REPORT SAMSO-TR-77-105

Fault-Tolerant Computers for Spacecraft

Advanced Programs Division
The Aerospace Corporation
El Segundo, Calif. 90245

6 June 1977

Prepared for

SPACE AND MISSILE SYSTEMS ORGANIZATION
AIR FORCE SYSTEMS COMMAND
Los Angeles Air Force Station
P.O. Box 92960, Worldway Postal Center
Los Angeles, Calif. 90009

This final report was submitted by The Aerospace Corporation, El Segundo, CA 90245, under Contract F04701-76-C-0077 with the Space and Missile Systems Organization, Deputy for Advanced Space Programs, P. O. Box 92960, Worldway Postal Center, Los Angeles, CA 90009. It was reviewed and approved for The Aerospace Corporation by G. W. Anderson and G. P. Millburn, Development Operations. Capt. Larry W. Kern, SAMSO/YAD, was the project officer.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication. Publication of this report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.

Larry W. Kern

Larry W. Kern, Capt, USAF
Project Officer
Computer Technology Division
Deputy for Advanced Space Programs

Frank P. Dyke

Frank P. Dyke, Lt Col, USAF
Chief, Computer Technology Division
Deputy for Advanced Space Programs

FOR THE COMMANDER

Floyd Stuart

Floyd Stuart, Col, USAF
Deputy for Advanced Space Programs

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: SAMSO TR-77-105	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER: Final Rept.	
4. TITLE (and Subtitle): Fault-Tolerant Computers for Spacecraft	5. TYPE OF REPORT & PERIOD COVERED: Interim Oct 76-Feb 77
	6. PERFORMING ORG. REPORT NUMBER: TR-0077(2161-02)-1
7. AUTHOR(s): H. Hecht	8. CONTRACT OR GRANT NUMBER(s): F04701-76-C-0077
9. PERFORMING ORGANIZATION NAME AND ADDRESS: The Aerospace Corporation, El Segundo, CA 90245	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Space and Missile Systems Organization, Los Angeles Air Force Station, Los Angeles, CA 90009	12. REPORT DATE: 6 June 1977
	13. NUMBER OF PAGES: 48
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): 49 p.	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Fault-tolerant computers
Spacecraft computers
Computer reliability
Reliability testing

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Recent advances in micro-electronics and in computer architecture have led to renewed interest in fault-tolerant computers for spacecraft applications. The general field of fault-tolerant computers is surveyed, and concepts of static and dynamic redundancy, error detection by code or multiple processing, and the significance of low-level redundancy are explained. Prior efforts in the design of fault-tolerant computers for space

DD FORM 1473 (FACSIMILE)

410 271

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19. KEY WORDS (Continued)

20. ABSTRACT (Continued)

applications are surveyed with processing capability, weight, and reliability estimates listed for a number of representative efforts.

The SAMSO Fault-Tolerant Spaceborne Computer (FTSC) is the most current of these developments, and it is described in greater detail together with the proposed reliability test methodology for the prototype. Recent developments in reliability estimation, employment of microprocessors, and fault-tolerant software that are related to this field are briefly discussed.

TABLE OF CONTENTS

SECTION I

I. INTRODUCTION

Fault-tolerance techniques for spacecraft computers have been under investigation since the early 1960s, motivated by the desire for greater reliability than can be provided in conventional (simplex) computers. Yet, to date, fault-tolerance techniques have gained only modest acceptance in operational spacecraft. The obstacles have been primarily the large development cost and the on-board resources required for such a computer. The computational needs have been met either by relegating most of the data processing to the ground or by using several on-board computers with the capability of switching from one to the other by ground command.

The obstacles to employment of fault-tolerant computers on spacecraft have now been considerably reduced due to advances in computer architecture and coding theory and particularly due to the tremendous progress in semiconductor technology that permits the required logic functions to be realized at much lower weight, power consumption, and cost. At the same time there is a greater need for fault-tolerant computers due to longer mission durations, much more demanding mission objectives (in spacecraft management and in payload data processing), and the desire for

greater autonomy. Thus, the results of previous research and development efforts on fault-tolerant computers for spacecraft applications are now bearing fruit.

Fault-tolerant computing for general applications is today a well-established discipline served by annual symposia[1]. A number of excellent survey documents are available[2,3,4]. Various techniques that can be applied to achieve fault tolerance in a digital component are described immediately below. This is followed by a summary of some major attempts to design or develop fault-tolerant computers for space applications*. At present, the furthest progress along the road to hardware realization has been achieved in the Fault-Tolerant Spaceborne Computer being developed for SAMSO, and this is described in a later section in some detail, together with an outline of how the reliability of such an essential component can be demonstrated at reasonable resource expenditure. The final section of this paper delineates some current topics in the field of spacecraft fault-tolerant computers.

---

*Space applications are throughout this article restricted to the spacecraft proper, excluding launch vehicles and intermediate stages.

SECTION II

II. CONCEPTS OF FAULT TOLERANCE

Fault-tolerance means the ability to render at least the essential level of service after occurrence of a fault. This implies a measure of redundancy in the broadest sense because there must have been initially some resources that were not fully utilized for the essential level of service. But redundancy itself (i.e., duplicating components) is not sufficient for fault tolerance; there must also be provisions for recognizing a fault and implementing restoration of the essential level of service. Specifically, in spacecraft applications, to be fault tolerant requires that the entire recovery from a fault be accomplished automatically on board. If error detection, reconfiguration, or recovery deperd on ground command, then it is a redundant installation rather than a fault-tolerant one. Redundant computer installations, in this narrower sense, are in use today in several spacecraft* (e.g., in the Viking lander and in some meteorological satellites). The fault-tolerant approach has the advantages of providing immediate response and reducing

---

*These computers incorporate automatic switchover provisions for some gross failures but comprehensive error detection and recovery depend on ground facilities.

dependence on ground support. This can be particularly important for spacecraft which are in ground contact for only a small portion of their orbit and for missions beyond earth-synchronous altitude.

In some cases, repair and restoration of service is almost inherent in the circuit structure itself: a pair of diodes in series will provide uninterrupted service after short-circuit failure of either one of them. Because no switching or reconfiguration is required in this case, it is termed static fault tolerance or static redundancy. By contrast, if two computers are operated together and, upon an inconsistency in their outputs, a test procedure is entered to identify the faulty one, this is termed dynamic fault tolerance because a change in configuration takes place upon occurrence of a fault.

This illustrates the two major forms of fault tolerance: static and dynamic. There is also a third one, hybrid fault tolerance, about which we will speak later on[15]. Static fault tolerance can be implemented at the part level or at a computer function level. A conceptual application of static fault tolerance at the part level is quad redundancy as shown schematically in Figure 1. In principle, such a structure provides continuous service after all first device failures and some second device failures. When applied to practical digital

devices, considerably more complexity is required to protect against single-point faults at the nodes[5,6,7]. Quad-redundant logic obviously requires four times the number of components, and it also incurs power and speed penalties, the latter because of the additional gate delays introduced. The greatest difficulty with quad-redundant logic is due to its fault-masking property. This makes it impossible to determine under ordinary operating conditions whether each one of the four devices is indeed operating correctly. Therefore, some circuit modification must be introduced to permit testing, and this becomes extremely difficult for the large number of gates involved in a modern computer.

Static fault tolerance at the functional level is represented by TMR, or triple modular redundancy, as shown conceptually in Figure 2[8,9]. The voter which is employed here furnishes an output that corresponds to the signal received from at least two out of the three inputs. The TMR configuration can tolerate any single fault but not usually a second solid fault. It requires slightly more than three times the resources of the non-redundant implementation, and it may incur a speed penalty if the clocks controlling the individual sections are not exactly synchronized.

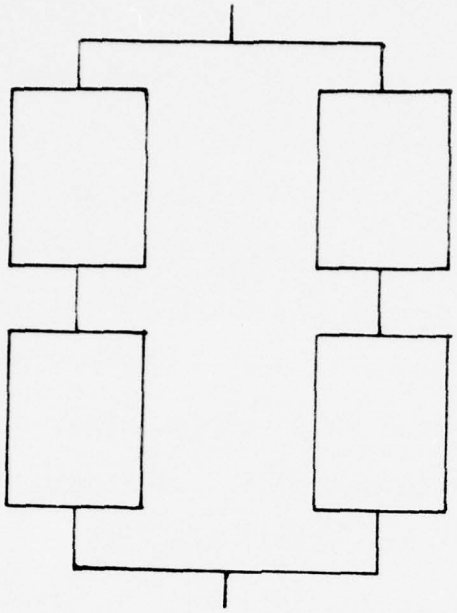

Figure 1. General Presentation of Quad Redundancy

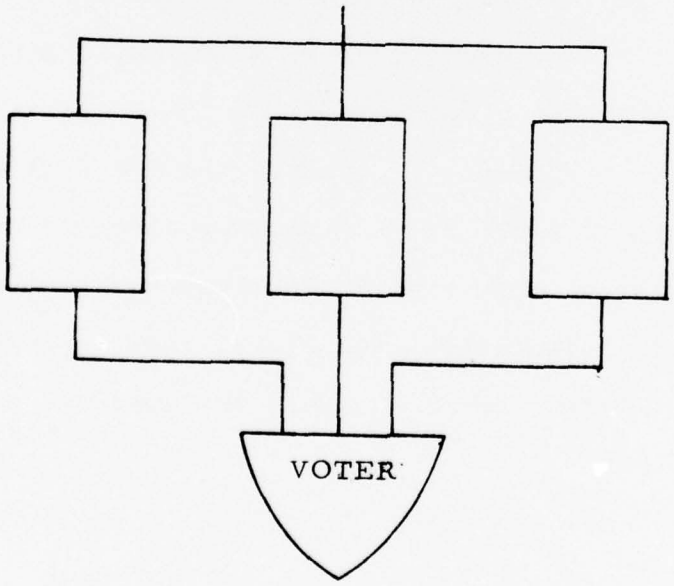

Figure 2. General Presentation of
Triple Modular Redundancy

A comparison of the reliability of a function implemented in TMR with that of a non-redundant implementation is shown in Figure 3. Note that TMR provides a significant reliability improvement only for mission times that are short compared to the mean-time-to-failure (MTTF) of the single constituent component. TMR for the entire computer structure is therefore seldom considered for spacecraft application, although it has been employed for launch vehicles which have shorter mission times (a notable example is the Saturn V). However, for a computer function which is of low complexity and therefore has a high MTTF, TMR is a viable fault-tolerance technique.

Dynamic fault tolerance requires separate provisions for error detection and for reconfiguration. It usually requires a more complex structure than static fault tolerance, but at the same time it is more flexible (e.g., spares can be unpowered, and extensive system reconfiguration is possible to cope with a fault). The principal error detection means are codes and multiple, usually dual, processing. Watchdog timers (that monitor receipt of a response) and voltage detectors are also sometimes used.

Codes are primarily useful for protecting information during its transmission and during storage into or recall from memory. Some codes also afford protection against arithmetic processing

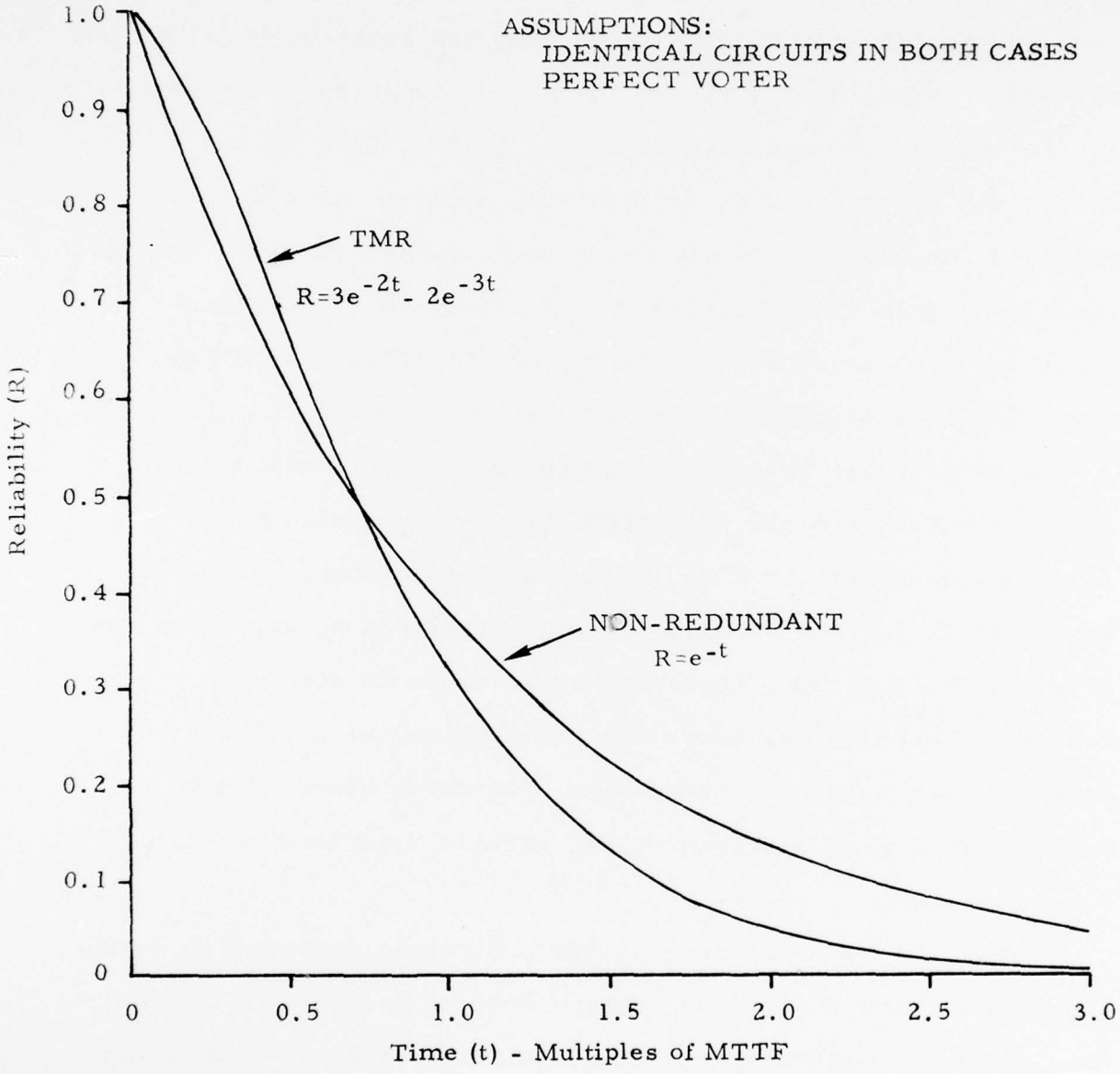

Figure 3. Reliability of a Non-Redundant Function and TMR

errors, but protection through all computer operations, particularly shifts and logical operations, is difficult[10,11,12]. For transmission and storage, error-detecting codes are frequently preferred to any other means of error detection because they require less redundant hardware. A single code bit (usually referred to as "parity bit") can detect the existence (but not the location) of any single error in an arbitrarily long word. Because a single hardware fault can frequently affect more than one bit of a word, dependence on a single error-detecting bit is not common in fault-tolerant computer design. A "deeper" code is usually employed, e.g., four bits to monitor a 32-bit word.

Somewhat longer codes have the interesting properties that they can not only establish that an error has occurred but can frequently locate it to the very bit that is affected. In a binary representation, once it has been established that a bit is faulty, it is immediately known that the correct representation is the complement of that bit, and these codes are in effect error correcting. The shortest error-correcting code for a 32-bit word requires six bits. It permits correction of any single error and detection of all two-bit errors and many multiple-bit errors. Practically all designs for fault-tolerant spacecraft

computers that are discussed here utilize codes for at least a portion of the error-detection task.

The logic circuitry for generating codes must be provided at both the transmitting and the receiving end. At the transmitting end the code is generated from the original information and appended to it for transmission. At the receiving end, the same code (or, more frequently, its complement) is again generated from the information part of the word and is then compared with the transmitted code. Encoding circuits typically require only a minor fraction of the semiconductor "real estate" of the computing or transmitting function with which they are associated. There is some delay associated with the coding or decoding process, but this is minor and can be reduced by use of "look-aside" coding techniques[13,14].

To economize on code length, codes in use in fault-tolerant computers do not provide 100 percent error detection. Most are insensitive to some types of multiple-bit errors. The probability that an error is detected is termed the "coverage" of a code[10]. The term coverage is also used in a broader sense in the fault-tolerant computer field to denote the conditional probability that, given a fault has occurred and spares are available, adequate restoration of computing service will be effected. If the probability distribution of errors is known, a

quantitative expression for coverage can be generated in a rather straightforward manner. At present there are no data available from which to construct a fault probability distribution pertinent to the type of computer used in spacecraft. Nevertheless, some assumptions about fault distribution are sometimes made to study variations in coverage from one implementation of a function to another.

The alternative to use of codes for error detection is multiple processing, storage, or transmission. Dual processing or transmission is usually considered fully effective for detecting errors (100 percent coverage), but it does not by itself provide identification of the correct information. However, temporary (intermittent) faults can be corrected by retransmission or repeat of an operation. If that does not clear the error, an arbitrary sequence of reconfigurations can be entered until correct results are obtained. Thus, explicit fault isolation can sometimes be omitted. Where such an approach is not permissible, triplication or a higher multiple of processing or storage is required. The implementation of these error-isolation techniques is straightforward, but, in the spacecraft computer environment, the additional hardware is objectionable. Therefore, dual or triplicated computer operations are employed only where there are no alternatives or where the total resources

involved are so small that provisions for multiple hardware are tolerable.

The time interval required for fault isolation, hardware reconfiguration, and software restoration ranges from several milliseconds to about a second, with the upper bound being approached only when memory refresh is involved. For on-orbit operation, a lapse of computer availability of this order is not objectionable. However, if the spacecraft computer is also utilized for boost guidance, the error-detection and recovery techniques have to be selected to keep the outage in the 0.05- to 0.1-second range, and TMR is then a preferred fault-tolerance implementation.

There are some fault-tolerant arrangements of computer functions that combine features of static and dynamic redundancy and that are therefore called hybrids. One of these is hybrid TMR, also known as TMR with spares[15]. The conventional voter used in TMR is here supplemented by logic that detects if one of the inputs disagrees in more than a transient manner from the other two. The element responsible for the faulty input is then switched out and replaced by a spare. Provision of even a single spare can significantly improve the reliability for long mission durations compared to conventional TMR. The technique can obviously be extended to voters that receive more than three

inputs and is then referred to as hybrid NMR (N-modular redundancy).

Another fault-tolerance technique that combines voting with isolation of a faulty element is the self-purging structure[16]. It usually starts out as an NMR arrangement with N greater than 3. The logic associated with the voter identifies a permanently faulty input and switches it out (but does not replace it). This assures that a subsequent intermittent or permanent fault will not produce a majority of voter inputs that represent the wrong logic level.

A derivative of self-purging logic can be applied to TMR. In addition to switching out the faulty input, the logic can serve as a disagreement detector among the two remaining signals into the voter. In case of disagreement, random selection will have at least some chance of providing continued computer operation. Another possibility is to revert to simplex operation after the first TMR input failure. This has greater success probability than operation with two inputs but, of course, it sacrifices error detection.

Most of the techniques mentioned here can be applied at several levels: for the computer as a whole, for major computer functions (memory, processor, etc.), or for individual logic circuits or interconnections (buses). It can be shown in a

general sense that the overall reliability of an item is increased when fault tolerance is applied at low levels (i.e., the item is decomposed into a series of fault-tolerant structures)[17]. On the other hand, the transition from one fault-tolerant structure to another always involves joining and redistributing of signals which entail delays and power loss and can introduce single-point fault possibilities. Therefore, the design of a fault-tolerant computer requires a careful tradeoff regarding the level at which fault-tolerant techniques shall be applied. Spacecraft computers represent a rather extreme case in this respect because of the need for long, unattended operation and, hence, the need to tolerate multiple faults. The advantages of low-level redundancy are particularly significant in this application.

A fault-tolerant computer can be constructed out of redundant conventional (simplex) computers, followed by a voter or comparison and selection circuits. The need for special computer design and development is thereby minimized, and advantage can be taken of the production experience and production base of the conventional computer. Unfortunately, this is not a viable approach for long-duration space missions. This is illustrated in Figure 4 by a comparison of the reliability functions of three computer configurations that

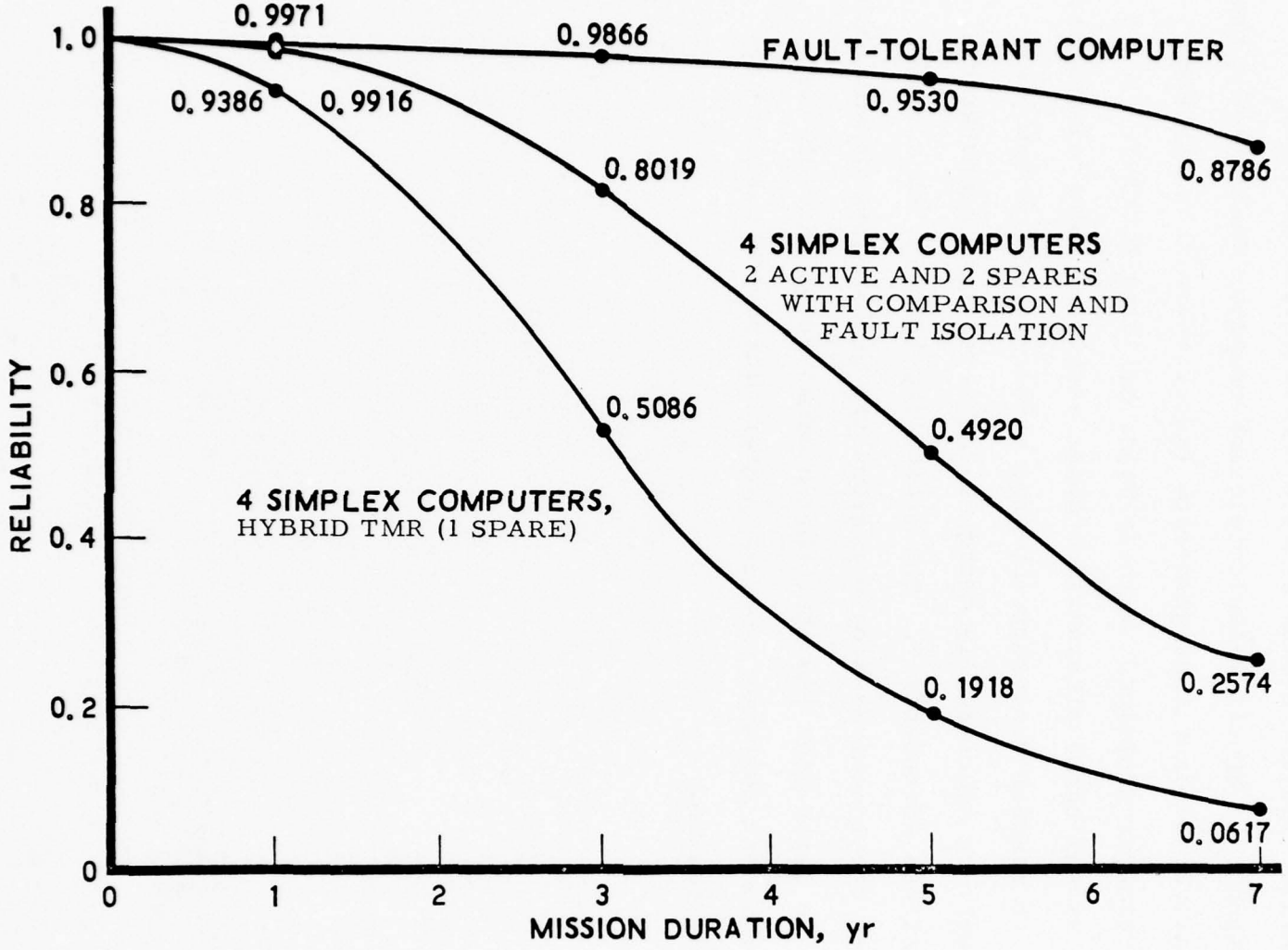

Figure 4. Reliability for Equivalent Computer Resources

utilize the same semiconductor types, have the same weight and performance, and are evaluated with consistent reliability assumptions: a specific fault-tolerant computer design[18] and two externally redundant configurations (i.e., those that utilize entire simplex computers as their major building blocks). It is seen that the three alternatives offer about the same reliability for short mission durations, but for mission times of two years or longer the specifically designed fault-tolerant computer has unquestioned advantages[19]. The reason for this is, of course, that for the longer mission durations the probability of multiple faults is quite high and that the internal redundancy of the fault-tolerant computer provides higher protection against such multiple faults.

SECTION III

III. SPACECRAFT APPLICATIONS

The first fault-tolerant digital equipment (not a stored program computer) to be developed for spacecraft, and the only one that has flown so far, is the Primary Processor and Data Storage (PPDS) for the NASA Orbital Astronomical Observatory (OAO)[20]. Design of this equipment was begun in 1960 at the IBM Electronics System Center under contract to Grumman Aircraft Corporation. It employed quad-redundant logic, TMR delay lines, and duplexed memory with error-detecting code. (This eclectic use of fault-tolerance techniques also characterizes each of the other designs described below.) Pertinent characteristics of the PPDS are shown in Table 1. The computer was comprised of over 72,000 discrete parts. Before committing to the use of quad-redundant logic in this computer, it was estimated that the single-string version would require 15,000 discrete parts in the processor proper (exclusive of memory) and that the reliability of the single-string processor might be as low as one percent for a year in space.

The first spacecraft computer design that made extensive use of dynamic fault-tolerance techniques was the JPL-STAR (Self-Test and Repair) computer[21,22,23]. The concept originated in 1961 but

TABLE 1

PPDS CHARACTERISTICS

Memory	:	8192 data words (25 bits) 256 or 512 command words (30 bits)
Clock Speed	:	50 KHz
Weight	:	106 Kg
Volume	:	0.19 m^3
Power	:	82 W average
Predicted Reliability	:	0.7 for one-year mission
Achieved Operating Life	:	4 years on orbit

major activity on the project was concentrated between the mid-60s and the early 70s. During that period there was much interest in missions to the outer planets that involved spacecraft operation of up to 10 years. Only very limited communication with the earth was possible during these missions, and a computer was required for navigation, experiment control, and housekeeping. Some characteristics of the STAR computer are shown in Table 2.

TABLE 2

JPL-STAR CHARACTERISTICS

Memory	:	Up to 64K words (32 bits)
Clock Speed	:	500 KHz
Weight	:	18 Kg design goal
Power	:	40 W
Predicted Reliability	:	0.95 for 100,000 hours (over 10 yr)

A breadboard of the computer was built which demonstrated the value of many innovative fault-tolerance techniques as well as overall system capabilities. Among features of the STAR computer that have been carried over into later designs are a modular, bus-oriented architecture and a separate restoring function, here called the TARP (Test-and-Repair Processor), that activates and deactivates modules and controls the program recovery following a hardware failure. Due to a sharp curtailment in the budget for outer planetary missions, the STAR never progressed to a flightworthy implementation.

In 1966, personnel of the MIT Instrumentation Laboratory (now the C. S. Draper Laboratory) began work on a fault-tolerant computer for the general aerospace environment, including spacecraft applications[24]. Significant innovative features were the use of a multiprocessor organization and single-instruction restart capability. In a multiprocessor, as the name implies, tasks are shared among a number of processors. If one of the processors becomes faulty, and this will of course have to be detected by one of the means described in the preceding section, the computational tasks can be transferred to the remaining processors. A software scheduler can assure that the most essential computing tasks are given the highest priority when such a fault occurs. In this way, faults result in graceful

degradation rather than in complete loss of computing capability. Another way of looking at the multiprocessor capability is to regard it as a fault-tolerant computer in which spares are being used to process optional tasks while they are not being required as replacements for processors executing the essential tasks.

The single instruction restart capability is provided by having three independent scratch-pad memories associated with each processor that contain the temporary data for the program then executing. The scratch-pad output is voted and, upon detection of an error, auxiliary memory capabilities can be brought into play. Triplication assures that no temporary data is ever lost, and reexecution can start at the last completed instruction. Through involvement in programming of the Apollo guidance computer, personnel at the Draper Laboratory had become particularly aware of the importance of preserving navigation data in case of computer restart. (The Apollo computers, while not hardware fault-tolerant, had restart capability to cope with temporary disturbances, a capability that was put to good use during several missions.)

The program was partially sponsored by the NASA Manned Spacecraft Center, and some of the concepts developed by it have influenced the design for the shuttle computer configuration. So

far no specific application of this design for satellites has developed.

The concept of a multiprocessor that can provide graceful degradation after faults was also at the heart of a computer design sponsored by NASA Marshall Space Flight Center as part of the SUMC (Space Ultra-Reliable Modular Computer) program. This particular phase was named Automatically Reconfigurable Modular Multiprocessor Systems (ARMMS)[25,26]. A later version under the acronym ARMS was intended as a feasibility model, and in it the multiprocessor capability was deleted. In both versions, TMR with voting was applied at the level of major computer functions (memory, central processor, and I/O processor). The design was to be applicable to launch vehicles, space stations, and deep-space probes. As far as is known, no application of this concept for satellite computing is presently active.

In 1974, the European space agency sponsored a study by SAAB-Scania to define a fault-tolerant spacecraft computer that would make maximum use of hardware under concurrent development for a simplex on-board computer[27]. Two candidate configurations were defined: one utilizing duplex processing for error detection, and a second one utilizing a multiprocessor configuration with software error detection. The study concluded that differences between the candidate configurations were too

small to be useful for selecting the most suitable (least expensive) one but that both candidates were well suited for the application. The characteristics shown in Table 3 below reflect the study results for the multiprocessor organization. As of January 1977 there had been no hardware implementation in this effort.

TABLE 3

SAAB-SCANIA COMPUTER CHARACTERISTICS

Memory	:	Up to 64K words (16 bits)
Throughput	:	400K adds/sec
Weight	:	18.3 Kg (with reduced memory)
Volume	:	0.025 m^3
Power	:	26.2 W
Predicted Reliability	:	0.82 for 4 years

SECTION IV

IV. THE FAULT-TOLERANT SPACEBORNE COMPUTER (FTSC)

The FTSC is being developed by the U. S. Air Force Space and Missile Systems Organization (SAMSO) in support of long-duration space missions. The overall objective is to provide a five-year on-orbit capability to perform the computational tasks shown in Table 4. Memory and timing estimates shown there include considerable growth allowance and can be taken as realistic requirements for the computer. Some of the tasks may require additional mass memory (now realizeable through magnetic bubble technology). A number of technical papers relating to the computer and preceding studies are listed in the bibliography following the references. A brassboard (a packaged and transportable breadboard) was completed in late 1976 by Raytheon Corporation.

The principal characteristics of the FTSC are shown in Table 5. A block diagram of the prototype configuration is shown in Figure 5. At the user interfaces, internal buses, and in memory the information is protected by an error-detecting and correcting code. Of the seven main memory modules, four are normally designated as active memories and three as spares that can be activated for memory failures not corrected by use of the

TABLE 4

ON-ORBIT COMPUTATION REQUIREMENTS

Task	Program and Data Memory - words	Computation Rate ops/sec
Attitude Control and Pointing	2,343	68,250
Telemetry	1,281	13,692
Commanding	912	19,500
Supervisor	2,909	33,042
Subsystem Management and Recovery	876	1,875
Subtotal	8,321	136,359
Payload Processing	6,500	60,000
Subtotal	6,500	60,000
Memory Size Without Payload Processing	10K	---
Memory Size With Payload Processing	16K	---
Maximum Computation Rate	---	200,000

codes. In the central processing unit, dual operation is utilized for error detection, and a rotating replacement is used for reconfiguration. The timing modules and the power modules each carry redundant internal monitors that detect deviation from specified output. In these modules, as well as in the interface

TABLE 5

FTSC CHARACTERISTICS

Memory	:	Up to 64K words (32 bits)
Throughput	:	200Kops (navigation instruction mix)
Weight	:	23 Kg with 60K memory (15 active modules, 9 spares)
		14 Kg with 16K memory (4 active modules, 3 spares)
Volume	:	Less than $0.028m^3$
Power	:	25 W

Predicted Reliability: 0.95 for 5 years

modules (DIU and DMAs), detection of a failure will cause switchover to the standby spare. The configuration control unit (the ultimate control for error recovery), the circumvention unit, and the hardened timer are operated in TMR. As may be surmised from the nomenclature attached to the latter two items, the computer is designed to operate through radiation and spacecraft discharge events. The ability to operate in high-radiation environments also makes this computer a prime candidate for outer planetary missions. All fault scenarios considered in an extensive study were shown to be tolerated by this computer configuration.

The predominant semiconductor type utilized in the FTSC is CMOS/SOS (complementary metal-oxide semiconductor, silicon on sapphire). This was selected because it provides adequate speed at very low power consumption. The memory utilizes a technology which offers non-volatility and non-destructive readout. Due to

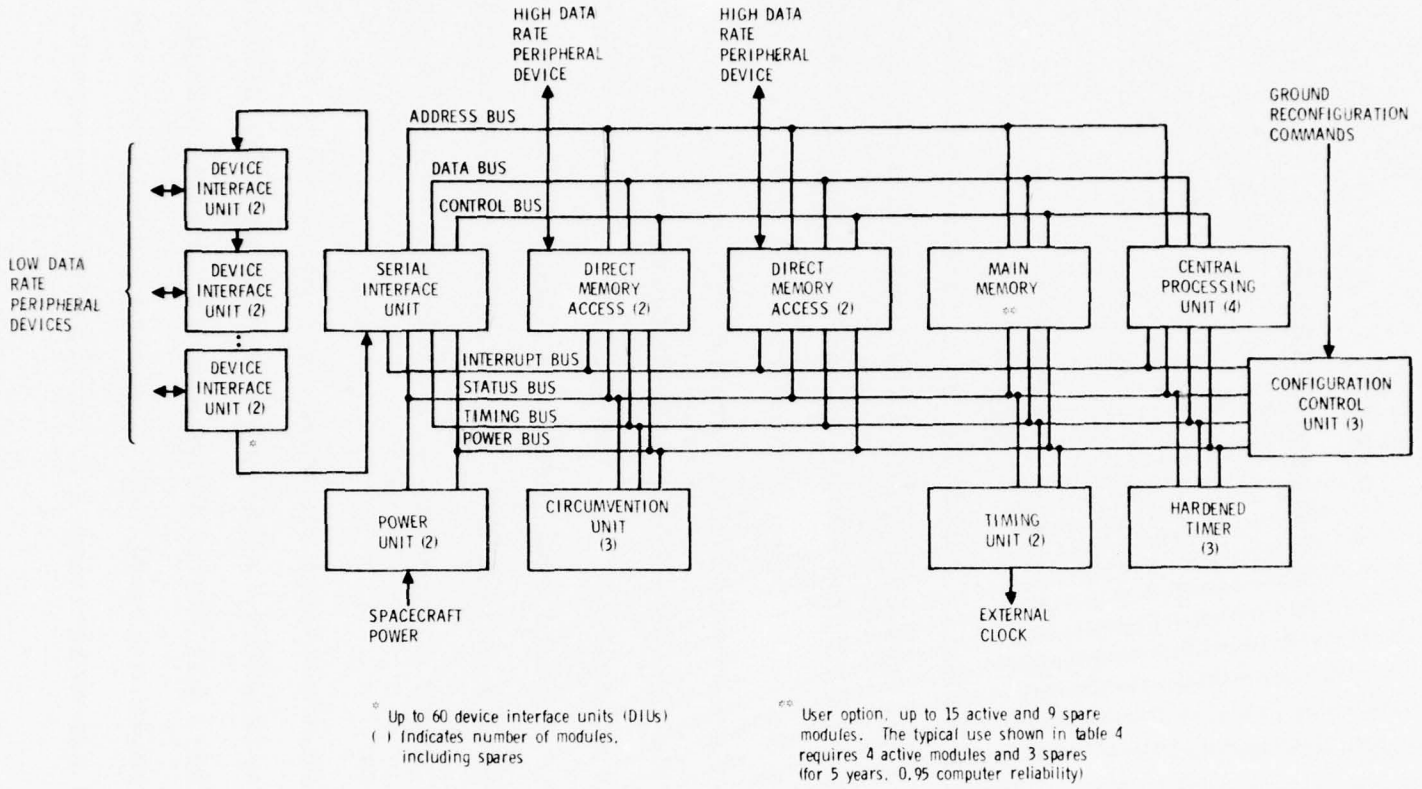

Figure 5. FTSC Block Diagram

non-volatility (memory content is retained while the memory is unpowered) memory modules can be powered down when not in active use. The non-destructive readout prevents garbling due to transients while a read operation is in progress.

The instruction set of the FTSC supports 32-bit fixed-point and floating-point arithmetic as well as vector operations that are particularly useful for navigation and pointing applications. Direct and indirect addressing with predecrement and postincrement are available. All these features permit writing very compact code and thus reduce the storage requirements. Some instructions are specifically oriented for testing the fault-tolerant features. Thus, data words can be stored with bad parity in order to exercise error-detection and correction in the memory modules, and a deliberate disagreement in the outputs from the two central processing units can be commanded. The ground override can disconnect individual elements of the configuration control unit and thus permit thorough checking of this vital part of the computer.

Many application programs in the fault-tolerant computer service closed-loop systems, and in these temporary loss of data may be tolerated. E.g., an attitude control system can usually recover from lack of data or even bad data existing during one or two computing cycles. Data for these applications need no

specific protection. On the other hand, certain navigation quantities cannot be recovered autonomously within the spacecraft if they are destroyed due to a memory failure. For these, there is a "STORE DOUBLE" instruction that permits virtually simultaneous access to the same location in two independent memory modules. These data are thus protected in case of a memory module failure, regardless of how extensive.

When reconfiguration following a failure is complete the computer resumes operation at a memory location whose address is stored in a special location, and this address is called the rollback point. Computations supporting closed-loop routines or housekeeping only need one rollback point at the start of each task. In a few cases, e.g., updating of position by velocity increments, complete repetition of a task could produce undesirable double incrementing. To prevent this, the programmer can insert a rollback point immediately after a position update. Upon subsequent recovery from failure, the program will restart there, thus assuring accurate position information. Rollback locations are considered particularly critical information and can be multiply stored, again by a single instruction.

SECTION V

V. RELIABILITY TESTING OF FAULT-TOLERANT COMPUTERS

The development of the FTSC is, of course, accompanied by an extensive test program starting with qualification and screening tests of the semiconductor parts, going through functional tests at the module and computer level, and culminating in a qualification test that stresses the hardware above anticipated launch and on-orbit thermal and mechanical environments. The requirement that is not directly addressed by this test program is that of reliability.

The predicted reliability of 0.95 for a five-year mission duration is equivalent to an MTBF of 100 years (assuming an exponential reliability function). Reliability demonstration, even at low confidence levels, requires that the product of the number of test articles and test time be roughly equal to the MTBF. A test of 10 computers carried on for a period of 10 years would meet this criterion but it is obviously impractical in both quantity and duration. Accelerated testing, by imposing stresses above levels expected for the service application, can be applied at the part level where failure modes are reasonably well defined. The computer as a whole has an unknown number of failure modes, and there is no realistic basis for assuming that

failure frequency could be increased by a specific factor if the testing were done under vibration, etc.

The difficulty in arranging a meaningful reliability demonstration is not unique to the FTSC or to fault-tolerant computers; it arises whenever we deal with a costly and highly reliable component. The nature of a fault-tolerant computer, however, permits reduction of test time by conducting the reliability demonstration at the module level. It is almost inherent in the concept of fault-tolerant computers that individual modules are less reliable than the computer as a whole.

In the FTSC the most costly and reliability-critical items are the memory modules. The module hazard is 4.43×10^{-6} per hour or 0.0388 per year. This implies a module MTBF of slightly in excess of 25 years. This number is still quite formidable, but it will be recalled that there are seven memory modules provided in the configuration that was discussed in the previous section. Thus, testing for the equivalent MTBF of a memory module can be accomplished with the memory complement of 3-1/2 computers during a period of one year. This still requires considerable resources but it is a vast improvement over the initial 100 computer-years! A specific calculation has shown that the required reliability can be demonstrated at the 50

percent confidence level by a test of 13 modules if no failures are encountered[28].

The module test establishes only that the probability of parts failures does not exceed an acceptable level. The reliability of the overall fault-tolerant computer requires, in addition, that error detection, reconfiguration, and software recovery are carried out correctly. The module test must therefore be supplemented with testing at the computer level as shown in Table 6.

TABLE 6

TESTS FOR SYSTEMS RELIABILITY

Test	Objective	Test Vehicle
Error Detection	Establish adequacy of coverage	Brassboard computer, ICS, emulation
Reconfiguration	Verify computer performance after a fault	Brassboard, ICS, prototype computer
Recovery	Verify program and data integrity after a fault	ICS, emulation, spacecraft simulation
Computer System	Establish absence of undesired interactions, validate all prior tests	Emulation, prototype computer

It is particularly advantageous that most of the tests listed there can be carried out with use of the brassboard, Interpretive Computer Simulation (ICS), emulation, and spacecraft simulation. These tests can therefore be performed concurrent with the design of the prototype, and their findings can be incorporated before the hardware is "frozen". Moreover, they do not tie up costly prototype hardware and test facilities. By the combination of module and systems-oriented tests, it is thus possible to gain confidence in the reliability of the computer before committing it to control of an expensive spacecraft.

SECTION VI

VI. CURRENT TOPICS

The FTSC is expected to meet the needs of a large variety of Air Force and NASA missions in the post-1980 time period. The computer provides adequate performance and reliability for these missions without imposing burdensome weight and power requirements. The modular nature of the computer permits tailoring memory size and sparing for all functions to a variety of user needs. The incorporation of the fault-tolerance provisions into a single component (where they can be tested at the factory) simplifies design, development, and test of the spacecraft compared to an assembly of separately packaged components with external redundancy provision. There are, however, a number of areas in which research and advanced development are required, either to fully exploit the capabilities of the FTSC or to lay the groundwork for more extensive applications of fault-tolerant data processing for future spacecraft.

Reliability calculations for the FTSC, as well as for all the other computers mentioned here, have been carried out by assuming an exponential reliability model at the part level. While the validity of this model has been questioned for some

time, the deviations from it were felt to be quite tolerable in view of the simple calculations that resulted from the use of the exponential assumption and the extensive modeling capabilities that existed specifically for the exponential failure law. Recent investigations of on-orbit failure rates have shown such drastic deviations from the exponential failure law that its use for predicting reliability in long-duration space missions should be avoided[29]. The existing data point to a high failure rate during the first day on orbit, followed by an appreciably lower failure rate during the first month, and successively lower rates for each six-month interval up to three years. While the general implications of this observation bode well for achieving very long mission durations, there is nevertheless the problem that new analysis and modeling tools are required. Most of all, a concensus as to the form of a reliability function for predicting on-orbit success for long-duration missions must be generated. Some research and considerable organizational effort in this area are required in order to fully exploit the capabilities that will be available from components currently under development.

Another important issue relates to the software that is to be executed on a fault-tolerant computer for spacecraft. Given that the hardware reliability can indeed be demonstrated at a level that will permit critical spacecraft operations to be

entrusted to computer control, can we really take that step without equivalent fault-tolerance provisions in the software? Neither testing nor formal verification methods can give assurance of the total correctness of software, particularly when it is operating under exception conditions[30,31]. Fault-tolerant software techniques are available but admittedly costly to implement[32,33]. A major deterrent to their present use is that they require additional memory, and this is one of the most expensive resources on a spacecraft computer. However, with the advent of magnetic bubble memories, it may be possible to keep alternate programs in a backup store at much lower cost, and this may indeed facilitate the employment of fault-tolerant software techniques for essential program segments.

The scope of computer applications is undergoing rapid change under the influence of microprocessors. While in some areas they may replace central computers, the more usual finding is that they supplement and adapt central computers to specific user needs and thus lead to new computer applications. This may also be the case in regard to the employment of microprocessors on spacecraft. Two specific areas of application come to mind: to offload a central fault-tolerant computer for high-throughput, simple calculations; and to provide for more flexible fault-tolerance provisions at the peripheral/computer interface.

Examples of high-throughput processing applications are in frequency control of communications equipment and in sensor data compression. In both cases, a simple instruction repertoire suffices, and a minimum of local storage is required. Yet these tasks need to be carried out at such high speeds that they pose a burden on the fault-tolerant computer or are completely impossible to service in that manner. Microprocessors seem well suited to these applications and can be made fault tolerant by being periodically tested and, if necessary, replaced under control of a program residing in the fault-tolerant computer.

An example of flexible fault-tolerance provisions at the computer interface arises in connection with the magnetic bubble memory. These memories offer comparatively inexpensive non-volatile storage and are admirably suited to contain backup copies of programs running in the computer, alternates to these programs for fault tolerance (as explained above), or overlays to cope with programming requirements during different mission phases. These memories are organized into cells, and the more likely failure modes all involve partial or complete outage of a single cell. If only a small number of bits for each word are stored in a given cell the information content might be reconstructed after cell failure by use of a high-level error-correcting code. The error correction could, of course, be

carried out under program control in the fault-tolerant computer. But how about the case when access to the bubble memory is required exactly because of a fault in the fault-tolerant computer? A microprocessor imbedded in the bubble memory can obviously be of considerable help in those circumstances, and it can also rearrange storage after a cell has been found faulty. Furthermore, it provides other desirable autonomy features for the bubble memory. Preliminary investigations in these areas are currently in progress and are expected to considerably enlarge the applicability of spacecraft computers in general and of fault-tolerant computers in particular.

REFERENCES

1. International Symposium on Fault-Tolerant Computing, held annually since 1971. Digest or Proceedings available from IEEE. Latest is Proceedings, FTCS-6 IEE Cat. No. 76CH1094-2C (June 1976).
2. A. Avizienis, "The Methodology of Fault-Tolerant Computing," Proceedings, First USA-Japan Computer Conference, Tokyo (October 1972), pp. 405-413.
3. J. Goldberg, P. G. Neuman, and J. H. Wensley, Survey of Fault-Tolerant Computing Systems, Stanford Research Institute, Menlo Park, CA (August 1972).
4. D. Rennels, Fault Detection and Recovery in a Redundant Computer Using Standby Spares, UCLA-ENG-7355, Computer Science Dept., UCLA, Los Angeles, CA (July 1973). Also published by National Science Foundation as NSF-OCA-GJ33007X-73553.
5. J. G. Tryon, "Quadded Logic," in Redundancy Techniques for Computing Systems, edited by R. H. Wilcox and W. C. Mann, Spartan Books, Washington, DC (1962), pp. 205-228.
6. P. A. Jensen, "Quadded NOR Logic," IEEE Trans. on Reliability (September 1963), pp. 22-31.

7. Ralph E. Kuehn, "Computer Redundancy: Design, Performance and Future," IEEE Transactions on Reliability (February 1969), pp. 3-11.

8. R. E. Lyons and W. Vanderkul, "The Use of Triple Modular Redundancy to Improve Computer Reliability," IBM Journal of Research and Development 6 (2) (April 1962), pp. 200-209.

9. J. E. Anderson and F. J. Macri, "Multiple Redundancy Applications in a Computer," Proceedings, 1967 Annual Symposium on Reliability, Washington, DC (January 1967), pp. 553-562.

10. W. C. Carter and P. R. Schneider, "Design of Dynamically Checked Computers," Proceedings, Information Processing 68, North Holland Publishing Co., Amsterdam (1969), pp. 878-883.

11. A. Avizienis, "Arithmetic Error Codes: Cost and Effectiveness Studies for Application in Digital Systems Design," IEEE Transactions on Computers (November 1971), pp. 1322-1331.

12. W. W. Peterson and M. O. Rabin, "On Codes for Checking Logical Operations," IBM Journal of Research and Development (April 1959), pp. 163-168.

13. W. C. Carter, K. A. Duke, and D. C. Jessep, Jr., "Look Aside Techniques for Minimum Circuit Memory Translators," IEEE Transactions on Computers (March 1973), pp. 283-289.

14. W. C. Carter, "Implementation of an Experimental Fault-Tolerant Memory System," IEEE Transactions on Computer (June 1976), pp. 557-568.

15. F. P. Mathur and A. Avizienis, "Reliability Analysis and Architecture of a Hybrid Redundant Digital System: Generalized Triple Modular Redundancy with Self-Repair," AFIPS Conference Proceedings 36 (Spring Joint Computer Conference, 1970), AFIPS Press, Montvale, NJ.

16. J. Losq, A Highly Efficient Redundancy Scheme: Self-Purging Logic, Technical Report 62, Digital Systems Laboratory, Stanford University, (July 1975).

17. I. S. Reed and D. E. Brimley, On Increasing the Operating Life of Unattended Machines, Memorandum RM-3338-PR, RAND Corporation, Santa Monica, CA (1962).

18. R. B. Conn, N. A. Alexandridis, and A. Avizienis, "Design of a Fault-Tolerant, Modular Computer with Dynamic Redundancy," AFIPS Conference Proceedings 41, Part II (1972 Fall Joint Computer Conference, Anaheim, CA), pp. 1057-1067.

19. H. Hecht, A Comparison of Fault-Tolerant and Externally Redundant Computers, SAMSO-TR-74-66 (January 1974).

20. A. E. Cooper and W. T. Chow, "Development of On-Board Space Computer Systems," IBM Journal of Research and Development (January 1976), pp. 5-19.

21. A. Avizienis, G. C. Gilley, F. P. Mathur, D. A. Rennels, J. A. Rohr, and D. K. Rubin, "The STAR (Self-Testing and Repairing) Computer: An Investigation of the Theory and Practice of Fault-Tolerant Computer Design," IEEE Transactions on Computers (November 1971), pp. 1312-1321.

22. A. Avizienis and D. A. Rennels, "Fault-Tolerance Experiments with the JPL STAR Computer," Digest of COMPCON 1972, San Francisco, CA, pp. 321-324.

23. J. A. Rohr, "STAREX Self-Repairing Routines: Software Recovery in the JPL-STAR Computer," Digest of 1973 International Symposium on Fault-Tolerant Computing, Palo Alto, CA (June 1973), pp. 11-16.

24. A. L. Hopkins, Jr., "A Fault-Tolerant Information Processing Concept for Space Vehicles," IEEE Transactions on Computers (November 1971), pp. 1394-1403.

25. R. A. Easton (ed.), Design of a Modular Digital Computer System, DRL4 and 5, Final and Phase III Report, Hughes Aircraft Company, Fullerton, CA (December 1973) (prepared under Contract NAS8-27926).

26. J. L. Bricker, "A Unified Method for Analyzing Mission Phase Reliability for Standby and Multiple Modular Redundant Computing Systems which Allows for Degraded Performance," IEEE Transactions on Reliability (June 1973), pp. 72-77.

27. J. Torin, et al., Fault-Tolerant Computing for On-Board Spacecraft Applications, RA-OBC-5-088 (2 volumes), SAAB-Scania, Sweden (September 1975). Prepared for European Space Agency under ESTEC Contract 2032/73/AK.

28. H. Hecht, Reliability Testing of the Fault-Tolerant Spacecraft Computer, SAMSO-TR-74-259 (September 1974).

29. A. R. Timmins, A Study of the Total Space Life Performance of GSFC Spacecraft, NASA TN-D-8017 (July 1975).

30. R. E. Keirstead, On the Feasibility of Software Certification, Final Report, SRI Project 2385, Stanford Research Institute, Menlo Park, CA (no date).

31. S. L. Gerhart and L. Yelowitz, "Observations of Fallibility in Applications of Modern Programming Methodologies," IEEE Transactions on Software Engineering (September 1976), pp. 195-207.

32. B. Randell, "System Structure for Software Fault Tolerance," Proceedings International Conference on Reliable Software, IEEE CAt. No. 75CH0940-7CSR (April 1975), pp. 437-449.

33. H. Hecht, Fault-Tolerant Software for Spacecraft Applications, SAMSO-TR-76-40 (1976).

BIBLIOGRAPHY

Agnew, P. W., et al., An Architectural Study for a Self-Repairing Computer, IBM Corporation, Rockville, MD (November 1965). (Available from Clearinghouse for Federal Scientific and Technical Information under AD 474 976.)

Burchby, D. D., Kern, L. W., and Sturm, W. A., "Specification of the Fault-Tolerant Spaceborne Computer," Proceedings of the 1976 International Symposium on Fault-Tolerant Computing, IEEE Cat. No. 76CH1094-2C), pp. 129-133.

D'Angelis, D., and Lauro, J., "Software Recovery in the Fault-Tolerant Spaceborne Computer," Proceedings of the 1976 International Symposium on Fault-Tolerant Computing, IEEE Cat. No. 76CH1094-2C, pp. 143-147.

O'Brien, F. J., "Rollback Point Insertion Strategies," Proceedings of the 1976 International Symposium on Fault-Tolerant Computing, IEEE Cat. No. 76CH1094-2C, pp. 138-142.

Parke, N. G., and Barr, P. C., "The SERF Fault-Tolerant Computer, Part II: Implementation and Reliability Analysis," Digest of Papers, International Symposium on Fault-Tolerant Computing, IEEE Cat. No. 73CH0772-4C (June 1973), pp. 27-38.

Roth, J. P., Bouricius, W. G., Carter, W. C., and Schneider, P. R., Phase II of an Architectural Study for a Self-Repairing Computer, IBM Corporation, T. J. Watson Research Center (November 1967). (Also issued as U. S. Air Force SAMSO-TR-67-106.)

Stiffler, J. J., "The SERF Fault-Tolerant Computer, Part I: Conceptual Design," Digest of Papers, International Symposium on Fault-Tolerant Computing, IEEE Cat. No. 73CH0772-4C (June 1973), pp. 23-26.

Stiffler, J. J., "Architectural Design for Near-100% Coverage," Proceedings of the 1976 International Symposium on Fault-Tolerant Computing, IEEE Cat. No. 76CH1094-2C, pp. 134-137.