in-flight maintenance could play--that is, whether the man on board could actually try to fix a detected fault--Chaffee's answer was direct: "Not in the electronics. As you know, these systems are pretty complex...and there are quite a few integrated circuits...."

On the other hand, even though there may be some argument that the man may accomplish some of the physical aspects of repair, e.g., by adjusting control switches, there seems to be general agreement that he can play almost no role in the analytical functions that are required to detect and locate troubles that may occur in the spacecraft electronics. Indeed, even the possibility of using plug-in modules has been questioned because of the unreliability of connectors.[3] In the same reference it is asserted that much has already been done in the provision of automatic malfunction detection and switchover logic--although so far mainly in the control of relatively large subsystems, and certainly not on the small module or component level.

Thus, at least in the Apollo program, the concept of manual in-flight maintenance seems to be discarded as too demanding on the astronaut's time, and has been replaced by an automatic self-repair approach that uses a dual form of redundancy. In these dual circuits, a fault-detection unit continuously monitors the operational unit and automatically switches power to the auxiliary unit whenever a fault occurs.[129]

We shall not attempt to document similar problems that still pertain to ground-based systems and conventional general-purpose computer installations, except to note that they exist; nor shall we consider the application of redundancy to nondigital systems, e.g., continuous-control systems, except to note that similar kinds of redundant structures and approaches apply.[175] Even the Soviet Union, which hardly makes the activities of its space programs a part of the accessible literature, has admitted the general need for improved reliability in its main-line conventional computer systems. Indeed it has been admitted that some of their present computers, e.g., the BESM, STREL, and URAL computers, are simply not reliable enough.[191]

Thus it is clear that the problem of attaining reliable digital systems continues to be paramount in the tasks of designers of spaceborne equipment. A quotation from a recent book* on the multitude of factors involved in space exploration is appropriate: "Before the nearest stellar systems can be probed, one of two technological breakthroughs must take place. Either probe equipment, including electronics and power supplies, must be given lifetimes on the order of ten years, or self-diagnosing, self-repairing automata must be developed. Both avenues will certainly be attempted."

b. On the Analysis of Reliable Systems--Bibliographies

Both in this country and in the Soviet Union there has been a great deal of reported effort devoted strictly to the analytical problem of calculating the probability of failure of a given network configuration, given the failure probabilities of its component parts. These analyses vary according to the different assumptions made about the failure law, about the nature of the replacement-and-repair process, if any, and about the nature of the structural or time redundancy provided. With few exceptions, a fundamental shortcoming concerning these analyses is the basic assumption concerning the independence of events--that is, the assumption that one failure is in no way conditioned upon the occurrence of another.

---

* Ref. 49, p. 34.

There have also been a large number of survey papers which purport to provide fundamentals to anyone new to the field; some of them are quite good--including the several books that have been devoted exclusively to the subject of reliability and redundancy--and provide a rich trove for one who wants to delve into the subject for the first time, or for the specialist who wants to broaden his understanding of the field as a whole. Finally, there have been several good bibliographies published on the subject.

An interesting qualitative discussion of the improvement in computer reliability through redundancy techniques is found in a nontechnical summary by Pierce.[245] This article includes a discussion of many of the pertinent structures--such as restoring organs, threshold gates, vote taking, and the like--all in terms that can be easily understood by the nonspecialist. Herwald presents a concise summary of the state of reliability theory,[131] while Aroian[12] gives a summary of the basic formulas needed for the calculation of the reliability of redundant systems. A survey of the various redundancy techniques that have been proposed is given by Teoste,[300] including detailed descriptions of the mathematical models for estimating the reliability improvement and for comparing the relative advantages of the several techniques including Moore-Shannon, majority circuits, and other kinds of redundancy structures. He concludes that where it is applicable, the Moore-Shannon type of redundancy provides the most significant improvement. Other similar treatments are available.[53,266,209] In particular, Creasey's paper[53] includes a review of the role that can be played by the application of error-correcting codes, while Fedderson and Shershin[350] describe the problem of determining the optimum number of redundant elements when various restraining factors such as cost, weight, and volume are taken into account. A more advanced, theoretical treatise that surveys the mathematical models useful in solving reliability problems, but from a mathematical point of view, is found in Barlow and Proschan.[21] Also worth mentioning is the generally high-caliber collection edited by Wilcox and Mann,[329] which brings together a great many contributions to reliability theory, ranging from the quite scholarly to the eminently practical.

The purely statistical approach to reliability theory is summarized in the collection edited by Zelen[336] which cites many interesting problems and contains papers concerned with statistical models, maintenance and replacement policies, confidence limits, and the like. In particular, a very readable review of the literature is in the paper by Weiss,[325] including topological and time-dependent aspects of reliability models. The statistical basis for the exploration of redundancy systems is also treated by Moscowitz[219] and by Kuznetsov[173] who presents a method for determining the reliability of a system from the results on tests made on part of the system (applying standard statistical techniques to the assumed situation where complete testing is impractical because of the amount of test equipment required and the limited amount of time available). Drenick[67] introduces the notion of expected economic gain, which is a random function depending upon the number of failures and their times of occurrence, as well as upon the particular replacement policy. He also is concerned with the probability laws by which equipments fail[66] and purports to show that the time between failures tends toward an exponential distribution as the number of components grows large.

The relevance of these models depends strongly, of course, on the accuracy of the assumptions on which they are based. In an early paper, Creveling[54] points out, among other things, the need to design circuits so that the assumption of independence of faults is justified. Either this approach is necessary, or the statistical models must be made more complex; for, as Pollyak[249] demonstrates, an incorrect statistical-independence assumption will indeed lead to errors in calculating reliability. (He gives examples of calculations showing the effect upon series-connected and parallel-connected components in particular.)

Virene[317] has discussed this problem rather directly in his presentation of nonparametric life testing--that is, the use of statistics in which no assumption is made concerning the underlying distribution that characterizes the operating life of an equipment--although he warns of the possible losses of efficiency when such methods are used.

Many other discussions of the application and estimation of system reliability have appeared,[16,218,250] including attempts to relate the probability characteristics of system reliability to the detailed failure-distribution law of the parameters of the components[269] and discussions of the usefulness of statistical assumptions concerning independence, probability distributions, and the like.[222]

Most of the papers in this field give adequate reference to previous and related publications; this is certainly true of most of the documents cited above. In addition several lengthy bibliographies have appeared. Of these we mention Balaban[18] and certainly the useful assemblies by Jensen.[138,137]

c. Optimum Redundancy and Other Considerations

An area of inquiry that is closely related to the previously discussed papers concerned with the various aspects of general reliability theory, yet that asks a more system-oriented question, is that represented by a number of papers on determining the "optimum" value, or ratio, of redundancy. These usually take into consideration other environmental factors such as power consumption, volume, weight, etc., affected by the extra components in the redundant system. We have already mentioned one of these,[350] which attempts to formulate redundancy as a function of total costs, individual element probabilities, etc., for various abstract configurations. Not surprisingly, the techniques of dynamic programming are among the tools presented for the solution of such multiple-constraint problems.

An approach along these lines is given by Webster,[322] who bases his work on the theorem that making a low-reliability part redundant causes a larger numerical increase in reliability than making a high-reliability part redundant. Hence redundancy should be progressively applied, starting with the less reliable portions of the system, until some system constraint--such as power consumption--is exceeded. In a later paper[323] he demonstrates the practical application of these procedures to a system composed of 14 subsystems.

More abstract approaches to the question are presented by Barlow et al.[22] and by Pierce,[242] who advances a procedure for synthesizing a system to obtain the greatest reliability corresponding to a set of fixed costs, or alternatively to obtain a given, fixed reliability specification with the minimum set of costs. In a similar vein, Barlow and Hunter[20] obtain relations for determining the number of components that maximizes the expected life of a circuit, given an exponential failure law.

On a different question, Esary and Proschan[77] have considered the relation between the failure rate of a system of (identical) components and the failure rates of the components themselves. In particular, they have treated the important case of "k out of n" circuits--that is, circuits which function properly if any k of the n components are still functioning.

In yet another area, an interesting point of view is represented by the work of Merekin,[204] Malyugin,[193] and Muroga,[224] who show how to form expressions for the reliability of certain types of combinational circuits directly from their switching functions, without analysis of the circuits themselves. Gendler,[99] on the other hand, analyzes a very particular circuit logic structure by considering the probability that a given threshold function is indeed realized by a threshold device that suffers statistical variations in its weights and in its threshold.

4. Discussions of Static Redundancy Applications

a. Fault-masking Techniques

In a broad sense the terms "static redundancy" and "fault masking" may be considered synonomous; as pointed out in this report the basic concept embraced by both terms is that the redundancy is provided as an internal, integral, autonomous part of the network and operates without intervention through the input-output terminals--at least until the network fails completely because the number of faults has become too large to be covered by the masking provisions. In this sense it matters not what the form of the redundancy is--whether a multiple-line voting scheme, series-parallel configurations, an internal error-correcting code process,

or some other scheme. For the purpose of this exposition, however, we shall reserve the term "fault masking" for the obvious structural type of replication represented by voting schemes, for example. We shall treat the signal type of redundancy based upon coding theory in the following section. This is strictly for convenience, however, as close inspection reveals that the distinction is quite artificial. Furthermore, we shall separate the structural fault-masking techniques into two parts: voting schemes and nonvoting schemes, primarily because such a split roughly represents an equally weighted division of the work that has been done in the field.

1) Nonvoting Schemes

For our purposes, we shall consider the basic paper by Moore and Shannon[214] to be the starting point for nonvoting redundancy schemes. Here relay-contact networks are explicitly considered as the components of interest in primarily combinational network construction. It is shown that as long as the failures in contacts can be considered statistically independent, then arbitrarily reliable networks can be built regardless of how unreliable the individual contacts may be. The construction proceeds by an iterative process wherein the individual contacts of the number are replaced by a certain network of contacts. Moore and Shannon showed the number of contacts required to achieve a given reliability as a function of the individual relay characteristics. Kochen[165] has directly extended their results by showing that the required redundancy in such networks is also a function of the particular logic function being realized. Further extensions to more general types of networks, including the important "k-out-of-n" structures, have been made by Esary et al.,[76,29,78] who also consider the case where the components may have differing reliabilities. Many other papers have appeared that are concerned with various other extensions or special network properties.[172,62,118,132,234,111] Asymptotically, it turns out that the correction of any single fault in a contact network does not increase the complexity of the network; this is shown for the case of shorted contacts in a paper by Potapov and Yablonskii[252] and for contacts that fail by opening by Madatyan.[189] For two or more shorts or opens, however, the

asymptotic complexity of the network must increase. In a similar fashion, it can be shown that the asymptotic complexity of two-level rectifier gate circuits must increase with the number of faults to be tolerated.[227] Some economies can be achieved by taking advantage of "don't-care" conditions, however, as is the case with straightforward switching-function realizations. Dunning et al.[69] consider the specific logic-block case of general NOR-gate trees (including considerations and comparisons of quadding, voting, etc., in such networks), while Weinstock[324] has treated the mathematical problem of reducing arbitrary network structures to a series-parallel form resulting in systematic methods for deriving the reliability parameters of any network involving a flow of information between two terminals.

Another important core technique for implementing a nonvoting kind of redundancy is due to Tryon,[305] who introduced what he calls "quadded" logic--a circuit construct wherein the components appear in quadruplicate so that errors are corrected one or two levels downstream from their inception by a mixing of good signals from neighboring units. Many papers have been devoted to extending Tryon's original results (e.g. Ref. 139 for NOR-gate networks). Of especial note are the valuable extensions and generalizations developed by Pierce[244,243] in his "interwoven" logic, where Tryon's work was shown to apply to other than the AND/OR/NOT logic blocks, and to much more general patterns of correction. In some cases double, triple, and other errors can be corrected, as well as single errors.

Taking a quite different approach, but in a similar vein--namely, the replacement of each component by a network of components--Urbano[310,311,312] has investigated what he calls polyfunctional networks. These are an iterated network construction in which each component is replaced by a copy of the entire network, and the iteration may be carried on to any desired degree. Questions such as the stability of such networks, and their convergence to a fixed function set, are examined. Sethares[281] has also examined the characterization of functions in polyfunctional networks.

In yet another direction, Muchnik and Gindikin[220] have examined the question of whether the conditions which establish the completeness of a set of logical primitives need to be modified when some of the primitives are unreliable. They find that some modification is necessary, and determine the new conditions.

2) Voting Schemes

The large number of papers on vote-taking redundancy can be traced back to the fundamental paper of Von Neumann,[318] where multiple-line redundancy was first established as a mathematical reality for the provision of arbitrarily reliable systems. In this paper it was demonstrated that arbitrarily specified reliability could be achieved using unreliable components (of bounded unreliability, however). It was essentially a mathematician's answer to the question of whether redundancy could indeed pay off. A spate of engineering attempts have followed in an attempt to adapt the fundamental results to practical computing systems. In some sense, the voting schemes can be distinguished from the nonvoting schemes (such as quadding) by the fact that the restoration unit, i.e., the vote-taker, can be physically distinguished from the units actually performing the logic. In essence, voting schemes simply replicate the function to be realized in several different "lines" and then takes a weighted measure, usually a majority vote, over the outputs of the independent lines.

Subsequent papers treating such systems in general are numerous.[33,156,38,196] Jensen has developed a lower bound on the reliability of multiple-line networks,[140] using a minimal-cut concept to describe the system and analyze its failures. The fact that the reliability of majority voting circuits is sensitive to the particular modes of failure (i.e., whether the probabilities of failing to zero and to one are different) is discussed by Rhodes;[264] an extension to components that can exhibit three possible states of behavior is discussed by Rau.[262] Lyons and Vanderkulk[188] consider voting circuits in which the vote-takers are assigned either to the inputs of the logic modules or to the outputs. It is shown that the output voting scheme is generally to be preferred,

from the standpoints both of improved reliability and of reduced redundancy ratio. The important concept of adaptive vote-taking, in which inputs are weighted according to their error history, was conceived by Pierce[240] and is discussed by Angell[9] and in Sec. II-A-2-b of the body of this report.

A number of papers have been concerned with the detailed design of the all-important vote-taker itself.[80,195] Farrell[80] also considers the problem of the optimum decomposition of a system, i.e., exactly where the vote-takers are to be placed, as does Cohn,[47] who determines the optimum level for a system composed of identical components. If the vote-takers and connecting wires are also vulnerable to failures (an often ignored assumption), it can be shown that it is not always best to make systems redundant on the component level. Knox-Seith[164] carries on in this tradition and determines the best placement and resulting reliability for cascaded logic systems. Gurzi[115] also has some recent results on the cascaded three-line situation; she derives upper and lower bounds on the possible improvements through triplication, and compares the single-voter and three-voter schemes. Jensen et al.[141] and Rubin[272] have also developed synthesis techniques which determine the minimum-cost voter placement locations, as well as new techniques for estimating the reliability of the resulting systems.

Many papers have been devoted to comparing the results that can be obtained for the various schemes, voting and otherwise (e.g., Refs. 300 and 37). Attempts in this direction have also been made by Domanitskii and Prangishvili,[63] who compare the failure probabilities of systems using multiple-line voting structures with those of systems using more elemental replication on the component level, specifically for the case of transistor NOR circuits. Their results indicate that as the reliability of the individual components increases, so does the preference for replication at the component level. In the same vein, but for different redundancy procedures, Polovko and Zaynashev[251] compare fault-masking schemes with replacement schemes; they observe that the situation is ambiguous (perhaps not surprisingly) and that for one set of assumptions one approach is preferable to the other, and vice versa. They

conclude that the widest range of tolerance is achievable by using a combined scheme employing both masking and replacement strategies--certainly consistent with the conclusions of this report.

Variations on the majority scheme have also been mentioned in the literature. For example, Depian and Grisamore[58] discuss the restoring element which averages its inputs, rather than taking a majority, and conclude that the averaging method can provide greater reliability in some cases. On the other hand, Lowrie questions the efficiency of the triple-line scheme itself[186] and proposes instead the use of a duplex approach. In essence this is simply the parallel operation of two identical logic circuits or computer subsystems, with associated circuitry to detect any discrepancy between the two. The system is not truly fault-masked, however, for at this point the redundancy becomes dynamic--not only must the discrepancy be detected but it must then be responded to by external diagnostic equipment to localize the faulty system and to initiate switching action to disconnect the offending system. Nonetheless it is in the replicated-line family and it is claimed that in some cases it can achieve a higher reliability than triple-line systems, while using fewer parts. This assertion obviously stands or falls on the complexity of the associated equipment needed for detection, diagnosis, and switching, and on whether the lost time is acceptable within the operation scheme of the system.

b. Applications of Coding Theory

The connection between error-detecting and correcting codes, which have been largely developed and formalized under the aegis of communication theory for noisy-channel applications, and the application of redundancy in computing networks has long been recognized, and a numerous literature exists which formalizes the relation. These involve both abstract determinations of bounds and specific implementations to various network types. We shall point to a few of these contributions, recognizing that the differentiation between static and dynamic systems becomes a little hazy at this point. We also wish to mention fault masking in sequential machines in a separate, subsequent section, and it

is obvious that we shall suffer some overlap, since clearly one very direct application of error-correcting codes is in the judicious encoding of the states of sequential machines in order to obtain certain properties of the code for the machine behavior. Nonetheless, coding is such a well-developed formalism in its own right, as well as in its easily identifiable relevance to computing networks, that we shall risk the possible redundancy in this discussion.

An early, elementary discussion of the use of checking codes in digital-computer applications is presented by Diamond;[60] in this paper, for example, he pointed out the required distance properties for various applications. Shortly thereafter, Elias[71] considered the problem of reliable computation with a model resembling a noisy channel and on the basis of his assumptions shows that truly arbitrary reliable computation may be obtainable only at the expense of reducing the computational capacity, in an information-theoretic sense, to zero.

A summary of the applications of error-correcting codes is found in Massey,[200] as well as some original work on "reversible" codes which make it possible to read data from memory starting with either end of a data block. Peterson[238] has a useful chapter on arithmetic codes, and Kautz[148] presents an evaluation of the use of several code families in digital systems, including a consideration of some new codes, and of the simplifications that are possible if simply detection is of interest and correction is not.

Mandelbaum[194] applies cyclic codes for burst-error detection to the case of arithmetic operations, and attempts to accommodate single, double, triple, and double-burst errors, while Armstrong treats the use of nonbinary single error-correcting codes,[10] developing both bounds on the required redundancy and some specific codes that meet the bounds. Ray-Chaudhuri[263] has extended Armstrong's work and has shown that multiple error-correcting codes can be applied to the correction of failures in several different units. Many other papers are available that treat various aspects of the problem (e.g., Refs. 293, 304, and 237), and much work has gone into the development of codes that are peculiarly well-suited for certain arithmetic operations (e.g., Refs. 28, 31, 239, 89, 90).

Avizienis[13,15] has pointed out the use of codes for diagnostics in a dynamic sense, as well as in the static sense.

Homan has described the logical design of an adder used in the IBM Stretch computer[134] which utilizes four-bit groupings that are tied together by a carry-look-ahead system. This paper is one of the first to describe high-speed (25 ns for the add operation) checking circuits for arithmetic operations. The automatic correction of burst errors originating in a computer memory has been described by Daher[55] in connection with the automatic error-correction unit for a disk memory.

The Soviets have also been active in the application of codes. For examples following Gavrilov's lead[95,96,97,98] several attempts have been made to apply error-correcting codes to the design of switching networks.[64,172,234,315,296] For sequential circuits, Kurdyukov[172] has developed a single fault-correcting counter based upon a Hamming code, and Gavrilov[96,98] has designed some redundant sequential relay networks that are insensitive to the malfunction of any one relay. A similar approach has been taken by Svechinskii[296] and his use of error-correcting codes for redundant state assignment. Sagalovich[275] has also recently considered the application of codes to the state-encoding problem in automata so that failures can be tolerated.

Rubio[273] has also considered the design of self-correcting counters using coding results. Other more abstract treatments of the general interrelevance of coding theory and redundant networks are available.[203,332,333]

c. Static Redundancy in Sequential Machines

There have been a number of more or less abstract papers dealing with the careful encoding of the secondary state assignments of sequential machines, often directly utilizing the properties of error-correcting codes. These machines are encoded redundantly, hence have redundant states, and if properly designed can possess any of several error-tolerant properties. For example, the assumption of a redundant state may signal an error-detection circuit that an improper transition has occurred; or the redundant states may be so organized that the machine

returns in a finite number of steps to the proper primary state, thus truly masking the error (after a period of time, during which the output behavior may be insensitive to the fact that something was in error).

Levenshtein[181] has made an attempt to define the class of sequential circuits which are insensitive to an accidental change of state, in the sense that they will return to proper state behavior after a finite amount of time following the occurrence of an internal error. Thus the present state of the network is dependent only on the inputs and a finite amount of past history. This situation has also been investigated by Dauber[56] and may be compared with the model described by Winograd[331] and Harrison,[119] where the errors are due to accidental input changes.

We have already noted the contributions of Sagalovich[275] and Svechinskii[296] in the use of error-correcting codes to make secondary assignments such that failures can be tolerated. Another paper which considers a similar model that tolerates error in its state transitions is that of Frank and Yan.[85] The special problems in relay machines have been discussed by Mullin[223] including the particular problems of analyzing the reliability of such machines.

The general problem of analysis of such machines is still difficult and not well understood. Tsertsvadze[306,307] has suggested the use of a stochastic automaton model as the appropriate general approach to the design of finite automata with specified reliability characteristics, composed of unreliable components. His methods, which are based on Von Neumann, as well as Moore-Shannon models, appear to be addressed both to permanent and to intermittent faults within a system.

Several papers have been directed toward the design of specific machine types; in particular, counters serve as a convenient vehicle. We have mentioned Rubio's[273] paper in this regard; Russo[274] has also used codes to design counters--using distance-three codes in the input equations, thus assuring a tolerance to single-bit errors, either permanent or transient.

5. Discussions of Dynamic Redundancy Applications

a. Approaches to Fault Diagnosis

For systems where dynamic redundancy is used to increase the reliability, it follows by definition that auxiliary equipment is involved in the response of the overall system to a fault in one of its subsystems. Thus the terminals of the subsystem are involved, with the help of equipment external to the subsystem proper, in the detection, location, and analysis of the nature of the fault (which we lump under the term *diagnosis* for this discussion). Secondly, external equipment (perhaps with human intervention) decides the nature of the response and carries through on its execution. Diagnostic procedures are reviewed here; the second phase is reviewed in the next section. In the last section we mention complete system organizations that have been proposed to facilitate self-repair.

All structural redundancy is, after all, in some sense simply the provision of spare parts in a mechanism to enable it to tolerate failures. In static redundancy these spare parts are permanently part of the mechanism and the toleration (up to some saturation point) takes place autonomously (and anonymously) within the mechanism itself. In dynamic redundancy, although the spare part may be fully powered--even performing the same job--there must be a switchover process on the advent of a failure. Hence the task of diagnosis must be no more than to determine the replaceable unit--whether that be an individual component, a network, or a subsystem. Thus, depending on the system, simple fault detection may be all the diagnostics that are needed if the replaceable unit is the entity signalling the detected fault. On the other hand, the fault detector may embrace a number of replaceable units, or modules, in which case fault-location routines must be instigated in order to find the particular offending module that must be removed. Finally, if the replaceable units are the actual components within a physically contained module, then what is more conventionally described as a diagnostic test routine must be used to locate the particular component that has failed.

Thus all of the terms "detection," "location," and "diagnosis," which cause some semantic confusion, we identify as describing general diagnosis activities.

The formal problem statement and solution of the fault-diagnosis situation for combinational networks has been presented in this report, as well as in an earlier summary.[149] Given a network with inputs and outputs, a fault can be diagnosed if and only if it produces an output column in the fault table that is distinguishable from all other columns (including the column for the network without faults. The problem then evolves into one of determining which rows of the fault table to use as inputs in a test schedule so that some criterion of the schedule is optimized (e.g., perhaps the maximum length is important, or perhaps the average length). Kautz' procedures[149] will formally give minimal schedules but become unwieldy for large networks. He has also introduced the notion of serial test schedules and has shown that these may result in much shorter routines.

The similar problem in the Soviet Union stems from an early work by Chegis and Yablonskii[41,335] who first introduced the notions of the fault table and the use of a minimum-length test schedule for the isolation of single faults (opens and shorts) in combinational relay-contact networks. They proposed the use of special switching functions, and showed how manipulations of these expressions could be considerably simplified when the network is planar, by working with the dual network. Alexander[4] in this country has also shown that special switching-circuit notations can be applied to the drawing up of test and maintenance schedules.

Later Soviet work has relied heavily on the earlier work; for example Kogan[167] improved the Chegis-Yablonskii procedure somewhat by using the notion of proper-cut sets of the graph of the contact network, and also evaluated some special networks based upon disjunctive and conjunctive normal-form expansions. Kogan,[168] in another paper, and later Vaksov,[313] showed how to minimize the length of the test schedule for the location of any number of shorts and opens in the class of nonrepetitious networks (i.e., those networks having just a single contact in

each input variable). Later, Glagolev[106] specialized the Chegis-Yablonskii approach to iterative contact networks, and others[291,287] have extended the concepts to gate-type networks, including location to either the faulty gate or the faulty subsystem. The possibility of utilizing auxiliary inputs is examined by Karibskii,[145] and a special form of serial testing is proposed by Kinsht[162] in an attempt to shorten the average test-schedule length.

On the level of entire digital systems, there have been two interesting analyses; one by Fleyshman,[83] who studied the self-monitoring concept and concluded that such a system needs an ultrareliable "hard core" which can be used to check out in succession increasingly larger portions of the system hierarchy, and one by Lyabutov,[187] who solved the general problem of optimizing a sequence of tests on a multimodule system, given for each module its probability of failure, the length of time required to test the module, and the probability that the test will not reveal a fault within the module. (An optimal sequence is defined here to be one whose average length is minimal.) In addition, the area of programming diagnostics has been examined for particular systems, e.g., the URAL computer.[79] There have also been more sophisticated diagnostic approaches which involve diagnosing over successively less gross portions of a computer until finally the particular offending gate is isolated.[316] This sort of approach appears closely related to similar philosophies of serial testing that have been mentioned previously.

Several papers have discussed the importance of diagnostic procedures in complex systems (e.g., Refs. 182 and 136), the nature of the diagnostic equipment, and routines (e.g., Refs. 34 and 87). Lee[180] proposes a method for the situation where normal operation can be interrupted, while Schneider and Wagner[276] are concerned with the case where uninterrupted operation is crucial. Several papers concern themselves with diagnostic programs in particular machines,[279,158,101] and Tsiang and Ulrich[308,309] have described the routines used in the modern electronic central office of a telephone system. The operation appears to have the same flavor as the duplex-redundancy scheme previously mentioned; that is, the central-office control is duplicated; both units

work together on the same input and the resulting outputs undergo continual comparison. Any discrepancy calls for fault-diagnosis routines to be started during the next available period.

Poage[246] has also considered efficient schedules for combinational networks, and has proposed an analytical procedure that yields an output expression which reflects network structure; he then uses this output expression to develop tests to detect single faults, and also extends the method to multiple faults as well. Chang has also investigated the problem of efficient diagnostic tests for combinational networks[40] and develops an algorithm for reducing the redundancy in the rows of the fault table selected for a schedule. Armstrong[11] uses a "path-sensitizing" concept in the treatment of the class of faults wherein connections become stuck at logical one or zero, and describes a procedure that can be applied to larger networks than can be treated by exact methods. Roth[270] has also developed diagnostic algorithms for faults in combinational networks. One way of developing a set of tests is to simply simulate the network on a computer and let the computer generate the failure cases; this approach is proposed by Seshu and Freeman[280] who use a decision tree to successively isolate the fault to a smaller and smaller set. Johnson considers the cost of a given test and the information gained by it to develop a figure of merit to help in developing the most efficient test procedure.[142]

Kautz[147] has described constructive procedures for the design of networks that provide an output to indicate a failure within the network--that is, an automatic fault detection signal--while Kilmer[159] has discussed an idealized computer organization composed entirely of fault-detecting circuitry of the fault-detecting type which can correct a bounded number of transient failures which might occur anywhere within the computer.

Thus the area of combinational network diagnostics would appear to be very well represented, and the formal problems seem to be well understood. There are still problems in large networks, in the handling of transient errors in general, in the utilization of auxiliary inputs and outputs, etc., but much comprehensive work has been done and directions for further research are relatively clear.

Such is not the case for the diagnosis of sequential machines, however. Some very excellent work has been done, but it is still largely of a theoretical nature and certainly is not practicable for machines larger than a very few states.

In one very basic approach, the problem is formulated in a fashion rather analogous to the fault-table representation for combinational networks. If the given machine is designated by S, then the class of faults, k in number say, corresponds to faulty machines S_1, ..., S_k. The obvious approach, then, is to determine a set of inputs so that the resulting output sequences will distinguish all the S_i from each other, as well as from the good machine S. This is essentially the formulation proposed by Gill[103] and by Poage and McCluskey,[247] wherein procedures for developing appropriate input sequences are presented. It is reiterated that the procedures rapidly become unwieldy, even with computer aids, beyond a few states and a small set of faulty machines.

Hennie[127] takes a similar approach, related to early work by Moore,[213] and asks for appropriate "checking experiments" (i.e., input sequences again) which will allow the determination of whether the assumed state table for the machine is actually the one being traversed by the machine being checked. An important consideration is whether the given machine has a "synchronizing" sequence, that is, a sequence which will return it to a given starting state--not all machines do. Kime[160,161] proposes a specific testing method based on Hennie's testing philosophy and considers the problem of designing circuits that have "distinguishing" sequences.

Soviet treatment of the problem of diagnosing faults in sequential networks is also rather skimpy. We mention only Karibskii et al.[146] where Gill's procedure[103] has been successfully generalized and a few bounds on the lengths of the test schedules are given. However, the problem is far from solved, and there is much work to do.

b. Spare-Equipment Considerations

There has also been a great number of papers concerned with the statistical aspects of systems with standby parts, under various assumptions on the lifetime of such spares, on their number, on their

repairability, and so on. It is convenient to mention, in this same context, questions concerning maintenance procedures and the effects that various maintenance policies have on system life, depending on various assumptions on checking periods.

Weiss[326] has considered the determination of the optimum checkout interval for a system which does not exhibit the usually assumed exponential failure characteristics. Flehinger[81] provides a treatment of a number of different probabilistic models corresponding to different preventive-maintenance policies.

Teoste[299] presents the design of a computer which can continue operating satisfactorily even while a part of it is being repaired. He provides a general discussion of machines of this type; the first gross approach is to consider several computers all operating on the same inputs, with the output taken from one unit till it fails, then from the next, and so on; then the system can be broken into smaller parts and essentially the same process applied to these parts, etc. Gaver[94] also discusses the time to failure and the availability of redundant systems in which repair is permitted. He assumes that failures are immediately identifiable, and again takes primarily a duplex approach. In a later paper[93] he loosens the assumptions somewhat and analyzes the failure time of such a system when the two units are not the same--in particular, when they possess different kinds of statistical failure properties. He develops explicit formulas and approximations for the mean time to failure, and shows how these are affected by various repair-time distributions. Other papers that consider various assumptions on the number and failure distributions of spares include Refs. 207, 217, 163, 327, and 334. Rosenheim and Ash[268] assume that the redundancy in spares extends to entire machines, rather than just to components or small units. Flehinger[82] has presented a comparison between, on the one hand, the situation where the redundancy is applied over complete independent machines, and on the other hand, the provision of spares on the basis of much smaller units. She calculates the resulting reliability in each case as a function of the number of units in the nonredundant machine and of the degree of replication.

Other special assumptions are reported by Sinitsa,[289] who includes the possibility of statistical dependence between the active and spare components of a system, and by Muth,[225] who assumes limited repair capabilities and plots expected time to failure as a function of the capability for one repair, two repairs, and so on. Nerber[228] and Buckley[35] both examine the necessity of including failure rates for the idle standbys, even in the power-off mode, and including the problem of detecting whether a turned-off part has failed.

The Soviets have also been fairly aggressive in examining these areas. General but fairly conventional formulations, concerned with determining the probability of failure-free operation of systems which are operable until a given number of failures have occurred, seem to be well represented.[154,226,235,260] These are generally concerned with the calculation of the mean time to failure under different probability assumptions. Closely related to these papers is the work of Shcherbakov[282] who investigates the characteristics of the "up-time" distribution of systems under various assumptions on the time for repair once a breakdown occurs. Also worth noting in this regard is Zhozhikashvili and Raikin's[337] extension to include the consideration of systems in which there is an elemental amount of self-diagnostic capability included in order to signal the loss of some fault-masking capabilities as a system progressively deteriorates. On the other hand, Malev[190] assumes that maintenance procedures are instituted periodically and calculates the average period of correct operation under this regimen, while the effects of incorporating both standby-reserve equipment and a maintenance-and-repair schedule are considered by Zubova[351] and by Raikin et al.[258] Zubova assumes duplicate systems, with one in idle standby, under different assumptions on failure times for each system, and different times within which a system is assumed to have fully recovered. Similar studies of standby systems have been reported by Gnedenko[107,108] and others.[23,102,257] The possibility of using one of the computers in an interconnected system to check out the other for faults automatically has been recognized, and programs for accomplishing this possibility have been developed.[231]

In a sequence of papers Raikin,[255,256,259] Kel'mans,[155] Smolitskii and Chukreev,[290] and Alekseev and Yakushev[2] have studied the problem of determining the optimal number of spare modules of each type in a modular system, taking into account the "cost" of each type of spare module, the probability of failure of each module, and whether or not the module may fail while it is a spare. Several algorithms for the optimization are discussed in these papers. Gertsbakh[102] and Barzilovich[23] have investigated the same type of system, under the condition of minimizing the average loss which occurs between the times of fault occurrence and fault repair, and algorithms are derived for optimal maintenance based upon this condition. Raikin[257] has extended his results on the average number of standby modules that remain available as a function of time.

Korman[169] has addressed himself to the inverse of the problem of determining the optimal reserve for a modular system; in particular he assumes a given ratio of spare modules and determines the necessary reliability characteristics of the module itself in order to achieve a specified overall system probability of trouble-free operation.

The possibility of using memories which are segmented into units is an important aspect of complex reconfigurable systems. In this regard, we note that in studies using the URAL-1 and the M-20 computers[170] it has been concluded that the use of substitute, switchable memory spares is feasible, even under the assumption that the switching system itself is no more reliable than the balance of the system. In a related approach, a duplexed memory system has been reported[261] which may be operated in either the duplex mode--with both memories being read simultaneously and updated by the particular section being used--or in a simplex mode. In the simplex mode, both memories are actively used and the choice of mode is determined by the nature of the mission.

Cosgrove and Masters,[50] and other workers at Westinghouse[341] have considered an interesting self-repair variation on the conventional multiple-line voting scheme: they make provision for the shifting around of blocks of circuitry as a function of the failure history. As failures occur and leave certain networks more vulnerable than others to

succeeding failures, circuit blocks are switched about in order to increase the reliability of the more vulnerable networks. In computer simulation programs they produce results that show that only a modest amount of switching is necessary to produce significant reliability gains.

Goldberg[109] has developed several network schemes that use both static-masking techniques and adaptive-replacement techniques to augment the reliability of multiple-line systems. (See also Sec. II-A-2-b of the body of this report.)

c. System Organizations that Facilitate Self-Repair

There is also a growing number of papers concerned with the organization of entire systems that optimize the capability for self-diagnosis and repair, reconfigurability, or modularity. For example, Terris and Melkanoff[303] describe a self-repairing system with a switching mechanism for the replacement of failed circuitry which has been automatically diagnosed. They utilize a "master machine" to aid in the control functions, and find that a thirty-percent increase in required equipment results in a fourfold increase in the mean lifetime between failures.

Landers[174] attempts to categorize self-repairing systems in a general way, notes that very little reduction to practice has been made of the many concepts that have been formulated, and offers the debatable opinion that the difficulties stem from the fact that such concepts border on self-reproducing systems--and hence share their fundamental difficulties. Doyle[65] describes a program that is used to enable a digital system to repair itself, and in application to the SAGE system it is asserted that the system provided automatic recovery from over 90 percent of the failures occurring during the period of study. Kruus[171] has also studied self-repairing systems composed of a number of identical machines, spare parts, and the necessary interconnecting mechanisms. He has simulated machines operating in parallel, with faults being detected by observing a difference in outputs. He also notes the problem of the initial setting of a newly substituted machine in order that its outputs agree with the outputs of the operating machines.

Agnew et al.[1] have presented the design of a hypothetical aerospace computer; they use a partitioning technique to determine the appropriate diagnostic subsystems. Interestingly enough, they conclude that diagnosis and self-repair (i.e., dynamic redundancy) are not sufficient for maximum system availability, but that static redundancy techniques must be used also in order to achieve the optimum configuration. Avizienis[14] has presented complete processor organizations using his codes for continuous generation of real-time diagnostic information in order to initiate repair, replacement, or reorganization of the system.

Forbes et al.[84] describe the organization of a computer specifically designed to maximize up-time through self-diagnostic routines, but consider only manual replacement of faulty modules. Manning[197] has reported on an extension of earlier work[198] on the self-diagnosis problems in a single-processor machine to those in a large multiprocessor machine. Joseph[143] also reports on a specific multiprocessor configuration, and England[74] describes a space-guidance computer configuration that exhibits both static and dynamic redundancy techniques at different hierarchical levels.

6. Peripheral Considerations

Redundancy techniques can be applied systems other than computing networks: for example, to power supplies, mechanical systems, etc. There have also been a number of papers concerned with the other environmental aspects exhibited by redundant systems: the increased volume and weight requirement, the adjustment of loads within a system when fault-masked components fail, and so on. These problem areas were not directly attacked in developing this survey, but several relevant articles can be noted that were accumulated while searching for articles pertinent to the main body of interest.

It has already been noted that redundancy techniques can be transformed directly to application in continuous systems.[175] Raikin has investigated the problem of redistribution of loads in a series or parallel-connected system; if an element fails during operation the remaining elements must take over its load or voltage, and specific reliability formulas are derived for given dependencies of the damage incurred.

Herron[128] considers weight as a constraint in maximizing reliability, and shows how his method can be extended to include other environmental constraints as well.

Finally, Paynter and Mathis[236] consider the problem of redundancy in power-supply design; they treat replication of entire units, as well as component redundancy within units. The important point is made that while a component failure in a signal-processing unit need not be catastrophic, a failure in a power supply usually is--hence, under the philosophy that the weakest unit should be strengthened first by the use of redundant structures, the attainment of power-supply reliability is extremely important. Techniques for this purpose are considered in Appendix B of this report.

REFERENCES

1. Agnew, P. W., D. H. Rutherford, R. J. Suhocki, and C. M. Yen, "An Architectural Study for a Self Repairing Computer," Final Technical Documentary Report No. SSD-TR-65-159, U.S. Air Force, Space Systems Division, Air Force Systems Command (November 1965), (AD 474 976).

2. Alekseyev, O. G., and V. L. Yakushev, "Algorithm for Optimum Reserve (Redundancy) of an Apparatus, "Engr. Cyb., No. 3, pp. 55-61 (1964).

3. Alelyunas, Paul, "Checkout: Man's Changing Role," Space/Aeronautics, Vol. 44, No. 7, pp. 66-73 (December 1965).

4. Alexander, S., "Application of Boolean Notation to the Maintenance of Switching Circuits," Electrical Eng., Vol. 33, No. 6, pp. 372-374 (June 1961).

5. Allen, C., "Design of Digital Memories That Tolerate All Classes of Defects" SEL Technical Report No. 4662-1, Stanford Electronics Laboratories, Stanford, California (May 1966).

6. Amarel, S., and J. A. Brzozowski, "Theoretical Considerations on Reliability Properties of Recursive Triangular Switching Networks", Redundancy Techniques for Computing Systems, R. H. Wilcox and W. C. Mann, eds. (Spartan Books, 1962).

7. Amarel, S., G. Cooke, and R. O. Winder, "Majority Gate Networks," IEEE Trans. on Electronic Computers, Vol. EC-13, No. 1, pp. 4-13 (February 1964).

8. Angell, J. B., "The Need and Means for Fault Detection in Redundant Systems," Working Paper presented at the Workshop on the Organization of Reliable Automata, Pacific Palisades, California (2-4 February 1966).

9. Angell, J. B., "The Need and Means for Self-Repairing Circuits," IEEE Int. Conv. Record, Part 2, pp. 193-199 (March 1963).

10. Armstrong, D. B., "A General Method of Applying Error Correction to Synchronous Digital Systems," Bell System Tech. J., Vol. 40, No. 2, pp. 577-593 (March 1961).

11. Armstrong, D. B., "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets," IEEE Trans. on Elec. Computers, Vol. EC-15, No. 1, pp. 66-73 (February 1966).

12. Aroian, L. A., "The Reliability of Serial Systems and Redundant Systems," Proc. 10th Ann. Symp. on Reliability and Quality Control, pp. 174-185 (January 1964).

13. Avizienis, A., "Detection and Correction of Failure in Digital Arithmetic Units," Space Programs Summary 37-25, Vol. IV, pp. 21-24, Jet Propulsion Laboratory, Pasadena, California (February 1964).

14. Avizienis, A., "A Diagnosable Arithmetic Processor," Working paper presented at the Workshop on the Organization of Reliable Automata, Pacific Palisades, California (2-4 February 1966).

15. Avizienis, A., "A Set of Algorithms for a Diagnosable Arithmetic Unit," Tech. Report No. 32-546, Jet Propulsion Laboratory, Pasadena, California (1964).

16. Baechler, D. O., and D. E. Van Tijn, "Computer Reliability Study," Final Report No. 240011374, Arinc Research Corp. (July 1963).

17. Baer, J. A., and C. H. Heckler, Jr., "Research on Reliable and Radiation Insensitive Pulse Drive Sources for All-Magnetic Logic Systems," Final Report, Contract 950104 under NASw-6, SRI Project 3729, Stanford Research Institute, Menlo Park, California (June 1962).

18. Balaban, H. S., "A Selected Bibliography on Reliability," IRE Trans. on Reliability and Quality Control, Vol. RQC-11, pp. 86-103 (July 1962).

19. Barker, R. C., "Computer Magnetics--A Selective Review of Periodical Literature," Electro-Technology, Vol. 69, No. 3, pp. 74, 75, 140 (March 1962).

20. Barlow, R. E., and L. C. Hunter, "Criteria for Determining Optimum Redundancy," IRE Trans. on Reliability and Quality Control, Vol. 9, pp. 73-77 (April 1960).

21. Barlow, R. E., and F. Proschan, Mathematical Theory of Reliability (John Wiley & Sons, New York, N.Y., 1965).

22. Barlow, R. E., L. C. Hunter, and F. Proschan, "Optimum Redundancy when Components are Subject to Two Kinds of Failure," J. SIAM, Vol. 9, No. 1 pp. 64-73 (March 1963).

23. Barzilovich, Ye. Yu., "Determination of Optimum Periods of Preventative Maintenance in Automatic Systems," Engr. Cyb., No. 3, pp. 32-39 (1964).

24. Benes, V. E., Mathematical Theory of Connecting Networks and Telephone Traffic (Academic Press, New York, N.Y., 1965).

25. Benes, V. E., "Permutation Groups, Complexes, and Rearrangeable Connecting Networks," Bell System Tech. J., Vol. 43, No. 4, part 2, pp. 1619-1656 (July 1964).

26. Bennion, D. R., and H. D. Crane, "All-Magnetic Circuit Techniques," Advances in Computers, Vol. 4, pp. 53-133 (Academic Press, New York, N.Y. 1963).

27. Bennion, D. R., H. D. Crane, and D. C. Engelbart, "A Bibliographical Sketch of All-Magnetic Logic Schemes," IRE Trans. on Electronic Computers, Vol. EC-10, No. 2, pp. 203-206 (June 1961).

28. Bernstein, A. J., and W. H. Kim, "Linear Codes for Single-Error Correction in Symmetric and Asymmetric Computational Processes," IRE Trans. on Information Theory, Vol. IT-8, No. 1, pp. 29-34 (January 1962).

29. Birnbaum, Z. W., J. D. Esary, and S. C. Sanders, "Multicomponent Systems and Structures and Their Reliability," Technometrics, Vol. 3, No. 1, pp. 55-77 (February 1961).

30. Bonn, T. H., "Magnetic Computer Has High Speed," Electronics, Vol.30, No. 8, pp. 156-160 (August 1957).

31. Brown, D. T., "Error Detecting and Correcting Binary Codes for Arithmetic Operations," IRE Trans. on Elec. Computers, Vol. EC-9, No. 3, pp. 333-337 (September 1960).

32. Brown, J. N., and E. E. Newhall, "The Storage and the Gating of Information Using Balanced Magnetic Circuits," Proc. of INTERMAG Conf., pp. 11-1-1 to 11-1-5 (April 1964).

33. Brown, W. G., J. Tierney, and R. Wasserman, "Improvement of Electronic Computer Reliability Through the Use of Redundancy," IRE Trans. on Elec. Computers, Vol. EC-10, No. 3, pp. 407-416 (September 1961).

34. Brule, J. D., R. A. Johnson, and E. J. Kletsky, "Diagnosis of Equipment Failures," IRE Trans. on Reliability and Quality Control, Vol. RQC-9, pp. 23-34 (April 1960).

35. Buckley, J. J., "The Non-Operating Failure Rate--a Valuable Reliability Tool," Proc. Eleventh Natl. Sym. on Reliability and Quality Control, pp. 434-438, (January 1965).

36. Bukharaev, R. G., "Computation of Contact-Circuit Reliability," Automation and Remote Control, Vol. 25, No. 8, pp. 1085-1090 (August 1964).

37. Burt, M. W., and D. C. James, "How Much Does Redundancy Improve Reliability?" Control Engineering, Vol. 10, No. 6, pp. 71-76 (June 1963).

38. Buzzell, G., W. Nutting, and R. Wasserman, "Majority Gate Logic Improves Digital System Reliability," IRE Natl. Conv. Record, Pt. 2, pp. 264-270 (1961).

39. Campbell, R. L., and W. Thomis, "A New Approach to System Maintenance," Bell Laboratories Record pp. 251-255 (June 1965).

40. Chang, H. Y., "An Algorithm for Selecting an Optimum Set of Diagnostic Tests," IEEE Trans. on Elec. Computers, Vol. EC-14, No. 5, pp. 705-711 (October 1965).

41. Chegis, I. A., and S. V. Yablonskii, "Logical Methods for Controlling the Operation of Electrical Networks," Trudy Matematicheskvo Instituta imeni V. A. Steklova, Vol. 51, pp. 270-360 (1958). Trans: No. 60-41069, Office of Technical Services U.S. Dept. of Commerce, Washington, D.C. (1960).

42. Clift, R. A., "Power Switching Trims Digital System Weight, Cost," Electronics, Vol. 39, No. 12, pp. 135-138 (June 1966).

43. Coates, C. L., and P. M. Lewis II, "A Threshold Gate Computer," IEEE Trans. on Elec. Computers, Vol. EC-13, No. 3, pp. 240-247 (June 1964).

44. Cobham, A., R. Fridshal, and J. H. North, "An Application of Linear Programming to the Minimization of Boolean Functions," Proc. 2nd Annual Symp. Switching Circuit Theory and Logical Design, AIEE Special Publication S-134, pp. 3-9, (September 1961).

45. Cochran W. G., and G. M. Cox, Experimental Designs (John Wiley & Sons, New York, N.Y., 1950).

46. Cohen, J. J. and L. A. Whitaker, "An Approach to Diagnostic Programming," 3rd. Annual Meeting ACM, 23-25 August 1960.

47. Cohn, M., "Redundancy in Complex Computers," 1956 IRE National Conference Proc. on Aeronautical Elec., pp. 231-235 (May 1956).

48. Condon, D. C., "Atomic Reactor Control System," Final Report Contract N259-1577, SRI Project 4529, Stanford Research Institute, Menlo Park, California (October 1963).

49. Corliss, William R., Space Probes and Planetary Exploration (D. Van Nostrand Co., Inc., Princeton, N. J., 1965).

50. Cosgrove, M. R., and C. G. Masters, "Self-Repair Techniques for Failure-Free Systems," Special Technical Report 2, Westinghouse Elect. Corp. (September 1963).

51. Crane, H. D., "Design of an All-Magnetic Computing System: Part II-Logical Design, IRE Trans. on Electronic Computers, Vol. EC-10, No. 2, pp. 222-232 (June 1961).

52. Crane, H. D., and E. K. Van De Reit, "Design of an All-Magnetic Computing System: Part I-Circuit Design," IRE Trans. on Electronic Computers, Vol. EC-10, No. 2, pp. 207-220 (June 1961).

53. Creasey, D. J., "Redundancy Techniques for Use in an Air Traffic Control Computer," In Microelectronics and Reliability, Vol. 3, (Pergamon Press, MacMillan Co., New York, N.Y., 1964).

54. Creveling, C. J.; "Increasing the Reliability of Electronic Equipment by the Use of Redundant Circuits," Proc. IRE, Vol. 44, No. 4, pp. 509-15 (April 1956).

55. Daher, P. R., "Automatic Correction of Multiple Errors Originating in a Computer Memory," IBM J. Res. Dev., Vol. 7, No. 4, pp. 317-324 (October 1963).

56. Dauber, P. S., "An Analysis of Error in Finite Automata," Inf. and Control, Vol. 8, No. 3, pp. 295-303 (June 1965).

57. Davis, R. A., "A Checking Arithmetic Unit," Proc. of the Fall Joint Computer Conference, Vol. 27, Part 1, pp. 705-713 (1965).

58. De Pian, L., and N. T. Grisamore, "Reliability Using Redundancy Concepts," IRE Trans. on Reliability and Quality Control, Vol. RQC-9, pp. 53-60 (April 1960).

59. De Pian, L., and N. T. Grisamore, "Two Approaches to Incorporating Redundancy into Logical Design," Redundancy Techniques for Computing Systems, R. H. Wilcox and W. C. Mann, eds. (Spartan Books, Washington, D.C., 1962).

60. Diamond, J. M., "Checking Codes for Digital Computers," Proc. IRE, Vol. 43, No. 4, pp. 487-488 (April 1955).

61. Dickinson, M. M., J. B. Jackson, and G. C. Randa, "Saturn V Launch Vehicle Digital Computer and Data Adapter," Proc. of the Fall Joint Computer Conference, pp. 501-516 (1964).

62. Dickinson, W. E., and R. M. Walker, "Reliability Improvement by the Use of Multiple-Element Switching Circuits," IBM J. Res. Dev., Vol. 2, No. 2, pp. 142-147 (April 1958).

63. Domanitskii, S. M., and I. V. Prangishvili, "Reliable Logic Elements and Output Amplifiers with Redundant Structure," Automation and Remote Control, Vol. 25, No. 4 pp. 511-515 (April 1964).

64. Domanitskii, S. M., and I. V. Prangishvili, "An Accurate Design Method for Semiconductor Switching Circuits Used in Industrial Automation," Automation and Remote Control, Vol. 24, No. 5, pp. 606-613 (1963).

65. Doyle, R. H., R. A. Meyer, and R. P. Pedowitz, "Automatic Failure Recovery in a Digital Data Processing System," IBM J. Res. Dev., Vol. 3, No. 1, pp. 2-12 (January 1959).

66. Drenick, R. F., "The Failure Law of Complex Equipment," J. Soc. Indust. Appl. Math., Vol. 8, No. 4, pp. 680-690 (December 1960).

67. Drenick, R. F., "Mathematical Aspects of the Reliability Problem," J. Soc. Indust. Appl. Math., Vol. 8, No. 1, pp. 125-149 (March 1960).

68. Dreyfus, P., "Programming Design Features of the Gamma 60 Computer," Proc. of the Eastern Joint Computer Conference (December 1958).

69. M. Dunning, B. Kolman, and L. Steinberg, "Reliability and Fault Masking in n-Variable NOR Trees," Proc. of the 6th Annual Symposium on Switching Circuit Theory and Logical Design, IEEE Publication 16c13, pp. 126-142 (October 1965).

70. Eckert, J. P., J. C. Chu, A. B. Tonik, and W. F. Schmitt, "Design of UNIVAC LARC System: I," Proc. of the Eastern Joint Computer Conference, pp. 59-65 (1959).

71. Elias, P., "Computation in the Presence of Noise," IBM J. Res. Dev., Vol. 2, No. 4, pp. 346-353 (October 1958).

72. Elspas, B., "Design and Instrumentation of Error-Correcting Codes," Final Report, Contract AF 30(602)-2327, SRI Project 3318 Stanford Research Institute, Menlo Park, California (October 1962).

73. Elspas, B., et al., "Investigation of Propagation Limited Computer Networks," Final Report, SRI Project 4523, pp. 68-75, Stanford Research Institute, Menlo Park, California (April 1964) AD-603 165.

74. England, W. A., "Improving Reliability by the Practical Application of Selected Redundant Techniques," Working paper presented at the Workshop for the Organization of Reliable Automata, Pacific Palisades, California (2-4 February 1966).

75. Ergott, H. L., and D. P. Rozenberg, "On the Analysis of Reliability Improvement through Redundancy," Report No. 62-825-494, IBM Federal Systems Div., Oswego, New York (1964).

76. Esary, J. D. and F. Proschan, "Coherent Structures of Non-Identical Components," Technometrics, Vol. 5, pp. 191-209 (May 1963).

77. Esary, J. D., and F. Proschan, "Relationship between System Failure Rates and Component Failure Rates," Technometrics, Vol. 5, pp. 183-189 (May 1963).

78. Esary, J. D. and F. Proschan, "The Reliability of Coherent Systems," in Redundancy Techniques for Computing System (Spartan Books, Washington, D.C., 1962).

79. Fakharov, V. V., "An Overall Diagnostic Routine for the "URAL-1" Computer," Dokl. (3rd) Sibirsk, Konferentsii po Matem. i Mekhan Tomsk, p. 267 (1964).

80. Farrell, Edward J., "Improving the Reliability of Digital Devices with Redundancy; An Application of Decision Theory," IRE Trans. on Reliability and Quality Control, Vol. RQC-11, No. 1, pp. 44-50 (May 1962).

Fedderson, A. P.--see item 350 of this Bibliography.

81. Flehinger, B. J., "A General Model for the Reliability Analysis of Systems Under Various Preventive Maintenance Policies," Ann. Math. Statistics, pp. 137-156 (March 1962).

82. Flehinger, B. J., "Reliability Improvement Through Redundancy at Various System Levels," IBM J. Res. Dev., pp. 148-158 (April 1958).

83. Fleyshman, B. S., "The Statistical Theory of Reliable Functioning in Finite Automata During Interferences," In Relay Systems and Finite Automata, (Burroughs Corp., Paoli, Pa., 1964).

84. Forbes, R. E., D. H. Rutherford, C. B. Streglitz, and L. H. Tung, "A Self-Diagnosable Computer," Proc. of the Fall Joint Computer Conference, Vol. 27, Part 1, pp. 1073-1086 (1965).

85. Frank, H. and S. S. Yau: "Improving Reliability of a Sequential Machine by Error-Correcting State Assignments," IEEE Trans. on Electronic Computers, Vol. EC-15, No. 1, pp. 111-113 (February 1966).

86. Frankel, S. P., "On the Minimum Logical Complexity Required for a General Purpose Computer, IRE Trans. on Electronic Computers, Vol. EC-7, No. 4, pp. 282-285 (December 1958),

87. Galey, J. M., R.E. Norby, and J. P. Roth, "Techniques for the Diagnosis of Switching Circuit Failures," IEEE Trans. on Comm. and Electr. Vol. 83, No. 74, pp. 509-514 (September 1964).

88. Gallager, R. G., "Low-Density Parity-Check Codes," IRE Trans. on Information Theory, Vol. IT-8, No. 1, pp. 21-28 (January 1962).

89. Garner, H. L., "Error Codes for Arithmetic Operations," working paper presented at the Workshop on the Organization of Reliable Automata, Pacific Palisades, California (2-4 February 1966).

90. Garner, H. L., "Generalized Parity Checking," IRE Trans. on Electronic Computers, Vol. EC-8, No. 1, pp. 25-30 (March 1959).

91. Garner, H. L., "The Residue Number System," IRE Trans. on Electronic Computers, Vol. EC-8, No. 2, pp. 140-147 (June 1959).

92. Garner, H. L., et al., "A Study of Iterative Circuit Computers," Tech. Doc. Report AL-TDR-64-24, Air Force Avionics Laboratory, Research and Technology Division, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio, (April 1964) AD-601 212.

93. Gaver, D. P., "Failure Time for a Redundant Repairable System of Two Dissimilar Elements," IEEE Trans. on Reliability and Quality Control, pp. 14-22 (March 1964).

94. Gaver, D. P., "Time to Failure and Availability of Paralleled System with Repair," IEEE Trans. on Reliability and Quality Control, p. 30 (June 1963).

95. Gavrilov, M. A., "Structural Redundancy and Reliability in the Operation of Relay Circuits," Izdatelstvo Akademii Nauk, SSSR (Moscow) pp. 16 (1960).

96. Gavrilov, M. A., "Structural Redundancy and Reliability of Relay Circuits," Automatic and Remote Control Proceedings of the First International Congress of the International Federation of Automatic Control (IFAC) Moscow, 1960, Vol. 2, pp. 838-844 (Butterworths, London, 1961).

97. Gavrilov, M. A., "The Synthesis of Remote Control Signals by Means of the Combinational Use of Pulse Modes," Automation and Remote Control, Vol. 17, No. 12 (1956).

98. Gavrilov, M. A., et al. "The Realization of Digital Corrector Networks," Automation Express, Vol. 1, No. 8, pp. 10-12 (1959); Soviet Phys.--Doklady, Vol. 3, No. 6, pp. 1102-1105 (1958).

99. Gendler, M. B., "Analysis of Conditions of Reliable Realizability of Threshold Functions Using Real Threshold Elements," Engineering Cybernetics Vol. 1, pp. 29-38 (1965).

100. Germeroth, J. H., "Casting Out Three in Binary Numbers," IRE Trans. on Electronic Computers, Vol. EC-9, No. 3, p. 373 (September 1960).

101. Gerrand, F. and H. S. Rasmussen, "Self-Correction in Large-Scale Digital Computers," Proc. 7th Intl. Symp. on Reliability and Quality Control, Philadelphia, pp. 351-359 (9-11 January 1961).

102. Gertsbakh, I. B., "Optimum Rule for Maintenance of a System with Many States," Engineering Cybernetics (1964).

103. Gill, A., Introduction to the Theory of Finite-State Machines (McGraw-Hill Book Co., New York, N.Y., 1962).

104. Gill, A., "Minimum-Scan Pattern Recognition, "IRE Trans. on Information Theory, Vol. IT-5, No. 2, pp. 52-58 (June 1959).

105. Gimpel, J. F., "A Reduction Technique for Prime Implicant Tables," IEEE Trans. on Electronic Computers, Vol. EC-14, No. 4, pp. 535-541 (August 1965).

106. Glagolev, V. V., "Formulation of Tests for Iterative Networks," Soviet Phys.--Doklady, Vol. 7, No. 6, pp. 480-482, (1962).

107. Gnedenko, B. V., "Duplication with Repair" Engr. Cyb. Vol. 5, pp. 102-108 (1964).

108. Gnedenko, B. V., "Idle Duplication," Engr. Cyb. Vol. 4, pp. 1-9 (1964).

109. Goldberg, J., "Network Schemes for Combined Fault Masking and Replacement," Working paper presented at the Workshop on the Organization of Reliable Automata, Pacific Palisades, California (2-4 February 1966).

110. Goldberg, J., J. A. Baer, and R. C. Minnick, "Development of Techniques for Improving the Reliability of Digital Systems Through Logical Redundancy," Final Report, Phase III, Stanford Research Institute, Menlo Park, California (August 1963).

111. Goldberg, J., R. C. Minnick and W. H. Kautz, "Development of Techniques for Improving the Reliability of Digital Systems Through Logical Redundancy," Final Report, Phase II, Stanford Research Institute, Menlo Park, California (May 1962).

112. Grasselli, A., "The Design of Program-Modifiable Microprogrammed Control Units," IRE Trans. on Electronic Computers, Vol. EC-11, No. 3 (June 1962).

113. Green, M. W., "High Speed Components for Digital Computers," Final Report, Part A, Contract AF 33(616)-5804, SRI Project 2548, Stanford Research Institute, Menlo Park, California (December 1959).

114. Gunderson, D. C., C. W. Hastings, and G. J. Penn, "Spaceborne Memory Organization," Interim Report, Contract NAS 12-38, Honeywell Systems & Research Division, Minneapolis, Minnesota (15 December 1965).

115. Gurzi, K. J., "Estimates for Best Placement of Voters in a Triplicated Network," IEEE Trans. on Electronic Computers, Vol. EC-14, No. 5, pp. 711-717 (October 1965).

116. Haavind, R., "Reaching for the Moon--and Beyond" Electronic Design, Vol. 14, No. 7, pp. 30-37 (29 March 1966).

117. Hamming, R. W., "Error Detecting and Error Correcting Codes" Bell System Tech. J. Vol. 29, pp. 147-160 (1950).

118. Harrison, M. A., Introduction to Switching and Automata Theory (McGraw-Hill Book Co., New York, N.Y., 1965).

119. Harrison, M. A., "On the Error-Correcting Capacity of Finite Automata," Inf. & Control pp. 430-450 (August 1965).

120. Haynes, J. L., "Logic Circuits Using Square-Loop Magnetic Devices: A Survey," IRE Trans. on Electronic Computers, Vol. EC-10, No. 2, pp. 191-203 (June 1961).

121. Haynes, J. L., and R. C. Minnick, "Magnetic Core Access Switches," Tech. Supplement to RADC-TR-61-117A, Stanford Research Institute, Menlo Park, California (May 1961).

122. Heckler, C. H. Jr., and J. A. Baer, "PCM Telemetry: A New Approach Using All-Magnetic Techniques," Phase I Report, Contract NAS1-3380, SRI Project 4687, Stanford Research Institute, Menlo Park, California (June 1964). Also issued as NASA CR-229 (May 1965).

123. Heckler, C. H., Jr., and J. A. Baer, "Feasibility Breadboard of an All-Magnetic PCM Telemetry System," Phase II Report, Contract NAS1-3380, SRI Project 4687, Stanford Research Institute, Menlo Park, California (June 1965).

124. Heckler, C. H., Jr., and J. A. Baer, "All-Magnetic PCM Telemetry: A Review of the System Breadboard," Phase III Report, Contract NAS1-3380, SRI Project 4687, Stanford Research Institute, Menlo Park, California (October 1965). Also issued as NASA CR-435 (April 1966).

125. Henderson, D. S., Logical Design for Arithmetic Units, Ph.D. Thesis (Harvard University, Cambridge, Mass.; May 1960).

126. Henderson, D. S., "Residue Class Error Checking Codes," Paper presented at 16th Annual Meeting of the ACM (1961).

127. Hennie, F. C., "Fault-Detecting Experiments for Sequential Circuits," Proc. of the 5th Annual Symp. on Switching Circuit Theory and Logical Design, Princeton, pp. 95-110 (October 1964).

128. Herron, D. P., "Optimizing Trade-offs of Reliability vs. Weight," IEEE Trans. on Reliability, Vol. R-12, pp. 50-54 (December 1963).

129. Hersperger, S. and C. Caudill, "Apollo PCM Subsystems Come of Age" Missiles & Rockets, pp. 36-39 (28 February 1966).

130. Herwald, S. W., "Evaluation and Reliability, Elect. Eng., Vol. 81, No. 8 pp. 614-618 (August 1962).

131. Herwald, S. W., "Reliability as a Parameter in the Systems Concept," in Systems: Research and Design, by D. P. Eckman, ed. (Proc. of the First Systems Symposium at Case Institute of Technology; John Wiley & Sons, New York, N.Y. 1961).

132. Hill, J. "Failsafe Circuits, "Report 120, Digital Computer Laboratory, University of Illinois, Urbana, Illinois (June 1962).

133. Holland, J., "A Universal Computer Capable of Executing an Arbitrary Number of Subprograms Simultaneously," Proc. of the Eastern Joint Computer Conference, p. 108 (December 1959).

134. M. E. Homan, "A 4-Megacycle 24-bit Checked Binary Adder, Comm. and Elec., pp. 443-450 (September 1961).

135. Osborn, P., "Ferroresonant Flip-Flops," Electronics, pp. 121-123 (April 1952).

136. James, D. C., A. H. Kent, and J. A. Holloway, "Redundancy and the Detection of First Failures," IRE Trans. on Reliability and Control, Vol. RQC-11, pp. 8-27 (October 1962).

137. Jensen, P. A., Bibliography of Redundancy Techniques," in Failure Tolerant Computer Design, W. H. Pierce, (Academic Press, New York, N.Y. 1965).

138. Jensen, P. A., "Bibliography on Redundancy Techniques," in Redundancy Techniques for Computing Systems, R. H. Wilcox and William C. Mann, eds. (Spartan Books, Washington, D.C., 1962).

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

140. Jensen, P., "The Reliability of Redundant Multiple-Line Networks," IEEE Trans. on Reliability, Vol. R-13, pp. 23-33 (March 1964).

141. Jensen, P. A., W. C. Mann, and N. R. Cosgrove, "The Synthesis of Redundant Multiple-Line Networks," First Annual Report Contract Nonr. 3842(00) Westinghouse Elec. Corp. Baltimore, Maryland (May 1963).

142. Johnson, R. A., Abstract of "Information Theory Approach to Diagnosis," IRE Trans. on Reliability and Control, p. 35 (1960).

143. Joseph, E. C., "Self Repairing Multiprocessor Design," Working paper presented at the Workshop on the Organization of Reliable Automata, Pacific Palisades, California (2-4 February 1966).

144. Kampe, T. W., "The Design of a General-purpose Microprogam-Controlled Computer with Elementary Structure," IRE Trans. on Electronic Computers, Vol. EC-9, No. 2 pp. 208-213 (June 1960).

145. Karibiskii, V. V., "Analysis of Systems for Checking Operability and Diagnosing Faults," Automation and Remote Control, Vol. 26, No. 2, pp. 305-310 (1965).

146. Karibskii, V. V., P. P. Parkhomenko and E. S. Sogomonyan, "Some Problems in Checking the Performance and Locating Failures in Finite Automata," Soviet Phys.--Doklady, Vol. 10, No. 3, pp. 182-184 (1965).

147. Kautz, W. H., "Automatic Fault Detection in Combinational Switching Networks," Proc. of the 2nd Annual Symposium on Switching Circuit Theory and Logical Design, Detroit, Michigan, pp. 195-214 (1961).

148. Kautz, W. H., "Codes and Coding Circuitry for Automatic Error Correction within Digital Systems," in Redundancy Techniques for Computing Systems, Wilcox and Mann, eds. (Spartan Press, Washington, D. C. 1962).

149. Kautz, W. H., "Fault Diagnosis in Combinational Networks," Working Paper presented at the Workshop on the Organization of Reliable Automata, Pacific Palisades, California (2-4 February 1966).

150. Kautz, W. H., "Totally Sequential Switching Circuits," in Switching Theory in Space Technology, Aiken and Main, eds., (Stanford University Press, Stanford, Calif. 1963).

151. Kautz, W. H., "Unit Distance Error-Checking Codes," IRE Trans. on Electronic Computers, Vol. EC-7, No. 2 (June 1958).

152. Kautz, W. H., and R. C. Singleton, "Non-Random Superimposed Codes," IEEE Trans. on Information Theory, Vol. IT-10, No. 4, pp. 363-377 (October 1964).

153. Keir, Y. A., P. W. Cheney, and M. Tannenbaum, "Division and Overflow Detection in Residue Number Systems," IRE Trans. on Electronic Computers, Vol. EC-11, No. 4. pp. 501-507 (August 1962).

154. Kel'mans, A. K., "Some Problems of Network Reliability Analysis," Automation and Remote Control, Vol. 26, No. 3, pp. 564-573 (March 1965).

155. Kel'mans, A. K., "On Certain Optimal Problems in the Theory of Reliability for Information-Transmission Systems." Automation and Remote Control, Vol. 25, No. 5, pp. 601-607 (May 1964).

156. Kemp, J. C., "Redundant Digital Systems," in Redundancy Techniques for Computing Systems, Wilcox and Mann, eds. (Spartan Books, Washington, D.C., 1962).

157. Kemp, J. C., W. R. Hiatt and T. M. Grenewitz, "Automatic Recovery from Transient Malfunctions in Space Computers," Tech. Documentary Report No. AL TDR 64-142, General Electric Advanced Electronics Center, Ithaca, N.Y. (August 1964), AD-604 064.

158. Kidd, M. C., "Self-Test of Digital Evaluation Equipment," IEEE Trans. on Aerospace Vol. AS-1, No. 2, pp. 1283-1296 (August 1963).

159. Kilmer, W. L., "An Idealized Overall Error-Correcting Digital Computer having only an Error-Detecting Combinational Part" IRE Trans. on Electronic Computers, Vol. EC-8, No. 3, pp. 321-325 (September 1959).

160. Kime, C. R., "A Failure Detection Method for Sequential Circuits," Technical Report 66-13, Dept. of Elec. Eng., University of Iowa City, Iowa (January 1966).

161. Kime, C. R., "An Organization for Checking Experiments on Sequential Circuits," IEEE Trans. on Electronic Computers, Vol. EC-15, No. 1, pp. 113-115 (February 1966).

162. Kinsht, N. V., "One Procedure for Malfunction Search," Novosibrosk Avtometriya, No. 3, pp. 34-38 (1965). Abstract: USSR Scientific Abstracts, C. C. A. T., No. 11, p. 134.

163. Kletsky, E. J., "Upper Bounds on Mean Life of Self-Repairing Systems," IRE Trans. on Reliability and Quality Control, pp. 43-48 (October 1962).

164. Knox-Seith, J. K., "Improving the Reliability of Digital Systems by Redundancy and Restoring Organs," Tech. Report No. 4816-2 SU-SEL-64-094, Stanford Electronics Laboratory, Stanford University, Stanford, California (August 1964).

165. Kochen, M., "Extension of Moore-Shannon Model for Relay Circuits," IBM J. Res. Dev., Vol. 3, No. 2, pp. 169-186 (April 1959).

166. Kodis, R. D., "Telemetering Buffer Storage for Space Vechicles," Data Systems Engineering, Vol. 18, No. 8, pp. 10-14 (October 1963).

167. Kogan, I. V., "Monitoring the Operation of Logical Devices," Doklady na Mezhdynarodnom Simposiume po Teorii Relenykh Ustroistv i Konechnykh Avtomatov. (Moscow, 1962); Trans: Relay Systems and Finite Automata, pp. 116-137 Burroughs Corp., Paoli, Pa., 1964).

168. Kogan, I. V., "Tests for Nonrepetitious Contact Networks," Problemy Kibernetiki, Vol. 12, pp. 39-44 (1964).

169. Korman, A. G., "Optimal Reserve of Equipment," Engr. Cyb. Vol. 4, pp. 21-33 (1964).

170. Korolenok-Gorskii, L. K., "Some Problems in the Reliability Analysis of Memories," Izd-vo Energiya (Leningrad), pp. 15-27 (1965); Abstract; USSR Scientific Abstracts, C. C. A. T., No. 11, p. 131.

171. Kruus, J., "Upper Bounds for the Mean Life of Self-Repairing Systems," Coordinated Science Laboratory Report R-172, University of Illinois, Urbana, Illinois (July 1963) AD-418 174.

172. Kurdyukov, K. P., "The Reliability of Relay-Contact Networks," Automation and Remote Control, Vol. 21, No. 4, pp. 366-371 (1960). See also Automation Express, Vol. 2, No. 4, pp. 30-34 (1960).

173. Kuznetsov, S. M., "Estimating the Reliability of Automatic Systems from the Results of Testing in Incomplete Set of Equipment," Automation and Remote Control, Vol. 22, No. 8, pp. 990-998 (February 1962).

174. Landers, R. R., "Achieving Higher Reliability Through Self-Repair," IEEE Trans. on Aerospace, Vol. AS-1, pp. 735-46 (August 1963).

175. Lawrence, L. A. J., High-Security Control through Redundant Channels," Control Engineering, Vol. 12, No. 3, pp. 74-79 (March 1965).

176. Lechner, J. A., "Simplified Reliability Calculations for Complicated Systems," IEEE Trans. On Systems Science and Cybernetics, Vol. SSC-1, No. 1, pp. 31-36 (November 1965).

177. Ledley, R. S., and J. B. Wilson, "Integrated Circuitry Implications in Logical Design," in Microelectronics and Large Systems Mathis, Wiley and Spandorfer, eds. (Spartan Books, Washington, D.C., 1965).

178. Lee, C. Y., "Representation of Switching Circuits by Binary Decision Programs," Bell System Tech. J., Vol. 38, pp. 985-999 (1959).

179. Lee, C. Y., and Paul, M. C., "A Content Addressable Distributed Logic Memory with Applications to Information Retrieval," Proc. IEEE, Vol. 51, pp. 924-932 (June 1963).

180. Lee, F. "An Automatic Self-Checking and Fault-Locating Method," IRE Trans. on Electronic Computers, Vol. EC-11, No. 5, pp. 649-654 (October 1962).

181. Levenstein, V. I., "Concerning Self-Adjusting Automata," Doklady na Mezhdynarodnom Simposiume po Teorii Relenykh Ustroistv i Konechnykh Avtomatov (Moscow) pp. 147-192 (1962).

182. Liddell, D. W., "Integration and Automatic Fault Location Techniques in Large Digital Data Systems," Proc. Spring Joint Computer Conference, pp. 213-224 (1962).

183. Liu, C. L. and J. W. S. Liu, "On a Multiplexing Scheme for Threshold Logical Elements," Information and Control, Vol. 8, pp. 282-294 (1965).

184. Lofgren, L., "Automata of High Complexity and Methods of Increasing their Reliability by Redundancy," Information and Control, Vol. 1, pp. 126-147 (1958).

185. Lowenschuss, O., "Restoring Organs in Redundant Automata," Information and Control, Vol. 2, pp. 113-136 (1959).

186. Lowrie, R. L., "High-Reliability Computers Using Duplex Redundancy" Electronic Industries, pp. 116-128 (August 1963).

187. Lyabutov, Ya. V., "Optimal Procedures for Localization of Breakdown in a Modularized Radioelectronic System," Engr. Cyb., No. 4, pp. 14-21 (1964).

188. Lyons, R. E, and W. Vanderkulk, "The Use of Triple-Modular Redundancy to Improve Computer Reliability," IBM J. Res. Dev., Vol. 6, No. 2, pp. 200-209 (April 1962).

189. Madatyan, Kh. A., "Synthesis of Circuits which Correct the Breaking of Contacts," Soviet Phys.--Doklady, Vol. 9, No. 11, pp. 949-951 (May 1965).

190. Malev, V. V., "Reliability of Reserve (Redundant) Systems with Periodic Maintenance," Engr. Cyb., No. 3, pp. 46-49 (1964).

191. Malikov, I. M., Rokhmistrov, A. N., Determining Computer Reliability, Leningradskii Inzhenerno-Ekonomicheskii Institut Trudy, No. 55 (1965); Abstract: USSR Scientific Abstracts, C.C.A.T., No. 13, p. 95.

192. Maling, K., and E. L. Allen, "A Computer Organization and Programming System for Automated Maintenance," IEEE Trans. on Electronic Computers, pp. 887-895 (December 1963).

193. Malyugin, V. D., "Reliability of Switching Circuits," Automation and Remote Control Vol. 25, No. 9, pp. 1235-1242 (September 1964).

194. Mandelbaum, D., "Arithmetic Error-Detecting Codes for Communications Links Involving Computers," IEEE Trans. on Communications Technology, Vol. COM 13, pp. 165-171 (June 1965).

195. Mann, W. C., "Restorative Processes for Redundant Computing Systems," in Redundancy Techniques for Computing Systems, Wilcox and Mann, eds., pp. 267-284 (Spartan Books, Washington, D.C., 1962).

196. Mann, W. C., "Systematically Introduced Redundancy in Logical Systems," IRE International Conv. Record, Pt. 2, pp. 241-63 (March 1961).

197. Manning, E., "On Self-Diagnosis of Large Multiprocessor Computers," working paper presented at Workshop on the Organization of Reliable Automata, Pacific Palisades, California (2-4 February 1966).

198. Manning, E. G., "Self-Diagnosis of Electronic Computers--An Experimental Study," Coordinated Science Laboratory Report R-259, University of Illinois, Urbana, Illinois (May 1965).

199. Marcus, I. R., "Transistor Flip-Flop and Ring Counter with Nonvolatile Memory," TR-1198, Harry Diamond Laboratory, Washington, D. C., HDL Project 46300, Army Project 1P523801A300 (February 1964).

200. Massey, J. L., "Error-Correcting Codes Applied to Computer Technology," Proc. National Electronics Conference, Vol. 19, pp. 142-147 (1963).

201. Massey, J. L., "Survey of Residue Coding for Arithmetic Errors," ICC Bulletin Vol. 3, No. 4, (October 1964).

Massey, J. L., Threshold Decoding, see item 352 of this Bibliography.

202. Masters, C. G., "Reliability Estimation for Redundant Systems," Working Paper presented at the Workshop on the Organization of Reliable Automata, Pacific Palisades, California (2-4 February 1966).

203. Maxwell, L., "Synthesis of Contact Networks from Prescribed Probability Functions," J. Franklin Inst., Vol. 28, No. 3, pp. 214-234 (March 1966).

204. Merekin, Yu. V. "Arithmetical Forms of Writing Boolean Expressions and Their Use in Calculating Circuit Reliability," Vyshislitelnye Sistemy, No. 7, pp. 13-23 (1963).

205. McCluskey, E. J., Jr., "Determination of Redundancies in a Set of Patterns," IRE Trans. on Information Theory, Vol. IT-3, No. 2, p. 167 (June 1957).

206. McCluskey, E. J., Jr., "Minimization of Boolean Functions" Bell System Tech. J., Vol. 35, pp. 1417-1444 (November 1956).

207. Mikhaelov, L. N., "On the Reliability of Multichannel Systems with Continuous Service," Automation and Remote Control, Vol. 23, No. 11 (November 1962).

208. Mina, K. V.,and E. E. Newhall, "Magnetic Circuits as Logis Packages," presented at the 1966 INTERMAG Conference, Stuttgart, Germany (Publication in IEEE Trans. on Magnetics in 1966 expected).

209. Mine, H., "Reliability of Physical Systems," IRE Trans. on Circuit Theory, Vol. CT-6 (Special Supplement), p. 138 (May 1959).

210. Minnick, R. C., "Cobweb Cellular Arrays," Proc. Fall Joint Computer Conference, Vol. 27, Part 1, pp. 327-341 (1965).

211. Minnick, R. C., "Cutpoint Cellular Logic," IEEE Trans. on Electronic Computers, Vol. EC-13, No. 6, pp. 686-698 (December 1964).

212. Minnick, R. C., et al., "Cellular Arrays for Logic and Storage," Final Report, Contract AF 19(628)-4233, SRI Project 4087, Stanford Research Institute, Menlo Park, California (April 1966).

213. Moore, E. F., "Gedanken-Experiments on Sequential Machines" Automata Studies, pp. 129-153 (Princeton University Press, Princeton, N.J., 1956).

214. Moore, E. F., and C. E. Shannon, "Reliable Circuits Using Less Reliable Relays" J. Franklin Inst., Vol. 262, pp. 191-208 (September 1956); pp. 281-297 (October 1956).

215. R. C. Moore, "Application of Residue Class Codes to the Cape System," Tech. Doc. RORD 6359, Minneapolis-Honeywell Inc., Military Products Group Research Laboratory, St. Paul, Minnesota (September 1964).

216. R. C. Moore, "An Investigation of Residue Check Symbols to Improve Residue Computer Reliability," Tech. Doc.R-Rd 6352, Minneapolis-Honeywell Inc., Military Products Group Research Laboratory, St. Paul, Minnesota (December 1964).

217. Morrison, D. F., and H. A. David, "The Life Distribution and Reliability of a System with Spare Components," Ann. Math. Stat., Vol. 31, pp. 1084-1094 (December 1960).

218. Moscowitz, F., "The Analysis of Redundancy Networks," AIEE Trans. on Comm & Electr, No. 39, p. 627-632 (November 1958).

219. Moscowitz, F., "The Statistical Analysis of Redundant Systems," IRE International Convention Record, Part 6, pp. 78-89 (1960).

220. Muchnik, A. A., and S. G. Gindikin, "The Completeness of a System of Unreliable Elements which Realize Functions in Logical Algebra," Soviet Phys.--Doklady, Vol. 7, No. 6, pp. 477-479 (1962).

221. Muller, D., "Asynchronous Logic and Application to Information Processing," Switching Theory in Space Technology, Aiken and Main, eds. (Stanford University Press, Stanford, California, 1963).

222. Mullin, A. A., "On the Nature of Reliability of Automata," in Redundancy Techniques for Computing Systems, Wilcox and Mann, eds. (Spartan Books, Washington D.C., 1962).

223. Mullin, A. A., "Reliable Stochastic Sequential Switching," Communication and Electronics (November 1958).

224. Muroga, S., "Preliminary Study of the Probabilistic Behavior of a Digital Network with Majority Decision Elements," Rome Air Development Center, Technical Note RADC-TN-60-146 (August 1960).

225. Muth, E. J., "Reliability of a System Having Standby Spare Plus Multiple-Repair Capability," IEEE International Convention Record, Part 10, pp. 9-15 (1965).

226. Naumchenko, V. V., "The Reliability of Ideally Redundant Systems," Automation and Remote Control, Vol. 25, No. 3, pp. 376-378 (March 1964).

227. Nechiporuk, E. I., "Self-Correcting Diode Networks," Soviet Phys.--Doklady, Vol. 9, No. 6, pp. 422-425 (December 1964).

228. Nerber, P. O., "'Power-off' Time Impact on Reliability Estimates," IEEE Int. Conv. Rec. Part 10, pp. 1-8, (March 1965).

229. Nichols, A. J., III, "Modular Synthesis of Sequential Machines," IEEE Conf. Record on Switching Circuit Theory and Logical Design, pp. 62-70 (October 1965).

230. Nitzan, D. "Flux-Switching in Multipath Cores," Report 1, Contract 950095 under NASw-6, Jet Propulsion Laboratory, Pasadena, Calif., SRI Project 3696, Stanford Research Institute, Menlo Park, California (November 1961).

231. Olefir, A. K., "Search for Faulty Elements in a System of Two Computers" Vychislitelnye Sistemy, No. 6, pp. 21-31 (1963). Abstract: USSR Scientific Abstracts, C.C.A.T., No. 9, p. 59.

232. Ord-Smith, R. J., "An Extension of Block Design Methods and an Application in the Construction of Redundant Fault Reducing Circuits for Computers," Computer Journal, Vol. 8, No. 1, pp. 28-32 (April 1965).

233. Ore, O., Graphs and Their Uses (L. W. Singer Co., New York, N.Y. 1963).

234. Ostianu, V. M., "Binary Signal Correction Networks," Automation and Remote Control, Vol. 21, No. 5, pp. 426-431 (1960); Automation Express, Vol. 3, No. 2, pp. 12-15 (1960).

235. Paderno, I. P., "Reliability of Systems Containing Reserve (Redundant) Equipment," Eng. Cyb., No. 2, pp. 33-37 (1963).

236. Paynter, D. A., and V. P. Mathis, "Redundancy Techniques in Reliable Power Supply Design" Proc. 1961 International Solid State Circuit Conf., pp. 50-51.

237. Peterson, W. W., "Binary Controls for Error Control," AIEE Trans. on Communications and Electronics, pp. 648-652 (January 1962).

238. Peterson, W. W., Error-Correcting Codes (John Wiley and Sons and MIT Press, New York, N.Y., 1961).

239. Peterson, W. W., "On Checking an Adder," IBM J. Res. Dev., Vol. 2, No. 2, pp. 166-168 (April 1958).

240. Pierce, W. H., "Adaptive Vote-Takers Improve the Use of Redundancy" in Redundancy Techniques for Computing Systems, pp. 229-250 (Spartan Books, Washington, D.C., 1962).

241. Pierce, W. H., "A Proposed System of Redundancy to Improve the Reliability of Digital Computers," Appl. Electr. Lab Report No. TR 1552-1, Stanford University, Stanford, California (July 1960).

242. Pierce, W. H., "Asymptotic Properties of Systems Synthesized for Maximum Reliability," Inf. & Contr., Vol. 7, No. 3, pp. 340-359 (September 1964).

243. Pierce, W. H., Failure-tolerant Computer Design (Academic Press, New York, N.Y., 1965).

244. Pierce, W. H., "Interwoven Redundant Logic," J. Franklin Institute, Vol. 277, No. 1, pp. 55-85 (January 1964).

245. Pierce, W. H., "Redundancy in Computers," Scientific American, Vol. 210, pp. 103-109 (February 1964).

246. Poage, J. F., "Derivation of Optimum Tests to Detect Faults in Combinational Circuits" Proceedings of the Symposium on Mathematical Theory of Automata, pp. 483-528 (Polytechnic Press, New York, N.Y., April 1963).

247. Poage, J. F., and E. J. McCluskey, "Derivation of Optimum Test Sequences for Sequential Machines," Proc. of the 5th Annual Symp. on Switching Circuit Theory and Logical Design, Princeton, pp. 121-132 (October 1964).

248. Pohm, A. V., R. J. Zingg, J. H. Hoper and R. M. Stewart, Jr., "Analysis of 10^8 Element Magnetic Film Memory Systems," Proc. Intermag Conf., pp. 5-3-1 to 5-3-4 (1964).

249. Pollyak, Yu. G., "On Error in Forecasting Reliability based on the Statistical Relationship between Failures of Elements." Elektrosvyaz, Vol. 17, No. 4, (April 1963).

250. Polovko, A. M., "Calculating the Dependability of Complex Automatic Systems," Izv. AN SSR, Ofn. Energetika i Avtomatika, Issue No. 5, pp. 174-178 (1960); translated in ASTIA 267736.

251. Polovko, A. M., Zaynashev, N. K., "Increasing the Reliability of Apparatus by a Combination of General Reserve by Replacement and a Separate System with a Constantly Connected Reserve." Engr. Cyb., No. 5, pp. 114-120 (1960).

252. Potapov, Yu. G., and S. V. Yablonskii, "On the Synthesis of Self-Correcting Relay Circuits," Soviet Physics Doklady, Vol. 5, No. 51, pp. 932-935 (1960).

253. Potter, G. B., and J. Mendelson, "Integrated Scratch Pads Sire New Generation of Computers," Electronics, pp. 118-126 (April 1966).

254. Pyne, I. B., and E. J. McCluskey, Jr., "The Reduction of Redundancy in Solving Prime Implicant Tables," IRE Trans. on Electronic Computers, Vol. EC-11, No. 4, pp. 473-482 (August 1962).

255. Raikin, A. L., "The Problem of Synthesizing a Surplus Structure in the Presence of Restrictions in the Form of Linear Inequalities," in Relay Systems and Finite Automata, pp. 531-538 (Burroughs Corp., Paoli, Pennsylvania, 1964).

256. Raikin, A. L., "Determining the Optimal Reserve for a System While Taking Into Account the Damage of Blocks in the Reserve Mode," Automation and Remote Control, Vol. 23, No. 11, pp. 1437-1442 (1963).

257. Raikin, A. L., "Additional Estimates for a Fractional Redundance Scheme," Automation and Remote Control, Vol. 25, No. 4, pp. 533-535 (April 1964).

258. Raikin, A. L., A. F. Rubtsov, and V. S. Penin, "Problem of Reliability of Technical Systems with Regularly Renewable Reserve" Engr. Cyb., No. 4, pp. 9-14 (1964).

259. Raikin, A. L., "Redundancy Optimization in the Presence of Constraints," Automation and Remote Control, Vol. 26, No. 2, pp. 381-392 (February 1965).

260. Raikin, A. L., "Reliability of Passive Redundancy Circuits with Permanently Connected Redundant Elements in the Case of Redistribution of Loads or Voltages." Automation and Remote Control, Vol. 24, No. 4, pp. 517-521 (April 1963).

261. Randa, G. C. and C. V. McNeil, "Self-Correcting Memory--The Basis of a Reliable Computer," Electronic Design, Vol. 13, No. 18, p. 28 (1965).

262. Rau, J. G. "Redundancy and Trichotomous Systems," J. Soc. Ind. Appl. Math., Vol. 12, No. 4, pp. 827-837 (December 1964).

263. Ray-Chaudhuri, D. K., "On the Construction of Minimally Redundant Reliable System Designs," Bell System Tech. J., Vol. 40, pp. 595-611 (March 1961).

264. Rhodes, L. J., "Effects of Failure Modes on Redundancy," Proc. 10th Natl. Symp. on Rel. and Qual. Control, pp. 360-364 (Washington, D.C., January 1964).

265. Rieff, G. A., "Interplanetary Spacecraft Telecommunication Systems," IEEE Spectrum (April 1966).

266. Robins, R. S., "On Models for Reliability Prediction" IRE Trans. on Reliability and Quality Control, Vol. 11, No. 1, pp. 33-43 (May 1962).

267. Rogers, J., and J. King, "The Case for Magnetic Logic," Electronics, Vol. 34, No. 17, pp. 40-47 (June 1964).

268. Rosenheim, D. E., and R. B. Ash, "Increasing Reliability by Use of Redundant Machines," IRE Trans. on Electronic Computers, Vol. EC-8, pp. 125-130 (June 1959).

269. Rostkovskaya, S. E., "On the Probability Characteristics of Component Reliability," Automation and Remote Control, Vol. 22, No. 11 (April 1962).

270. Roth, P. J., "Diagnosis of Automat Failures: A Calculus and a Method," IBM J. Res. Dev., Vol. 10, No. 4, pp. 278-291 (July 1966).

271. Rothstein, J., "Residues of Binary Numbers Modulo Three," IRE Trans. on Electronic Computers, Vol. EC-8, No. 2, p. 229 (June 1959).

272. Rubin, D. "Placement of Voters in Modularly Redundant Digital Systems," Working paper presented at the Workshop on the Organization of Reliable Automata, Pacific Palisades, California, (2-4 February 1966).

273. Rubio, J., "A Study of Some Self-Correcting Sequential Networks," Philips Res. Reports, Vol. 17, No. 4, pp. 315-328 (August 1962).

274. Russo, R. L., "Synthesis of Error-Tolerant Counters using Minimum Distance Three State Assignments:, IEEE Trans. on Electronic Computers, Vol. EC-14, pp. 359-366 (June 1965).

275. Sagalovich, Yu L. "A Method of Increasing the Reliability of Finite Automata," Problemi Peredachi Informatsii, Vol. 1, No. 2, pp. 27-35 (1965). Abstract: USSR Scientific Abstracts, C.C.A.T., No. 11, p. 134.

276. Schneider, S. and D. H. Wagner, "Error Detection in Redundant Systems," Proc. of the Western Joint Computer Conference, pp. 115-21 (1957).

277. Schultz, W. R., and J. H. Brenner, "Application of Multi-Aperture Devices in Space-borne Digital Control Equipment," Conference Proceedings - National Convention on Military Electronics, (IEEE), pp. 425-429 (September 1963).

278. Schwartz, Samuel A., "High-Efficiency Power Supply Uses Micrologic," Electronic Industries (February 1966).

279. Seshu, S. "On an Improved Diagnosis Program" IEEE Trans. on Electronic Computers, Vol. EC-14, No. 1, pp. 76-79 (February 1965).

280. Seshu, S., and Freeman, D. N., "The Diagnosis of Asynchronous Sequential Switching Systems": IRE Trans. on Electronic Computers, Vol. EC-11, pp. 459-465 (August 1962).

281. Sethares, G. "Closed Sets of Boolean Functions and the Reliability Problem for Polyfunctional Nets," IEEE Trans. on Electronic Computers, Vol. EC-15, No. 1, pp. 115-117 (February 1966).

282. Shcherbakov, O. V., "Determination of the Reliability of Systems with Arbitrary Laws Governing Repair," Automation and Remote Control, Vol. 25, No. 8, pp. 1091-1095 (August 1964).

283. Short, R. A., "A Theory of Relations Between Sequential and Combinational Realizations of Switching Functions," No. 098-1, Stanford Electronics Laboratories, Stanford University, Stanford, California (1960).

284. Short, R. A., "The Design of Complementary-Output Networks," IRE Trans. on Electronic Computers, Vol. EC-11, pp. 743-752 (1962).

285. Short, R. A., "Two-Rail Cellular Cascades," Proc. of the 1965 Fall Joint Computer Conference, AFIPS, Vol. 27, Part 1, pp. 355-369 (1965).

286. Short, R. A., and W. H. Kautz, "A Survey of Soviet Activities in Reliability," Working paper presented at the Workshop on the Organization of Reliable Automata, Pacific Palisades, California (2-4 February 1966).

287. Sindeev, I. M., "Synthesizing Logical Schemes for Failure Detection and Control of Complex Systems," Eng. Cyb., No. 2, pp. 16-23 (1963).

288. Slotnick, D. L., W. C. Borck and R. C. McReynolds, "The Solomon Computer," Proc. Fall Joint Computer Conference (AFIPS) Vol. 22, p. 97 (1962).

289. Sinitsa, M., "On Reservation by Replacement Method," IRE Trans. on Reliability and Quality Control, Vol. 9, pp. 6-13 (April 1960).

290. Smolitskii, Kh. L., and P. A. Chukreev, "The Problem of Optimum Redundancy of an Apparatus," Izvestiya Akademii Nauk SSSR, Otdelenie Teknicheskikh Nauk, Energetika i Avtomatika, No. 4 (1959).

291. Sogomonyan, E. S., "Monitoring Operability & Finding Failures in Functionally Connected Systems," Automation and Remote Control, Vol. 25, No. 6, pp. 874-882 (June 1964).

292. Spitzer, C. F., "The Ferroresonant Trigger Pair: Analysis and Design," Communication and Electronics, No. 26, pp. 407-416 (September 1956).

293. Steinbuch, K. and F. Zendeh, "Self-Correcting Translator Circuits," Proc. IFIP Congress, 1962, pp. 359-365 (North Holland Publishing Co., Amsterdam, Holland, 1963).

294. Stuart-Williams, R., "Magnetic Cores, Characteristics and Applications," Automatic Control, Vol. 14, pp. 44-47 (April 1961).

295. Susskind, A. K., D. R. Haring, C. L. Liu, and K. S. Menger, "Synthesis of Sequential Switching Networks," Report ESL-FR-216, Contract AF 33(657)-11677, MIT Research Laboratory of Electronics, Cambridge, Mass. (1964), AD-608 881.

296. Svechinskii, V. B., "Self-Correcting Design of Finite Automata," Automation and Remote Control, Vol. 25, No. 5, pp. 623-628 (1964) and Automation Express, Vol. 7, No. 1, p. 24.

297. Svoboda, A., and M. Valach, "Operational Circuits (Operatorove Obovody), Stroje Na Zpracovani Informaci, Vol. III, Prague, Czechoslovakia, pp. 247-295 (1955) (In Czechoslovakian). Air Technical Information Center, WADD Translation No. F-TS-10126/V.

298. Szabo, N., "Sign Detection in Nonredundant Residue Systems," IRE Trans. on Electronic Computers, Vol. EC-11, No. 4, pp. 494-500 (August 1962).

299. Teoste, R., "Design of a Repairable Redundant Computer," IRE Trans. on Electronic Computers, Vol. EC-11, No. 5, pp. 643-649 (October 1962).

300. Teoste, R., "Digital Circuit Redundancy," IEEE Trans. on Reliability, Vol. R-13, pp. 42-61 (June 1964).

301. Teoste, R., "Reliability of Redundant Computers," Lincoln Lab Report No. 21G-0029 (ASTIA 260494), MIT, Lexington, Mass. (March 1961).

302. Terris, I., "Self-Repairing Digital Computers," Hughes Aircraft Co. Technical Report No. SSD-TR-65-52 (June 1965), AD 466123.

303. Terris, I., and M. A. Melkanoff, "Investigation and Simulation of a Self-Repairing Digital Computer," 1965 IEEE Conf. Record on Switching Circuit Theory and Logical Design (October 1965).

304. Tooley, John, "Network Coding for Reliability," Comm. Elec., pp. 407-413 (January 1963).

305. Tryon, John, "Quadded Logic," in Redundancy Techniques for Computing Systems, Wilcox and Mann, eds., pp. 205-228 (Spartan Books, Washington, D.C., 1962).

306. Tsertsvadze, G. N., "Stochastic Automata and the Problem of Constructing Reliable Automata from Unreliable Elements I," Automation and Remote Control, Vol. 25, No. 2, pp. 198-210 (February 1964).

307. Tsertsvadze, G. N., "Stochastic Automata and the Problem of Constructing Reliable Automata from Unreliable Elements II," Automation and Remote Control, Vol. 25, No. 4, pp. 458-464 (1964).

308. Tsiang, S. H., and W. Ulrich, "Automatic Trouble Diagnosis of Complex Logic Circuits," Comm. Elec. pp. 575-583 (January 1963).

309. Tsiang, S. H., and W. Ulrich, "Automatic Trouble Diagnosis of Complex Logic Circuits," Bell System Tech J., Vol. 41, No. 4, pp. 1177-1200 (July 1962).

310. Urbano, R. H., "Matrix Criteria for Arbitrary Reliability in Iterated Neural Nets," IEEE Trans. on Electronic Computers, Vol. EC-14, pp. 627-629 (August 1965).

311. Urbano, R. H., "Reliability, Redundancy, Capacity and Universality in Polyfunctional Nets," Draft Paper presented at the "Workshop on the Organization of Reliable Automata," Pacific Palisades, California, 2-4 February 1966.

312. Urbano, R. H., "Some New Results on the Convergence, Oscillation, and Reliability of Polyfunctional Nets," IEEE Trans. on Electronic Computers, Vol. EC-14, No. 6, pp. 769-781 (December 1965).

313. Vaksov, V. V., "On Tests for Irreversible Switching Circuits," Automation and Remote Control, Vol. 26, No. 3, pp. 515-518 (March 1965).

314. Van De Riet, E. K., D. R. Bennion, and J. M. Yarborough, "Feasibility Study for Reliable Magnetic Connection Switch," Final Report--Phase I, Contract 951232 under NAS7-100, Jet Propulsion Laboratory, Pasadena, California, SRI Project 5669, Stanford Research Institute, Menlo Park, California (February 1966).

315. Varshamov, R. R. and V. M. Ostianu, "Application of the Theory of Finite Fields to the Synthesis of Relay Mechanisms Using Structural Redundancy," Relay Systems and Finite Automata, pp. 369-372 (Burroughs Corp., Paoli, Pennsylvania, 1964).

316. Vedeshnikov, V. A., A. F. Volkov, V. D. Zenkin, V. A. Trapeznikov, and T. A. Turkovskaya, "A Digital Computer with Programmed Control," Byulletin Izobretenii i Tovarnykh Znakov, No. 18, pp. 91-92 (1964).

317. Virene, E. P., "Reliability Abstracts and Technical Reviews," Vol. 5, No. 7, Serial Numbers 2051-2100 NASA (July 1965).

318. Von Neumann, J., "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," Annals of Mathematical Studies, No. 34, pp. 43-98 (Princeton University Press, Princeton, New Jersey, 1956).

319. Walendziewicz, "The D210 Magnetic Computer," Proceedings 1962 IRE Spaceborne Computer Engineering Conference, pp. 117-127 (1962).

320. Watson, R. W., "Error Detection and Correction and other Residue Interacting Operations in a Redundant Residue Number System," Ph.D. Thesis, University of California, Berkeley (1965).

321. Watson, R. W., "Preliminary Report on Operations in Redundant Residue Number System," Tech. Document R-RD 6330, Honeywell Military Products Group, St. Paul, Minn. (April 1964).

322. Webster, L. R., "Choosing Optimum System Configurations, Proc. Tenth Annual Symposium on Reliability and Quality Control, pp. 345-359 (1964).

323. Webster, L. R., "Optimum Redundancy for a Satellite Power System," Proc. Eleventh Annual Symposium on Reliability and Quality Control, pp. 568-574 (1965).

324. Weinstock, G. D., "Topological Analysis of Non-Series-Parallel Redundant Networks," IEEE Intl. Conv. Record, Part 6, pp. 34-40 (1963).

325. Weiss, G. H., "A Survey of Some Mathematical Models in the Theory of Reliability," in Statistical Theory of Reliability, Marvin Zelen, ed. (University of Wisconsin Press, Madison, Wis., 1963).

326. Weiss, G. H., "Optimum Periodic Inspection Programs for Randomly Failing Equipment," J. Res. National Bureau of Standards, Vol. 67B, pp. 223-228 (1963).

327. Weiss, G. H., "Reliability of a System in Which Spare Parts Deteriorate in Storage," J. Res. National Bureau of Standards, Vol. 66B, No. 4, pp. 157-168 (1962).

328. Welch, P. D., "On the Reliability of Polymorphic Systems," IBM Systems Journal, Vol. 4, No. 1, pp. 43-52 (1965).

329. Wilcox, R. H. and W. C. Mann (eds) Redundancy Techniques for Computing Systems (Spartan Book Co., Washington, D.C., 1962).

330. Wilkes, M. V., W. Renwick, and D. J. Wheeler, "The Design of the Control Unit of an Electronic Digital Computer," Proc. IEE, Vol. 105, Pt. B., pp. 121-128 (March 1958).

331. Winograd, S., "Input-Error-Limiting Automata," J. Assoc. Comp. Mach., Vol. 11, pp. 338-351 (1964).

332. Winograd, S., "Redundancy and Complexity of Logical Elements," Information and Control, Vol. 5, pp. 177-194 (1963).

333. Winograd, S. and J. D. Cowan, Reliable Computation in the Presence of Noise (MIT Press, Cambridge, Mass., 1963).

334. Winter, B. B., "Introduction to Cyclic Replacement Systems," IEEE Trans. on Reliability, Vol. R-12, pp. 36-40 (December 1963).

335. Yablonskii, S. V., and I. A. Chegis, "Tests for Electrical Networks," Uspekhi Matematicheskii Nauk, Vol. 10, No. 4, pp. 182-184 (1955).

336. Zelen, Marvin, ed., Statistical Theory of Reliability (University of Wisconsin Press, Madison, Wis., 1963).

337. Zhozhikashvili, V. A., and A. L. Raikin, "Analysis of Reliability of Systems with Fault Signaling," Automation and Remote Control, Vol. 23, No. 3, pp. 352-357 (March 1962).

338. "Astronavigation Computer Research," ASD-TDR-63-337, Contract AF 33(616)-8268, Sperry Gyroscope Co., Great Neck, N.Y., for Research and Technology Division, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio (October 1963), AD-423 704.

339. "Data Sheets for BR-801 Regulator," Bunker-Ramo Corporation, Canoga Park, California (June 1966).

340. "Final Report for a Magnetic Computer Study," SSD-TR-64-208, Contract AF 04(695)-320, Burroughs Corporation for Space Systems Division, Air Force Systems Los Angeles Air Force Station, Los Angeles, California (September 1964), AD 447 910.

341. "Final Report on Phase II of Research on Failure Free Systems," NASA CR-343, Contract NASw-572, Westinghouse Electric Corporation, Baltimore, Maryland (December 1965).

342. "Magnetic Logic in Space (A Report of Applications and Reliability)," NASA Accession Number N64-21384, DI/AN Controls, Inc. (January 1964).

343. "Micropower Functional Electronic Blocks," Technical Reports 2 and 3, Fairchild Semiconductor Division, Syosset, N.Y. (January and March 1966).

344. "Research on Failure Free Systems," NASA CR-105, Contract NASw-572, "Westinghouse Electric Corporation, Baltimore, Maryland (November 1964).

345. "Survey/Study of the Interconnection Problems in Microelectronics," Final Report of First Phase, NASA Contract NASw-919, Moore-Peterson Associates, Santa Barbara, California (20 July 1965).

346. "Technology Survey--Microelectronics in Space Research," NASA SP-5031, Research Triangle Institute (August 1965).

347. "The Development and Use of Multiaperture-Core Flux Logic Devices to Perform Logical Functions in Digital Data Processing," Vol. 1, WADC-TR-59-648, Contract AF 33(600)-31315, International Business Machines Corp., Owego, N.Y. (December 1959), AD-231 176.

348. "The Development and Use of Multiaperture-Core Logic Devices to Perform Logical Functions in Digital Data Processing," Vol. 2, WADC-TR-59-648, Contract AF 33(600)-31315, International Business Machines Corp., Owego, N.Y. (December 1959). AD-231 177.

349. "The RW-400--A New Polymorphic Data System," Datamation, Vol. 6, No. 1, pp. 8-14 (January/February 1960).

350. Fedderson, A. P., and A. C. Shershin, "Redundancy Concepts and Optimality Considerations," TM 024-43-RSA-16, Autonetics Div., Downey, California (18 March 1946), AD-459 712.

351. Zubova, A. F., "On a Method of Calculating the Reliability of Redundant Systems," Automation and Remote Control, Vol. 26, No. 4, p. 701 (April-September 1965).

352. Massey, J. L., Threshold Decoding (MIT Press, Cambridge, Massachusetts, 1963).

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Stanford Research Institute 333 Ravenswood Avenue Menlo Park, California 94025	UNCLASSIFIED
	2b. GROUP n/a

3. REPORT TITLE

TECHNIQUES FOR THE REALIZATION OF ULTRA-RELIABLE SPACEBORNE COMPUTERS

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Final Report--Phase 1

5. AUTHOR(S) (First name, middle initial, last name)

Jacob Goldberg, Karl N. Levitt, Robert A. Short

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
September 1966	400	352

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
NAS 12-33 b. PROJECT NO.	Final Report--Phase 1 SRI Project 5580
c. d.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	National Aeronautics and Space Administration, Electronics Research Center, 575 Technology Square, Cambridge, Mass. 02139

13. ABSTRACT

This is a report of a study of techniques for the realization of ultrareliable, high-performance, spaceborne computers. The study included the evaluation of, and several new contributions to, the most significant known techniques and the proposal and investigation of several promising new techniques. The state of the art of existing redundancy techniques for fault-detecting and fault-masking is assessed, with special emphasis on multiple-line voting redundancy, error-correcting codes, and redundant-state schemes for sequential networks. A number of directions for the improvement of these techniques are described. Significant potential improvements in reliability are available in designs allowing for a high degree of reconfigurability in structure and programs, and system schemes and design techniques needed for such behavior are proposed and investigated. In particular, we discuss the design of minimal test schedules for fault detection and diagnosis, the design of highly modular processing networks and of programmable interconnection networks, and the overall organization of maintenance and computation functions in a computer system. The application of error control techniques to memory systems and to power supplies is considered, and the possible use of all-magnetic logic networks is examined. Included in the report is a critical and selective survey of the literature that is relative to the attainment of reliable systems and networks through the judicious use of redundant structures. Finally, recommendations are made for further research into the development of techniques for ultrareliable system design.

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
spaceborne computers						
reliability						
redundancy						
fault detection						
fault diagnosis						
fault masking						
reconfigurable computers						
error-correcting codes						
reliable power supplies						
reliable memories						
literature on redundancy						