Fault-Tolerant Computing

Wing N. Toy, AT&T Bell Laboratories
Michele Morganti, Central R&D Laboratories

Fault tolerance and fault-tolerant computing—or, more generally, the ability to produce correct results even in the presence of faults, errors, and other anomalous or unexpected conditions—is in no way a new discipline. On the contrary, it evolved simultaneously with the computer, and has steadily evolved to support the increasing complexity of computing techniques and tools. For example, error-detecting techniques such as "double checking," "check-summing," and "check-pointing" existed well before the first electronic computer was made, and, with minor adaptations, still exist in all current, ultrareliable computers.

With the advent of electronic computers, however, came the increased interest in fault-tolerant computing that we know today; at that time, FT was at the mercy of the explosive forces of the hardware and software revolutions.

We can appreciate the results of those revolutions when we compare FT "then and now." The first generation of electronic computers was characterized by a relatively low architectural complexity (if measured in number of gates or gate connectivity), small program size, high hardware costs, and extremely high unreliability. FT research focused on the development of sophisticated space- and time-redundancy techniques that would enable a system to operate economically and successfully in spite of frequent hardware malfunctions. At the same time, however, a more important and dangerous assumption was implicitly being made: The design process had to be free of faults and, therefore, a fault-*in*tolerant approach had to be taken against design errors in both hardware and software.

A few years later the scenario changed completely: hardware reliability improved dramatically with the advent of semiconductor technology. Meanwhile, software complexity—and with it, software failures—increased at an unexpectedly high rate as a consequence of the low cost of software compared with hardware. Hence, FT was *de facto* relegated to critical applications only—such as space, telecommunications, and nuclear energy—while most of the interest (and relevant investments) focused on development methodologies, high-level languages, and validation and verification techniques that assumed the necessity of fault-free design. Eventually, the approach proved inadequate and models appeared that allowed for an "acceptable" rate of software-originated failures and compensated for software design errors as well as hardware component failures.

Today, the FTC evolution is still very active, propelled by two major forces: VLSI technology and the rapid expansion of office automation. In particular, VLSI technology has made hardware so inexpensive and reliable that a complex system, and the fault-effected design problems that go with it, can now be created on silicon chips. However, correction releases and on-line patching techniques, which are used in software for fixing errors and "bugs," are neither simply nor economically transferrable to hardware. The transfer problem justifies the increased and definite interest in the very few fault-tolerant techniques (such as *n*-version design) that attempt to obtain correct results and successful operation from systems that are known to be imperfect.

At the same time, the spread of office automation systems and techniques has dramatically extended the domain of *computer-dependent work*—that is, that portion of everyday work that depends completely on the availability of computer systems. A survey of ongoing research and market trends seems to indicate that two distinct approaches have been taken to deal with this problem. The first is to introduce traditional FTC methodologies (characterized by high redundancy levels), in the commercial computer market (the appearance of AT&T's 3B line is a clear indication of this trend). The second is to fully exploit the high level of distribution permitted by personal computing to guarantee high service availability in spite of multiple workstation failures. Although the second approach appears more promising, both are usually combined in present solutions, since the effective distribution of large databases is still an open and controversial problem.

In this issue, with the help of our contributors, we document this analysis, illustrate for our readers the state of FT today, and demonstrate how we expect its importance to grow in the future.

In the first article, Daniel P. Siewiorek at Carnegie-Mellon University provides an overview of fault-tolerance techniques. Touching on the differences between fault-avoidance and fault-tolerance approaches to system design, Siewiorek names major redundancy techniques currently used by system designers to increase reliability, availability, and serviceability. He then analyzes the benefits of a few of these techniques.

The evolution of FT techniques in accordance with changing system architectures is an interesting subject which we have merely touched upon in our introduction. Omri Serlin of ITOM International provides a more thorough look at the development of present-day FT design methods, citing several systems and discussing the design approaches used in developing them. He concludes that work on FT hardware is already well advanced and software now demands attention.

As hardware systems have become more reliable, overall systems designs, particularly software systems, have become more complex. Consequently, downtime due to hardware faults and failures has decreased, while for software faults and failures, it has increased. John J. Wallace and Walter W. Barnes of AT&T Bell Laboratories explore the possibilities of developing failure-free systems through the approach of "design for ultra-availability." In their discussion of AT&T's 3B20 Duplex computer running on the Unix RTR operating system, Wallace and Barnes stress the importance of software systems that implement intelligent fault recovery through central configuration management. This system offers ultrahigh availability through the accurate diagnosis of failures and the effective reconfiguration of data paths—at the expense of some increase in development and testing costs.

Dave Johnson of Intel, on the other hand, describes the fault-tolerant VLSI architecture of Intel's 432. His aim is to illustrate the possibilities of generic FT chips. Although multipurpose chips are yet to be developed, Intel can now offer users of its 432 systems three levels of fault tolerance, all based on self-checking and redundancy. Johnson shows how basic Intel FT design configurations can be upgraded as system downtime becomes more critical.

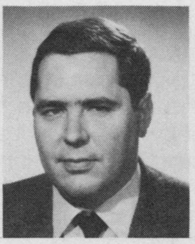
David Sarrazin and Miroslaw Malek at the University of Texas, Austin, focus their attention on fault-tolerant memory. Recognizing that chip storage technology opens certain FT design opportunities, they investigate coding, replication, and reconfiguration of RAM chips in order to extend the mean time between memory failures.

In autonomous decentralized systems, LSI technologies and specialized algorithms have been used to enhance the control and coordination of subsystems and improve functional availability. Hirokazu Ihara and Kinji Mori of Hitachi provide several theorems and algorithms used in Hitachi's autonomous, decentralized system and loop network system. This networked FT control has evolved beyond the experimental stage and is now being implemented in the Kobe and Tokyo subway systems.

Finally, replication and redundancy techniques often compound system faults and increase the need for on-line maintenance while lowering the overall average of system downtime. As an alternative, Algirdas Avižienis and John P. J. Kelly of UCLA propose that fault tolerance can be attained more effectively through design diversity. In their article, they discuss various concepts and experiments in design diversity.

Wing N. Toy is a supervisor in the Exploratory Switching Network Department at AT&T-Bell Laboratories in Naperville, Illinois, where he has been involved in the design of highly reliable processors for Bell System electronic switching systems and other telephone-related applications for the past 32 years. He was on the faculty of the Computer Science Division of the University of California, Berkeley, Engineering Department, as a visiting MacKay lecturer during the 1973-74 academic year. He is a fellow of IEEE and was cited for contributions to the conception, design, and development of fault-tolerant computers for electronic telephone switching and telecommunication systems. Also, he was made an AT&T-Bell Laboratories' fellow in 1983.

Toy received his BSEE and MSEE from the University of Illinois in 1950 and 1952 respectively and his PhD from the University of Pennsylvania in 1969.



Michele Morganti is head of the Automation and Information Systems Department at Central R&D Laboratories Italtel, Italy. His current responsibilities include directing of computer-aided hardware and software design methodologies from high-level specifications to volume production. Before working for Italtel he was with Telettra Switching Division, also in Italy, where he was deeply involved in the hardware and software design of digital switching systems.

Morganti received his doctorate in engineering from the Politecnico di Milano in 1973. He has been a member of the IEEE since 1977, and is an active member of the IEEE Computer, Communications, and Reliability Societies and of the IFIP WG 10.4 on Fault-Tolerant and Reliable Computing.

Questions about this special issue can be directed to Wing N. Toy, AT&T Bell Laboratories, Naperville-Wheaton Rd., Naperville, IL 60566.