THE EVOLUTION OF FAULT TOLERANT COMPUTING AT THE JET PROPULSION LABORATORY & AT UCLA 1960-1986

Algirdas Avizienis
David A. Rennels

June 1987
CSD-870022

FOREWORD

This report presents an overview of the origins and evolution of fault-tolerant computing during the years 1960-1986 at the Jet Propulsion Laboratory, Pasadena, California, and at the University of California, Los Angeles, California, U.S.A. A preliminary version of this report was presented at the Symposium on the Evolution of Fault-Tolerant Computing that was organized by IFIP Working Group 10.4 "Reliable Computing and Fault Tolerance" and took place on June 30, 1986 in Baden, Austria.

The entire text (except Appendix C) will appear as a chapter in the book "The Evolution of Fault-Tolerant Computing" that will be published by Springer-Verlag, Vienna, Austria in 1987. Appendix C consists of the paper "Design of Fault-Tolerant Computers" (reference [Aviz 67a]) that was presented at the 1967 Fall Joint Computer Conference in Anaheim, California. To the extent of our knowledge, this paper introduced the term "fault-tolerant computer" and the concept of fault tolerance into technical literature.

Algirdas Avižienis
David A. Rennels
June 5, 1987

THE EVOLUTION OF FAULT TOLERANT COMPUTING AT THE JET PROPULSION LABORATORY AND AT UCLA: 1960 - 1986

Algirdas Avižienis

David A. Rennels

UCLA Dependable Computing and Fault-Tolerant Systems Laboratory
University of California, Los Angeles, CA 90024, U.S.A.

1. Early Efforts at JPL

The Jet Propulsion Laboratory (JPL) is a research facility in Pasadena, California, that was founded by Professor Theodore von Karman of the California Institute of Technology as a test site in 1936 and was supported by the U.S. Army until October 1958, when it was transferred to the recently founded NASA, the U.S. National Aeronautics and Space Administration. The primary mission of JPL within the NASA structure is to develop unmanned interplanetary spacecraft and to conduct scientific investigations of the other planets of our solar system. Unmanned investigations of the Moon by Ranger spacecraft were the first step in the series of space exploration missions that have continued with the Mariner, Viking, and Voyager series of interplanetary spacecraft that thus far have reached Mercury, Venus, Mars, Jupiter, Saturn and Uranus.

Until 1958 the task of JPL was the development of guided rockets for the U.S. Army (the Corporal and Sergeant series) that offered interesting dependability problems of their own. Upon arriving at JPL in July, 1955 as a recent M.S. graduate from the University of Illinois, A. Avižienis was assigned to solve the problem of dependable guidance parameter insertion in the new Sergeant system. The previous Corporal system employed hand-set dials within the rocket's guidance compartment that did not provide any verification of settings. The design for the new Sergeant system employed a system that read a punched card in a remote command post, transmitted the readings to control a stepping motor in the guidance compartment, and then independently read back the settings to the command post, where they were matched against the card to verify that the proper setting had been accomplished. This custom-tailored error detection mechanism employed relay logic for its control and identified several dependability issues that reemerged a few years later in the context of spacecraft guidance and control computing.

Dependability issues for spacecraft computing emerged at JPL after its transfer to NASA in 1958. A. Avižienis encountered them soon after returning to JPL from a four-year educational leave with a Ph.D. earned in May, 1960 at the Digital Computer Laboratory of the University of Illinois, where he designed parts of the ILLIAC II arithmetic unit and did thesis research on high-speed signed-digit arithmetic that did not address dependability problems. The problem that was perceived at JPL was the need for longevity of the spacecraft computer: unmanned interplanetary missions of 1 to 10 years duration were being considered, while the mean-time-to-failure of contemporary guidance computers was measured in fractions of a year. The NASA-specified OAO satellite on-board processor employed component redundancy that was not suitable for integrated circuits, while the SATURN V guidance computer employed a TMR CPU and was designed for a mission length of 250 hours [Kueh 69]. Both designs from the IBM Federal Systems Division at Owego, N.Y., provided valuable insights, but did not offer a solution to the problem of unattended operation for several years, with the critical performance demands coming at the very end of a long mission that leads to the flyby or orbiting of a planet.

Late 1960 marked the beginning of a multi-year research effort at JPL. It was the search for an effective solution to the problem of building a long-life onboard information processing system for unmanned spacecraft. A. Avižienis was the only full-time researcher for the first three years, working with the support of research group supervisor John J. Wedel, Jr. and with expert advice on spacecraft design, guidance, control, and computing from members of JPL Section 341 (''Flight Computers and Sequencers'') and other members of JPL technical staff.

Valuable information on the SATURN V computer was gained during visits to NASA Marshall Spaceflight Center in Huntsville, AL, and to IBM Federal Systems in Owego, N.Y. Supporting research on the application of error detecting and correcting codes in system design was done under a contract with the Stanford Research Institute, Menlo Park, CA., by Dr. William H. Kautz and others. Two meetings also provided new breadth of viewpoint: a small Conference on Diagnosis of Failures in Switching Circuits (17 talks, 54 participants) in May, 1961 at Michigan State University, and a well-attended (over 500 participants) Symposium on Redundancy Techniques in Computing Systems held in Washington, D.C., on February 6-7, 1962. The latter consisted of 23 presentations that ranged from solid results to sketchy suggestions and philosophical discourses [Wilc 1962].

2. The STAR Computer at JPL

Existing theoretical studies of the long-life problem indicated that large numbers of spares offered a promise of longevity, given that *all* spares could be successfully employed in sequence [Reed 62]. The JPL problem was to translate the idealized ''spare replacement'' system model into a flightworthy implementation of a spacecraft guidance and control computer. About one year after rejoining the JPL staff, A. Avižienis submitted on October 6, 1961 a ten-page JPL Interofffice Memo to Henry A. Curtis, the Manager of Section 341, outlining the design of ''a Self-Testing-And-Repairing System for Spacecraft Guidance and Control'', designated by the acronym ''STAR''. The proposal was fully supported by JPL and NASA research management, and the research effort continued for more than ten years, culminating with the construction and demonstration of the laboratory model of the JPL-STAR computer [Aviz 71a].

The above reference and many other publications have adequately documented the final design of the JPL-STAR computer, and it shall not be detailed here. However, an interesting view of the STAR system evolution may be gained from the October 6, 1961 JPL memo that is reproduced as Appendix A to this paper. The memo has been used as evidence for the patent application filed by JPL in 1967 that resulted in U.S. Patent No. 3, 517, 171, "Self Testing and Repairing Computer", granted on June 23, 1970 to A. Avižienis and assigned to NASA, but it has not been previously published.

The 1961-1965 interval of STAR research was dedicated to the evolution of system organization concepts and an in-depth study of error detection techniques that led to the evolution and adoption of low-cost arithmetic error codes in the STAR design. The AN, residue, and inverse residue arithmetic errror codes were investigated and developed [Aviz 71b]. In 1963, Allen D. Weeks joined the STAR effort as a logic designer and John Buchok as a senior technician. In June of 1965 David A. Rennels, then a doctoral student at Caltech, accepted a summer position with the STAR project and worked on the design and construction of a byte-serial arithmetic unit for "AN"-coded 32-bit operands with the check constant A = 15 [Aviz 73]. D. A. Rennels rejoined the STAR project in 1966 to serve as the principal hardware designer, and later as the main evaluator of the STAR laboratory model, conducting numerous coverage evaluation experiments [Aviz 72], [Renn 73a].

The initial goal of the STAR design was to place the test and repair features into hardware; however, as the design evolved, it became apparent that a hardware-software interface had to be devised to enable the implementation of program rollbacks. To develop the software, the STAR team recruited John A. Rohr, then a Ph.D. student and ILLIAC III software designer at the University of Illinois. He joined JPL in September, 1967 and served as the principal software designer to the end of the STAR project. His work produced a programming system (assembler, loader, simulator, executive), a programming manual, test and demonstration programs, and culminated with the development of STAR rollback techniques and a system executive program that interfaced with the hardware features [Rohr 73a], [Rohr 73b]. The first program was successfully executed on the STAR laboratory "breadboard" model in 1969.

Publication of STAR papers was held back until the design was well defined; the first description was presented on request at a NASA-sponsored meeting in October, 1966 [Aviz 66]. The Central Control Unit described there and later renamed as the Test-And-Repair Processor TARP [Aviz 71a] was quite probably the first appearance of the Service Processor concept. The characterization of the computer as "self-repairing" appeared to be too specific and the term "fault-tolerant" was devised for all subsequent descriptions, beginning with a paper on "Design of Fault-Tolerant Computers" at the 1967 Fall Joint Computer Conference [Aviz 67]. This paper introduced the term "fault-tolerant" and the concept of fault tolerance into technical literature. Several technical papers dealing with various aspects of the STAR design followed during the next six years [Aviz 68], [Aviz 71a], [Gill 72], [Renn 73b], [Rohr 73b], [Aviz 73].

Additional research efforts were initiated in 1968, with Francis P. Mathur undertaking reliability modeling studies [Math 70a], [Math 70b], [Math 71] and George C. Gilley investigating the systematic extension of STAR techniques for the automatic maintenance of an entire autonomous interplanetary spacecraft [Gill 70], [Gill 72]. An adaptation of the STAR design to a specific application was carried out for the JPL Thermoelectric Outer Planet Spacecraft (TOPS) that was intended for the 15-year "Grand Tour" flyby of four outer planets of the solar system [A&A 70]. David K. Rubin led the effort, with support from the entire STAR team. Two elements of the STAR breadboard were designed outside of JPL. A highly reliable magnetic power switch was developed by the Stanford Research Institute, Menlo Park, Calif., and a fault-tolerant read-only memory was built by the M.I.T. Instrumentation Laboratory, Cambridge, Mass., under contracts from JPL.

Regrettably, deep cuts in NASA budgets for unmanned space exploration led to the termination of the TOPS project in 1972, and research on the STAR system breadboard itself was ended in 1973. The STAR system remained in the laboratory as a utility computer for subsequent research until July, 1977, when it was "mothballed" in a JPL storage facility. The California Museum of Science and Industry requested the STAR computer for an exhibit in 1983; however, it could not be located in storage, and its fate remains unknown.

The only apparent direct descendant of the JPL-STAR fault-tolerant architecture was the Fault-Tolerant Spaceborne Computer (FTSC) [Burc 76]. The architecture of FTSC is the joint result of two architectural studies [Conn 72], [Stif 73]. The STAR contribution came through the study done at Ultrasystems, Inc., Irvine, CA. [Conn 72]. The effort was directed by R. B. Conn, while STAR designers A. Avižienis, D. A. Rennels, and J. A. Rohr served as principal consultants in this study, sponsored by the U.S. Air Force. This study updated the STAR architecture by taking into account the increasing levels of circuit integration and combined all processing into a single replaceable CPU. The TARP was significantly refined and designated as the CCU (Configuration Control Unit). A second iteration of the design led to a duplication of the CPU for improved error detection. The complexity of the CCU was reduced by using the CCUs to monitor the pair of CPUs, and relegating to the CPUs the monitoring and recovery management of the rest of the system.

The study done at Raytheon [Stif 73] contributed a single-error correcting and burst-error detecting code in the memory and for bus transfers, as well as single bit-plane replacement within a memory module by means of a "rippler" switch. The STAR approach to pooled memory sparing and assignment was retained. Two "brassboard" models of FTSC were built by Raytheon and were evaluated by extensive fault injection tests [Burc 76]. However, the design needed about 20 custom-designed and radiation-hardened LSI chips. The rapid development of microprocessors made the FTSC single-processor custom design excessively costly to implement for space applications, and the project was terminated before any flight-qualified machines could be built.

D. A. Rennels took over the lead role in fault tolerance research at JPL in 1972. At that time, A. Avižienis initiated a major research project in fault-tolerant computing at UCLA, which is described in the second half of this paper. He also remained in a supporting role as an Academic Member of Technical Staff at JPL until 1981, collaborating with D. A. Rennels and others on some of the subsequent projects that are described below. Contributions of the STAR project were noted by the awarding of the biannual Information Systems Award of the American Institute of Aeronautics and Astronautics in 1979 and of the NASA Exceptional Service Medal in 1980. While both awards were given to A. Avižienis, they actually recognized more than a decade of dedicated effort by a diversely talented group of contributors, who demonstrated the feasibility of building fault-tolerant long-life systems for autonomous operation.

3. After STAR: Fault-Tolerant Distributed Systems at JPL

In the STAR computer hundreds of SSI/MSI chips were needed to make a single processor. Therefore the processor was subdivided into three simpler functional units (each with several hundred SSI/MSI chips), and the memory was subdivided into 4096-word modules. By 1970, major improvements were anticipated in component technology, and it was clear that single-chip processors would soon be available for flight computers. JPL spacecraft were already being designed with several independent digital controllers for command, telemetry and attitude control processing. It could be expected that small general purpose computers would take over these functions and also be embedded in various payload subsystems. It was clear that the next step in computing technology would lead to distributed systems for spacecraft on-board computing.

3.1 The Unified Data System (UDS)

The Unified Data System (UDS) project was begun in the early 1970s as an attempt to define and breadboard a distributed computer architecture to be used on the next generation of spacecraft. At that time the STAR investigation had been concluded, and A. Avižienis was directing a research effort at UCLA. The principal architects of the UDS system were D. Rennels, V. Tyree and B. Riis-Vestergaard. The architecture consisted of a set of computer modules connected by a redundant set of serial busses. Computer modules were of two types, High Level Modules (HLM) and Terminal Modules (TM). This approach made an important distinction between those (HLM) computers which supplied shared global functions (e.g. system executive, telemetry formatting, data analysis) and those (TM) computers which would be embedded in spacecraft subsystems and used for local control and data handling. HLMs were connected only to the buses, were non-dedicated and could be backed up by a common pool of spares. TMs had specific I/O lines to the subsystems in which they were embedded and had to be backed up by dedicated spares which had the same custom I/O connections [Renn 76].

The spacecraft for which this system was designed operated with a synchronous executive. All activities were time-driven. Each computing function was carried out in periodic intervals and all functions were synchronized much like gears on a clock. For example, a television picture was taken every 48 seconds, another instrument would go through eight 6-seconds cycles every picture cycle, etc.

A number of novel features were included in the design of UDS to enhance its reliability [Renn 78a]. A key element was to make its operation simple, predictable, and verifiable. We observed that although the system was required to satisfy hard real-time timing constraints, the minimum resolution on timing of inputs and outputs of various subsystems was a multiple of a basic period of several milliseconds. A real-time interrupt was supplied to all computer modules to define the minimum RTI interval. Inputs were sampled and held over specified intervals (of one or more RTIs) and all outputs were changed only at the end of the intervals. As a result, collections of computations could be treated as atomic events in the specified time intervals. They could be executed in any order with exactly the same results, and unused time could be inserted in each interval to allow an error recovery to occur without changing the input-output characteristics of the computations. Previous spacecraft computers were interrupt driven, making simulation expensive and the system response much less predictable because it was more dependent on the ordering of closely timed events. This approach made the system more predictable and simplified verification of its programs.

Another basic approach was to view the purpose of subsystem-embedded computers as a mechanism to create virtual subsystems which simplified system interfaces and reduced timing constraints on the buses interconnecting the computers. To guard against error propagation due to software errors, the subsystem-embedded Terminal Modules (which were expected to be supplied by subcontractors) were not allowed to initiate bus transmissions. Instead, the High Level Modules moved data between designated memory areas in the TMs on a periodic basis and checked on its validity. A software executive was developed by H. F. Lesh which provided scheduling of tasks in the various computer modules. A PDL-like specification language was used, augmented by timing constructs such as WAIT, WHEN, BACKSTART to suspend and reactivate programs in appropriate synchronization [Lesh 76].

A distributed microcomputer breadboard was constructed and programmed to simulate a Mariner-class spacecraft. There were two HLMs: one dedicated to spacecraft system executive and command, and another to telemetry and control of data movement between computers. TMs were implemented to control a flight television camera, a tape recorder, and the uplink-downlink functions. The breadboard was demonstrated taking a picture of an image across the lab, shipping it out as a digital telemetry stream, and displaying it on a monitor. This research breadboard was a precursor to a distributed computer system which will soon fly on the Galileo mission to Jupiter.

This research resulted in a better understanding of the system-level issues in a distributed real-time system. The approaches used for a fault-tolerant bus system, the techniques for fault containment, the hierarchic control strategies developed to simplify and reduce time-criticality of intercommunications, and special features of the system executive all apply to contemporary spacecraft systems. They were used later in the JPL Fault-Tolerant Building Block Computer (FTBBC). Due to limited resources, the UDS design assumed concurrent fault detection in processors but did not attempt to implement it in the breadboard. The subsequent FTBBC program addressed this issue.

3.2 The Fault-Tolerant Building Block Computer

In the mid 1970s, Reeve Peterson of the Naval Ocean Systems Center (NOSC) was managing a program in VLSI development, and was interested in developing circuits for fault-tolerant systems. He was interested in the possibility of producing fault-tolerant VLSI-based computers which could last throughout the operational life of their host systems and provide "maintenance-free missions." A joint program sponsored by NOSC and NASA was initiated. An architecture was developed by D. Rennels which resulted in a breadboard of the FTBBC [Renn 78b].

The JPL Fault-Tolerant Building-Block Computer (FTBBC) architecture was designed to use a small set of VLSI building-block circuits to interconnect existing microprocessor and memory chips to form Self-checking Computer Modules (SCCM). The SCCMs were designed to contain redundant communications interfaces which allowed them to be connected with other active and spare (SCCM) computers to form a fault-tolerant distributed system. Self-checking (morphic) logic design was used throughout the SCCM design to provide concurrent fault detection in each SCCM computer module. Four building-block circuits were designed: (1) a Memory Interface building block, (2) an I/O building block, (3) a Bus Interface building block which allowed the SCCMs to be connected with similar SCCMs into a network, and (4) a Core building block which compared the outputs of two (duplicate) processors, checked information on internal buses for proper coding, and collected fault messages from other building blocks. After detecting a fault the Core could initiate a program rollback to correct transient faults, and disable the SCCM if the fault persisted, indicating a permanent fault. A breadboard SCCM was constructed, and experimental fault insertion was carried out which verified the concurrent fault detection capabilities of the SCCM. Faults were inserted into both the operational logic and the check circuits by shorting randomly selected wires to ground [Renn 81].

The design used a redundant set of MIL-STD-1553A buses for fault-tolerant intercommunication, and a local executive similar to the UDS design was employed. I/O and software scheduling was synchronized by a real-time interrupt as in UDS. The majority of SCCMs were standby redundant, but the system executive function was duplicated in two SCCMs, so that if one failed, the dedicated backup could continue system control without excessive delays and loss of system state information. Only a single SCCM was constructed, therefore the distributed system has not been tested experimentally. An SCCM with a backup spare was used as a baseline fault-tolerant processor in an autonomous satellite study conducted by JPL for the US Air Force. Fault detection mechanisms and redundant elements were added to a USAF satellite design, and automatic recovery algorithms were written for the SCCM [Aren 83]. David Eisenman was responsible for much of the algorithm development and software architecture of this system.

3.3 Current Research Issues

The FTBBC design is characterized by the use of redundancy at several levels. Spare (memory and processor) chips can be employed within SCCMs to enhance their reliability. Similarly, redundant I/O and memory modules can be employed within each SCCM. Fault tolerance is achieved by also employing spare SCCMs within the network. The use of redundancy at several levels has been found to be necessary in order to achieve long unattended life with a moderate amount of spare hardware. This presents difficult reliability modeling problems, since subsystems with internal redundancy no longer have constant failure rates, and Markov models become very large. Modeling multi-level redundant systems has been shown to be an important new research issue. Similarly, the implementation of concurrent fault detection using self-checking logic raises new VLSI design issues.

In the early 1980s, a program was initiated at JPL to develop methodologies to design self-checking VLSI circuits which also are self-exercising. The goal was to develop a computer which in addition to having concurrent fault detection, also exercises its internal circuits in such a way as to flush out latent faults within milliseconds (concurrently with normal operation). The approach was to modify generic VLSI circuits used in self-checking computer design and add internal self-exercising features to expose both transient errors and permanent faults quickly. Existing circuits for concurrent fault detection were used to detect the faults which are exposed. A self exercising memory design was presented at FTCS-16 [Renn 86a]. Preliminary results indicate that, given that concurrent fault detection is implemented, the additional use of self-exercising design is both effective and relatively inexpensive.

In 1986, NASA support was shifted from fault-tolerant computing to the development of a dataflow machine, so at the current time JPL is assisting other government agencies in the development of fault-tolerant computers. This program involves assisting in program planning and conducting small technology development activities at JPL. Current activities involve evaluating the fault tolerance potential of several existing high-performance architectures, as well as developing and evaluating design approaches for implementing fault tolerance in them. An evaluation of fault-tolerance issues in Hypercube architectures was completed and alternative design approaches have been proposed [Renn 86b].

4. Fault Tolerance and Dependable Computing at UCLA: 1962-1986

4.1 The First Decade: 1962-1972

By 1960 Professor Gerald Estrin at the University of California, Los Angeles, (UCLA) had initiated research on the very stimulating and advanced concept of the "Fixed-plus-Variable" computer architecture. After a few exchanges of visits and presentations between JPL and UCLA, A. Avižienis joined the UCLA Department of Engineering faculty and Dr. Estrin's project in September of 1962 and started teaching undergraduate and graduate courses on computer system design and computer arithmetic, while also directing research at JPL. He presented the first formal course on fault-tolerant computer design at UCLA in 1966, and this

course has been offered annually since then. A second, advanced graduate seminar course was established in 1975, and a seminar on fault-tolerant software was started in 1983. About 500 graduate students have taken at least one of these courses, and over 800 practicing designers and other computer professionals have taken a "short course" version offered annually through UCLA Extension, and also presented in other U.S. cities, in London, in Paris, and in Tokyo.

Research activities in fault tolerance began at UCLA in late 1962, with the first Master's thesis "A Study of Redundant Switching Circuits" by K.B. de Graaf being completed in June, 1964. This thesis was followed by 14 more M.S. and 18 Ph.D. theses on various aspects of fault tolerance, supervised by A. Avižienis, who also authored or co-authored over 100 publications in this area. Several more M.S. and Ph.D. theses were supervised and papers published by other UCLA faculty, as discussed later.

Fault tolerance research at UCLA during the 1963-72 decade was characterized by a very effective collaboration with the STAR project at JPL. The excellent laboratory facilities and expert technicians at JPL enabled the design, construction, and evaluation of the experimental STAR computer, while the academic environment at UCLA provided the opportunity to the researchers to present their results and insights through the rigorous form of Ph.D. dissertations. The STAR computer research led to the UCLA Ph.D. theses by F. P. Mathur [Math 70b], G. C. Gilley [Gill 70], and D. A. Rennels [Renn 73a], all directed by A. Avižienis, and the University of Illinois Ph.D. thesis by J. A. Rohr [Rohr 73a]. The immediate supervision of this research was delegated to A. Avižienis by the thesis committee chairman, Prof. J. A. Robertson of the University of Illinois.

4.2 Two Early Formative Meetings

The first organizational manifestation of UCLA activities in the fault tolerance field was initiated by A. Avižienis in 1965. It was the Workshop on the Organization of Reliable Automata, held in Pacific Palisades, California on February 2-4, 1966 and co-sponsored by the UCLA Department of Engineering and the Technical Committee on Switching Circuit Theory and Logic Design of the IEEE Computer Group. The organizing committee consisted of Dr. Raymond E. Miller (IBM Research, Yorktown Heights, N.Y.), Dr. Robert A. Short (Stanford Research Institute, Menlo Park, CA), and was chaired by Professor Algirdas Avižienis, UCLA. The event attracted 43 participants, many of whom later formed the nucleus of the IEEE CG Technical Committee on Fault-Tolerant Computing that was founded in 1969. Presentations were given by 30 speakers. Texts of the talks were not published as a volume, but the workshop speakers provided three sessions (12 papers) at the First Annual IEEE Computer Conference held on September 6-8, 1967 in Chicago [IEEE 67]. Since a published reference to this 1966 Workshop does not exist, the program is reproduced as Appendix B to this paper.

The success of the Workshop and the continued support and interest of several participants led A. Avižienis to propose to the IEEE Computer Group (IEEE-CG) in early 1969 that a Technical Committee on Fault-Tolerant Computing (TC-FTC) should be formed to promote further activities in this field. The approval of the IEEE-CG Administrative Committee was

granted on November 18, 1969. A letter from Computer Group Chairman E. J. McCluskey, dated November 20, 1969, appointed A. Avižienis to serve as the first Chairman of the new TC-FTC and requested him to invite the founding members. The 18 initial members were: A. Avižienis, W. G. Bouricius, W. C. Carter, H. Y. Chang, J. Goldberg, A. L. Hopkins, E. C. Joseph, E. J. McCluskey, E. G. Manning, F. P. Mathur (TC Secretary), G. Metze, C. V. Ramamoorthy, J. P. Roth, R. A. Short, C. V. Srinivasan, S. A. Szygenda, C. Tung, and S. S. Yau. The new TC met for the first time on May 5, 1970 during the Spring Joint Computer Conference in Atlantic City, New Jersey.

The first objective of the new TC-FTC was the establishment of a technical conference, since an open conference dedicated to the theory and design of fault-tolerant computers had not been held since the 1962 Symposium on Redundancy Techniques for Computing Systems in Washington, D. C. [Wilc 62]. Co-sponsorship of the new meeting and strong organizational support was provided by JPL, and the initial International Symposium on Fault-Tolerant Computing took place on March 1-3, 1971 at the Huntington-Sheraton Hotel in Pasadena, California, with A. Avižienis serving as Symposium Chairman, and W. C. Carter as Program Chairman. A total of 251 participants registered for the meeting, representing the following countries: USA 230, Canada 9, France 4, Japan 3, England 3, Federal Republic of Germany and Italy, 1 each.

The program consisted of 33 papers (including three from France, one from England, and one from Japan) arranged in six sessions, and a panel discussion on diagnosis and testing [Gill 71]. The session titles were: test generation and diagnosis, fault location and testing, reliability modeling and analysis, architecture and design, error protection and recovery, and software reliability. At the conference banquet, the distinguished space scientist and Director of JPL, Dr. William C. Pickering, addressed the participants, outlining plans for future exploration of the planets and noting the key role of fault-tolerant computing in this endeavour. Many participants of the Symposium subsequently visited JPL for a tour and a demonstration of the STAR computer that included fault injection tests during program execution.

The annual series of comprehensive symposia on fault-tolerant computing took off with an auspicious start, and the TC-FTC began building an international membership of fault tolerance experts. The Call for Papers of the second annual Symposium, to be held in the Boston, Massachusetts area on June 19-21, 1972 was distributed at the Pasadena meeting.

4.3 The Second Phase at UCLA: Scope of Activities, 1972-1986

A major new research effort in fault-tolerant computing began at UCLA in July of 1972, when A. Avižienis became the Principal Investigator for a five-year, $887,900 research grant "Fault-Tolerant Computing" from the U.S. National Science Foundation (NSF) and established the Reliable Computing and Fault Tolerance research group that has recently evolved into the Dependable Computing and Fault-Tolerant Systems (DC & FTS) Laboratory. Further research grants and contracts from NSF, the Office of Naval Research, the Federal Aviation Administration, NASA, the State of California, and industry have raised the total funding received for research support since 1972 to over $3.5 million.

Faculty participation in research related to fault tolerance also grew steadily: beginning with two co-investigators (Professors W. W. Chu and D. F. Martin) on the initial NSF grant in 1972, ten more regular faculty members and several visiting professors and research scientists have taken part in research projects during the 1972-1986 period. Graduate student participation has also been very strong. It is estimated that about 200 publications, 30 Ph.D. dissertations, and 20 M.S. theses have resulted from the research on dependable computing and fault-tolerant systems that has been carried out by faculty and students associated with the projects of the Reliable Computing and Fault-Tolerance research group and its successor, the DC & FTS Laboratory, established in July, 1985.

A broad range of research problems have been addressed, including fault-tolerant architectures for distributed systems, supercomputers, and real-time applications, modeling and evaluation of fault-tolerant systems, fault tolerance in associative processors and database machines, fault-tolerant VLSI design, arithmetic error detecting and correcting codes, design of self-checking PLA's, fault-tolerant computer communications, software reliability, and design methodologies for fault-tolerant systems. Research on the tolerance of design faults by design diversity was initiated in 1975, and has resulted in a series of N-version programming experiments, as well as the implementation of the DEDIX distributed supervisory system for N-version software, and the development of a design paradigm for diverse multichannel systems.

4.4 Methodology Research and a Design Paradigm

The specification and design of the STAR computer at JPL involved much improvisation and experimentation with design alternatives. It became apparent that the lessons learned during this process could serve as the foundation for a more orderly approach that would utilize a set of guidelines for the choice of fault masking, error detection, diagnosis, and system recovery techniques.

The first effort to present such guidelines appeared in the 1967 Fall Joint Computer Conference paper "Design of Fault-Tolerant Computers" [Aviz 67]. This paper introduced the concept of a "fault-tolerant system", presented a classification of faults, and outlined the alternate forms of masking, detection, diagnosis, and recovery techniques along with some criteria for choices between "massive" (i.e., masking) and "selective" application of redundancy. The design of the JPL STAR computer was used to illustrate the application of these criteria in choosing the fault tolerance techniques for a spacecraft computer that had long life and autonomy requirements with strict weight and power constraints. The 47 references covered the most relevant published work to mid-1967.

The earlier book by W. H. Pierce "Failure-Tolerant Computer Design" [Pier 65] served as an important reference; however, it must be noted that Pierce's definition of "failure tolerance" corresponded exactly to fault masking in logic circuits, including voting, adaptive, and interwoven logic, redundant relay contact networks, and application of error correcting codes as a masking technique. It is a definitive work on masking forms of redundancy that were known at that time. However, neither error detection, nor fault diagnosis, nor recovery techniques were included as elements of Pierce's "failure-tolerant" computers.

The 1967 paper was the first of a sequence of publications intended to formulate an evolving view of dependable computing as the consequence of a judicious introduction of fault tolerance and fault avoidance during system design. Two different classes of faults - those due to physical causes, and those due to human mistakes and oversights are considered. This evolving view has been presented in a series of papers on the techniques, scope, and aims of fault tolerance. The key contributions to this series are: [Aviz 71c], [Aviz 72b], [Aviz 75a], [Aviz 75b], [Aviz 76], [Aviz 77c], [Aviz 78a], [Aviz 79], [Aviz 82a], [Aviz 82b], [Aviz 84b], [Aviz 86a].

The unifying theme of these papers has been the evolution of a design paradigm for fault-tolerant systems that guides the designer to consider fault tolerance as a fundamental issue throughout the design process. The series shows a progressive refinement of concepts and an expansion of the scope to include the tolerance of "human-made" design and interaction faults. Other recent themes are the balancing of performance and fault tolerance objectives during system partitioning and the integration of subsystem recovery procedures into a multi-level recovery hierarchy. Strong emphasis is also placed on the application of design diversity in all parts of a multichannel system in order to attain tolerance of design faults. Very valuable support for this research effort has come from the author's participation in the activities of IFIP Working Group 10.4, and quite especially from the discussions of fundamental concepts of fault tolerance that have been taking place since the very first meeting of the WG 10.4 in 1981. Most specifically, the work of Dr. J.-C. Laprie has been of great value, especially through collaboration during his stay as a Visiting Professor at UCLA in 1985 [Aviz 86a].

A closely related current effort is the development of a paradigm for the qualitative evaluation of the fault tolerance attributes of complex system designs. This "inverse" of the design paradigm is being developed as part of the research related to the Advanced Automation System for air traffic control in the U. S. [Aviz 87].

4.5 Fault-Tolerant System Design and Analysis

A major research effort in the design of fault-tolerant systems has been a natural consequence of the design methodology research described above. Results in several areas that have been addressed at UCLA since 1972 are summarized below.

Fault-Tolerant High Speed Systems. The emphasis in this area has been on the introduction of low-cost error detection, fault diagnosis, reconfiguration, and recovery techniques into large multiprocessor and "supercomputer" architectures. The results consist of three Ph.D. dissertations [Vine 71], [Thom 77], [Bond 81], as well as several publications [Vine 73], [Sylv 74], [Aviz 74a], [Sylv 75], [Thom 75], [Baqa 76a], [Baqa 76b], [Thom 76], [Aviz 77a], [Aviz 77b], [Aviz 78b], [Aviz 83a], [Ragh 84]. Faculty collaborators in this effort were Profs. M. D. Ercegovac and T. Lang. Under direction of Prof. W. W. Chu, performance and fault tolerance of multiport memories was studied in the Ph.D. thesis of P. Korff [Korf 76]. More recently, D. Rennels and A. Avižienis have initiated a study of the issues involved in implementing fault-tolerance in highly parallel multicomputers. Recommendations for implementing fault-tolerance in hypercube connected systems (e.g., the JPL Hypercube) have been presented [Renn 86b].

Associative Processors and Database Machines. Here the emphasis has been on introducing fault tolerance in a systematic manner and assessing the cost and the effectiveness. The associative processor work includes the Ph.D. thesis by B. Parhami [Parh 73c] and related papers that considered fault tolerance issues in this class of machines [Parh 73b], [Parh 74]. The later database machine work, done in collaboration with Prof. A. F. Cardenas, consisted of one Ph.D. thesis [Alav 81] and two papers [Card 83], [Aviz 84a].

Error Detection Methods. Majority of the research in error detection has dealt with continuing investigation of arithmetic error detecting codes. Previous arithmetic code work had introduced the concepts of "low-cost" arithmetic codes [Aviz 64], inverse residue codes, and multiple residue and AN codes with "low-cost" and "hybrid-cost" variations [Aviz 65], [Aviz 67b], [Aviz 69], [Aviz 71b]. Later results were: algorithms for coded operands [Aviz 73], applications to storage errors [Parh 73a], [Parh 78], coding and algorithms for signed-digit representations [Aviz 81a], and two-dimensional residue codes [Aviz 83a], [Aviz 85a], [Aviz 86b]. Other studies considered external monitoring [Aviz 81b] and diagnosis [Ng 77c]. Further work on arithmetic error codes was contributed by Prof. A. Svoboda [Svob 78], and a Ph. D. dissertation on error-coded algorithms for on-line arithmetic was done by A. Gorji-Sinaki [Gorj 81] under direction of Prof. M. Ercegovac.

Design of Distributed Systems. The "building block" approach to fault-tolerant distributed system design was pioneered by D. A. Rennels [Renn 78a], [Renn 78b], [Grey 84], [Renn 86a], [Renn 86b]. Much of the work in this area has focused on fault-tolerant real-time space systems. D. Rennels has supervised several research studies on the architectures required for the next generation of on-board computer systems [Renn 81a], [Renn 81c], [Depa 82], [Renn 84]. In the area of ground-based distributed systems a study was completed by C. Covey under the direction of D. Rennels which examined hardware augmentations to speed up functions for maintaining consistency and synchronization when data is replicated at several sites [Cove 82]. The Ph.D. thesis of B. Grey examined the potential use of highly fault-tolerant shared storage servers for large distributed systems. A preliminary architecture was completed, and highly secure capabilities-based storage management techniques were explored [Grey 85].

Under direction of A. Avižienis, Ph.D. dissertations were done on interconnection networks [CheH 81], on distributed architectures for N-version software execution [Maka 82a], [Maka 84], and on communication architectures [Ragh 82a], [Ragh 82b], [Ragh 85]. Professor D. S. Parker directed an investigation of distributed operating system and application algorithms, with emphasis on distributed concurrency control [Park 81]. The problem also was investigated in the Ph.D. thesis by R. A. Ramos [Ramo 82], [Park 82a]. A new type of network, called the Gamma network, which is a multi-processor interconnection network with redundant paths was introduced and analysed using redundant number systems [Park 82b], and regular networks were investigated [Malo 82]. Distributed communication systems were investigated under the direction of Prof. M. Gerla. A new fault-tolerant ring architecture was developed which consists of two interleaved rings [Grna 80a]. Computationally efficient techniques for reliability evaluation of a network in which both nodes and links can fail with given probabilities were devised [Grna 79], [Grna 81b]. An extended model of distributed systems, Stochastic Petri Nets, was developed and its properties in both discrete and continuous time as well as some

practical applications were studied in the Ph.D. dissertation of M. Molloy, which demonstrated the capabilities of the Stochastic Petri Net models to analyze systems for both correctness and performance [Moll 81]. Approximations and bounds on the performance of multibus interconnection schemes were derived [Mars 82]. Collaborating faculty included Profs. M. D. Ercegovac, M. Gerla, D. S. Parker, and B. Bussell.

Fault Tolerance Aspects of VLSI Design. This research included studies of self-checking design [OryC 73], [Sum 75], [Sum 76] and the Ph.D. dissertation by S. L. Wang on totally self-checking PLAs [Wang 79], [Wang 81]. The Ph.D. thesis of M. W. Sievers explored computer-aided design of totally self-checking logic [Siev 80], [Siev 81]. Yield-improving designs were investigated in Ph.D. research by T. E. Mangir [Mang 81], [Mang 82]. An investigation into the feasibility of a circuit-oriented approach in enhancing testability of VLSI chips by dynamically controlled partitioning was performed as a Ph.D. thesis by V. G. Oklobdzija, directed by Prof. M. Ercegovac [Oklo 82a], [Oklo 82b]. In 1984, D. A. Rennels initiated studies to develop VLSI circuits which are both self-checking and provide concurrent self-testing during normal operation. This work is being conducted with S. Chau who is nearing completion of a Ph.D. dissertation [Renn 86a]. Dr. Yuval Tamir joined the UCLA faculty in 1985 after receiving his Ph.D. at UC Berkeley [Tami 85]. His Ph.D. thesis addressed a number of problems in the implementation of fault-tolerant VLSI circuits and their application in fault-tolerant multicomputer architecture [Tami 83, 84a, 84b, 84c]. Recently, he has addressed recovery issues in large multicomputer systems [Tami 87]. A study led by Y. Tamir and D. Rennels is also under way to reduce error checking delays in high speed VLSI processors by pipelining the error checks and providing several cycle rollback capability to compensate for delayed error signals.

Modeling and Evaluation of Fault-Tolerant Systems. Directed by A. Avižienis, early work in this area was done by F. P. Mathur [Math 70a], [Math 70b], including the CARE reliability modeling program [Aviz 71a]. Further work, including experimental evaluation of the JPL-STAR computer and the RMS modeling system, was done by D. A. Rennels [Aviz 72a], [Renn 73a], [Renn 73b]. A major advance in Markov modeling was contributed through the Ph.D. dissertation of Y. W. Ng [Ng 76a], who devised an unified model [Ng 73], [Ng 75], [Ng 80], that introduced transient faults [Ng 76b], degradability, and repair [Ng 77a]. The ARIES 76 reliability modeling system (written in APL) contained all these features [Ng 77b], [Ng 78], [Ng 80], and found wide acceptance for education, research, and in industry. The sucessors to ARIES 76 were the ARIES 81 [Maka 82b] and ARIES 82 [Maka 82c] systems that were written in the language C and introduced the model of a "periodically renewed" fault-tolerant system [Maka 81]. ARIES 82 is still widely used in research and industry. Under the direction of D. Rennels, several reliability models were developed by A. DePaula to deal with the use of multilevel redundancy and systems with time-varying failure rates. These were based on recursive integral formulations, with closed form solutions in some cases and numerical integrations in more complex cases to evaluate systems in which Markov matrices become unwieldy [Depa 82]. A separate effort addressed the modeling of transient faults in TMR systems [Merr 75].

4.6 Tolerance of Design Faults by Design Diversity and N-Version Software

By early 1970s significant progress had been made in the tolerance of physical faults, and it became clear that design faults, especially as represented by software "bugs", presented the next challenge to the researchers in fault tolerance. A research effort to attain tolerance of design faults by means of multi-version software was started by A. Avižienis at UCLA in early 1975. The method was first described as "redundant programming" at the April 1975 International Conference on Reliable Software in Los Angeles [Aviz 75a], and was renamed as "N-version programming" in the course of the next two years [Aviz 77d]. The entire UCLA design diversity research effort through mid-1985 has been summarized in [Aviz 85b]. The name "Multi-Version Software" (MVS) is also used.

The N-version programming approach to fault tolerant software systems employs functionally equivalent, yet independently developed software components. These components are executed concurrently under a supervisory system that uses a decision algorithm based on consensus to determine final output values. From its beginning in 1975, the fundamental conjecture of the MVS approach at UCLA has been that errors due to residual software faults are very likely to be masked by the correct results produced by the other versions in the system. This conjecture does not assume independence of errors, but rather a low probability of their concurrence and similarity. MVS systems achieve reliability improvements through the use of redundancy and diversity. A "dimension of diversity" is one of the independent variables in the development process of an MVS system. Diversity may be achieved along various dimensions, e.g., specification languages, specification writers, programming languages, programmers, algorithms, data structures, development environments, and testing methods.

The scarcity of previous results and an absence of formal theories on N-version programming in 1975 led to the choice of an experimental approach: to choose some conveniently accessible programming problems, to assess the applicability of N-version programming, and then to proceed to generate a set of programs. Once generated, the programs were executed as N-version software units in a simulated multiple-hardware system, and the resulting observations were applied to refine the methodology and to build up the concepts on N-version programming. The first detailed assessment of the research approach and a discussion of two sets of experimental results, using 27 and 16 independently written programs, obtained from Prof. D. Berry's software engineering class, was published in 1978 [CheL 78b]. The detailed results appear in the Ph.D. thesis by Liming Chen [CheL 78a].

The preceding exploratory research demonstrated the practicality of experimental investigation and confirmed the need for high quality software specifications. As a consequence, the first aim of the next phase of UCLA research (1979-82) was the investigation of the relative applicability of various software specification techniques. Other aims were to investigate the types and causes of software design faults, to propose improvements to software specification techniques and their use, and to propose future experiments for the investigation of design fault tolerance in software and in hardware.

To examine the effect of specification techniques on multi-version software, an experiment was designed in which three different specifications were used. The first was written in the formal specification language OBJ [Gogu 79b]. The second specification language chosen was the non-formal PDL that was characteristic of current industry practice. English was employed as the third, or "control" specification language, since English had been used in the previous studies [CheL 78b]. The detailed description of the experiment has been reported in the Ph.D. dissertation by J. P. J. Kelly [Kell 82], and the main results have been presented in [Aviz 82c], [Kell 83] and [Aviz 84b].

In parallel with the experiment, a general model for unified interpretation of N-Version Programming and Recovery Block methods was developed. The same model allows modeling of sequential and parallel N-Version Programming (NVP) as well as of the Recovery Block scheme. Following this model, queueing models have been developed to analyze the performance of the fault tolerance techniques. The average segment processing time (average throughput) and reliability were the performance measures. The same queueing models were used for examination of both performance measures as functions of system parameters which include: average segment processing time, recovery rate, repair rate and segment failure probability. The obtained results and a comparison of the two fault tolerance techniques were published in [Grna 80b], [Grna 80c].

The NASA Langley Research Center is sponsoring the NASA Four-University experiment in fault tolerant software which has been underway since 1984. During the summer of 1985, the NASA experiment employed 40 graduate students at four universities to design, code and document 20 diverse software versions of a program to manage redundancy and to compute accelerations for a redundant inertial measurement unit. The analysis of this software currently engages researchers at six sites: UCLA, the University of Illinois at Urbana-Champaign, North Carolina State University, and the University of Virginia, as well as the Research Triangle Institute (RTI), and Charles River Analytics (CRA). Empirical results from this experiment will be jointly published by the cooperating institutions after the verification, certification, and final analysis phases are complete. While the joint results still await publication, some independent results from the UCLA effort led by John P. J. Kelly have been reported in [Kell 86].

In the course of the experiments at UCLA it became evident that the usual general-purpose campus computing services were poorly suited to support the systematic execution, instrumentation, and observation of N-version fault-tolerant software. In order to provide a long-term research facility for experimental investigations of design diversity as a means of achieving fault-tolerant systems, the UCLA Reliable Computing and Fault Tolerance research group designed and implemented the prototype DEDIX (DEsign DIversity eXperiment) system [Aviz 85c], a distributed supervisor and testbed for multiple-version software, at the UCLA Center for Experimental Computer Science. DEDIX is supported by the Center's Olympus Net local network, which utilizes the UNIX-based LOCUS distributed operating system to operate a set of VAX 11/750 computers. The purpose of DEDIX is to supervise and to observe the execution of N diverse versions of an application program functioning as as fault-tolerant N-version software unit. DEDIX also provides a transparent interface to the users, versions, and the input/output system, so that they need not be aware of the existence of multiple versions and recovery algo-

rithms. The prototype DEDIX system has been operational since early 1985. Several modifications have been introduced since then, most of them intended to improve the speed of the execution of N-version software. The first major test of DEDIX that is currently taking place is the experimentation with the set of 20 programs produced by the NASA-sponsored four-university project discussed earlier. At the same time, a formal specification effort for the DEDIX is being initiated.

The past experience at UCLA has pinpointed an effective specification as the keystone of success for N-version software implementation [Aviz 84b]. Significant progress has occurred in the development of formal specification languages since our previous experiments. Our current goal is to compare and assess the ease of use by application programmers of several formal program specification methods. Presently the first choice is the Larch specification language family [Gutt 85]. The NASA Four-University experiment software that was originally specified in English has been specified in Larch [Tai 86], and work has been initiated on specifying parts of DEDIX in Larch as well. Valuable advice and support in the efforts have been received from the originators of Larch, Prof. J. V. Guttag of M.I.T., Dr. J. J. Horning of the DEC Systems Research Center, and Prof. J. M. Wing of Carnegie-Mellon University.

In an MVS system, several versions of a program are executed, usually in parallel, and their intermediate results are compared. In this way, faults in the individual versions are masked by a consensus. Without recovery the accumulation of failures is eventually large enough to saturate the fault-masking ability, and the entire system fails. It is therefore essential to recover these versions as they fail, and to transform the erroneous state of the failed versions to an error-free state from which normal execution can continue. The method of Community Error Recovery (CER) developed at UCLA in the Ph.D. dissertation of K. S. Tso [Tso 87a] makes use of the assumption that at any given time during execution a majority of good versions exists which can supply information to recover the failed versions. Experimental evaluation of the CER method has been performed, using the DEDIX supervisory system and the five diverse programs written at UCLA for the NASA experiment described previously [Kell 86]. A summary of the implementation and modeling of the CER method has been presented in [Tso 86], and a discussion of the evaluation appears in [Tso 87b].

Related research on the NASA experiment programs (in two completed M.S. theses) has considered statistical data on coincident and similar errors [Dora 86], and the branch coverage test has been applied to study various aspects of the programs [Swai 86]. In late 1986 a new experiment was initiated with support from the Sperry Commercial Flight Systems Division of Honeywell, Inc., Phoenix, AZ., in which six teams of two programmers each will use six different programming languages (Pascal, C, Modula-2, Ada, Lisp, and Prolog) to write a flight control program. The goal is to study the diversity that can be attained by the use of very different languages.

4.7 Reliable Software, Formal Specification, and Program Correctness

With the start of the 1972-78 NSF grant, Professor David F. Martin took charge of a study on various aspects of attaining reliable software. This research focused on semantic and pragmatic issues of the correct implementation of programming languages. The investigations carried out comprised a balanced combination of foundational theoretical studies and practical implementations. During the 1972-78 period, Ph.D. dissertations were completed and contributions were made in the areas of acyclic parallel program schemata [Hadj 75], compiler correctness [Chir 76], portable translator writing systems [Heis 76], high-level microprogramming languages and the synthesis of correct microprograms [Patt 77], and the design and proof of correct implementation of an expression-oriented, microcomputer-based high-level programming language [Clea 78].

Beginning in 1973, Professor Joseph A. Goguen conducted algebraic research on program semantics, specification and synthesis [Gogu 79a]. Results included the development of the basic mathematical definitions and results which underlie the algebraic approach to abstract data types [Gogu 78a], a method to introduce abstract error messages into abstract data types and programs [Gogu 77], and the new programming specification language OBJ [Gogu 78b]. An interactive implementation called OBJ-T was done on UCLA's PDP-10 [Gogu 79b]. Professor R. Burstall from the University of Edinburgh and J. Goguen developed an algebraic specification language named CLEAR [Burs 77].

In 1979, a three-year NSF grant "Improvement of the Reliability of Computing," was received that supported the N-Version fault-tolerant software research as well as the continuation of above discussed research concerned with the development of techniques for formal algebraic specification of semantics, and studies of techniques and tools to assure the correct implementation of programming languages. Two Ph.D. dissertations were completed under the guidance of Prof. D. F. Martin. The thesis research conducted by M. Zamfir, developed a mathematical model of concurrent computing agents, the *flow net*. Parallel programming languages can also be defined in this model in the usual syntax-directed fashion [Zamf 82]. The principal accomplishment toward correct implementation of programming languages was the Ph.D. dissertation by R. Bigonha which produced the design of a language and system for the modular specification of the denotational semantics of programming languages [Bigo 82].

Directed by Prof. J. Goguen, Ph.D. thesis research by K. Parsaye-Ghomi produced a theory of higher order data types, i.e., abstract data types with higher order operations and equations [Pars 82].

5. Acknowledgments

The evolution of fault-tolerant computing at JPL and at UCLA began late in 1960. During the past twentysix years, numerous individuals have made significant contributions to these efforts. Many of them are recognized by citations of publications; we also wish to thank those contributors whose names appear in the acknowledgments of the referenced papers, and also to those colleagues whose valuable contributions have remained less evident.

An especially important and pleasant aspect of our research has been the opportunity to welcome at UCLA and to work with visiting colleagues from abroad. We sincerely thank them all and look forward to many return visits: T. Anderson (U.K.), J. Arlat, J.-P. Blanquart, M. Buchwalter, J. P. Chinal, A. Costes, J.-C. Laprie, P. Traverse (France), W. Giloi, W. Goerke, U. Voges (Fed. Rep. Germany), H. Kopetz (Austria), M. Ajmone Marsan, L. Strigini (Italy), P. Gunningberg (Sweden), Y. Hatakeyama, H. Ihara, H. Mori, T. Sasada (Japan), R. Huslende (Norway), A. Grnarov (Yugoslavia), R. Jasinevičius, J. Mockus, V. Statulevičius (Lithuania), C.-L. Chen (P. R. China).

It has been a special pleasure to put together this overview of past efforts as part of the IFIP WG 10.4 Baden Symposium honoring our dear friend and colleague, Dr. William C. Carter. Ever since our first meeting in 1965, Bill has been a constant source of advice, support, and inspiration. Thank you, Bill, for your personal standard of integrity and excellence that set an admirable example for us to follow.

6. References

Note: The single asterisks at the references, e.g. [Alav 81]*, designate Ph.D. dissertations, and the double asterisks (**) designate M.S. theses completed at UCLA.

[A&A 70] "TOPS Outer Planet Spacecraft," *Astronautics and Aeronautics* (Special Issue), Vol. 8, No. 9, September 1970.

[Alav 81]* Alavian, F., "Database Recovery and Fault-Tolerance Analyses in Parallel Associative Database Processors," *Ph.D. dissertation,* UCLA Computer Science Department, University of California, Los Angeles, December 1981; also *Technical Report No. CSD-820318,* March 1982.

[Aren 83] Arens, W., Rennels, D. A., "A Fault-Tolerant Computer for Autonomous Spacecraft," *Digest of FTCS-13, the 13th International Symposium on Fault-Tolerant Computing,* Milano, Italy, June 1983, pp. 467-470.

[Aviz 64] Avižienis, A., "A Set of Algorithms for a Diagnosable Arithmetic Unit," Jet Propulsion Laboratory, Pasadena, California, *Technical Report 32-546,* March 1, 1964.

[Aviz 65] Avižienis, A., "A Study of the Effectiveness of Fault-Detecting Codes for Binary Arithmetic," Jet Propulsion Laboratory, Pasadena, California, *Technical Report 32-711,* September 1, 1965.

[Aviz 66] Avižienis, A., "System Organization of the JPL Self-Testing and Repairing Computer and Its Extension to a Multiprocessor Configuration," *Proceedings of the NASA Seminar on Spaceborne Multiprocessing,* October 1966, Boston, pp. 61-66.

[Aviz 67a] Avižienis, A., "Design of Fault-Tolerant Computers," *AFIPS Conference Proceedings, 1967 Fall Joint Computer Conference*, Vol. 31, Washington, D. C.: Thompson, 1967, pp. 733-743.

[Aviz 67b] Avižienis, A., "Concurrent Diagnosis of Arithmetic Processors," *Digest of the 1st Annual IEEE Computer Conference*, Chicago, IL, September 1967, pp. 34-37.

[Aviz 68] Avižienis, A., "An Experimental Self-Repairing Computer," in *Information Processing '68, Proceedings of the IFIP Congress 1968*, Vol. 2, pp. 872-877.

[Aviz 69] Avižienis, A., "Digital Fault Diagnosis by Low-Cost Arithmetical Coding Techniques," *Proceedings of the Purdue University Centennial Year Symposium on Information Processing*, April 1969, pp. 81-92.

[Aviz 71a] Avižienis, A., Gilley, G. C., Mathur, F. P., Rennels, D. A., Rohr, J. A, Rubin, D. K., "The STAR (Self-Testing-and-Repairing) Computer: An Investigation of the Theory and Practice of Fault-Tolerant Computer Design," *IEEE Trans. on Computers*, Vol. C-20, No. 11, November 1971, pp. 1312-1321; also in *Digest of the 1971 International Symposium on Fault-Tolerant Computing*, Pasadena, CA, March 1971, pp. 92-96.

[Aviz 71b] Avižienis, A., "Arithmetic Error Codes: Cost and Effectiveness Studies for Application in Digital System Design," *IEEE Trans. on Computers*, Vol. C-20, No. 11, November 1971, pp. 1322-1330; also in *Digest of the 1971 International Symposium on Fault-Tolerant Computing*, Pasadena, CA, March 1971, pp. 118-121.

[Aviz 71c] Avižienis, A., "Fault-Tolerant Computing, An Overview," *IEEE Computer*, Vol. 4, No. 1, February 1971, pp. 5-8.

[Aviz 72a] Avižienis, A., Rennels, D. A., "Fault-Tolerance Experiments With The JPL STAR Computer", in *Digest of COMPCON '72 (Sixth Annual IEEE Computer Society Int. Conf.)*, San Francisco, California, 1972, pp. 321-324.

[Aviz 72b] Avižienis, A., "The Methodology of Fault-Tolerant Computing," *Proceedings of the 1st USA-Japan Computer Conference*, Tokyo, October 1972, pp. 405-413.

[Aviz 73] Avižienis, A., "Arithmetic Algorithms for Error-Coded Operands", *IEEE Trans. on Computers*, Vol. C-22, No. 6, June 1973, pp. 567-572; also in *Digest of FTCS-2, the 2nd International Symposium on Fault-Tolerant Computing*, Newton, MA, June 1972, pp. 25-29.

[Aviz 74a] Avižienis, A., Parhami, B., "A Fault-Tolerant Parallel Computer System for Signal Processing," *Digest of FTCS-4, the 4th International Symposium on Fault-Tolerant Computing*, Champaign, IL., June 1974, pp. 2-8 - 2-13.

[Aviz 75a] Avižienis, A., "Fault-Tolerance and Fault-Intolerance: Complementary Approaches to Reliable Computing," *Proceedings of the 1975 International Conference on Reliable Software*, Los Angeles, April 1975, pp. 458-464.

[Aviz 75b] Avižienis, A., "Architecture of Fault-Tolerant Computing Systems," *Digest of FTCS-5, the 5th International Symposium on Fault-Tolerant Computing*, Paris, June 1975, pp. 3-16.

[Aviz 76] Avižienis, A., "Fault-Tolerant Systems," *IEEE Trans. on Computers*, Vol. C-25, No. 12, December 1976, pp. 1304-1312.

[Aviz 77a] Avižienis, A., "Fault-Tolerance and Longevity: Goals for High-Speed Computers on the Future," *Proceedings of the Symposium on High-Speed Computer & Algorithm Organization*, University of Illinois at Urbana-Champaign, April 1977, Academic Press, pp. 173-178.

[Aviz 77b] Avižienis, A., Ercegovac, M., Lang, T., Sylvain, P., Thomasian, A., "An Investigation of Fault-Tolerant Architectures for Large-Scale Numerical Computing," *Proceedings of the Symposium on High-Speed Computer & Algorithm Organization*, University of Illinois at Urbana-Champaign, April 1977, Academic Press, pp. 173-178.

[Aviz 77c] Avižienis, A., "Fault-Tolerant Computing: Progress, Problems, and Prospects," *Information Processing 77, Proceedings of the IFIP Congress 1977*, Toronto, August 1977, pp. 405-420.

[Aviz 77d] Avižienis, A., Chen, L., "On the Implementation of N-version Programming for Software Fault Tolerance During Execution," *Proceedings COMPSAC 77*, (First IEEE-CS International Computer Software and Applications Conference), Chicago, November 1977, pp. 149-155.

[Aviz 78a] Avižienis, A., "Fault-Tolerance: The Survival Attribute of Digital Systems," *Proceedings of the IEEE*, October 1978, 66-10, pp. 1109-1125.

[Aviz 78b] Avižienis, A., Bond, J. W. III, "Fault Tolerance in Large Computing Systems," *Proceedings of the 3rd Jerusalem Conference on Information Technology*, Jerusalem, August 1978, pp. 9-16.

[Aviz 79] Avižienis, A., "Toward a Discipline of Reliable Computing," *Proceedings of the EUROIFIP 79*, (European Conference on Applied Information Technology), London, September 1979, pp. 701-706.

[Aviz 81a] Avižienis, A., "Low Cost Residue and Inverse Residue Error-Detecting Codes for Signed-Digit Arithmetic," *Proceedings of the 5th IEEE Symposium on Computer Arithmetic*, Ann Arbor, MI, May 1981, pp. 165-168.

[Aviz 81b] Avižienis, A., "Fault Tolerance by Means of External Monitoring of Computer Systems," *AFIPS Conference Proceedings*, Vol. 50, May 1981, pp. 27-40.

[Aviz 82a] Avižienis, A., "The Four-Universe Information System Model for Fault-Tolerance," *Digest of FTCS-12, the 12th International Symposium on Fault-Tolerant Computing*, Santa Monica, California, June 1982, pp. 6-13.

[Aviz 82b] Avižienis, A., "Design Diversity - the Challenge of the Eighties," *Digest of FTCS-12, the 12th International Symposium on Fault-Tolerant Computing*, Santa Monica, California, June 1982, pp. 44-45.

[Aviz 82c] Avižienis, A., Kelly, J. P. J. "Fault-Tolerant Multi-Version Software: Experimental Studies of a Design Diversity Approach," *Proceedings of 6th International Conference on Software Engineering (Poster Sessions)*, Tokyo, Japan, September 1982, pp. 101-102.

[Aviz 83a] Avižienis, A., Raghavendra, C. S., "Applications for Arithmetic Error Codes in Large, High-Performance Computers," *Proceedings of the 6th IEEE Symposium on Computer Arithmetic*, Aarhus, Denmark, June 1983, pp. 169-173.

[Aviz 84a] Avižienis, A., Cardenas, A. F., Alavian, F., "On the Effectiveness of Fault Tolerance Techniques in Parallel Associative Database Processors," *Proceedings of the IEEE 1984 International Conference on Data Engineering*, April 1984, pp. 50-59.

[Aviz 84b] Avižienis, A., Kelly, J. P. J., "Fault Tolerance by Design Diversity: Concepts and Experiments," *Computer*, Vol. 17, No. 8, August 1984, pp. 67-80.

[Aviz 85a] Avižienis, A., "Arithmetic Algorithms for Operands Encoded in Two-Dimensional Low-Cost Arithmetic Error Codes," *Proceedings of the 7th IEEE Symposium on Computer Arithmetic*, Urbana, Illinois, May 1985, pp. 285-292.

[Aviz 85b] Avižienis, A., "The N-Version Approach to Fault-Tolerant Software", *IEEE Transactions on Software Engineering*, Vol. SE-11, No. 12, December 1985, pp. 1491-1501.

[Aviz 85c] Avižienis, A., Gunningberg, P., Kelly, J. P. J., Strigini, L., Traverse, P. J., Tso, K. S., Voges, U., "The UCLA DEDIX system: a Distributed Testbed for Multiple-Version Software," *Digest of FTCS-15, the 15th International Symposium on Fault-Tolerant Computing*, Ann Arbor, Michigan, June 1985, pp. 126-134.

[Aviz 86a] Avižienis, A. Laprie, J. C., "Dependable Computing: From Concepts to Design Diversity," *Proceedings of the IEEE*, Vol. 74, No. 5, May 1986, pp. 629-638.

[Aviz 86b] Avižienis, A., "Two-Dimensional Low-Cost Arithmetic Residue Codes: Effectiveness and Arithmetic Algorithms," *Digest of FTCS-16, the 16th International Symposium on Fault-Tolerant Computing*, Vienna, Austria, July 1986, pp. 330-336.

[Aviz 87] Avižienis, A., Ball, D. E., "On the Achievement of a Highly Dependable and Fault-Tolerant Air Traffic Control System," *Computer*, Vol. 20, No. 2, February 1987, pp. 84-90.

[Baqa 76a]** Baqai, R. M., "The Reliability Aspects and Interconnection Network Strategies for ILLIAC IV type Like Array Processors," *M.S. thesis*, UCLA Computer Science Department, University of California, Los Angeles, March 1976.

[Baqa 76b] Baqai, I., Lang, T., "Reliability Aspects of the ILLIAC IV Computer," *Proceedings of the 1976 International Conference on Parallel Processing*, August 1976, pp. 123-131.

[Bigo 82]* Bigonha, R. S., "A Denotational Semantics Implementation System," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, March 1982; also *Technical Report No. CSD-820317*, March 1982.

[Bond 81]* Bond, J. W., III, "A Comparison of Fault-Tolerance in Large Scale Scientific Computer Systems," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, September 1981; also *Technical Report No. CSD-810601*, June 1981.

[Burc 76] D. D. Burchby et al., "Specification of the Fault-Tolerant Space-Borne Computer (FTSC)," *Digest of FTCS-6, the 6th International Symposium on Fault-Tolerant Computing*, Pittsburgh, PA, June 1976, pp. 129-133.

[Burs 77] Burstall, R. M., Goguen, J. A., "Putting Theories Together to Make Specifications," *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, (MIT, Cambridge, Mass.) 1977, pp. 1045-1058.

[Card 83] Cardenas, A. F., Alavian, F., Avižienis, A., "Performance of Recovery Architectures in Parallel Associative Database Processors," *ACM Transactions on Database Systems*, Vol. 8, No. 3, September 1983, pp. 291-323.

[CheH 81]* Chen, H. P. D., "The Analysis and Synthesis of Interconnection Networks for Distributed Computer Systems," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, June 1981; also *Technical Report No. CSD-820203*, February 1982.

[CheL 78a]* Chen, L., "Improving Software Reliability by N-version Programming," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, June 1978; also *Technical Report No. UCLA-ENG-7843*, June 1978.

[CheL 78b] Chen, L., Avižienis, A., "N-version Programming: A Fault Tolerance Approach to Reliability of Software Operation," *Digest of FTCS-8, the 8th International Symposium on Fault-Tolerant Computing*, Toulouse, France, June 1978, pp. 3-9.

[Chir 76]* Chirica, L. M., "Contributions to Compiler Correctness," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, September 1976; also *Technical Report No. UCLA-ENG-7697*, October 1976.

[Clea 78] Cleaveland, J. C., "Design, Implementation and Correctness of an Expression-Oriented Language for Microcomputers," UCLA Computer Science Department, *Technical Report No. UCLA-ENG-7837*, July 1978.

[Conn 72] Conn, R. B., Alexandridis, N. A., Avižienis, A., "Design of a Fault-Tolerant Modular Computer with Dynamic Redundancy," *AFIPS Conference Proceedings*, Vol. 41, Fall JCC 1972, pp. 1057-1067.

[Cove 82] Covey, C., Rennels, D. A., "Hardware Support Mechanisms for Concurrency Control in Local Computer Networks," *Digest of International Workshop on High Level Language Architecture*, Fort Lauderdale, Florida, December 1982.

[Depa 82]* DePaula, A., "Evaluation and Reliability Estimation of Distributed Architectures for On-Board Computers," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, September 1982; also *Technical Report No. CSD-821205*, December 1982.

[Dora 86]** Dorato, K., "Coincident Errors in N-Version Programming," *M.S. thesis*, UCLA Computer Science Department, University of California, Los Angeles, June 1986.

[Gill 70]* Gilley, G. C., "Automatic Maintenance of Spacecraft Systems for Long-Life, Deep-Space Missions", *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, September 1970.

[Gill 71] Gilley, G. C., Editor, *Digest of the 1971 International Symposium on Fault-Tolerant Computing*, Pasadena, California, March 1971.

[Gill 72] Gilley, G. C., "A Fault-Tolerant Spacecraft", in *Digest of FTCS-2, the 2nd International Symposium on Fault-Tolerant Computing*, Newton, MA, June 1972, pp. 105-109.

[Gogu 77] Goguen, J. A., "Abstract Errors for Abstract Data Types," *Proceedings IFIP Working Conference on Formal Description of Programming Concepts*, (ed. J. Dennis), MIT, 1977, pp. 21.1-21.32; also in *Formal Description of Programming Concepts*, (ed. E. Neuhold), North Holland, 1978.

[Gogu 78a] Goguen, J. A., Thatcher, J. W., Wagner, E. G., "An Initial Algebra Approach to the Specification, Correctness, and Implementation of Abstract Data Types," *Current Trends in Programming Methodology*, Vol. 4, *Data Structuring* (ed. R. Yeh), Prentice Hall, 1978, pp. 80-149.

[Gogu 78b] Goguen, J., "Some Design Principles and Theory for OBJ-0, A Language for Expressing and Executing Algebraic Specifications of Programs," *Proceedings, International Conference on Mathematical Studies of Information Processing*, Kyoto, Japan, 1978, pp. 429-475.

[Gogu 79a] Goguen, J. A., "Algebraic Specification," *Research Directions in Software Technology* (ed. P. Wegner), MIT Press, 1979, pp. 370-376.

[Gogu 79b] Goguen, J. A., Tardo, J. J., "An Introduction to OBJ: A Language for Writing and Testing Formal Algebraic Program Specifications," *Proceedings of the Conference on the Specification of Reliable Software*, Cambridge, MA, April 1979, pp. 170-189.

[Gorj 81]* Gorji-Sinaki, A., "Error-coded Algorithms for On-line Arithmetic," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, February 1981; also *Technical Report No. CSD-810303*, March 1981.

[Grey 84] Grey, B. O., Avižienis, A., Rennels, D. A., "A Fault-Tolerant Architecture for Network Storage Systems," *Digest of FTCS-14, the 14th International Symposium on Fault-Tolerant Computing*, Kissimmee, Florida, June 1984, pp. 232-239.

[Grey 85]* Grey, B. O., "FTSS: A Fault-Tolerant Storage System Supporting High Availability and Security in a Distributed Processing Environment," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, March 1985.

[Grna 79] Grnarov, A., Kleinrock, L., Gerla, M., "A New Algorithm for Network Reliability Computation," *Proceedings Computer Networking Symposium*, Gaithersburg, Maryland, December 1979.

[Grna 80a] Grnarov, A., Kleinrock, L., Gerla, M., "A Highly Reliable Distributed Loop Architecture," *Digest of FTCS-10, the 10th International Symposium on Fault-Tolerant Computing*, Kyoto, Japan, October 1980, pp. 319-324.

[Grna 80b] Grnarov, A., Arlat, J., Avižienis, A., "Modeling of Software Fault-Tolerance Strategies," *Proceedings of the 11th Annual Pittsburgh Modeling & Simulation Conference*, University of Pittsburgh, Pennsylvania, Vol. 11, Part 2, May 1980, pp. 571-578.

[Grna 80c] Grnarov, A., Arlat, J., Avižienis, A., "On the Performance of Software Fault-Tolerance Strategies," *Digest of FTCS-10, the 10th International Symposium on Fault-Tolerant Computing*, Kyoto, Japan, October 1980, pp. 251-253.

[Grna 81b] Grnarov, A., Gerla, M., "Multiterminal Analysis of Distributed Processing Systems," *Proceedings of the International Conference on Parallel Processing*, August 1981.

[Gunn 85] Gunningberg, P., Pehrson, B., "Protocol and Verification of a Synchronization Protocol for Comparison of Results," *Digest of FTCS-15, the 15th International Symposium on Fault-Tolerant Computing*, Ann Arbor, Michigan, June 1985, pp. 172-177.

[Gutt 85] Guttag, J. V., Horning, J. J., Wing, J. M., "Larch in Five Easy Pieces," Report No. 5, Digital Equipment Corporation Systems Research Center, Palo Alto, California, July 24, 1985.

[Hadj 75]* Hadjioannou, M., "Acyclic Parallel Program Schemata," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, March 1975; also *Technical Report No. UCLA-ENG-7521*, April 1975.

[Heis 76]* Heiser, J. E., "METAFOR - A Verified, Portable Translator Writing System," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, 1976.

[IEEE 67] *Digest of the 1st Annual IEEE Computer Conference*, Chicago, IL, September 1967.

[Kell 82]* Kelly, J. P. J., "Specification of Fault-Tolerant Multi-Version Software: Experimental Studies of a Design Diversity Approach," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, June 1982; also *Technical Report No. CSD-820927*, September 1982.

[Kell 83] Kelly, J. P. J., Avižienis, A., "A Specification-oriented Multi-version Software Experiment," *Digest of FTCS-13, the 13th International Symposium on Fault-Tolerant Computing*, Milano, Italy, June 1983, pp. 120-126.

[Kell 86] Kelly, J. P. J., Avižienis, A., Ulery, B. T., Swain, B. J., Lyu, R. T., Tai, A., Tso, K. S., "Multi-Version Software Development," *Proceedings IFAC Workshop SAFECOMP 86*, Sarlat, France, October 1986, pp. 43-49.

[Korf 76]* Korff, P., "A Multiaccess Memory," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, June 1976; also *Technical Report UCLA-ENG-7607*, July 1976.

[Kueh 69] Kuehn, R. E., "Computer Redundancy: Design, Performance, and Future", *IEEE Transactions on Reliability*, Vol. R-18, Feb. 1969, pp. 3-11.

[Lesh 76] Lesh, H. F., et. al., "Software Techniques for a Distributed Real-time Processing System," *Proceedings of the IEEE National Aerospace and Electronics Conference (NAECON)*, Dayton, Ohio, pp. 290-295, May 1976.

[Maka 81] Makam, S., Avižienis, A., "Modeling and Analysis of Periodically Renewed Closed Fault-Tolerant Systems," *Digest of FTCS-11, the 11th International Symposium on Fault-Tolerant Computing*, Portland, Maine, June 1981, pp. 134-144.

[Maka 82a]* Makam, S. V., "Design Study of a Fault-Tolerant Computer System to Execute N-Version Software," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, December 1982; also *Technical Report No. CSD-821222*, December 1982.

[Maka 82b] Makam, S., Avižienis, A., "ARIES 81: A Reliability and Life-Cycle Evaluation Tool for Fault-Tolerant Systems," *Digest of FTCS-12, the 12th International Symposium on Fault-Tolerant Computing*, Santa Monica, California, June 1982, pp. 267-274.

[Maka 82c] Makam, S., Avižienis, A., Grusas, G., "ARIES 82 User's Guide," *Technical Report No. 820830*, UCLA Computer Science Department, University of California, Los Angeles, August, 1982.

[Maka 84] Makam, S. V., Avižienis, A., "An Event-Synchronized System Architecture for Integrated Hardware and Software Fault-Tolerance," *Proceedings of the 4th International Conference on Distributed Computing Systems*, San Francisco, California, May 1984.

[Malo 82]** Malony, A. D., "Regular Interconnection Networks," *M.S. thesis*, UCLA Computer Science Department, University of California, Los Angeles, September 1982; also *Technical Report No. CSD-820825*, August 1982.

[Mang 81]* Mangir, T. E., "Use of On-Chip Redundancy for Fault-Tolerant VLSI Design," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, June 1981; also *Technical Report No. CSD-820201*, February 1982.

[Mang 82] Mangir, T. E., Avižienis, A., "Fault-Tolerant Design for VLSI: Effect of Interconnect Requirements on Yield Improvement of VLSI Designs," *IEEE Transactions on Computers*, Vol. C-31, No. 7, July 1982, pp. 609-616.

[Mars 82] Marsan, A. M., Gerla, M., "Markov Models for Multibus Multiprocessor Systems," *IEEE Transactions on Computers*, Vol. C-31, No. 3, March 1982, pp. 239-248.

[Math 70a] Mathur, F. P., Avižienis, A., "Reliability Analysis and Architecture of a Hybrid-Redundant Digital System: Generalized Triple Modular Redundancy with Self-Repair", in *Proceedings of the Spring Joint Computing Conference, AFIPS Conference Proceedings*, Vol. 36. Montvale, N. J.: AFIPS Press, 1970, pp. 375-383.

[Math 70b]* Mathur, F. P., "Reliability Modeling and Estimation of Fault-Tolerant Digital Computers," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, June 1970.

[Math 71] Mathur, F. P., "On Reliability Modeling and Analysis of Ultra-Reliable Fault-Tolerant Digital Systems," *IEEE Transactions on Computers*, Vol. C-20, No. 11, November 1971, pp. 1376-1382.

[Merr 75] Merryman, P. M., Avižienis, A., "Modeling Transient Faults in TMR Computer Systems," *Proceedings of the 1975 Reliability and Maintainability Symposium*, Washington, D.C., January 1975, pp. 333-339.

[Moll 81]* Molloy, M., "On the Integration of Delay and Throughput Measures in Distributed Processing Models," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, June 1981; also *Technical Report No. CSD-810921*, September 1981.

[Ng 73]** Ng, Y. W., "Reliability Modeling for Fault-Tolerant Computers," *M.S. thesis*, UCLA Computer Science Department, University of California, Los Angeles, December 1973.

[Ng 75] Ng, Y. W., Avižienis, A., "A Unifying Reliability Model for Closed Fault-Tolerant Systems," *Digest of FTCS-5, the 5th International Symposium on Fault-Tolerant Computing*, Paris, June 1975, pp. 224.

[Ng 76a]* Ng, Y. W., "Modeling and Analysis of Fault-Tolerant Computers," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, September 1976; also *Technical Report No. UCLA-ENG-7698*, September 1976.

[Ng 76b] Ng, Y. W., Avižienis, A., "A Model for Transient and Permanent Fault Recovery in Closed Fault Tolerant Systems," *Digest of FTCS-6, the 6th International Symposium on Fault-Tolerant Computing*, Pittsburgh, June 1976, pp. 182-188.

[Ng 77a] Ng, Y. W., Avižienis, A., "A Reliability Model for Gracefully Degrading and Repairable Fault-Tolerant Systems," *Digest of FTCS-7, the 7th International Symposium on Fault-Tolerant Computing*, Los Angeles, June 1977, pp. 22-28.

[Ng 77b] Ng, Y. W., Avižienis, A., "ARIES - An Automated Reliability Estimation System for Redundant Digital Structures," *Proceedings of the 1977 Annual Reliability and Maintainability Symposium*, Philadelphia, January 1977, pp. 108-113.

[Ng 77c] Ng, Y. W., Avižienis, A., "Local Irredundancy in Combinational Circuits," *Digest of FTCS-7, the 7th International Symposium of Fault-Tolerant Computing*, Los Angeles, June 1977, pp. 109-113.

[Ng 78] Ng, Y. W., Avižienis, A., "ARIES 76 User's Guide," *Technical Report No. UCLA-ENG-7894*, UCLA Computer Science Department, University of California, Los Angeles, December 1978.

[Ng 80] Ng, Y. W., Avižienis, A., "A Unified Reliability Model for Fault-Tolerant Computers," *IEEE Transactions on Computers*, Vol. C-29, No. 11, pp. 1002-1011, November 1980.

[Oklo 82a]* Oklobdzija, V. G., "Design for Testability of VLSI Structures through the Use of Circuit Techniques," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, 1982; also *Technical Report No. CSD-820820*, August 1982.

[Oklo 82b] Oklobdzija, V. G., Ercegovac, M. D., "Testability Enhancement of VLSI Using Circuit Structures," *Proceedings of IEEE 1982 International Conference on Circuits and Computers*, New York, 1982.

[OryC 73]** Ory-Cristelly, R., "Design of a Dynamically Checked, Signed-Digit Arithmetic Unit," *M.S. thesis*, UCLA Computer Science Department, University of California, Los Angeles, November 1973; also *Technical Report No. UCLA-ENG-7366*, November 1973.

[Parh 73a] Parhami, B., Avižienis, A., "Application of Arithmetic Error Codes for Checking of Mass Memories," *Digest of FTCS-3, the 3rd International Symposium on Fault-Tolerant Computing*, June 1973, pp. 47-51.

[Parh 73b] Parhami, B., Avižienis, A., "Design of Fault-Tolerant Associative Processors," *Proceedings of the 1st Annual Symposium on Computer Architecture*, Gainesville, FL., December 1973, pp. 141-145.

[Parh 73c]* Parhami, B., "Design Techniques for Associative Memories and Processors," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, March 1973; also *Technical Report No. UCLA-ENG-7321*, March 1973.

[Parh 74] Parhami, B., Avižienis, A., "A Study of Fault-Tolerance Techniques for Associative Processors," *AFIPS Conference Proceedings*, Vol. 43, National Computer Conference, Chicago, May 1974, pp. 643-652.

[Parh 78] Parhami, B., Avižienis, A., "Detection of Storage Errors in Mass Memories Using Low-Cost Arithmetic Codes," *IEEE Transactions on Computers*, Vol. C-27, No. 4, April 1978, pp. 302-308.

[Park 81] Parker, D. S., Popek, G. J., et al., "Detection of Mutual Inconsistency in Distributed Systems," *Proceedings of the 5th Berkeley Workshop on Computer Networks and Distributed Data Management*, Emeryville, California, February 1981.

[Park 82a] Parker, D. S., Ramos, R., "A Distributed File System Architecture Supporting High Availability," *Proceedings of the 6th Berkeley Workshop on Distributed Data Management & Computer Networks*, Asilomar, California, February 1982, pp. 161-183.

[Park 82b] Parker, D. S., Raghavendra, C. S., "The Gamma Network: A Multiprocessor Interconnection Network with Redundant Paths," *Proceedings of the 9th Annual Symposium on Computer Architecture*, Austin, Texas, April 1982, pp. 73-80.

[Pars 82]* Parsaye-Ghomi, K., "Higher Order Data Types," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, January 1982; also *Technical Report No. CSD-820112*, January 1982.

[Patt 77] Patterson, D. A., "Verification of Microprograms," *Technical Report No. UCLA-ENG-7707*, UCLA Computer Science Department, January 1977.

[Pier 65] Pierce, W. H., "Failure-Tolerant Computer Design," Academic Press: New York and London, 1965.

[Ragh 82a]* Raghavendra, N., "Fault Tolerance in Computer Communication Architectures," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, June 1982; also *Technical Report No. CSD-820928*, September 1982.

[Ragh 82b] Raghavendra, C. S., Gerla, M., Avižienis, A., "Reliability Optimization in the Design of Distributed Systems," *Proceedings of the 3rd International Conference on Distributed Computing Systems*, Miami, Florida, October 1982, pp. 388-393.

[Ragh 84] Raghavendra, C. S., Avižienis, A., Ercegovac, M. D., "Fault Tolerance in Binary Tree Architectures," *IEEE Transactions on Computers*, Vol. C-33, No. 6, June 1984, pp. 568-572.

[Ragh 85] Raghavendra, C. S., Gerla, M., Avižienis, A., "Reliable Loop Topologies for Large Local Computer Networks," *IEEE Transactions on Computers*, Vol. C-34, No. 1, January 1985, pp. 46-55.

[Ramo 82]* Ramos, R. A., "High Reliability, Availability and Consistency in Distributed Systems," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, September 1982; also *Technical Report No. CSD-821214*, December 1982.

[Reed 62] Reed, I. S., Brimley, D. E., "On Increasing the Operating Life of Unattended Machines", RAND Corp., Memo. RM-3338-PR, November 1962.

[Renn 73a]* Rennels, D. A., "Fault Detection and Recovery in Redundant Computer Using Standby Spares", *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, June 1973; also *Technical Report No. UCLA-ENG-7355*, July 1973.

[Renn 73b] Rennels, D. A., Avižienis, A., "RMS: A Reliability Modeling System for Self-Repairing Computers," *Digest of FTCS-3, the 3rd International Symposium on Fault-Tolerant Computing*, June 1973, pp. 131-135.

[Renn 76] Rennels, D.A., et al., "The Unified Data System: A Distributed Processing Network for Control and Data Handling on a Spacecraft," *Proceedings of the IEEE National Aerospace and Electronics Conference (NAECON)*, Dayton, Ohio, May 1976, pp. 283-289.

[Renn 78a] Rennels, D. A., "Architectures for Fault-Tolerant Spacecraft Computers", *Proceedings of the IEEE*, October 1978, Vol. 66, No. 10, pp. 1255-1268.

[Renn 78b] Rennels, D. A., Avižienis, A., Ercegovac, M., "A Study of Standard Building Blocks for the Design of Fault-Tolerant Distributed Computer Systems", *Digest of FTCS-8, the 8th International Symposium on Fault-Tolerant Computing*, Toulouse, France, June 1978, pp. 144-149.

[Renn 81a] Rennels, D. A., et. al., *Fault-Tolerant Computer Study Final Report*, JPL Publication 80-73, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, February 1981.

[Renn 81b] Rennels, D. A., "Some Past Experiments and Future Plans in the Experimental Evaluation of Fault-Tolerance", *UCLA Computer Science Department Quarterly*, vol. 9, no. 2, Spring 1981, University of California, Los Angeles, pp. 91-98.

[Renn 81c] Rennels, D. A., DePaula, A., Fremont, M., "Fault-Tolerant Design Considerations for Future Spacecraft Computer Systems," *UCLA Report Prepared for the Aerospace Corporation*, El Segundo, California, Aerospace Library Call Number A81-04858, October 1981.

[Renn 84] Rennels, D. A., "A Building Block Architecture for a High-Speed Distributed Processing System," *Digest GOMAC Government Microcircuits Applications Conference*, Las Vegas, Nevada, November 1984.

[Renn 86a] Rennels, D. A., Chau, S., "A Self-Exercising Self-Checking Memory Design", *Digest of FTCS-16, the 16th International Symposium on Fault-Tolerant Computing*, Vienna, Austria, July 1986, pp. 358-363.

[Renn 86b] Rennels, D. A., "On Implementing Fault Tolerance in Binary Hypercubes," *Digest of FTCS-16, the 16th International Symposium on Fault-Tolerant Computing*, Vienna, Austria, July 1986, pp. 344-349.

[Rohr 73a] Rohr, J. A., "System Software for a Fault-Tolerant Digital Computer," *Ph.D. dissertation*, Department of Computer Science, University of Illinois, Urbana, Illinois, February 1973.

[Rohr 73b] Rohr, J. A., "STAREX Self-Repair Routines: Software Recovery in The JPL-STAR Computer", *Digest of FTCS-3, the 3rd International Symposium on Fault-Tolerant Computing*, Palo Alto, California, June 1973, pp. 11-16.

[Siev 80]* Sievers, M. W., "Computer-Aided Design and Reliability of a General Logic Structure for Custom VLSI," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, June 1980; also *Technical Report No. CSD-820111*, Janu-

ary 1982.

[Siev 81] Sievers, M., Avižienis, A., "Analysis of a Class of Totally Self-Checking Functions Implemented in a MOS LSI General Logic Structure," *Digest of FTCS-11, the 11th International Symposium on Fault-Tolerant Computing*, Portland, Maine, June 1981, pp. 256-261.

[Stif 73] Stiffler, J. J., Parke IV, N. G., Barr, P. C., "The SERF Fault-Tolerant Computer," Parts I and II, *Digest of FTCS-3, the 3rd International Symposium on Fault-Tolerant Computing*, Palo Alto, California, June 1973, pp. 23-31.

[Sum 75]** Sum, E., "Evaluation Techniques for Self-Checking Logic Circuits," *M.S. thesis*, UCLA Computer Science Department, University of California, Los Angeles, June 1975.

[Sum 76] Sum, E. K. S., Avižienis, A., "A Probabilistic Model for the Evaluation of Signal Reliability of Self-Checking Logic Circuits," *Digest of FTCS-6, the 6th International Symposium on Fault-Tolerant Computing*, Pittsburgh, June 1976, pp. 83-87.

[Svob 78] Svoboda, A., "Arithmetic Circuit Fault Detection by Modular Encoding," *Proceedings of the Fourth IEEE Symposium on Computer Arithmetic*, Santa Monica, California, October 1978, pp. 208-219.

[Swai 86]** Swain, B., "Group Branch Coverage Testing of Multi-Version Software," *M.S. thesis*, UCLA Computer Science Department, University of California, Los Angeles, December 1986; also *Technical Report No. CSD-860013*, December 1986.

[Sylv 74]** Sylvain, P., "Evaluating the Array Machine," *M.S. thesis*, UCLA Computer Science Department, University of California, Los Angeles, June 1974.

[Sylv 75] Sylvain, P., Vineberg, M., "The Design and Evaluation of the Array Machine: A High-Level Language Processor," *Proceedings of Second Annual Symposium on Computer Architecture*, Houston, TX, January 1975, pp. 119-125.

[Tai 86]** Tai, A. T., "A Study of the Application of Formal Specification for Fault-Tolerant Software," *M.S. thesis*, UCLA Computer Science Department, Los Angeles, California, June 1986.

[Tami 83] Tamir, Y., Séquin, C. H., "Self-Checking VLSI Building Blocks for Fault-Tolerant Multicomputers," *International Conference on Computer Design*, Port Chester, New York, November 1983, pp. 561-564.

[Tami 84a] Tamir, Y., Séquin, C. H., "Design and Application of Self-Testing Comparators Implemented with MOSPLAs," *IEEE Transactions on Computers*, Vol. C-33, No. 6, June 1984, pp. 493-506.

[Tami 84b] Tamir, Y., Séquin, C. H., "Error Recovery in Multicomputers Using Global Checkpoints," *13th International Conference on Parallel Processing*, Bellaire, Michigan, August 1984, pp. 32-41.

[Tami 84c] Tamir, Y., Séquin, C. H., "Reducing Common Mode Failures in Duplicate Modules," *International Conference on Computer Design*, Port Chester, New York, October 1984, pp. 302-307.

[Tami 85] Tamir, Y., "Fault Tolerance for VLSI Multicomputers," *Ph.D. dissertation*, also *CS Division Report No. UCB/CSD 86/256*, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, California, August 1985.

[Tami 87] Tamir, Y., Gafni, E., "A Software-Based Hardware Fault Tolerance Scheme for Multicomputers," *Proceedings of the 16th Parallel Processing Conference*, St. Charles, Illinois, August 1987.

[Thom 75] Thomasian, A., Avižienis, A., "Dynamic Scheduling of Tasks Requiring Multiple Processors," *Proceedings of the 11th Annual IEEE Computer Society Conference*, Washington, DC, September 1975, pp. 76-80.

[Thom 76] Thomasian, A., Avižienis, A., "A Design Study of a Shared-Resource Computing System," *Proceedings of the 3rd Annual Symposium on Computer Architecture*, Clearwater, FL., January 1976, pp. 105-112.

[Thom 77]* Thomasian, A., "A Design Study of a Shared Resource Array Processing System," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, June 1977; also *Technical Report No. UCLA-ENG-7702*, August 1977.

[Tso 86] Tso, K. S., Avižienis, A., Kelly, J. P. J., "Error Recovery in Multi-Version Software," in *Proceedings IFAC Workshop SAFECOMP'86*, Sarlat, France, October 1986, pp. 35-41.

[Tso 87a]* Tso, K. S., "Recovery and Reconfiguration in Multi-Version Software," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, March 1987; also *Technical Report No. CSD-870013*, March 1987.

[Tso 87b] Tso, K. S., Avižienis, A., "Community Error Recovery in N-Version Software: A Design Study with Experimentation," *Digest of FTCS-17, the 17th International Symposium on Fault-Tolerant Computing*, Pittsburgh, Pennsylvania, July 1987.

[Vine 71]* Vineberg, M. B., "Implementation of a Higher Level Language on an Array Machine," *Ph. D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, June 1971; also *Technical Report No. UCLA-ENG-7157*, 1971.

[Vine 72] Vineberg, M., Avižienis, A., "Implementation of a Higher-Level Language on an Array Machine," *Proceedings of COMPCON'72, 6th Annual IEEE Computer Society International Conference*, September 1972, pp. 37-39.

[Vine 73] Vineberg, M., Avižienis, A., "Implementation of a Higher-Level Language on an Array Machine," *Proceedings of the International Workshop on Computer Architecture*, Grenoble, France, June 1973.

[Wang 79] Wang, S. L., Avižienis, A., "The Design of Totally Self-Checking Circuits Using Programmable Logic Arrays," *Digest of FTCS-9, the 9th International Symposium on Fault-Tolerant Computing*, Madison, WI, June 1979, pp. 173-180.

[Wang 81]* Wang, S. L., "The Design of Totally Self-Checking Circuits by Using Programmable Logic Arrays," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, June 1981; also *Technical Report No. CSD-810608*, June 1981.

[Wilc 62] Wilcox, R. H., Mann, W. C., editors, "Redundancy Techniques for Computing Systems", Spartan Press, Washington, DC, 1962.

[Zamf 82]* Zamfir, M., "Syntax and Semantics of Concurrent Computing," *Ph.D. dissertation*, UCLA Computer Science Department, University of California, Los Angeles, June 1982; also *Technical Report No. CSD-820819*, August 1982.

APPENDIX A

The following text is an exact reproduction of the JPL Spacecraft Computers and Sequencers Section interoffice memorandum addressed to Section Manager Henry A. Curtis that initiated the JPL STAR computer research project in 1961. It is the earliest existing description of the STAR concept, and it was used as evidence to support the U.S. Patent application filed by JPL in 1967. Subsequently, U.S. Patent No. 3, 517, 671 "Self Testing and Repairing Computer" was granted to A. Avižienis (assigned to NASA) on June 23, 1970. Figures 1 and 2 are taken directly from the original.

JET PROPULSION LABORATORY **INTEROFFICE MEMO**

TO: H. A. Curtis **FROM:** A. A. Avizienis **DATE:** 10-6-61

SUBJECT: Preliminary Discussion of the Logical Design of a Self-Testing and Repairing System for Spacecraft Guidance and Control.

I. OBJECTIVE

The objective of this memo is to discuss the organization and logical design of a Self-Testing-And-Repairing (STAR) system which can perform the guidance computer and sequencer functions in a spacecraft. It is expected that the self-repair property will increase the probability of successful operation of the system on long-term missions. It may also be a contribution in advancing the state of the art in design of reliable computing systems.

II. CHARACTERISTICS

The following system characteristics are considered to be essential in the STAR system for spacecraft guidance computing and sequencing:

1. Self-repair (including input and output mechanisms).
2. Evolution from presently used hardware and techniques.
3. Flexibility and growth potential.
4. Linkage with ground-based computers.
5. Gradual ("graceful") degradation upon accumulation of failures.

Because of the special purpose of the STAR system, it is possible to make certain assumptions about its operational requirements and conditions. Several significant features are:

1. Speed requirement is relatively low.
2. Sequencer function as well as computing must be performed.
3. Available power is limited.
4. There may be long periods of idleness (standby).

5. Inputs are from transducers and radio link; outputs to radio link and actuators.

6. Operation may be linked with ground-based computer or operator.

7. Program is generally fixed for one mission; missions are variable in duration and purpose.

8. Self-repair must extend to input and output devices.

9. New techniques and components (adaptation, cryogenics, etc.) may become available.

10. Length of missions and reliability requirements are going to increase.

The listed objectives and properties form the basis for the following discussion of a possible configuration of the STAR system.

III. SELF-REPAIR

In order to achieve self-repair in space environment, the STAR system must be capable of auto-diagnosis or fault-masking. Auto-diagnosis must be followed by the replacement of a permanently defective part (to be called *failure*) or by a repetition of the operation if the result is diagnosed as invalid because of a transient malfunction (to be called *error*). Thus, errors are corrected by means of time redundancy, and failures are corrected by means of equipment redundancy.

Fault-masking is an alternative approach: here failure or temporary malfunction of an element does not produce an erroneous output because other (redundant) elements mask the effect of the failure. Masking will not be effective if an error (transient malfunction) is caused by external noise.

When information is coded in an error-detecting code, failures and both types of errors (of internal and external origin) will be detectable (subject to limitations of the code). The diagnostic (error-detecting) equipment itself, however, should not malfunction during diagnosis. For this purpose fault-masking must be incorporated with the diagnostic equipment. Furthermore, it must be especially protected (shielded) against externally originated errors. Automatic time redundancy (repetition of all diagnoses with voting) may be applicable here.

A review of above discussed characteristics leads to the preliminary choice of the organization of the STAR system which is shown in Figure 1.

The system consists of a central Diagnostic Control and of arrays of peripheral Function Units of several types. Each array of Function Units includes one or more reserve units; a reserve unit is chosen when the Diagnostic Control detects a failure in the (presently) operating unit. The failed unit is permanently disconnected and the next reserve unit is connected to Diagnostic Control.

The power consumption is least when the reserve units are stored "cold"; furthermore the system will be operative as long as at least one unit in an array remains operative. These reasons relegate "triplicated with voting" use of Function Units to second place. The number of identical Function Units in one array may vary according to the length of the mission, the relative importance of the unit in the system, weight limitations, and the reliability of the type of unit used. The internal design of the Function Units is relatively independent of the Diagnostic Control as long as the standard format of information is retained; this should facilitate the introduction of changes and improvements.

IV. THE DIAGNOSTIC CONTROL

The Diagnostic Control of the STAR system performs all functions of a control unit in a conventional computer. Furthermore, it evaluates the validity of all information which passes between the operating Function Units and performs the self-test and self-repair sequences when invalid information is detected.

Obviously, all information flow must be routed through the Diagnostic Control; this requirement limits the speed of the entire system. All information must be coded in an error-detecting code; either a uniform code must be used in all Function Units, or a code conversion must be performed by the Diagnostic Control (conversion in the Function Unit itself is an alternative). Other failure-indicating inputs from the operating Function Units may be useful, such as indication of power loss, etc.

The Diagnostic Control itself must be the most reliable part of the STAR system. Fault-masking and redundancy at component level offer the most attractive approach to make it reliable. Self-diagnosis during periods of idleness may be applicable; this, however, requires a spare control unit or complete transfer to ground control in case of a failure.

The Diagnostic Control receives an instruction from the Fixed Storage unit and evaluates its validity. If the validity is questionable, the instruction is obtained once more (time-redundant check for errors); if it is still invalid, a reserve Fixed Storage unit is consulted for the instruction.

If the instruction is valid, the proper Function Unit is instructed to execute it; the result is then brought into the Diagnostic Control and tested for validity. Again, in case of indicated invalidity, the operation is repeated (one or more times); if the result is still invalid, a failure of the Function Unit is assumed. The Diagnostic Control contains a switching arrangement which is now actuated and the next Function Unit in the array is activated and connected to the Diagnostic Control. The failed unit is permanently deactivated and disconnected.

A flow diagram of the procedure described above is shown in Figure 2. A special "degradation procedure" is necessary when the reserve Function Units of one type are exhausted, or when no valid instructions are available.

The Diagnostic Control is also the most likely location for the sequence generator and the clock of the STAR system. Input, output, arithmetic, and storage functions are relegated to the Function Units. It is desirable to keep the Diagnostic Control unit as small as possible, since it is the most vulnerable part of the STAR system. An effort should be made to incorporate most functions into the Function Units. A single general-purpose Diagnostic Control unit then could be used as center of many STAR systems of varying size, capacity and purpose.

V. THE FUNCTION UNITS

The most obviously needed types of Function Units are (1) Fixed Store, (2) Arithmetic, (3) Memory, (4) Input and (5) Output. Others may be found necessary, for instance, Clock and Sequencer, or Scientific Data Reducer units.

A. Fixed Store Function Units

These units contain programs for guidance computations, and all emergency, diagnostic and other internal procedures. Scientific data reduction program may also be contained here. The program data (instructions) are coded for detection of errors and failures. The redundancy is in the form of reserve units; time redundancy (repeated readout) is also to be used. An independent indication of the origin of the instruction delivered to the Diagnostic Control may be necessary.

B. Arithmetic Function Units

The Arithmetic Function Units must have a provision to retain the error-detecting code in the results of arithmetical operations. Furthermore, an independent indication of which arithmetic operation has been performed is desirable with the result; in this manner improper interpretation of the instruction by the Arithmetic unit can be detected.

C. Memory Function Units

This type of unit is used for storage of intermediate results and other non-fixed information. Two units are likely to be used at once to avoid loss of information; an error-correcting code is an alternative. An independent indication (to the Diagnostic Control) of the address from which (or into which) information was delivered would avoid errors due to incorrect address decoding.

D. Input Function Units

These units are the various sensors and transducers on the spacecraft and the radio receiver. Input data must appear in digital form and be coded in an error-detecting code. Such code may appear on shaft-position digitizers (code wheels), etc. Validity check procedures must be available for each input, either in form of a preliminary and terminal checkout, or by repeated measurements (by the same or different instruments). Spare input devices should be available for self-repair.

The radio receiver is expected to deliver coded information which can be checked for validity; a spare receiver is desirable.

E. Output Function Units

These units are the various actuators and the radio transmitter. An independent feedback of the output information as delivered by the output units will provide a check of their performance. Voting-type redundancy of actuators should be considered. A spare radio transmitter is considered desirable.

VI. OTHER CONSIDERATIONS

The design of an initial breadboard Diagnostic Control and Function Units would utilize the hardware and techniques which are most readily available, and have been used in earlier flights. There is no conceptual difficulty in starting a design with magnetic and semiconductor elements; adaptive switching circuits and new components may be introduced at a suitable time. The logical design of the STAR system is relatively independent of any specific set of circuits or components.

The separation into Function Units allows most flexibility in assembling a complete STAR system; only the format of information must be retained. Thus, many different systems may be put together from the set of units.

The design problem is also subdivided into: (1) development of the STAR system concept, (2) development of the information format; (3) development of the Diagnostic Control unit, (4) development of the various Function Units. It is necessary to consider the division of computational requirements between Earth-based computers and the STAR system in the spacecraft. The relative reliability and accuracy must be evaluated; however, duplicate operations with an assigned priority and comparison offer the greatest flexibility.

The ultimate use of the spacecraft for long exploratory flights and orbiting of planets favors a complete STAR system in the computer; however, provisions should be made to utilize ground support and also to achieve "graceful" degradation by relinquishing functions to ground computing systems if a non-repairable failure occurs in the spacecraft STAR system.

The degradation of the STAR computer should occur in an orderly manner, according to a special program in the Fixed Store. Preferably, control and execution of lost functions is requested from the ground. The communication with ground may be continuous and used as a back-up check while the STAR system is functioning properly.

The redundancy of communication equipment is implied by the above requirement; the level of redundancy remains to be established as well as the procedure of switchover (if needed).

VII. CLOSING REMARK

The above presented concept of the STAR system has been put down on paper for the first time; inevitably it is uncertain and too general on many important problems. However, it appears to offer an interesting and potentially very reliable computing system for the 1965-70 period of space exploration.

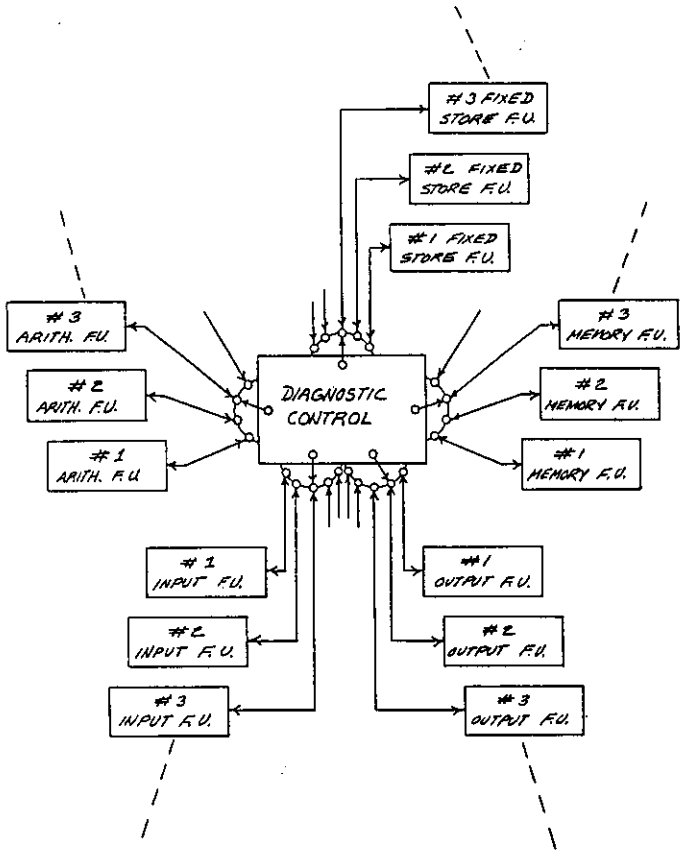

FIG. 1 BLOCK DIAGRAM OF "STAR" SYSTEM

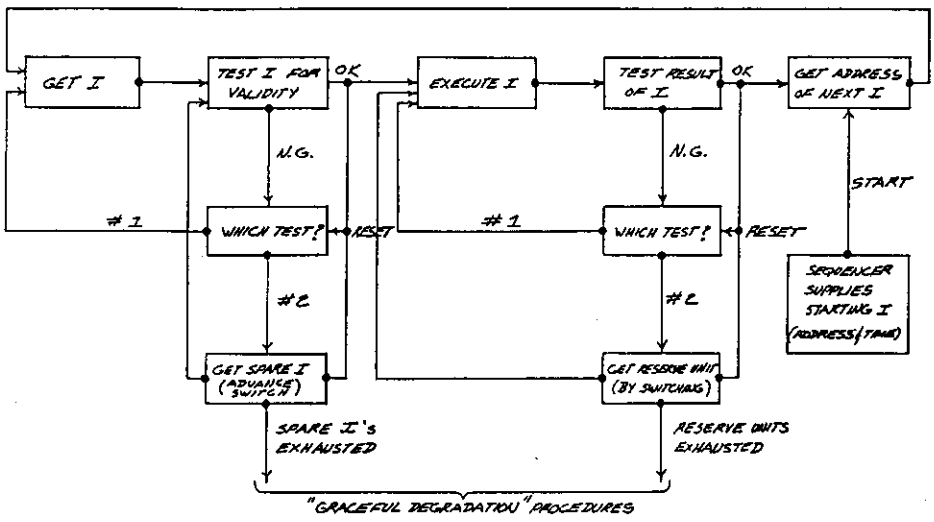

FIG. 2 DIAGNOSTIC CONTROL FLOW DIAGRAM FOR "STAR" SYSTEM

APPENDIX B

This is the program of a Workshop held on February 2-4, 1966 that served as a major stimulus for the subsequent formation of the IEEE Computer Group Technical Committee on Fault-Tolerant Computing in 1969.

WORKSHOP ON THE ORGANIZATION OF RELIABLE AUTOMATA

Terrace Room, Santa Ynez Inn, 17310 Sunset Blvd., Pacific Palisades, California

Sponsored by:

The Switching Circuit Theory and Logical Design Committee, IEEE Computer Group

Department of Engineering, University of California, Los Angeles

Engineering Extension, University of California Extension, Los Angeles

Wednesday, February 2, 1966

10:00 am	Registration and Coffee Hour	
12:00 noon	Opening Luncheon	
	Speaker: Professor C. M. Duke Chairman, Department of Engineering University of California, Los Angeles	
1:30 pm	*Introduction:*	
	"On the Problem of Reliable Automata"	A. Avižienis
	"A Survey of Soviet Activities in Reliability"	R. A. Short and W. H. Kautz
2:30 pm	*Redundancy Theory and Techniques*	S. Winograd, Chairman
	"Stability of Threshold Element Nets Subject to Common Shifts of Threshold" *(Paper was not presented - author did not attend)*	A. M. Andrew
	"The Need and Means for Fault Detection in Redundant Systems"	J. B. Angell
	"Reliability Estimation for Redundant Systems"	C. G. Masters
	"Placement of Voters in Modularly Redundant Digital Systems"	D. Rubin
	"Reliability, Redundancy, Capacity and Universality in Polyfunctional Nets"	R. H. Urbano
6:00 pm	Workshop Dinner	
	Keynote Speaker: Professor E. J. McCluskey, Jr. Chairman, Switching Circuit Theory and Logical Design Committee, IEEE Computer Group	

7:30 pm	*Theory of Diagnosis*	R. A. Short, Chairman
	"Methods for Finding Fault Detection and Diagnostic Tests"	D. B. Armstrong
	"Minimization of the Number of Fault Detection Tests"	H. Y. Chang
	"Evaluation of Computer Self-Test Process by Software Simulation"	J. W. Hirsch
	"Fault Diagnosis in Combinational Networks"	W. H. Kautz
	"Algorithms for the Diagnosis of Automaton Failures"	J. P. Roth

Thursday, February 3, 1966

8:00 am	Breakfast	
9:00 am	*Application of Coding and Automata Theory*	W. W. Peterson, Chairman
	"Reliability of Sequential Machines"	J. A. Brzozowski
	"Error Codes for Arithmetic Operations"	H. L. Garner
	"Memory Failures in Automata"	J. F. Meyer
	"Coded Redundancy in Logic Nets"	C. L. Sheng
	"I: On Active Self-Correcting Systems; II: On Error Correction in Memory Systems"	C. V. Srinivasan
	"Functional Coding in Redundancy Techniques"	S. Winograd
12:30 pm	Luncheon	
2:00 pm	*Redundant and Self-Diagnosing Systems*	R. E. Forbes, Chairman
	"A Diagnosable Arithmetic Processor"	A. Avižienis
	"Study of Aerospace Computer Concepts"	M. Ball and F. H. Hardie
	"A Self-Diagnosable Computer"	R. E. Forbes
	"On Self-Diagnosis of Large, Multi-Processor Computers"	E. Manning
	"An Algorithmic Approach to Self-Diagnosis"	R. A. Marlett
	"Self-Checking Microprograms"	R. W. Heckelman
5:30 pm	Social Hour	
6:30 pm	Dinner	
8:00 pm	Ad Hoc Discussion Groups	

Friday, February 4, 1966

8:00 am	Breakfast	
9:00 am	*Multiprocessor and Replacement Systems*	R. E. Miller, Chairman
	"Improving Reliability by the Practical Application of Selected Redundant Techniques"	W. A. England
	"Network Schemes for Combined Fault Masking and Replacement"	J. Goldberg
	"Some Aspects of Self-Repairing Automata"	E. C. Joseph
	"Evaluation of Logical and Organizational Methods for Improving the Reliability and Availability of a Computer"	D. E. Muller
	"Some Techniques in Designing Computer Subsystems for Automated Maintenance and Reliability"	C. V. Ramamoorthy
	"On a Study of Self-Repairing Digital Computers"	I. Terris
12:00 noon	*Closing Remarks:*	
	"A Review and an Extrapolation"	R. E. Miller
12:30 pm	Adjournment	

Workshop Participants

Dr. R. Alonso, M.I.T. Instrumentation Laboratory, Cambridge, Massachusetts

Prof. J. B. Angell, Stanford University, Stanford, California

Dr. D. B. Armstrong, Bell Telephone Laboratories, Inc., Murray Hill, New Jersey

Mr. M. Ball, IBM Space Guidance Center, Owego, New York

Prof. J. A. Brzozowski, University of California, Berkeley, California

Dr. H. Y. Chang, Bell Telephone Laboratories, Inc., Holmdel, New Jersey

Mr. C. Disparte, Hughes Aircraft Company, Culver City, California

Mr. W. A. England, Honeywell, Inc., St. Petersburg, Florida

Prof. G. Estrin, University of California, Los Angeles, California

Mr. R. E. Forbes, IBM Space Guidance Center, Owego, New York

Prof. H. L. Garner, University of Michigan, Ann Arbor, Michigan

Mr. J. Goldberg, Stanford Research Institute, Menlo Park, California

Mr. F. H. Hardie, IBM Space Guidance Center, Owego, New York

Mr. R. W. Heckelman, G. E. Electronics Laboratory, Syracuse, New York

Mr. J. W. Hirsch, Autonetics, Anaheim, California

Dr. E. C. Joseph, Univac, St. Paul, Minnesota

Prof. T. Kasami, Osaka University, Osaka, Japan

Dr. W. H. Kautz, Stanford Research Institute, Menlo Park, California

Mr. C. M. Klingman, BELCOM, Inc., Washington, D.C.

Mr. L. S. Levy, Aerospace Corporation, Los Angeles, California

Prof. E. J. McCluskey, Jr., Princeton University, Princeton, New Jersey

Prof. E. Manning, Massachusetts Institute of Technology, Cambridge, Massachusetts

Mr. R. A. Marlett, University of Illinois, Urbana, Illinois

Mr. C. G. Masters, Jr., Westinghouse Electric Corporation, Baltimore, Maryland

Prof. G. Metze, University of Illinois, Urbana, Illinois

Mr. J. F. Meyer, University of Michigan, Ann Arbor, Michigan

Prof. H. Mine. Kyoto University, Kyoto, Japan

Prof. D. E. Muller, University of Illinois, Urbana, Illinois

Prof. W. W. Peterson, University of Hawaii, Honolulu, Hawaii

Dr. C. V. Ramamoorthy, Honeywell, Inc., Waltham, Massachusetts

Dr. J. P. Roth, IBM Research Center, Yorktown Heights, New York

Mr. D. K. Rubin, Jet Propulsion Laboratory, Pasadena, California

Prof. C. L. Sheng, University of Ottawa, Ottawa, Canada

Dr. C. V. Srinivasan, RCA Laboratories, Princeton, New Jersey

Prof. A. Svoboda, University of California, Los Angeles, California

Dr. I. Terris, Hughes Aircraft Company, Culver City, California

Dr. R. H. Urbano, AF Cambridge Research Laboratories, Bedford, Massachusetts

Mr. J. J. Wedel, Jr., Jet Propulsion Laboratory, Pasadena, California

Dr. S. Winograd, IBM Research Center, Yorktown Heights, New York

Mr. H. S. Zieper, RCA Defense Electronic Products, Camden, New Jersey

Workshop Committee

Prof. A. Avižienis, UCLA, Los Angeles, and JPL, Pasadena, California, Chairman

Dr. R. E. Miller, IBM Research Center, Yorktown Heights, New York

Dr. R. A. Short, Stanford Research Institute, Menlo Park, California

APPENDIX C

Design of fault-tolerant computers

by ALGIRDAS AVIŽIENIS
University of California
Los Angeles, California, and
Jet Propulsion Laboratory
Pasadena, California

Causes and symptoms of logic faults in digital systems

Reliable performance of hardware has been a requirement for digital systems since the construction of the first digital computer. Improper functioning of the logic circuits in a digital system is manifested by *logic faults,* which are defined for this paper as "permanent or transient deviations of logic variables from the values specified in design."

Permanent faults are caused by physical changes in the components of a logic circuit which permanently alter the logic function specified by the designer. The most common permanent faults are the *determinate* faults of "stuck on zero" and "stuck on one" types. Less frequent is the *indeterminate* or "stuck on X" fault, in which the logic variable assumes both "zero" and "one" values improperly during a sequence of operations. Faults also differ in their extent: a *local* fault affects only one logic circuit, while a *distributed* (or catastrophic) fault occurs when one failure creates faults in several logic circuits of the same system.

Transient faults are caused by temporary changes in the properties of logic circuits, which lead to deviations from the specified values of logic variables. Such transient faults also belong to one of the above listed categories. They are caused either by external influences (electromagnetic interference, noise in power supply, etc.) or by temporary circuit malfunctions (overheating, overload conditions, etc.). External causes may simultaneously induce many faults throughout the system, therefore independent occurrence cannot be assumed for all transient faults.

At the system level a logic fault is manifested as an *error* in the program being executed by the system. Two types of errors may be distinguished. A *word error* occurs when a computer word (data word or instruction) is altered by a fault. A description of the alteration is called the *damage pattern* of the fault. A *logic error* occurs when an individual logic variable, which is not a part of a structured word, is altered by a fault. Examples of such variables are the various control signals in a computer. A logic error alters the algorithm being executed in some undersirable manner.

The present paper is concerned with the introduction of fault-tolerance in order to increase the reliability and availability of a digital system. We say that a system is fault-tolerant if its programs can be properly executed despite the occurrence of logic faults. All above discussed types of faults and errors need to be considered in the design of a fault-tolerant computer. Theoretical studies of fault-tolerance need a clear identification of the types of faults and/or errors which are to be tolerated.

Protective redundancy for fault-tolerance

Reliable performance of digital systems is usually attained by the systematic application of two techniques. The first is the selection of highly reliable components and the use of proven methods for their interconnection and packaging. The second technique is an extensive verification of the logic design, of the programs, and of the finished hardware, first by simulation and later by diagnostic and functional tests under expected environmental conditions. In spite of these reliability assurance techniques, the system may still fail during use because of uncontrollable or undetected faults. These are caused by undetected design errors, random failures of components or connections, and externally induced malfunctions during the operation of the system.

The effects of logic faults can be eliminated by the introduction of *protective redundancy* into the system. A computer system contains protective redundancy if faults can be tolerated because of the use of additional components or programs, or the use of more time for the computational tasks. These additional components, programs, and time are not required by the system in order to execute the specified tasks as long as faults

do not occur. The techniques of protective redundancy may be divided into two major categories: *massive* (also called *masking*) redundancy and *selective* redundancy.

In the massive (masking) redundancy approach the effect of a faulty component, circuit, signal, subsystem, or system is masked instantaneously by permanently connected and concurrently operating replicas of the faulty element. The level at which replication occurs ranges from individual circuit components to entire self-contained systems. Theoretical studies of massive redundancy were initiated in January 1952 by John von Neumann in a series of five lectures at the California Institute of Technology.[1] Other pioneering contributions in this field are due to C. J. Creveling,[2] and to E. F. Moore and C. E. Shannon.[3] The subject has attracted considerable attention in the past decade, and four principal techniques of massive redundancy may be distinguished among the published studies. These techniques are:

1. Replication of circuit components: e.g., "quadded" diodes, resistors, transistors, duplicated connections, etc.[2,4]
2. Replication or coding of logic signals: use of multiple channels and voting elements,[1,5,6] recursive nets,[3,7] interwoven logic,[8,9] variation-tolerant coded threshold element nets.[10]
3. Adaptive logic elements, e.g., voters with variable-weight inputs.[9]
4. Replication of entire systems with comparison or voting at system level.[11,12,13]

The category of selective redundancy encompasses redundancy techniques which fall outside the definition of massive redundancy. Since instantaneous masking is excluded in the selective technique, it is necessary to detect the presence of a fault. Subsequently, the fault is made harmless by a corrective action. The techniques of fault detection fall into two major categories:

1. *Concurrent diagnosis* by the application of error-detecting codes and special monitoring circuits. Detection occurs while the system is being used.[14,15,16,17,18]
2. *Periodic diagnosis* using diagnostic hardware and/or programs. Use of the system is interrupted for diagnosis.[19,20,21,22,23]

A variety of approaches exists in the implementation of both methods; furthermore, combinations of both techniques have been successfully employed.[24]

A *corrective action* which eliminates effects of the fault must follow the detection. The four principal techniques of correction are:

1. Correction of errors by the use of error-correcting codes and associated special purpose hardware and/or software (including recomputation).[15,16,24]
2. Replacement of the faulty element or system by a stand-by spare.[25,26,27]
3. Replacement as above, with subsequent maintenance of the replaced part and its return to the stand-by state.[28,29,30]
4. Reorganization of the system into a different fault-free configuration which can continue the specified task.[31,32,33]

It must be emphasized that the preceding classification of protective redundancy techniques is intended to facilitate a systematic approach to the study of fault-tolerant systems. In most practical cases a mixture of these techniques has been proposed or used in order to attain fault-tolerance. The references which are cited in this section contain fundamental contributions to the theory of protective redundancy, and have been chosen to serve as illustrations of the various approaches. However, this is not an exhaustive listing of all relevant contributions, and the reader is referred to bibliographies in the cited references for further papers. Especially the book by W. H. Pierce[9] and the recent study by J. Goldberg, et al.,[33] contain very extensive bibliographies on protective redundancy. The latter study includes numerous references to Russian publications in this field as well as reviews of various techniques.

State of the art in the design of fault-tolerant computers

The continuing increases in the speed and complexity of digital systems accelerate the demand for fault-tolerance. The cost of uncorrected errors is especially severe in large time-shared computer service systems and in situations in which a computer controls a very valuable system, and is not accessible to human repair. Examples are a real-time control computer and a spacecraft computer controlling an interplanetary mission. A second critical requirement for fault-tolerance exists when human lives may be affected by computer errors, e.g., in military defense systems and in control of high-speed transportation or of medical systems.

The most immediate solution in such critical applications has been the replication of entire systems,[11,12,13] frequently backed up by transfer of control to a human operator or to a separate, less precise backup system. The replication at system level becomes extremely costly when very large and fast systems (e.g., time-shared "computer utility" systems) must be replicated. Furthermore, occurrence of independent faults in two or more replicas is more probable as the systems become more complex or as the required unattended lifetime is increased. The need for lower cost of protective redundancy and for longer mean life values of protected

systems has stimulated studies of other methods of fault-tolerance.

In the early stages of development attention had been directed toward massive redundancy at the lowest level —the replication of individual components (resistors, transistors, etc.).[3,3,4] The principal example of its practical application is the primary processor of the Orbiting Astronomical Observatory.[4] The use of component redundancy has been limited by design difficulties and by new developments in component technology. The design of logic circuits becomes very difficult because the circuits must function correctly under wide variations of component values, caused by shorting or opening of individual components. The change from discrete components to integrated circuits has largely invalidated the assumption of independent component failures. Without it, the advantages of component redundancy are lost.

Considerable effort has been continuously directed toward practical use of massive triple modular redundancy (TMR) in which logic signals are handled in three identical channels and faults are masked by vote-taking elements distributed throughout the system.[5,6,9] Studies have considered optimization of voter placement and analyzed the gain in reliability or mean life of a TMR system.[34,35,36,37] In the most important application to this date, the guidance computer for the Saturn V spacecraft launch vehicle has been designed employing TMR techniques in its arithmetic and control sections.[6,38] This large-scale application of TMR may be expected to provide a practical assessment of its effectiveness.

TMR and related types of massive redundancy at the level of logic signals offer both obvious advantages and some serious drawbacks in a general comparison to selective redundancy. The principal advantages of massive redundancy are:

1. The corrective action is immediate and "wired-in," while it is delayed and may require switching in selective redundancy.
2. During operation there is no need for fault detection, which is essential in selective redundancy.
3. All parts of the system are equally protected; unprotected "hard core" elements may exist only at interfaces with other systems. In most selective redundancy schemes a "hard core" exists in the system.
4. The conversion of a non-redundant design to a massively redundant one is relatively straightforward, while more novel design techniques are demanded by the introduction of selective redundancy.

Compared to massive redundancy, the selective form requires several additional features: a system ability to tolerate interruptions for repair and to execute a "roll-back" for error correction, sophisticated diagnosis methods, protection for the "hard core," and trade-off studies between time, program, and hardware replication. The advantages of selective redundancy over the massive form are, however, also very significant in most applications:

1. Power is required by only one copy of each replaceable item in a replacement system; all parts require power in the massive form.
2. The replacement switch provides fault isolation between subsystems; such isolation is essential in the case of catastrophic failures. Massive redundancy usually assumes independent failures of logic elements; such independence requires isolation which is difficult to provide for batch-fabricated integrated circuit packages. The entire batch may possess the same defect; also, mechanical or thermal damage is likely to affect an entire package, rather than single logic circuits.
3. All spares can be utilized in selective redundancy; in the massive form a majority of faulty elements in a given region leads to system failure.
4. The designs of individual replaceable blocks may be altered, and the number of spares may be adjusted to a given requirement without changes in the system design in the case of selective redundancy; such changes are more difficult in the massive case.
5. The replication in massive redundancy frequently leads to increased fan-out and fan-in requirements for logic elements, or to increased tolerance limits in circuit design; such problems are avoided in the selective case.
6. Permanent connection of the redundant elements makes the initial check-out more difficult to implement in systems with massive redundancy; special circuits and system outputs are necessary.
7. Massively redundant systems with voting require synchronization of the separate channels at the voting elements; they also are susceptible to transient external influences (e.g., sparks) which alter logic signals in a majority of channels without leaving permanent damage. The delayed occurrence of diagnosis in the selective case allows detection of such transient changes in signals.

The most developed techniques of selective redundancy are fault detection by periodic diagnosis and the application of parity and similar error codes[6,14] to detect or correct errors in data transmission and storage. The periodic diagnosis techniques have progressed from exclusively software implementations to software combinations with special-purpose hardware[21,24] and to studies of system design methods which facilitate their self-diagnosis.[22,23] An extensive bibliography on periodic

diagnosis has been compiled by Breuer.[39] Except for parity checking, there have been relatively few studies of concurrent diagnosis in an entire computer including the processors,[17] or in a generalized logic net.[15,18]

Theoretical results on general models of replacement systems demonstrate gains in reliability and mean life for both unmaintained [25,26] and maintained[28,29,30] cases. More recently there have been proposals for designs of computers as replacement systems.[40,41] The following sections of this paper present a replacement system with concurrent diagnosis[27] which is presently being constructed. Reorganization of a system upon fault detection has also been discussed in recent publications;[31,32,33] however, much work remains to be done in order to arrive at complete system designs and to derive measures of expected effectiveness.[31,42]

In conclusion it is noted that the rapid development of integrated circuit technology is causing a shift of emphasis from massive to selective redundancy techniques in which functional units of a system are replaceable as single elements. It is also possible that massive redundancy will find a new application in large-scale integration, serving to mask manufacturing defects and thus increasing the yield of the manufacturing process.

Design considerations for a fault-tolerant spacecraft computer

Theoretical studies of selective redundancy, and specifically of replacement systems, indicate that a significant increase in the availability and in the mean life of a digital system may be attained without the high cost of complete replication and concurrent operation of several copies of a system. The challenge to the designer at the present time is to create computer systems which translate the theory into a working system which uses state-of-the-art components, meets current performance requirements, and attains the theoretically possible gains in reliability.

The choice of a method or of a combination of methods of redundancy for a particular computing system is influenced by the intended application. The present section considers the application of protective redundancy to a guidance and control computer for an unmanned interplanetary spacecraft. The computer may also be employed for the onboard processing of scientific data when guidance computation is no in progress. The guidance computer is required to survive space voyages to other planets which range up to several years in duration and to perform approach guidance and control computations at the end of the voyage. Continued control of the spacecraft after arrival, processing of scientific data collected, control of the landing and operation of a capsule, and guidance for a return voyage may also be required. Course corrections are to be computed one or more times during the mission; considerable time is available for this task. The computing at launch and in early stages of the mission may be performed or supported by computers on the ground and in the launch vehicle. The very long communication distances and possible occultation of the spacecraft make Earth-based support ineffective at the destination planet. The computations which are required at a remote destination present the most demanding problem to the spacecraft guidance and control computer.

The design of a spacecraft computer must be performed within the strict constraints of the available power, weight, and volume. The existence of these constraints indicates an advantage for selective redundancy, which does not necessarily require power for the spare replicas and which offers protection with the minimum of one spare for each operating element. An evaluation of relative advantages of the massive and selective redundancy approaches has led to the choice of selective redundancy for fault-tolerance in a spacecraft guidance computer which is being developed at the Jet Propulsion Laboratory. It will be called the "JPL Self-Testing And -Repairing" (abbreviated JPL-STAR) computer in this paper. The performance requirements demand a certain computing capacity at the end of a long voyage, and there are no requirements for a higher computing capacity at an earlier time during the mission. Under these conditions, a fixed-configuration replacement system possessing the required capacity is preferred over a reorganizable or "degradable" system which has a minimal configuration of the same capacity. The replacement system is a simpler solution, since it avoids the programs, switches, and control hardware which perform the reconfiguration and resulting rescheduling of operation.

A replacement system provides to the user one standard configuration of functional subsystems which has the required computing capacity. The standard computer is supplemented by one or more spares of each subsystem. The spares are held in a standby condition and serve as replacements of operating units when permanent faults are discovered. The presence of spares imposes additional requirements on the selectively redundant system. In addition to the ordinary functions of a computer, the system must incorporate some means of fault detection, a recovery procedure for the case of transient faults, a replacement procedure and a switch for the case of permanent faults, and a checkout procedure for all spares before the mission. The standard configuration itself must be designed as an array of self-contained functional subsystems with clearly defined interfaces for replacement switching. The hardware or software which controls the recovery and/or replacement must be fault-tolerant as well.

Early in the design fundamental choices must be made between hardware and software implementations of the fault detection and recovery procedures. The current generation of aerospace computers almost exclusively uses software techniques, supplemented by hardware for parity checking of data storage and transfer. The continuing decrease in the size and in power requirements of integrated electronic circuits, as well as the vulnerability of software techniques in the case of memory failures, led to the choice of a hardware implementation of fault detection and recovery in the JPL-STAR computer. The experimental breadboard Model I JPL-STAR system is expected to provide valuable operational experience about such an extensive use of hardware techniques in a replacement system. An actual hardware design rather than simulation was chosen in order to explore the circuit aspects of switching, fault detection, isolation of faulty subsystems, and recovery from transient faults.

Fault detection in digital circuits is implemented either by periodic or by concurrent diagnosis. The currently most common approach is periodic diagnosis which utilizes a diagnostic program stored in the memory. Computation is periodically interrupted and the diagnostic program is executed. Detection of a fault initiates the replacement procedure; the program is "rolled back" to a point preceding the previous (successful) diagnosis period. Errors in computation which have been caused by transient faults remain undetected in this approach. The diagnosis program itself is vulnerable to faults in the memory system. The cost of diagnosis consists of the storage used for the diagnostic program, of the time consumed by its execution and of the time needed for repair and repeated execution of the program segment which was run after the last diagnosis. Such time costs are very severe in re-entry and landing programs for guidance and control which require real-time computing. The alternate diagnosis method is concurrent diagnosis in which error-detecting codes and monitoring circuits are employed to show the presence of faults. The execution of every instruction is checked immediately; instead of the stored diagnostic program, the cost is in hardware and consists of the logic circuits which perform the code checking algorithm and the other monitoring circuits. Errors due to transient faults are detectable, and the immediate detection of a fault permits a relatively short rollback of the program. For these reasons concurrent diagnosis has been chosen for fault detection in the JPL-STAR computer. The simplest and most costly error-detecting code (100% redundancy) is the complete duplication of program and data words. Errors are indicated by the disagreement of two words; further diagnosis is needed to pinpoint the faulty source. Parity and other more complex codes which detect errors in the transmission of digital data have a lower redundancy, but are not suitable for the checking of arithmetic operations. In order to apply a uniform code in the entire system, arithmetical error-detecting codes were selected as a means of concurrent diagnosis for the JPL-STAR system. An extensive theoretical investigation of the effectiveness, cost and applicability of arithmetic codes was conducted prior to the system design of the JPL-STAR computer.[16,17] The results showed the existence of a class of low-cost codes with sufficient effectiveness of error detection for concurrent diagnosis. The code-checking circuits are supplemented by monitoring circuits which verify the synchronization of operation for the various subsystems. Other circuits compare duplicated critical functions of the subsystems and measure important circuit parameters (e.g., read and write currents in memory units). The monitoring circuits are included in order to detect the faults which are not always indicated by the code checking algorithm.

Recovery and replacement procedures require both software and hardware contributions. Consistently with the choice of hardware for fault detection, the JPL-STAR computer employs hardware implementation to the furthest possible degree in these procedures as well. The most fundamental hardware consideration in a replacement system is the method of switching and the nature of the switch which implements the replacement operation. The reliability of the switch is a limiting factor in the estimates of reliability for the entire system. Furthermore, the switch must provide complete isolation in the case of catastrophic failures occurring in the part of a computer which is to be replaced. The principal alternatives in the choice of a switching method are *information switching* and *power switching*. A study of switching techniques[18] has led to the conclusion that the switching of power to replaceable units offers strong isolation against catastrophic failures and minimizes the number of switches requiring extreme reliability. Furthermore, the data transmission speed within the computer is not affected by the circuit properties of the switch. A magnetic power switch for the JPL-STAR computer which is an integral part of a replaceable unit has been designed and constructed. The switch is a part of the unit's power supply and is designed to fail asymmetrically—in an open mode.

The use of a power switch requires that all unpowered copies of a replaceable unit should be permanently attached to the data transmission busses of the system. As a consequence, an unpowered unit is required to produce only logic signals of value "zero" on all of its output lines. Furthermore, all input and output lines of every replaceable unit are isolated from the busses

in order to prevent shorting of a bus by a short inside the unit.

Organization of the JPL-STAR computer, model I

The preceding section has outlined the principal alternatives which were considered in the choice of fault-tolerance techniques for the JPL-STAR computer. An experimental breadboard Model I of the JPL-STAR computer has been designed and is presently being constructed. The main objectives of the Mod I STAR computer are to gain experience with the hardware aspects of a replacement system and to conduct experiments with fault detection and recovery procedures. The performance specifications of the Mod I STAR computer are similar to those of many present-generation aerospace computers; they have not been matched to any specific application. The fundamental choices in fault-tolerance techniques are as follows:

1. All machine words (data and instructions) are encoded in an error-detecting code.
2. The computer is subdivided into several replaceable functional units.
3. Fault detection, recovery, and replacement are carried out by special-purpose hardware; software techniques may be added later to provide additional fault-tolerance features.
4. Replacement is implemented by power switching: units are removed by turning power off, and connected by turning power on.
5. The information lines of all units are permanently connected to the busses through isolating circuits; unpowered units produce only logic "zero" outputs.
6. The error-detecting code is supplemented by monitoring circuits which serve to verify the proper synchronization and internal operation of the functional units.

The Model I employs a 32-bit word length for its operands and instructions. Machine words are transmitted between the functional units in four-bit bytes, that is, in a series-parallel mode. The functional units contain their own sequence generators and possess identical input and output connections. A typical functional unit is shown in Figure 1. The "Info. Input" and "Info. Output" lines are connected to information busses. They receive and send coded machine words, one byte at a time. The "Switch Control" line supplies the "change position" command to the power switch, while the present switch position is shown by the "Switch Status" line. The other control input lines supply a "Clock" pulse train input, a synchronization test signal ("Sync"), and a "Reset" signal which places the functional unit into a standard state. There are also three more status output lines. The "Active" signal

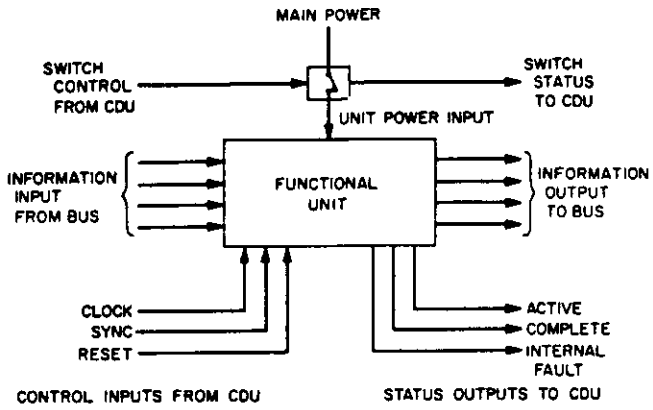

Figure 1—Typical JPL-STAR computer functional unit

indicates that at least one Info. Output line has an active (logic "one") output. The "Complete" signal occurs at the end of every subalgorithm being performed by the unit. The "Internal Fault" signal occurs when the internal monitoring circuits of the unit detect an abnormal condition. All status outputs are connected to the Control and Diagnosis Unit (CDU). The CDU also generates the four control input signals. The CDU initiates all recovery and replacement actions on the basis of the status signals reecived from the powered functional units in the system.

The block diagram of the JPL-STAR Model I computer is shown in Figure 2. It is a fixed-point, binary computer suitable for spacecraft guidance applications. Information words are transmitted on two busses in bytes of four bits each. The choice of the byte mode reduces the size of busses and simplifies the checkers, which are diagnostic hardware for error detection in the transmitted information words. An expansion to parallel operation is straightforward and will increase the computing speed at the cost of larger busses and more complex checkers. The replaceable functional units of Model I are:

1. a main arithmetic processor (MAP);
2. a control arithmetic processor (CAP);
3. a 16K read-only memory unit (ROM);
4. up to 12 read-write memory units, 4K each (RWM);
5. an input/output (buffer) unit (IOU);
6. a logic processor (LOP);
7. an interrupt unit (IRU);
8. a system clock unit (SCU);
9. two bus checkers (CH1, CH2);
10. a control and diagnosis unit (CDU).

Properties of these functional units are summarized in the next section.

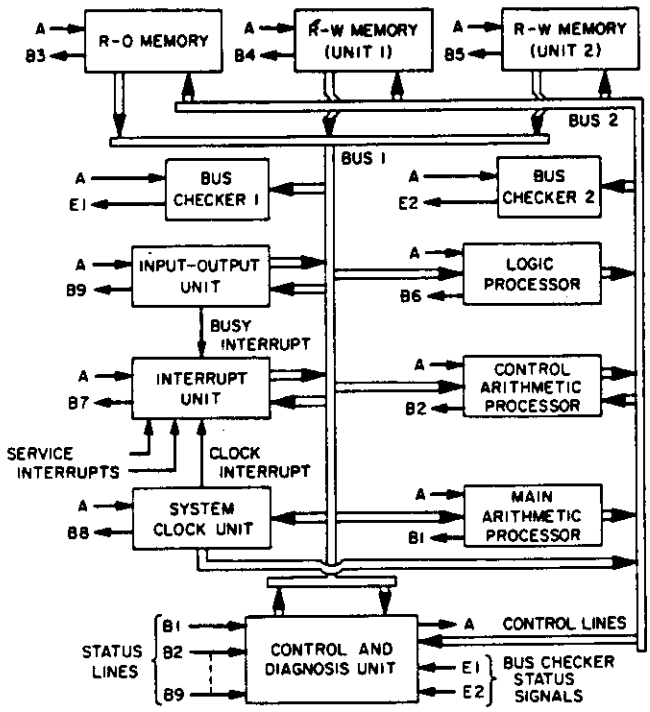

Figure 2—JPL-STAR model I computer block diagram

All information words in the JPL-STAR computer are encoded in an error-detecting code. In the case of numerical data words and addresses of instructions the code must be preserved during arithmetic operations. The two principal methods of arithmetic encoding are product (or "An") and residue codes.[16,17,44,45] In order to gain a better understanding of the relative virtues of these two methods, both are employed in the Model I: product coding for numeric operands, and residue coding for addresses. Figure 3 shows the formats of numeric operands and instructions.

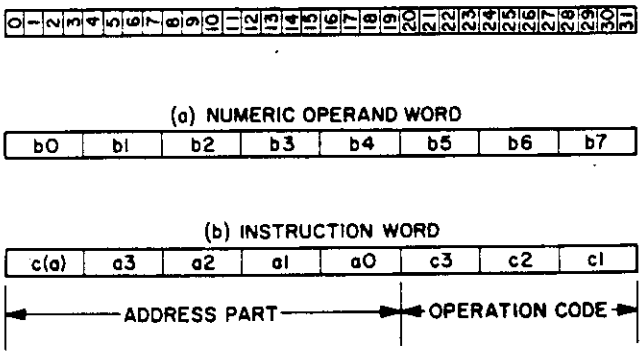

Figure 3—JPL-STAR model I computer word formats

The numeric operands (Figure 3a) are 32 bits long binary product-coded numbers with the check factor 15. Binary numeric operands x (28 bits long) are multiplied by 15 to obtain the product coded 32-bit operands 15x. The check factor 15 has been found to be especially effective in the case of series-parallel transmission and computing in bytes of 4 bits length.[16] The *checking algorithm* computes the modulo 15 residue of coded words which are transmitted on the busses. A zero residue (represented by 1111) indicates a coded word; all other values indicate a fault in the functional unit which delivered the word to the bus.

The 32-bit instruction words (Figure 3b) consist of a 12-bit operation code and a 20-bit address part. The address part is encoded in the residue code with the check modulus 15. An address part consists of a 16-bit binary address a and a 4-bit check symbol c(a). The check symbol c(a) has the value

$$c(a) = 15 - |a|_{15}$$

where $|a|_{15}$ is the modulo 15 residue of a. The checking algorithm computes the modulo 15 residue of an address and adds it (modulo 15) to the check symbol c(a).

A zero sum (represented by 1111) indicates a properly coded address part. The residue code is preferable for address parts over the product code because it is separable, and the address a is available to the memory address decoding circuits in its ordinary binary form. It is important to note that the "one's complement" $15 - |a|_{15}$ rather than the residue $|a|_{15}$ itself is used as the check symbol c(a). In this case the fault-detection effectiveness in byte-serial operation remains the same as for product-coded operands, while the use of $|a|_{15}$ as a check symbol gives a lower effectiveness. Furthermore, the bus checking algorithm is the same for product-coded operands and for address parts—it is simply a modulo 15 summation of all bytes and a test of the result for the zero value represented by 1111.

The *operation code* is divided into three bytes of four bits each. The operation code bytes are protected by a 2-out-of-4 encoding, which leaves six valid words in a four-bit byte. Such coding is most efficient for short words and is acceptable in a computer because operation codes are not subjected to arithmetic operations. It is evident that their validity must be tested by a separate checking circuit, since it cannot be verified by the modulo 15 checker (which is bypassed by the operation codes). The separation of the operation code into three separately encoded bytes facilitates the decoding and validity testing of op. codes received by the functional units. The 2-out-of-4 encoding gives a total of 216 distinct combinations for operation codes. The indication of index registers which are to be used

must be contained within the operation code. Since the Model I contains two index registers, every indexable operation code requires three distinct combinations, while non-indexable operations require only one each.

It is apparent that residue encoding with the check modulus 15 can be applied to the numeric operands and to the operation codes as well as to the address parts. Such use of a uniform residue code has the advantages of code separability and of identical check algorithms. In the case of operation codes, a modulo 15 residue-coded 12-bit number provides 256 distinct combinations. On the other hand, the 2-out-of-4 coding for individual bytes of the operation code permits validation and use of individual bytes. The choice of three different methods of encoding in the Model I was motivated by the need to gain detailed insight into their relative merits and shortcomings.

One instruction cycle is executed by the Model I JPL-STAR system in three steps. In the first step, the address of the instruction is sent from the Sequence Register in the Control Arithmetic Processor to the memory units; the transmission of the address is tested by the Bus Checker. In the second step, the addressed memory unit broadcasts the operation code and address to all functional units. The appropriate units recognize the code, accept the address, and initiate execution. In the third step (if needed) the instruction is executed, a result is placed on the bus and accepted by the destination unit. The Bus Checkers test every word on the busses for proper encoding.

Functional units of model I

The replaceable functional units of the computer have a standard format for their output words and have the same internal communication lines, as shown in Figure 1 and described previously. The Input/Output and Interrupt units also have external communication lines to the spacecraft. Brief descriptions of the functional units are given in this section.

The Main Arithmetic Processor (MAP) accepts the operands and delivers the results as 32-bit product-coded binary numbers. All arithmetic control is contained in the MAP; an input consists of an operation code (add, subtract, multiply, divide) followed by a coded operand, and the output is a coded result followed by a 2-out-of-4 Condition Code byte, indicating either one of three singularities (sum overflow, quotient overflow, zero divisor) or the type of a good result (positive, zero, negative). The good result codes are used by the CAP as data for conditional jump instructions. All partial and final results are delivered to the Bus Checker #2. A breadboard model of the MAP has been constructed and tested.[46] It is believed to be the first complete arithmetic processor for product-coded operands.

The Control Arithmetic Processor (CAP) contains two Index Registers (IR), the Sequence Register (SR), the Condition Code Register (CCR), and an adder. When the 16-bit index word x from the IR is added to an address a, its 4-bit check symbol c(x) is added modulo 15 to c(a). The indexed address and the new check symbol go past the Bus Checker #2 to the input lines of the memory units. The incrementing (by one) of the current address s in the SR is performed in exactly the same manner, with 1 added to s and 14 = c(1) added modulo 15 to c(s). The incremented address is returned to the SR. The presence of the Condition Code byte in the CAP permits fast execution of conditional jump instructions.

The Logic Processor (LOP) performs the bit-by-bit logic operations, shifts, and code conversions on input operand words. It contains the Logic Accumulator (LAR) and the Mask Register (MR). The arithmetic coding is removed from the operand before the operation, since arithmetic codes are not preserved during logic operations. The final result is again encoded; provisions exist for both product and residue codes. The LOP is therefore capable of encoding input words, removing code from output words, and executing conversions between product and residue codes. A four-bit adder is included for encoding and decoding operations. Functioning of the LOP is checked by operating two copies concurrently and comparing all results.

The Read-Only Memory (ROM) unit contains the permanent programs and the associated constants for a given mission. The experimental model provides 2^{14} words of 32 bits each, using a "braid" assembly of transformers and wires for the permanent storage of binary information.[47] The ROM also contains all necessary peripheral electronics: the op. code, address, and output registers, access circuits, drivers, sequence control, and monitoring circuits. All output words from the ROM are delivered past Bus Checker #1. The Model I JPL-STAR computer includes complete replicas of the ROM as replacements; the replacement of peripheral electronics without discarding the core and wire assembly is being studied.

The Read-Write Memory (RWM) units are self-contained 4096 word modules with the same peripheral electronics as the ROM. Direct addressing is provided for 64K words, including the 16K ROM; this permits up to 12 RWM units. Each RWM unit has three modes of operation. In the *standard* mode a RWM unit recognizes its own wired-in *unit address*, and is connected to both input and output busses. In the *auxiliary* mode, a RWM unit stores and assumes the unit address of another unit, to be called its *main* unit. The

auxiliary unit stores the same input words as its main unit. When the main unit reads out a word to the bus, the auxiliary unit reads out the same word internally and compares it to the word which is on the bus. If the words disagree, the auxiliary unit signals a comparison error to the CDU. A "reverse" diagnostic command causes a reversal of roles between an auxiliary unit and its main unit. The third mode of RWM operation is the *relocated* mode, which utilizes a stored unit address, but otherwise is identical to the standard mode. A RWM unit is placed into or released from the auxiliary or relocated mode by special instructions which are directed to its wired-in unit address. The auxiliary mode permits a redundant storage of machine words in one, two, or more separate units with continued comparison of readout supplementing the Bus Checker. The relocated mode permits a replacement of a failed RWM unit by a spare with the same unit address. The choice of employing redundant storage is left to the user.

The Interrupt Unit (IRU) and the Input-Output Unit (IOU) serve as interfaces with the external world. The IOU contains buffer registers for receiving and delivering machine words. The IRU receives commands from an outside operator and service requests from other parts of the spacecraft system. An interrupt is effected when the IRU places a properly coded instruction word on the bus, preempting the delivery of the next instruction specified by the Sequence Register. The details of interface protection remain to be established for specific spacecraft systems; in general, complete duplication will serve under most conditions.

The System Clock Unit (SCU) contains counters needed for the sequencing and timekeeping functions of the computer and the spacecraft. Counter settings are coded machine words. The SCU generates an internal interrupt request when a preset count has been reached.

The two Bus Checkers (CH1 and CH2) serve to check all machine words which are being transmitted on the busses for validity of encoding. The checking of arithmetic codes requires a four-bit Check Sum Accumulator (CSA) and a four-bit modulo 15 adder (with an end-around carry) which adds the bytes being transmitted to the word in the CSA. An error-status line to the CDU indicates whether the CSA contains an acceptable word (1111). The checking of the non-numeric 2-out-of-4 operation code bytes is carried out by a separate logic circuit which also has an error-status line to the CDU. The relatively small size of the Bus Checkers makes their duplication quite practical for fault-tolerance.

The Control-Diagnosis Unit (CDU) issues the control signals for timing and for replacement. It also initiates the recovery actions when a fault is indicated by status outputs of the powered functional units and bus checkers. In normal operation, the internal timing of the units is initiated for every instruction with the "Sync"signal and is tested by reception of the "Complete" signal. A copy of the current instruction is stored in the CIR register within the CDU and provides the data to verify all "Active" and "Complete" signals and the Bus Checker status signals. The CDU also contains the Rollback Point Register (RPR) which is loaded with an address under program control. When a fault is detected from the status signals, computing is "rolled back" to the instruction at this address and repeated in order to correct a transient fault. If a persistent fault is indicated by the same unit, a replacement is carried out, and the program is resumed at the rollback point. Software diagnosis may be employed to provide additional data on permanent faults. Periodic updating of the RPR is the responsibility of the programmer. For catastrophic transient faults (e.g., brief power loss) the CDU contains a wired-in "Restart" procedure.

The CDU acts as the "hard core" of the system and requires immediate fault-tolerance. The Model I STAR computer maintains four powered CDU copies. The CDU outputs are determined by a majority vote of three units; in the case of a disagreement, the minority unit is at once replaced by the operating fourth unit, and a new spare CDU is brought in and synchronized with the other powered units to act as an operating spare. Because of the four unit requirement, design effort has been concentrated on reducing the CDU to the least possible complexity. Experience with Model I is expected to yield further insights into this problem and to lead to modifications of the CDU design.

The performance of the JPL-STAR computer must be measured against an equivalent non-redundant computer. The important parameters are complexity, gain in mean life, and types of faults which can be tolerated. The first design objective has been the ability to tolerate a wide range of faults, including catastrophic (multiple dependent) transient and permanent faults in the functional units. The gain in mean life is being investigated by means of simulation (hardware and software) as well as by mathematical methods. The results of these studies will be reported upon completion of construction and hardware simulation.

ACKNOWLEDGMENT

The research described in this paper has been carried out at the Jet Propulsion Laboratory, Pasadena, California, under Contract NAS7-100, sponsored by the National Aeronautics and Space Administration. The author wishes to acknowledge the support and encouragement of W. F. Scott and discussions with J. J. Wedel and G. R. Hansen. System and logic design of

several functional units was performed by D. A. Rennels and A. D. Weeks, all of the Flight Computers and Sequencers Section, Guidance and Control Division, JPL.

REFERENCES

1 J VON NEUMANN
Probabilistic logics and the synthesis of reliable organisms from unreliable components
Automata Studies C E Shannon and J McCarthy eds Annals of Math Studies No 34 Princeton University Press 1956 p 43-98

2 C J CREVELING
Increasing the reliability of electronic equipment by the use of redundant circuits
Proceedings of the IRE 44 509-515 April 1956

3 E F MOORE C E SHANNON
Reliable circuits using less reliable relays
Journal of the Franklin Institute 262 Nos 9 and 10 191-208 and 281-297 Sept Oct 1956

4 T B LEWIS
Primary processor and data storage equipment for the orbiting astronomical observatory
IEEE Transactions on Electronic Computers EC-12 No 5 677-686 Dec 1963

5 W G BROWN J TIERNEY R WASSERMAN
Improvement of electronic computer reliability through the use of redundancy
IRE Transactions on Electronic Computers EC-10 No 3 407-416 Sept 1961

6 M M DICKINSON J B JACKSON G C RANDA
Saturn V launch vehicle digital computer and data adapter
AFIPS Conference Proceedings 26 501-516 Fall JCC 1964

7 R H URBANO
Some new results on the convergence oscillation and reliability of polyfunctional nets
IEEE Trans on Electronic Computers EC-14 No 6 769-781 Dec 1965

8 J G TRYON
Quadded logic
Redundancy Techniques for Computing Systems Spartan Press Inc Washington DC 1962 p 205-228

9 W H PIERCE
Failure-Tolerant Computer Design
Academic Press Inc New York 1965

10 S WINOGRAD J D COWAN
Reliable Computation in the Presence of Noise
The MIT Press Cambridge Mass 1963

11 J J DONEGAN C PACKARD P PASHBY
Experiences with the Goddard computing system during manned spaceflight missions
Proceedings of the 19th National Conference of the ACM p A2.1-1 to A2.1-8 1964

12 J E HAMLIN
A general description of the NASA real time computing complex
Proceedings of the 19th National Conference of the ACM p A2.2-1 to A2.2-22 1964

13 R T STEVENS
Norad's computers get all the facts
Electronics 40 No 4 p 113-118 Feb 20 1967

14 W W PETERSON
Error correcting codes
The MIT Press and John Wiley and Sons Inc New York 1961

15 W H KAUTZ
Codes and coding circuitry for automatic error correction within digital systems
Redundancy Techniques for Computing Systems Spartan Press Inc Washington DC 1962 p 152-195

16 A AVIZIENIS
A study of the effectiveness of fault-detecting codes for binary arithmetic
JPL Technical Report No 32-711 Jet Propulsion Laboratory Pasadena Calif Sept 1 1965

17 A AVIZIENIS
Concurrent diagnosis of arithmetic processors
Paper No 3.1 Conference Digest of the 1st Annual IEEE Computer Conference Chicago Illinois Sept 6-8 1967

18 D B ARMSTRONG
A general method of applying error correction to synchronous digital systems
Bell System Tech Journal 40 No 2 577-594 March 1961

19 S SESHU D N' FREEMAN
The diagnosis of asynchronous sequential switching systems
IRE Transactions on Electronic Computers EC-11 No 4 459-465 August 1962

20 J P ROTH
Diagnosis of automata failures: A calculus and a method
IBM Journal of Research and Development 10 No 4 278-291 July 1966

21 W C CARTER H C MONTGOMERY R J PREISS H J REINHEIMER
Design of serviceability features for the IBM System/360
IBM Journal of Research and Development 8 No 2 115-126 April 1964

22 R E FORBES D H RUTHERFORD C B STIEGLITZ L H TUNG
A self-diagnosable computer
AFIPS Conference Proceedings 27 Part 1 1073-1086 Fall JCC 1965

23 E MANNING
On computer self-diagnosis: Part I—Experimental study of a processor Part II—Generalizations and design principles
IEEE Trans on Electronic Computers EC-15 No 6 873-890 Dec 1966

24 R W DOWNING J S NOWAK L S TUOMENOKSA
No 1 ESS maintenance plan
The Bell System Technical Journal 43 No 5 Part 1 1961-2019 Sept 1964

25 B J FLEHINGER
Reliability improvement through redundancy at various system levels
IBM Journal of Research and Development 2 No2 148-158 April 1958

26 J E GRIESMER R E MILLER J P ROTH
The design of digital circuits to eliminate catastrophic failures
Redundancy Techniques for Computing Systems Spartan Press Inc Washington D C 1962 p328-348

27 A AVIZIENIS
Coding of information for a guidance computer with active redundancy
JPL Space Programs Summary No 37-22 Vol IV 9-12 Jet Propulsion Laboratory Pasadena Calif 1963

28 D E ROSENHEIM R S ASH
Increasing reliability by use of redundant machines
IRE Trans on Electronic Computers EC-8 125-130 June 1959

29 R TEOSTE
Design of a repairable redundant computer
IRE Trans on Electronic Computers EC-11 No 5 643-649 October 1962

30 J KRUUS
Upper bounds for the mean life of self-repairing systems
Coordinated Science Laboratory R-172 University of Illinois Urbana Illinois July 1963 AD-418 174

31 I TERRIS M A MELKANOFF
Investigation and simulation of a self-repairing digital computer
IEEE Conference Record on Switching Circuit Theory and Logical Design 264-272 1965

32 R L ALONSO A L HOPKINS JR
H A THALER
A multiprocessing structure
Paper No 6.2 Conference Digest of the 1st Annual IEEE Computer Conference Chicago Illinois Sept 6-8 1967

33 J GOLDBERG K N LEVITT R A SHORT
Techniques for the realization of ultra-reliable spaceborne computers
Final Report—Phase 1 Contract NAS12-33 Stanford Research Institute,Menlo Park Calif Sept 1966

34 R E LYONS W VANDERKULK
The use of triple-modular redundancy to improve computer reliability
IBM Journal of Research and Development 6 No 2 200-209 April 1962

35 H L ERGOTT D P ROSENBERG
Modular redundancy for spaceborne computers
Proceedings of IEEE Spaceborne Computer Engineering Conference 137-150 Anaheim California Oct 30-31 1962

36 K J GURZI
Estimates for best placement of voters in a triplicated logic network
Trans IEEE EC-14 No 5 711-716 October 1965

37 D K RUBIN
The approximate reliability of triply redundant majority-voted systems
Paper No 3.4 Conference Digest of the 1st Annual IEEE Computer Conference Chicago Illinois Sept 6-8 1967

38 J E ANDERSON F J MACRI
Multiple redundancy applications in a computer
Proc 1967 Annual Symposium on Reliability 553-562 Washington D C Jan 10-12 1967

39 M A BREUER
General survey of design automation of digital computers
Proc of the IEEE 54 1708-1721 Dec 1966

40 W G BOURICIUS W C CARTER J P ROTH
P R SCHNEIDER
Investigations in the design of an automatically repaired computer
Paper No 6.4 Conference Digest of the 1st Annual IEEE Computer Conference Chicago Illinois Sept 6-8 1967

41 P W AGNEW R E FORBES C B STIEGLITZ
An approach to selfrepairing computers
Paper No 6.3 Conference Digest of the 1st Annual IEEE Computer Conference Chicago Illinois Sept 6-8 1967

42 D EMULLER
Evaluation of logical and organizational methods for improving the reliability and availability of a computer
Paper No 6.1 Conference Digest of the 1st Annual IEEE Computer Conference Chicago Illinois Sept 6-8 1967

43 E K VAN de RIET D R BENNION J M YARBOROUGH
Feasibility study for a reliable magnetic connection switch
Final Report—Phase I Contract 951232 under NAS7-100 Stanford Research Institute Menlo Park Calif Feb 1966

44 D T BROWN
Error detecting and correcting codes for arithmetic operations
IRE Trans EC-9 333-337 1960

45 H L GARNER
Error codes for arithmetic operations
IEEE Trans EC-15 763-770 1966

46 A AVIZIENIS
The diagnosable arithmetic processor
JPL Space Programs Summary 37-37 IV Jet Propulsion Laboratory Pasadena Calif 1966 76-80

47 W H ALDRICH R L ALONSO
The 'braid' transformer memory
IEEE Trans EC-15 502-508 August 1966

Dr. Algirdas Avižienis: Technical Biography

Algirdas Avižienis was born in Kaunas, Lithuania in 1932 and has resided in the United States since 1950. He received the B.S., M.S., and Ph.D. degrees, all in electrical engineering, from the University of Illinois, Urbana-Champaign, in 1954, 1955, and 1960, respectively.

From 1956 to 1960 he was associated with the Digital Computer Laboratory at the University of Illinois as a Research Assistant and Fellow, participating in the design of the ILLIAC II computer and completing a Ph.D. thesis that developed the class of signed-digit number systems for fast digital arithmetic. In 1960 he joined the Spacecraft Computers section of the Jet Propulsion Laboratory (JPL), California Institute of Technology, and initiated research on reliability of computing systems that originated the concept of fault tolerance. He organized and directed the JPL STAR research project from 1961 to 1972. This effort resulted in the construction and evaluation of the experimental JPL STAR (Self-Testing-And-Repairing) computer, for which he received U.S. Patent No. 3, 517, 171, "Self-Testing and Repairing Computer," granted on June 23, 1970 and assigned to NASA. A paper that described the JPL STAR computer won the Best Paper selection of the *IEEE Transactions on Computers* in 1971.

Dr. Avižienis joined the faculty of the University of California, Los Angeles (UCLA) in 1962. Currently he is Professor and Director of the Dependable Computing and Fault-Tolerant Systems Laboratory in the Computer Science Department of UCLA, where since 1972 he has been Principal Investigator of continuing research projects on fault tolerant computing and system architectures, supported by about $3.5 million funding by NSF, ONR, NASA, FAA, and grants from the State of California and industry. Dr. Avižienis served as Chairman of the UCLA Computer Science Department from 1982 to 1985. He teaches courses in computer system architecture, computer arithmetic, fault-tolerant systems, and software fault tolerance. He has supervised 21 Ph.D. dissertations, 30 M.S. theses, and is the author or coauthor of over 120 publications in these fields of study. His present research at UCLA focuses on software fault tolerance, fault-tolerant distributed system architectures, and design methodology for fault-tolerant systems.

Dr. Avižienis has served as a consultant for studies of computer systems design and fault tolerance sponsored by the U.S. Air Force, U.S. Navy, NASA, The Federal Aviation Administration (FAA), and the National Bureau of Standards, as well as for industrial research in the U.S. and abroad. He has also served on several study groups and panels, including a three-year term as a member of the Advisory Panel on Computer Science and Engineering for the National Science Foundation (NSF), membership on the Hardware Systems Committee of the 1975-78 NSF Computer Science and Engineering Research Study, service as chairman of the Fault Tolerance Panel for the 1980 AFOSR and USAF Space Division Summer Study on "Autonomous Spacecraft Maintenance", and as the expert on fault tolerance for the U. S. Defense Science Board 1981 Summer Study on "Technology Base for the 1990's". In 1985, he was elected to serve a three-year term on the Computing Research Board, and since 1984 he serves as fault tolerance expert on the Advanced Automation System Technical Advisory Group of the FAA.

For his pioneering efforts in the field of fault-tolerant computing, Dr. Avižienis has received numerous honors and awards, among them the Honor Roll of the IEEE Computer Society in 1968; the NASA Apollo Achievement Award in 1969; election to Fellow of IEEE "for fundamental contributions in fault-tolerant computing" in 1973; the biannual AIAA Information System Award in 1979 with the citation, "for the original conception and development of fault-tolerant computing and leadership in the theoretical and practical application of highly reliable aerospace computers"; the NASA Exceptional Service Medal in 1980 "in recognition of outstanding achievements and widely recognized leadership in the field of fault-tolerant computing"; and the second annual IEEE Computer Society Technical Achievement Award "for sustained contributions to fault-tolerant computing" in 1985. In recognition of his scientific

accomplishments, Dr. Avižienis was awarded the honorary degree "Docteur Honoris Causa" by the Institut National Polytechnique of Toulouse, France in November, 1985. He was advanced to the rank of Faculty of Highest Distinction at UCLA in 1986, which is the top professorial rank in the University of California system.

As a member of the IEEE Computer Society, Dr. Avižienis founded and was the first Chairman of the Technical Committee on Fault-Tolerant Computing (1969-1973), and was the organizer and General Chairman of the First International Symposium on Fault-Tolerant Computing in 1971. He also served for four years (1971-1974) as a member of the Governing Board of the IEEE Computer Society. In international activities, he has served as the founding Chairman of the Working Group 10.4 on "Reliable Computing and Fault Tolerance" of IFIP (the International Federation for Information Processing) from 1980 to 1986. The General Assembly of IFIP presented its Silver Core award to Dr. Avižienis in 1986.

Dr. Avižienis has lectured and conducted joint research at the National Polytechnic Institute of Mexico, the University of Sao Paulo, Brazil, the Laboratoire d'Automatique et d'Analyse des Systemes (LAAS) in Toulouse, France, Keio University in Tokyo, Japan, the Innovative Computer Systems Center of the Technical University of Berlin, FRG, and the Microelectronics Research Institute in Lintong, Peoples' Republic of China. In 1974 he spent a five-month research visit, sponsored by the U.S. National Academy of Science, at the Institute of Mathematics and Cybernetics, Lithuanian Academy of Sciences, Vilnius, Lithuania, USSR.

Dr. David A. Rennels: Technical Biography

David A. Rennels was born in Terre Haute, Indiana in 1942. He received the BSEE degree from Rose Hulman Institute of Technology, Terre Haute, Indiana, in 1964, the MSEE from Caltech in 1965, and the PhD in Computer Science from UCLA in 1973.

In 1966 he joined the Spacecraft Computers Section of the Jet Propulsion Laboratory (JPL) of the California Institute of Technology and participated in the design and experimental validation of the JPL-STAR (Self-Testing and Repairing) Computer. In 1976 he initiated the Fault-Tolerant Building Block Computer project at JPL. This program developed an architecture which uses a small number of standard LSI-implementable building block circuits along with existing microprocessors and RAMS to construct fault-tolerant distributed computer systems for spacecraft. The architecture has been implemented as a breadboard, and it was used as the control computer for an autonomous satellite design conducted by JPL for the U.S. Air Force.

In 1978 Dr. Rennels joined the faculty of the University of California, Los Angeles (UCLA), where he is currently an associate professor. At UCLA he has been a co-principal investigator in the "Distributed Processing" research project sponsored by the Office of Naval Research, and was principal investigator of a program titled "Fault-Tolerant Local Networks using VLSI Building Blocks", sponsored by the National Science Foundation. He is currently carrying out research sponsored by Hughes Aircraft, TRW, and Rockwell International. He has served as a consultant to a number of private companies in the areas of fault-tolerant computer architecture. He remains a part-time academic member of the technical staff at JPL.

Dr. Rennels was the general chairman of the 12th International Symposium on Fault-Tolerant Computing in 1982, and is a member of the IFIP Working Group 10.4 on "Reliable Computing and Fault-Tolerance."