SEL-77-032

RESEARCH IN THE DIGITAL SYSTEMS LABORATORY:

AUGUST 1976 - JULY 1977

Technical Report No. 150

September 1977

Digital Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

Sponsors of research describedinthis report include the Air Force Office of Scientific Research, the Army Research Office - Durham, the Ballistic Missile Defense Systems Command, the Defense Supply Service (DARPA), the Energy Research and Development Administration, the Joint Services Electronics Program, the National Aeronautics and Space Administration, the National Institutes of Health Division of Research Resources, the National Science Foundation, the Office of Naval Research, and University of California Lawrence Livermore Laboratory. Specific grants and contracts are cited in each research section.

RESEARCH IN THE DIGITAL SYSTEMS LABORATORY: AUGUST 1976 - JULY 1977

Technical Report No. 150

Digital Systems Laboratory
Stanford University
Stanford California, 94305

ABSTRACT

This report summarizes the research carried out in the Digital Systems Laboratory at Stanford University during the period August 1976 through July 1977.

Research investigations were concentrated in the following areas: Computer Reliability and Testing, including detection of intermittent failures, testing for sequential circuits, self-checking linear feedback shift registers, simulation analysis of high-reliability systems, effects of failures on gracefully degradable systems, performance-related reliability measures, verifiably reliable computer systems, fault diagnosis in digital systems, and software reliability; Critical Fault-Pattern Determination; Computer Architecture, including trace facility, memory interleaving, and monitors for signal activity; Organization of Computer Systems, including an emulation research laboratory, emulators, and memory performance; Feasibility of Real-Time Emulation, including directly executable languages; Distributed Data Processing for Ballistic Missile Defense; Description Languages and Design for General-Purpose Computer Architectures, including evaluation of existing hardware description languages, development of a

structural description language, applications of the structural design language, bounds for maximal parallelism, and parallel information processing in biological systems; Computer Networks, including broadcast protocols in packet-switched computer networks and the optimal placement of dynamic-recovery checkpoints in recoverable computer systems; Design and Verification of Reliable Software including specifications and proofs for abstract data types in concurrent programs, specification and verification of monitors, and operating system design; Design Automation, including a language for describing the structure of digital systems, the SPRINT printed-circuit design system, computer-aided layout of large-scale integrated circuits, and an interactive system for design capture; Database, including studies in distributed processing and problem solving, a database matinenance system, and the implementation of database in medicine; and Digital Incremental Computers.

TABLE OF CONTENTS

page no.

A. COMPUTER RELIABILITY AND TESTING 1
1. Optimal Strategy for Detection of Intermittent Failures . 1
2. Random Compact Testing for Sequential Circuits. 1
3. Self-Checking Linear Feedback Shift Registers 3
4. Simulation Analysis of High-Reliability Systems 3
5. Effects of Failures on Gracefully Degradable Systems. . . 4
6. Performance-Related Reliability Measures. 5
7. Verifiably Reliable Computer Systems. 6
8. Fault Diagnosis in Digital Systems. 7
9. Software Reliability. 8

B. CRITICAL FAULT-PATTERN DETERMINATION 8

C. COMPUTER ARCHITECTURE. 10
1. Trace Facility. 10
2. Memory Interleaving 11
3. Monitors for Signal Activity. 12

D. ORGANIZATION OF COMPUTER SYSTEMS 13
1. Emulation Research Laboratory 13
2. Emulators 14
3. Memory Performance. 15

E. FEASIBILITY OF REAL-TIME EMULATION 16
1. Directly Executable Languages 16

F. DISTRIBUTED DATA PROCESSING FOR BALLISTIC MISSILE DEFENSE. . . 17
1. Distributed Computer Systems. 17

G. DESCRIPTION LANGUAGES AND DESIGN FOR GENERAL-PURPOSE CCMPUTER ARCHITECTURE 18
1. Evaluation of Existing Hardware Description Languages . . 18
2. Development of a Structural Description Language. 18
3. Applications of the Structural Design Language. 19
4. Bounds for Maximal Parallelism. 19
5. Parallel Information Processing in Biological Systems . . 20

page no.

H. COMPUTER NETWORKS 21

1. Broadcase Protocols in Packet-Switched Computer Networks 21

2. The Optimal Placement of Dynamic-Recovery Checkpoints in Recoverable Computer Systems. 22

I. DESIGN AND VERIFICATION OF RELIABLE SOFTWARE 22

1. Specifications and Proofs for Abstract Data Types in Concurrent Programs. 23

2. Specification and Verification of Monitors. 23

3. Operating System Design 24

J. DESIGN AUTOMATION 25

1. A Language for Describing the Structure of Digital Systems 25

2. The SPRINT Printed-Circuit Design System 25

3. Computer-Aided Layout of Large-Scale Integrated Circuits 26

4. Interactive System for Design Capture 26

K. DATABASE 26

1. Studies in Distributed Processing and Problem Solving . . 27

2. Database Maintenance System 30

3. Implementation of Database in Medicine 30

APPENDIX 31

Ph.D. Dissertations 31

Journal Articles, Books, Book Chapters. 31

Conference Proceedings. 33

Technical Reports 35

Technical Notes 38

Papers Accepted for Publication 40

Papers Accepted for Presentation. 40

DIGITAL SYSTEMS LABORATORY RESEARCH

AUGUST 1976 - JULY 1977

A. COMPUTER RELIABILITY AND TESTING

E. J. McCluskey

NSF Grants MCS76-05327 and MCS76-05327-A01
Contract AFOSR 77-3325

1. Optimal Strategy for Detection of Intermittent Failures (J. Savir)

Intermittent failures are physical failures that appear and disappear apparently at random. In integrated circuits, overspreading or loose particles of eutectic solder can make temporary faulty contacts on the silicon die. Another common type of intermittent failure is produced by weak bonding between the leads and circuit. Stresses break open the bonding, but a temporary contact can be reestablished by vibrations or similar effects.

Such failures in digital systems are especially difficult to detect because the periods during which they occur are unknown. A detailed analysis of the problem led to an optimal strategy to test combinational circuits for possible intermittent failures. The best test is designed to minimize the conditional probability that an intermittent failure escapes detection. A relatively simple procedure that determines the optimal strategy produces the set of input vectors for detection plus the relative frequencies with which these input vectors must be applied. Optimal testing is far more efficient than the more intuitive approach which consists of repeating n times the test for the permanent faults.

2. Random Compact Testing for Sequential Circuits (J. Losq)

With the increasing complexity of digital circuitry, the testing problem has become correspondingly more difficult and time-consuming. The methods that produce test sets by quickly analyzing the effects of

every possible failure on the circuit become prohibitively expensive. For large printed circuit boards with several hundred ICs, such methods may require a great deal of computer time to produce test sets that cover approximately 90 percent of all single failures. One method to overcome this complexity, known as random compact testing, uses random inputs as test vectors. The testing sequence is based on a long sequence of random input vectors; however, instead of keeping the complete output sequence (which may be several million characters), the output stream is compacted. Counting the number of logic 1's in the output sequence or the number of transitions are examples of types of compaction. The result, called the signature, is then used to detect the presence or absence of failures; the rationale is that faulty circuits will yield a different signature than fault-free circuits.

It has been demonstrated that compact testing detects most failures in combinational circuits. The goal of this research was to investigate its efficiency in detecting failures in sequential circuits which are far more difficult to test. The initial state, synchronizing sequences, and propagation of internal failures to the outputs were problems to be considered. The initial-state problem can be overcome simply because a long sequence of random inputs is sufficient for initialization. Most failures inside the memory elements can be detected by compact testing, and, most failures in the combinational logic that synthesizes the memory excitations and the outputs will change the circuit signature. The ability to detect small variations in signatures is directly dependent on the length of the test sequence. A longer sequence produces better detection.

As a result, compaction is believed to be an efficient and simple method for testing sequential circuits. The current emphasis of this work is oriented toward the determination of the best compaction scheme. Transition counting, l's counting, and compaction by means of a linear feedback shift register are being considered. It is hoped that such a study will provide a general and simple approach to achieving good fault detection (over 90 percent coverage) without the need for complex circuit analysis.

3. Self-Checking Linear Feedback Shift Registers (D. J. Lu)

In the application of cyclic redundancy codes, linear feedback shift registers (LFSRs) perform data encoding and error detection and correction functions. In testing digital systems, LFSRs are used to compact long data streams into short easily compared digital signatures; they also have applications as counters and pseudo-random sequence generators.

Because LFSRs are responsible for the integrity of data or test results, the proper operation of these circuits is very important. Analysis of the behavior of faulty LFSRs revealed that they are not self-checking. For certain stuck-at faults, the typical LFSR was observed to be neither fault-secure nor self-testing. Even for the application of cyclic redundancy code checkers, it became evident that the basic LFSR provides inadequate distinction between code data errors and circuit faults.

The implementation of totally self-checking LFSRs by duplication was investigated. The fault-secure property was verified, and the self-testing capabilities were evaluated on a quantitative scale. To reduce hardware and time-delay requirements in the comparison logic, the possibilities of reducing the number of comparison points were examined. It was discovered that, given a reasonable single-fault assumption, the comparison logic could be eliminated and that the totally self-checking LFSR can be implemented in two LSI chips.

4. Simulation Analysis of High-Reliability Systems (P. A. Thompson)

In previous work, a simulation technique was developed to evaluate the reliability of redundant digital systems. The behavior of the system in the presence of faults was configured into a simulation model into which randomly occurring faults could be injected. The evaluation process for one system required the simulation of many missions, with one mission determined to be all the events from time = 0 until the time at which the system fails. By obtaining the time-to-failure for many missions, an estimate of system reliability as a function of time could be computed. Unfortunately, this method is prohibitively expensive if

reliability is too close to 1.0 during the time interval being considered. For example, the values of reliability were so high in the dual computer system that approximately 10^9 missions would be required to obtain reasonably accurate results.

A method of estimation was developed, which greatly reduces the number of missions needed to achieve the desired accuracy in the reliability function. By taking advantage of several statistically stationary properties (such as the constant failure rates and system response to faults), it was possible to consider one simulated long (in time) mission as the probabilistic equivalent of many shorter missions. An unbiased estimator of the reliability function has been obtained for this method. An upper bound on the variance has been derived so that the minimum number of missions required for the desired level of accuracy can be computed.

This method of estimation was applied in the analysis of a dual computer system with a lo-hour reliability of approximately 1 to 10^{-7}. It reduced the total number of required missions and the cost of simulation by a factor of 2×10^3.

5. Effects of Failures on Gracefully Degradable Systems (J. Losq)

Gracefully degradable systems are multiprocessors that can continue to operate even after failure occurrences. When a failure is detected, the remaining fault-free subsystem takes the necessary recovery steps to stop the propagation of errors and proceeds to reconfigure the remaining system so that the faulty part can be isolated and normal computation can resume. The economic interest of gracefully degradable systems is self-evident--continuous service to the system users.

Many architectures have achieved graceful degradation, and widely diverse systems have been designed. A very general model was developed to evaluate the effects of failures on system performance. The model considers hardware, software, detection-recovery capabilities, and the type of applications for which these systems are used. Failures are classified according to their effects. Safe failures are immediately detected before the errors propagate; unsafe failures may produce errors for some time before they are detected. Consequently, recovery

from unsafe failures is far more difficult and, depending on the criticality of the application, it may be necessary to recheck every action taken by the system before the detection occurred.

The model partitions gracefully degradable systems into resources. Each resource, whether hardware or software, performs a given type of service. Resources are formed by functionally equivalent elements; processors for the processing resource is an example. For the system to be operational, each resource must provide a minimal level of service, and failures in the elements will decrease that level. Unsafe failures incapacitate the resource during the time required for detection and recovery. Total system performance depends directly on the state of the resources.

Using this model, it was possible to evaluate the sensitivity of system performance to failures. The merits of hardware- vs software-implemented fault-detection mechanisms were also compared. Many of the trade-offs and optimization problems have been documented. The importance of the software unreliability factor in the overall performance equation has led to a refocusing of our work toward modeling software-induced failures.

6. Performance-Related Reliability Measures (D. Beaudry)

Reliability measures for distributed or multiprocessor computing systems should consider total system performance. Traditional reliability calculations are applicable when redundancy is used to achieve fault tolerance. These measures, however, do not address the problems of diminished performance caused by the failure-induced degradation of system resources. Several measures have been developed that take into account the system's capacity to execute its computing tasks.[t] These measures, obtained by means of a Markov model of the system, can be applied to compare and evaluate computer-system architectures with different reliability and performance characteristics.

[t] M. D. Beaudry, "Performance-Related Reliability Measures for Computing Systems," Tech. Note No. 101, Digital Systems Laboratory, Stanford University, Stanford, Calif., 1977.

7. Verifiably Reliable Computer Systems (P. A. Thompson)

When designing computer systems for critical applications, it is necessary to be able to determine accurately the overall reliability of the system. Knowledge of reliability and mission times makes it easier to select the best design alternative among a large number of candidates. It is also often necessary to demonstrate with a given level of confidence that a complex redundant system meets the reliability criteria specified in the development contract. Unfortunately, the lack of accurate failure-rate data for components makes it impossible to obtain precise estimates of their reliability in most systems that use redundancy to enhance reliability. As a result, it would be useful to know for which redundant structures a precise determination can still be obtained without accurate data relating to failure and repair rates. Given a particular redundant system, how it should be partitioned (into modules or components) must also be determined so that knowing the function of each section guarantees, in theory, a possible estimation of overall system reliability.

The initial approach to this problem has assumed a digital system containing a fixed number of identically designed redundant modules. System behavior in the presence of any possible fault is known, as is the reliability function for each of the modules. It is further assumed that the failure/repair probability distributions associated with each type of fault that may occur within a module are not precisely known. It is desirable to identify the necessary and sufficient system characteristics so that the reliability of the whole system can be exactly determined under these constraints.

In systems with no more than two identical modules, it has been proved that, generally, it is impossible to exactly determine the overall reliability function. The study of systems with three or more identical modules is not yet complete, but intuitive arguments are based on the extension of this result to any number of modules. It is not known how additional assumptions, such as assuming constant (in time) failure rates, will affect this result.

Although it may never be possible to obtain the exact reliability function for a system, perhaps some redundant configurations will

produce better estimations than others. Future work will investigate the theoretical limits on the precision of such estimations, based on incomplete failure/repair data.

8. Fault Diagnosis in Digital Systems (M. L. Blount)

The problem of fault diagnosis in multiunit digital systems, particularly multiprocessor computers, has been investigated. Two models of system-level diagnosis in digital systems have appeared in the literature--the Preparata and the Kime models. In both, some idealistic assumptions were made concerning tests and the testing process.

A new general model for system-level diagnosis, called the probabilistic model, has been proposed in which the outcomes of tests are considered to be random rather than fully deterministic events. Compared to the Preparata and Kime models, the probabilistic model has the following advantages.

- The restriction that all tests must be complete tests is removed.
- Intraunit tests can be developed.
- It is possible to determine the extent to which faults in the testing units affect the test results and the effects of transient faults on the diagnostic procedure.
- The diagnosability measure is more general.

In addition, a diagnosis strategy has been derived to optimize this measure.

The probabilistic model is being used in a study of diagnostic strategies employed in high-availability multiprocessors. The goal is to compare the various strategies in terms of their effectiveness.

Research is also directed toward an efficient implementation of a diagnostic facility for high-availability multiprocessors. A distributed microcomputer-based design has been proposed. The facility also appears to be capable of handling the error-detection monitoring, retry, and reconfiguration required in a high-availability system.

9. Software Reliability (D. Beaudry)

Because software failures are a significant source of system failures in complex computing systems, they must be included in models developed to evaluate system reliability and performance characteristics. A statistical analysis of failure data obtained from the Stanford Linear Accelerator Center multiprocessor system revealed that software and hardware failures have different characteristics and cannot be modeled in the same way.† If variations in the demand for system resources (especially the operating system) are taken into account, however, a tractable model for software-induced system failures can be defined. This type of analysis can be used to evaluate the effects of changes in hardware or software system resources.

B. CRITICAL FAULT-PATTERN DETERMINATION

E. J. McCluskey

NASA Grant NSG 1410

A study has been undertaken to develop methods for determining all the fault patterns that can cause a highly reliable computer system to fail. For critical applications (such as airplane automatic flight control or landing), computer systems have been made highly reliable through the use of such redundancy techniques as triple modular redundancy. These highly redundant systems will recover from most failures, and only a few failures or a combination of failures (called fault patterns) will result in a system crash. Reliability-evaluation techniques based on a statistical evaluation of these critical fault patterns do not produce accurate results. The population of critical fault-patterns is generallytoosmall or too poorly behaved to enable meaningful statistical approaches.

†M. D. Beaudry, "A Statistical Analysis of Service Interruptions in the SLAC Multiprocessor," Tech. Rept. No. 141, in preparation.

Complete enumeration of the critical fault patterns produces much more useful information for improving and validating highly reliable systems. Three approaches are being investigated--penetration, simulation, and analytical methods.

The penetration method takes advantage of the superior human ability to pinpoint potential problems. By analogy with code-breaking techniques, a careful examination of such a highly reliable (and critical) computer can lead to very significant savings in time after the system is simulated or modeled. Many error-prone areas (such as specification completeness or timing problems) are extremely difficult to manage by simulation.

Simulation can determine the effects of failures on system behavior. A general simulator intended for reliability analysis has been developed at the Center for Reliable Computing, based on simulating the effects of each possible fault pattern to determine the critical ones. Because of the large amount of computing time required for such a global simulation, it is highly desirable to use it in interaction with the penetration approach. Because the simulator does not require that all parts of the system be described with the same level of detail, the most critical parts of the system can be defined in greater detail.

Investigation of an analytical approach is under way. By accurate modeling, it should be possible to identify the possible states of the system following a failure. The system will be represented by a graph where the nodes are the states and the edges correspond to failures. The critical fault patterns will correspond to a sequence of edges (failure occurrences)'that take the system from a fault-free state to a failed state. Such an approach overcomes the need to enumerate all possible fault patterns. The single failures that cause system crashes are the paths of length 1 between the fault-free and the failed state, the critical double failures are the paths of length 2, and so on.

Because of the different but complementary advantages (and drawbacks) of these three approaches, the best solution may involve an interaction between the analytical and simulation approaches. Human intelligence will close the loop.

c. COMPUTER ARCHITECTURE

E. J. McCluskey and S. S. Owicki

Contract N00014-67-A-0112-0601

1. Trace Facility (D. J. Rossetti)

The STRAP/370 instruction tracing facility is producing data for various studies in the areas of performance, architecture, and operating systems. A paper entitled, "The Design and Implementation of an Operating System Tracer," has been submitted for publication in the Communications of the Association for Computing Machinery. It describes the unique aspects of the tracer, provides examples of its use, and recommends further applications of the data and technique.

Cur approach is being extended into the microprocessor area; most of the desirable features of the original technique will be retained. Computer architectures will be evaluated from the standpoint of development and programming ease as opposed to processing speed because software development has become a major cost in microprocessor use. The principal tool is a detailed-level instruction tracer having the following characteristics.

- It is independent of the program being measured to the extent that one microprocessor can be used to measure another.
- The instruction stream is sampled in bursts at a rate determined by the user.
- Minimal perturbation is introduced into the measured system because the trace is collected with hardware assistance and not by interpretation.
- Actual systems and programs can be measured, including input/output and interrupts. The trace is not gathered in an artificial environment, such as a simulator.

The tracer is an extension and adaptation of the work of a graduate student. It has been used to obtain instruction frequency data for application programs in our microprocessor laboratory. Current plans are to collect extensive trace data for studies in architecture

evaluation and to gain understanding of microprocessor program and input/output behavior.

2. Memory Interleaving (F. W. Terman)

Models of interleaved memory systems have been investigated by means of a trace-driven simulation. The basic model extends the one developed by Burnett and Coffman[†] to the architecture of the IBM 360/370 with its variable-length instructions and operands. This basic simulation model also includes multibyte transfers per memory access and facilitates the study of the effects of channel interference, data-request reordering, and data queue emptying.

Memory requests for the simulation are obtained from two sets of instruction-by-instruction trace records. The first set, produced by the program TRACE/360, traces the problem state component of typical programs running on the IBM 360/370. The second set, produced by the program STRAP/370, traces samples of the total activity of the CPU, including the supervisor and problem states.

The theoretical predictions of Burnett and Coffman for the increase in memory bandwidth as a result of interleaving are found to fit well with the simulation results obtained for the fetching of instructions. The usefulness of fetching instructions in blocks is limited by the relatively high frequency of branches on the IBM 360/370. Consequently, an active channel has relatively little effect (less than 10 percent degradation) on the fetching of instructions from an interleaved memory.

For the transfer of operands to and from memory, the simulation results reveal only one-half the increase in memory bandwidth predicted by Burnett and Coffman, which indicates that data references on the IBM 360/370 are not random. The effect of an active channel is again small because of the number of idle modules in a typical memory cycle. A larger degradation (10 to 20 percent) occurs if the data

[†]G. J. Burnett and E. G. Coffman, Jr., "A Study of Interleaved Memory Systems," Proc. AFIPS Spring Joint Computer Conf., 36, AFIPS Press, Montvale, N.J., 1970, pp. 467-474.

requests are forced to "catch up" with the instruction requests before a successful branch is executed. This is a reasonable restriction because the branch address cannot be calculated with certainty until the values of the general-purpose registers are known. On the other hand, a significant improvement (15 to 30 percent) in the memory bandwidth can be obtained by allowing the data requests to be filled out of order. These effects are being investigated in detail and compared to the predictions of other theoretical models.

This analysis is being extended to include the effects of interference between multiple CPUs. The effects of the interdependencies between instructions and operands and between the operands themselves will also be examined.

3. Monitors for Signal Activity (R. C. Ogus)

Probabilistic models have been developed[†] to describe logic circuits where a probability can be assigned to a signal line that indicates that the signal is a logic "1" given the probability of the inputs of the circuit is a "1." Algorithms have been developed to derive the signal-probability expressions as functions of input signal probabilities. Signal probability is of value in computing the probability of detecting faulty behavior in circuits and in the calculation of the signal reliability of a circuit (the probability that the circuit output is correct). In addition, a scheme to select efficient test sets from random inputs has been described[‡] wherein the effect on a circuit output of exercising any input variable must be measured. In this way, the inputs can be weighted according to their importance. This requires measuring both the input and output signal probabilities. This scheme can also be applied in error-latency studies and in system testing.

[†] K. P. Parker and E. J. McCluskey, "Analysis of Logic Circuits with Faults Using Input Signal Probabilities," IEEE Trans. on Computers, C-24, 5, May 1975, pp. 573-578.

[‡] H. D. Schnurmann, E. Lindbloom, and R. G. Carpenter, "The Weighted Random Test Pattern Generator," IEEE Trans. on Computers, C-24, 7, Jul 1975, pp. 695-700.

Because, in many cases, circuits are too large for analytical analysis, it is desirable to develop a hardware monitor unit that can actually measure signal activity and display the signal probability. Two monitor designs have been proposed. The first is a simple stand-alone single probe that can directly measure and display the probability of a "1" on a signal line. The second is a more complex processor that can insert a number of probes into a circuit and measure the signal probabilities of the lines and also process the information to relate to a particular application.

A prototype of the first monitor has been constructed and is being exercised. The second design would be implemented on an IBM System/7 computer. These tools should be useful augmentations to the other monitors described above and should prove valuable in such applications as test generation and actual circuit testing.

D. ORGANIZATION OF COMPUTER SYSTEMS

M. J. Flynn

ERDA Contract EY76-S-03-0326 PA39

This project is directed toward general studies of the organization of computing machines, including parallel processors that attempt to manage a large number of data, logical resources, and/or tasks simultaneously. Detailed investigations of computer systems include

- computer-system resource use and specification by general models and simulation
- comparative study and evaluation of system-design architectures and concepts, including parallel and microprogrammed processors

1. Emulation Research Laboratory (C. J. Neuhauser)

The primary mission of the Stanford Emulation Laboratory is to study the processor instruction execution as it is represented by both physical and conceptual processors. A facility has been developed that may replace (or emulate) the instruction-processing activity of an

arbitrary machine. At the center of this facility is an interpretively programmed host processor. When properly programmed, it will exactly emulate the instruction fetching, decoding, and execution of a selected target machine (conceptual or physical). The software-oriented process of organizing the host-processor resources for a particular target-machine emulation is referred to as "microprogrammed interpretation."

The Emulation Laboratory is divided into two cooperating but independent subsystems--the emulation and experiment-control systems. The emulation system contains a special-purpose "host" (or interpretive) processor, control storage, main storage, and a common bus. To emulate a particular target machine, a microprogram interpretation of this machine is constructed and loaded into the control store of the host processor. During emulation, the control store and physical resources of the host processor serve as the data storage and manipulation resources of the target machine. The main memory system is similarly configured to represent the main memory system of the target machine. The experiment-control system consists of low-speed user-oriented I/O devices clustered around a general-purpose processor.

The two laboratory systems are coupled via the common bus and are organized so that the experiment-control system has direct access to the data storage of the host processor. This provides a convenient method of direct observation and control of the emulation experiment.

Host-machine microprograms are written in one of two especially developed languages.

2. Emulators (C. J. Neuhauser)

Several emulators have been written to interpret the codes of traditional machines. An emulator capable of processing the complete IBM 360 basic instruction set has been written. For typical instruction sequences, approximately 100,000 target-machine instructions are executed per second, which indicates that the emulator is operating at 70 percent of an actual IBM 360 Model 50. Comparable execution rates may be expected for similar second- and third-generation physical processors. Studies are under way to extend the 360 emulator so that detailed resource-usage statistics can be obtained during the emulation process.

We have written emulators for the PDP-11, CDC-6400/6600, RC-4000, and Intel-8080. These systems interpret target codes at 50 to 100,000 target-machine instructions per second. The lower performance is the result of machines with data paths in excess of 32 bits or that use complex register interpretation. We are continuing to expand the number of architectures we interpret and to obtain usage statistics from each of these machines.

3. Memory Performance (B. R. Rau)

The performance of the memory system is central to the performance of the entire computer system and, of the factors that affect memory performance, program behavior is, perhaps, the most important. The performance of the memory system is best measured by two figures of merit--access time and bandwidth. Accordingly, our work has concentrated on studying the effect of program behavior onthesetwo measures.

The most cost-effective way of reducing access time is to structure the memory hierarchically with a number of levels, each one faster and smaller than the one below it. Two aspects of program behavior that most directly impact the effectiveness of a memory hierarchy are temporal and spatial locality. The least-recently-used stack model (LRUSM) is a good model of temporal locality, but its analysis has been neglected; however, we have developed some properties and applications for it. Spatial locality is modeled by the spatial locality function based on the LRUSM and is observed to be invariant with the level in the hierarchy at which it is measured. This property is utilized in developing a procedure for analyzing multilevel memory hierarchy. The string of operand requests possesses sequentiality in a generalized sense, and this can be exploited to enhance performance at the very highest level of the hierarchy.

The cost-effective approach for increasing the bandwidth is to interleave the memory in a low-order manner. The LRUSM is an adequate model of program behavior in this context and can be used to analyze the bandwidth obtained with a highly overlapped uniprocessor. The bandwidth in a multiprocessor system is also being studied under fairly general assumptions concerning program behavior.

The fundamental issue of how program behavior should be modeled and analyzed is being examined in some detail. It is argued that the most successful approach is to permit a judicious choice of approximations and to balance the errors incurred in developing the model and in analyzing it subsequently.

E. FEASIBILITY OF REAL-TIME EMULATION

M. J. Flynn

Contract DAAG29-76-G-0001

The emphasis of this study is focused on the feasibility of advanced and more powerful host processors capable of processing 1 to 5 MIPS of target machine instruction.

Directly Executable Languages (L. W. Hoevel)

A DEL is an intermediate language tailored to a specific combination of source language, host machine, and user community. User-written source programs are generally evaluated by translating them into an equivalent DEL "surrogate" and then submitting them for execution, in place of the original source version, as often as a user desired. The DEL thus occupies the same place in a computing system as a traditional machine language; however, because it is specifically designed to couple well with the given constraints, the resulting system should be far more efficient.

We have been studying the potential space-time advantage of DEL-based systems for high-level source language/microprogrammable host-machine constraint combinations. In particular, we have developed an intermediate text for a basic FORTRAN/EMMY system that results in an average reduction factor of 7.5 in both space and time. This language, called DELtran,† has the following interesting properties.

†Technical Notes 108 and 130, Digital Systems Laboratory, Stanford Electronics Laboratories, Stanford University, Stanford, Calif.

- It is "invertible"; the original source text can be deduced from its DELtran surrogate and static symbol table--at least up to the level of redundant blanks and parentheses.

- No memory accessing "overhead" instructions need be inserted into the DELtran instruction stream (approximately three such "overhead" instructions are present in 370 instruction streams for each functional instruction).

- The result of any intermediate computations is pushed onto an internal LIFO evaluation stack; however, only intermediate values are pushed onto this stack, never the values of atomic variables.

F. DISTRIBUTED DATA PROCESSING FOR BALLISTIC MISSILE DEFENSE

M. J. Flynn

Contract DASG60-77-G-0073

The emphasis of this study is focused on the feasibility of advanced computer architecture by use of distributed computing elements.

Distributed Computer Systems (P. S. Yu)

Few areas of current and future architecture research and development hold as much potential promise for BMD applications as distributed data processing. The possibilities in terms of performance, systems integrity, and extensibility, if realized, may significantly impact overall BMD system-implementation philosophy.

To support ongoing BMD research in DDP, the objectives of the proposed research are

- development of analytical tools to evaluate the performance of distributed networks of computers

- development of analytical tools to evaluate the performance of nodal architectures (methods for evaluating and determining the performance of various configurations of processor architectures located at a geographically common site)

- investigation of novel approaches to distributed nodal computer architectures

The emphasis of this research is twofold: (1) to derive mathematical models to determine the performance of existing distributed data-processing approaches at both the network and nodal levels and (2) to initiate an investigation of novel approaches to the structure of such DDP architectures.

G. DESCRIPTION LANGUAGES AND DESIGN FOR GENERAL-PURPOSE COMPUTER ARCHITECTURES

M. J. Flynn, W. M. vanCleemput

Contract N00014-75-C-0601

The purpose of this study is to develop description languages to describe computer architectures and to develop a basis for understanding the limits of the design of such architectures (to establish the fastest speed of execution of a program).

1. Evaluation of Existing Hardware Description Languages (D. Hanson, W. vanCleemput)

To evaluate the feasibility of describing computer architectures by means of a hardware description language, several of these languages were analyzed. Three were selected for further study, based on the availability of a working compiler and simulator and on the suitability of the language for the multilevel description of digital systems. Compilers for the CDL (Chu, University of Maryland) and the CASSANDRE (Mermet, University of Grenoble, France) languages were implemented, and these languages were compared in terms of their descriptive power. A compiler for the DDL language (Dietmeyer, University of Wisconsin) was obtained and is being adapted for use on the IBM 370.

2. Development of a Structural Description Language (W. vanCleemput)

Most of the currently existing hardware description languages emphasize a behavioral description and neglect the structural aspects of a design. A behavioral description is often more than sufficient to describe an existing architecture. To aid in the design process, however,

it is necessary to obtain all available information (structure and intended behavior). The SDL language was developed to identify the structural properties of the system, and a compiler for the language is operational. A special characteristic of SDL is that it is hierarchical.

3. Applications of the Structural Design Language (T. Bennett, J. Hupp, K. Stevens, W. vanCleemput)

One of the obvious applications of a structural design language is to serve as the input medium for a physical design system. Such a system may consist of several subsystems for performing such tasks as printed-circuit layout, integrated-circuit layout, gate-level logic simulation, circuit-level simulation, fault test generation, and automated logic diagram generation. A printed-circuit layout system is being implemented, and a prototype system will soon be operational. An integrated-circuit layout system is in the design phase. Interfaces between the SDL language and several logic-level and circuit level simulators are in the planning stage.

4. Bounds for Maximal Parallelism (R. Lee, M. Flynn)

This is a study of the performance limits of a single program executing on a large number of identical processors operating in parallel in an MIMD (multiple instruction-multiple data) organization. With the rapidly decreasing cost of LSI microprocessors, it is now economically feasible to consider a whole army of processors within the computer architecture to speed up a computation, even at the reduced efficiency of each component processor. The processor is no longer the hallowed CPU or the most valuable resource to be utilized with the greatest efficiency. Some "acceptable" level of efficiency should be obtained; however, we must determine the type of speedup that can be expected by increasing the number of processors even if the problems of control and communication are ignored.

We first defined a general model computation on a p-parallel processor and distinguished between the logical parallelism (p^* processes) inherent in a computation and the physical parallelism (p

processes) available in the computer organization. We then identified such performance measures as execution time, **speedup,** efficiency, and space-time product from which we can evaluate the performance improvements (if any) of p-parallel processor systems over uniprocessor systems. We observe that performance generally depends on both computer architecture and computation.

We derived the necessary and sufficient conditions for the maximum attainable **speedup** of a p-parallel processor over a **uniprocessor** as

$$S_p < \min\left(\frac{p}{\ln p}, \frac{p^*}{\ln p^*}\right)$$

Despite the many views concerning the potential **speedup** of parallel processor systems, this bound has never before been established. In addition, with sufficient processors ($P \geq p^*$), the conditions under which the maximum attainable **speedup** of a computation is $p^*/\ln p^*$, maximum efficiency is $1/\ln p^*$, minimum execution time is $T_1 \cdot \ln p^*/p^*$, and the minimum space-time product is $T_1 \cdot \ln p^*$ (where T_1 is the execution time of the computation on a uniprocessor). The empirical speedups obtained for a large number of different types of computations revealed that 80 percent of all programs examined satisfied these conditions and had maximum speedups of less than $p^*/\ln p^*$.

5. Parallel Information Processing in Biological Systems (S. Wakefield, M. Flynn)

The interconnections and types of synapses between units of a particular neural subsystem (the stomatogastric ganglion of the lobster) have been extensively determined by biologists, as have the stereotyped motor patterns it produces. The exact mechanisms and sequence and duration bounds of impulse bursts that must underly the production of the coordinated muscle-activation patterns, however, are unknown. Such a mechanism would be analogous to the switching network responsible for the traversal of states in a digital sequential circuit. Because of this analogy, this and other similar simple biological information-processing subsystems are being investigated. In addition, the

speed, power requirements, size, and information capacity of individual neurons are being compared to those of electronic information-processing components.

H. COMPUTER NETWORKS

V. Cerf

Grant MDA903-76-C-0093, ARPA Order No. 2494

1. Broadcast Protocols in Packet-Switched Computer Networks (Y. K. Dalal)

This study investigates the design and analysis of broadcast-routing algorithms for store-and-forward packet-switched computer networks. Broadcast routing is defined here as multidestination routing in which a packet is delivered to all destinations rather than to some subset.

We have examined five alternatives to transmitting separately addressed packets from the source to the destinations. The algorithms are compared qualitatively in terms of memory requirements, ease of implementation, adaptiveness to changing network conditions, and reliability; they are also compared quantitatively in terms of the number of packet copies generated to perform broadcast and the delays to propagate the packet to all destinations. Lower bounds on the performance measures are determined by examining regular graphs.

Protocols that provide reliable communication using broadcast routing (such as broadcast protocols) are analogous to interprocess communication protocols except that communication is between one and many processes. The design of broadcast protocols is faced with problems similar to those in the design of interprocess communication protocols--addressing, sequencing, duplicate detection, and guarantee of delivery. This area presents many subjects for future research.

We have demonstrated how the catalog of a distributed file system could be structured simply if the system could make use of efficient reliable broadcast protocols. The properties of such protocols at the host level are based on their applications and on the reliability of

the routing algorithms. We have examined the trade-offs between global and subgroup broadcast routing. One conclusion is that communication subnets should support both capabilities in the form of multidestination addressing and reverse-path forwarding, respectively.

An outcome of this investigation is the formulation of two distributed (parallel) algorithms for constructing minimal spanning trees. We believe that these algorithms are the first of their kind. They can be used in broadcast routing and in such networks as the Packet Radio Network.

2. The Optimal Placement of Dynamic-Recovery Checkpoints in Recoverable Computer Systems (W. A. Warren-Angelucci)

Reliability is a major factor in any computer system because, no matter how carefully designed and constructed, these systems will fail. The rapid and systematic restoration of service after an error or malfunction is a significant design and operational goal. This study has developed a recovery method that guarantees that the computer system, its associated data bases, and communication transactions will be restored to an operational and consistent state within a given time and cost bound after the occurrence of a failure.

We have considered the optimization of a specific software strategy--rollback and recovery--within the framework of a graph model of program flow that encompasses communication interfaces and data-base transactions. Algorithms have been formulated that optimize the placement of dynamic-recovery checkpoints, and a run-time technique has been developed to determine the optimal placement of these checkpoints. A method has also been presented for statically precomputing a set of optimal decision parameters for the associated program model.

I. DESIGN AND VERIFICATION OF RELIABLE SOFTWARE

S. S. Owicki

Contract F49620-77-C-0045

1. Specifications and Proofs for Abstract Data Types in Concurrent Programs (S. S. Owicki)

Shared abstract data types, such as queues and buffers, are useful tools for building well-structured concurrent programs. This study has developed a method for specifying shared types to simplify concurrent-program verification. The specifications describe the operations of the shared type in terms of their effect on variables of the process invoking the operation. This makes it possible to verify the processes independently, thereby reducing the complexity of the proof. The key to defining such specifications is the concept of a private variable--a variable that is part of a shared object but belongs to just one process. Shared types can be implemented through an extended form of monitors, and proof rules will verify that a monitor correctly implements its specifications. Concurrent programs can be verified using the specifications of their shared types.

2. Specification and Verification of Monitors (S. S. Owicki)

A monitor is a programming language construction that defines a logically related group of shared data items and a set of operations on those items. The operations are the only means by which programs may access the shared data, and the monitor includes synchronization to ensure that processes do not interfere with each other as they perform operations. This limited access simplifies the verification task.

A monitor is specified by listing the shared data items and their initial values, plus the effects of each operation. Verification of programs that use the monitor is simplified by describing the operations in terms of variables that are private to the program invoking the operation. The relationship between the private and shared variables is expressed by an invariant relation which is true for the initial monitor values and is preserved by each operation.

There are two steps in verifying systems that use monitors. The first is proving that the monitor satisfied its specifications (that the operations preserve the invariant and have the required effect on private variables). The second is verifying the processes that call the monitor by using the specifications of the monitor operations. In some

instances, the correct behavior of one process depends on the actions of another; here, it is convenient to use a process invariant--an invariant relation on the private process variables. These tools appear to be powerful enough for most applications of concurrent programming with monitors. Further work is required to develop methods for treating dynamically allocated resources that do not fit the monitor pattern.

3. Operating System Design (A. Spector)

We have focused on the design of a paging interactive time-sharing system suitable for such computers as a **PDP-11/70.** Although we do not expect to include very complex I/O or protection mechanisms in our design, we are attempting to achieve a realistically usable system.

The system will be a hierarchy consisting of two levels--the kernel and the supervisor. The kernel is the more primitive and is the run-time support system that implements the basic operations of the high-level programming language used by the supervisor. More specifically, it contains the independent procedures of the machine-dependent operating system so as to provide concurrent processes and monitors. It also translates certain privileged hardware operations to language-level primitives. For example, because supervisor processes use monitors to communicate, the kernel must maintain the queues necessary to support critical sections and wait and signal primitives. An unusual feature is that the kernel maintains detailed per-process state information that can be used by such supervisor routines as scheduling and accounting.

Although the nucleus is small and can be considered a single entity, the supervisor performs many complex functions and is best thought of as a hierarchy. For example, the long-term process scheduler is a primitive operation of the supervisor and underlies the operation of the whole system. Our design requires this process to run synchronously and to communicate periodically to the kernel a set of processes that can be executed in some ensuing interval. The memory-allocation process (also one of the basic operations) contains the logic to allocate physical memory to processes and to support shared and nonshared pages. Scheduling and memory-allocation decisions can be made, based on state information recorded by the kernel and on data contained in

certain supervisor monitors. Higher levels of the supervisor hierarchy include sections to support I/O and user program calls.

The principal emphasis of our work has been directed toward the design of the most primitive portions of the system. We believe that the accurate assignment of meanings to language primitives and the proper definition of the boundary between the kernel and supervisor will have great influence on the outcome of our project. As a result, we are proceeding with great caution in the hope of preparing a sound base on which to build a carefully structured and verified system.

J. DESIGN AUTOMATION

W. M. vanCleemput

1. A Language for Describing the Structure of Digital Systems (W. M. vanCleemput)

A large number of languages have been developed for the description of digital systems; however, most of these describe only the behavior of a system, not its physical structure. The purpose of this research is to develop a powerful language for describing the physical structure of a digital system. Such a language can be used as the input to many design automation tools such as logic simulators, fault simulation, printed-circuit layout systems, automated logic diagram generators. A compiler has been developed to test the usefulness and feasibility of this language.

2. The SPRINT Printed-Circuit Design System (W. M. vanCleemput, K. Stevens, T. Bennett, J. Hupp, N. Yamada)

The objective of this system is to develop an interactive computer-assisted design of printed-circuit boards. The SPRINT allows for the manual placement of critical components and for automatic placement of other components such as 14- and 16-pin dual in-line packages. The interconnection routing module manually routes critical connections and automatically routes noncritical connections. The system is limited to two signal layers; however, expansion to multilayer boards is planned.

The input to the SPRINT is the SDL (structural description language). In the future, the SDL description will also be used as the input to a logic simulator, a fault test generation/simulation system, and an automatic logic diagram generation.

The current system is implemented in MORTRAN and FORTRAN IV on the IBM 370 at SLAC and makes use of a Tektronix 4013 terminal, and the SDL compiler is implemented in a SNOBOL dialect called SPITBOL. The output of the system is a CalComp plot in which the artwork must be generated manually.

3. Computer-Aided Layout of Large-Scale Integrated Circuits (W. M. vanCleemput, E. Slutz)

Although several systems exist for the automated layout of LSI circuitry, none of them obtains one that is comparable to a manually designed layout. The objective of this project is to design and implement a system based on algorithmic approaches wherein certain decisions become the responsibility of the human designer. It is expected that this system will reduce design time considerably, without the penalty of excessive silicon real estate, and will be able to make use of the hierarchical nature of a system.

4. Interactive System for Design Capture (W. M. vanCleemput)

In the current design system, all input is in the form of the SDL (structural design language); however, designers often prefer to use schematic diagrams. A system is being developed that will take as its input a schematic drawing from an interactive graphics terminal and will output an SDL description ready for further processing by the design-automation programs.

K. DATABASE

G. Wiederhold

Contract DAHC15-73-C-0435

1. Studies in Distributed Processing and Problem Solving

(G. Wiederhold, K. Knutsen, H. Garcia-Molina, E. Gilbert, R. G. Smith)

a. Technical Goals

The objective of this research is the development of an understanding and a methodology for the analysis of alternatives in distributed processing and problem solving. One of the primary reasons for interest in this area is its potential to break through the speed-limitation barrier found in uniprocessing systems. If such a breakthrough can be achieved, the viability of the methods being developed by other projects using the SUMEX-AIM resource will be enhanced.

The rapid growth of microprocessor and communications technology has resulted in an increased number of proposed implementations of networks employing multiple processors. The computations to which these distributed systems are to be applied include heuristic decision-making problems, mathematical modeling, data reduction, searching through large databases, and general-purpose multiaccess computing. There is, however, a lack of an adequate global understanding of the computational trade-offs implied by network architectures.

To complement the experimental results of other investigators and to broaden their applicability to the system-design decision-making process, we are developing a general framework of computation for the study of processor interaction. This framework consists of rules to obtain parameters from programs that specify the computations, rules to parameterize descriptions of networks of processors, and procedures to calculate expected system performance from these parameter sets. The procedures may be based on relationships between descriptive and outcome variables developed through the use of simulation of network models at a suitably high level of abstraction. The framework is to be sufficiently powerful so that, when it is validated, the methods will be able to assist in the a priori assessment of the potential performance of new system alternatives or of systems with improved components.

A number of large computational applications are being analyzed so as to assess their potential for decomposition into modules for distributed processing. The current candidate applications are

- programs based on heuristic methods in decision making
- programs using multifaceted databases to retrieve and abstract information
- programs that acquire data from multiple (possibly dissimilar) sensors and attempt to reduce these data to simpler hypotheses
- programs that solve large numerical problems

The first two applications have been investigated through simulations, and the results are being analyzed.

Simulation is not the end-product of this study but is a means to develop and assess the validity of our model of the interaction of computations and processor-network architecture. Where possible, mathematical results will be used to verify the validity of model simulations. Parameters used to describe the computations include

- computational kernel size--cycle and memory demand of a computational unit between interprocessor reference requirements
- computation definition message size--amount of data required to transmit sufficient information to initiate a computational kernel
- database size--amount of data or program text required to sustain a computational kernel and its availability and residence in the network

The behavior of the system can be varied through the adjustment of other parameters that may be set to reflect the architecture of specific hardware systems or may be varied to obtain optimal performance. In addition to obvious parameters (as the number and power of the processors), we expect the following factors to be significant in developing an understanding of the spectrum of multiprocessor architectures.

- Interconnection density--as the density decreases, message delay and congestion will increase. This parameter will provide a high-level abstraction of multiprocessor connectivity schemes. Geographical distribution will increase message delay and transmission cost.

- Computational locality--a high degree of locality (of database or procedural information in the network) will enhance the probability that relevant knowledge exists in closely linked nodes, thus counteracting the effects of a low interconnection density.

- Database viscosity--a database, including the programs required to carry out the computations at a node, may be more or less fixed to one specific node. This, therefore, encourages the use of certain nodes for specific functions. Many current multiprocessor networks are completely rigid, and an optimal initial program and database of dynamic-resource allocation are required to cope with changing loads and to enhance reliability. For this reason, this parameter must be included.

- Redundancy--to assess the cost and benefits in terms of responsiveness and reliability, the redundancy of the database and computations will be made a parameter. To utilize the redundancy well, the computational resources (programs or data) that affect system performance must be identifiable.

- Error rate--to test the effectiveness of reliability strategies, node failures will be simulated based on probability distributions.

An important aspect of this model is that we intend to maintain the abstractions at a sufficiently high level to allow analytical and intuitive verification of model behavior when applied to well-understood computations. Computations have been mapped into specific parallel machines, but these results are not easily transferred into new architectures. The multiprocessor systems now being built may have characteristics with unpredicted effects on system behavior. We expect to be able to use the model to determine potential bottlenecks, which then will define areas where additional design attention has a high payoff.

b. bong-Term Objectives

We do not intend to build hardware based on the abstract model. We would like to verify our results using existing multiprocessor systems and, assuming that our model (with appropriate parameters describing load and architecture) matches the given system, we hope to be able to advise on system utilization and development. A local resource

may be the Stanford I processor now being built under ERDA sponsorship.† If we determine that a certain architecture appears to be promising, we would like to encourage and participate in its implementation.

c. Significance

A broad model adequate for the prediction of approximate system performance when distributed techniques are applied to large computational applications will simplify decision making and reduce the time and work spent on approaches that are not likely to provide significant benefits. System development can be better directed toward specific applications.

2. Database Maintenance System (D. Borel, G. Wiederhold)

A system (DBMTNS) has been developed that uses an external specification of a database to control database-integrity auditing. In addition, DBMTNS will advise maintenance personnel concerning bypassing or repair of integrity failures.

3. Implementation of Database in Medicine (G. Wiederhold, G. Purdy et al.)

A session of the 1977 National Computer Conference was organized to bring together personnel who follow two competing database approaches used in medicine (MUMPS and CODASGL). A descriptive technique was used to illustrate communalities and differences. It is intended that the conclusions reached will be published in tutorial form in the ACM Sigbio Newsletter.

† F. Baskett, McWilliams, and C. Widdoes, "Stanford-l; Multiprocessor Preliminary Design," Artificial Intelligence Internal Report, Stanford University, Stanford, Calif., Oct 1976.

APPENDIX

Ph.D. DISSERTATIONS

Betancourt, Rodolfo - "Analysis and Synthesis of Sequential Circuits Using Clocked Flip-Flops," Dept. of Electrical Engineering, (Prof. Edward J. McCluskey)

Dalal, Yogen K. - "Broadcast Protocols in Packet Switched Computer Networks," Dept. of Electrical Engineering, (Prof. Vinton G. Cerf)

Knauer, Scott G. - "Real-Time Adaptive Digital Video Compression," Dept. of Electrical Engineering, (Prof. Allen M. Peterson)

Patel, Janak H. - "Improving the Throughput of Pipelines with Delays and Buffers," Dept. of Electrical Engineering, (Prof. Edward S. Davidson)

Rafii, Abbas - "Empirical and Analytical Studies of Program Reference Behavior," Dept. of Electrical Engineering, (Prof. Forest Baskett)

Stritter, Edward P. - "File Migration," Dept. of Computer Science, (Prof. Forest Baskett)

Thomas, A. Thampy - "Scheduling of Multiconfigurable Pipelines," Dept. of Electrical Engineering, (Prof. Edward S. Davidson)

JOURNAL ARTICLES, BOOKS, BOOK CHAPTERS

Flynn, M. J. and R. Kosaraju, "Processes and Their Interactions," Special Issue of Kybepnhthe, vol. 5, 1976, pp. 159463.

Flynn, M. J., "Non-Specific Computers," INFOTECH Future Systems, vol. II, . 1977, pp. 154-168.

JOURNAL ARTICLES, BOOKS, BOOK CHAPTERS (continued)

McCluskey, E. J., "Logic Design," Encyclopedia of Computer Science (Ralston and Meek, Eds.), Petrocelli/Charter, New York, pp. 809-813, 1976.

McCluskey, E. J., "A Survey of Research at the Center for Reliable Computing, Stanford University," J. Design Automation and Fault-Tolerant Computing, vol. 1, no. 1, pp. 85-90, Oct. 1976.

Parker, K. P., "Adaptive Random Test Generation,: J. Design Automation and and Fault-Tolerant Computing, vol. 1, no. 1, pp. 62-83, Oct. 1976.

Peterson, J. L., Computation Sequence Sets," J. Computer and System Sciences, vol. 13, no. 1, pp. l-24, Aug. 1976.

vanCleemput, W. M., Computer-Aided Design of Digital Systems: A Bibliography, vol. 2: 1975-76 Update, Computer Science Press, Woodland Hills, California, Oct. 1976.

vanCleemput, W. M., "Mathematical Models for the Circuit Layout Problem," IEEE Trans. Circuits and Systems, vol. CAS-23, no. 12, pp. 759-767, Dec. 1976.

vanCleemput, W. M., "Hypergraph Models for the Circuit Layout Problem," Applied Mathematical Modelling, vol. 1, no. 3, pp. 160-161, Dec. 1976.

Wakerly, J. F., "Checked Binary Addition with Checksum Codes," J. Design Automation and Fault-Tolerant Computing, vol. 1, no. 1, Oct. 1976.

Wakerly, J. F. and E. J. McCluskey, "Microcomputers in the Computer Engineering Curriculum," Computer, vol. 10, no. 1, pp. 32-38, Jan. 1977.

Wakerly, J. F., "Microprocessor Input/Output Architecture," Computer, vol. 10, no. 2, pp. 26-33, Feb. 1977.

Wakerly, J. F., "Documentation Standards Clarify Design," Computer Design, vol. 16, no. 2, pp. 75-85, Feb. 1977.

Wiederhold, G., "Applications of Computers in Medicine," Encyclopedia of Computer Science (Ralston and Meek, Eds.), Petrocelli/Charter, New York, 1976, pp. 611-612, 873-879.

Wiederhold, G., Database Design, McGraw-Hill, New York, (Computer Science Series), May 1977.

CONFERENCE PROCEEDINGS

Beaudry, M. D., "Performance Related Reliability Measures for Computing Systems," Proc. Seventh Int. Conf. on Fault-Tolerant Computing, Los Angeles, California, June 28-30, 1977, pp. 16-21.

Blount, M. L., "Probabilistic Treatment of Diagnosis in Digital Systems," Proc. Seventh Int. Conf. on Fault-Tolerant Computing, Los Angeles, California, June 28-30, 1977, pp. 72-77.

Dalal, Y., "A Distributed Algorithm for Constructing Minimal Spanning Trees in Computer Communication Networks," Proc. Fifth Texas Conf. on Computing Systems, Austin, Texas, October 18-19, 1976.

Flynn, M. J., "Some Remarks on High Speed Computers," Dig. of Papers, 14th IEEE Computer Society Int. Conf., San Francisco, California, (COMPCON Spring 77) Feb. 28-March 3, 1977, pp. 18-20.

Flynn, M. J., "Computer Organization and Architecture," Lecture Notes on Advanced Operating Systems, Springer-Verlag, Munich, Germany (in press), Advanced Course in Operating Systems, Munich, Germany, July 28-Aug. 5, 1977.

Flynn, M. J., "The Interpretive Interface: Resources and Program Representation in Computer Organization," Proc. Symp. on High Speed Computers and Algorithm Organization, University of Illinois, Champaign, Illinois, April 13-15, 1977, pp. 41-69.

Karp, R. A. and D. C. Luckham, "Verification of Fairness in an Implementation of Monitors," Proc. Second Int. Conf. on Software Engineering, San Francisco, California, Oct. 13-15, 1976, pp. 40-46.

Kolupaev, S., "Cascade Structure in Self-Checking Networks," Proc. Seventh Int. Conf. on Fault-Tolerant Computing, Los Angeles, California, June 28-30, 1977, pp. 150-154.

Lee, R., "Performance Bounds in Parallel Processor Organizations," Proc. Symp. on High Speed Computers and Algorithm Organization, University of Illinois, Champaign, Illinois, April 13-15, 1977, pp. 453-455.

Losq, J., "The Effects of Failures on Gracefully Degradable Systems," Proc. Seventh Int. Conf. on Fault-Tolerant Computing, Los Angeles, California, June 28-30, 1977, pp. 29-34.

Losq, J., "Efficiency of Compace Testing for Sequential Circuits," Proc. Seventh Int. Conf. on Fault-Tolerant Computing, Los Angeles, California, June 28-30, 1977, pp. 168-174.

McCluskey, E. J. and R. C. Ogus, "Comparative Architecture of High Availability Computer Systems," Dig. of Papers, 14th IEEE Computer Society Int. Conf., San Francisco, California (COMPCON Spring 77), Feb. 28-March 3, 1977, pp. 288-293.

Phillips, J. V. and T. H. Bredt, "Design and Verification of Real-Time Systems," Proc. Second Int. Conf. on Software Engineering, San Francisco, California, Oct. 13-15, 1976.

Rodnick, J. and G. Wiederhold, "A Review of Ambulatory Medical Record Systems in the United States: Charting Services that are of Benefit to the Physicians," MEDINFO-77 Proceedings (Shires and Wulfs, Eds.), North Holland Pub. Co., 1977, pp. 957-961.

CONFERENCE PROCEEDINGS (continued)

Savir, J., "Optimal Random Testing of Single Intermittent Failures in Combinational Circuits," Proc. Seventh Int. Conf. on Fault-Tolerant Computing, Los Angeles, California, June 28-30, 1977, pp. 180-185.

Saxena, A. R. and T. H. Bredt, "Verification of a Monitor Specification," Proc. Second Int. Conf. on Software Engineering, San Francisco, California, Oct. 13-15, 1976.

Shedletsky, J. J., "Random Testing: Practicality vs. Verified Effectiveness," Proc. Seventh Int. Conf. on Fault-Tolerant Computing, Los Angeles, California, June 28-30, 1977, 168-174.

vanCleemput, W. M., "A Hierarchical Language for the Structural Description of Digital Systems," Proc. 14th Design Automation Conf., New Orleans, Louisiana, June 20-22, 1977, pp. 379-385.

vanCleemput, W. M., "A Digital Automation Course for Logic Designers," Proc. Int. Conf. on Computer Aided Design Education, Middlesbrough, Teesside, England, July 13-15, 1977.

Widdoes, L. C., "Architectural Considerations for General Purpose Multiprocessors," Dig. of Papers, 13th IEEE Computer Society Int. Conf., Washington, D.C. (COMPCON Fall 76), Sept. 8-10, 1976, pp. 251-254.

TECHNICAL REPORTS

Cornouter Reliability

Thompson, P. A., "A Simulator for the Evaluation of Digital System Reliability," Tech. Rpt. no. 119, August 1976.

Usas, A., "Error Management in Digital Computer Input/Output Systems," Tech. Rpt. no. 122, May 1976.

TECHNICAL REPORTS (continued)

Computer Architecture

Hoevel, L., "Structure of DEL's: A New Theory of Interpretive System Support," Tech. Rpt. no. 130, March 1977.

Iliffe, J. K., "Interpretive Machines," Tech. Rpt. no. 149, June 1977.

Lee, R., "Performance Brounds for Parallel Processors," Tech. Rpt. no. 125, November 1976.

Rau, B. R., "The Exact Analysis of Models of Program Reference Strings," Tech. Rpt. no. 124, December 1976.

Rau, B. R., "Sequential Prefetch Strategies for Instructions and Data," Tech. Rpt. no. 131, January 1977.

Rau, B. R., "Program Behavior and the Performance of Interleaved Memories," Tech. Rpt. no. 138, May 1977.

Rau, B. R., "Properties and Application of the Least-Recently-Used Model of Program Behavior," Tech. Rpt. no. 139, June 1977.

Yu, P. s., "On Accuracy Improvement and Applicability Conditions of Diffusion Approximation with Applications to Modelling of Computer Systems," Tech. Rpt. no. 129, January 1977.

Yu, P. s., "Passage Time Distributions for a Class of Queueing Networks: Closed, Open, or Mixed, with Different Classes of Customers with Applications to Computer System Modeling," Tech. Rpt. no. 135, March 1977.

Yu, P. s., "Performance Analysis of Computer Communication Network Via . Random Access Channels," Tech. Rpt. no. 137, April 1977.

TECHNICAL REPORTS (continued)

Design Automation

vanCleemput, W. M., "A Structural Design Language for Computer Aided Design of Digital Systems," Tech. Rpt. no. 136, April 1977.

vanCleemput, W. M., T. C. Bennett, J. A. Hupp, and K. R. Stevens, "SPRINT - An Interactive System for Printed Circuit Board Design User's Guide," Tech. Rpt. no. 143, June 1977.

Computer Networks

Dalal, Y., "Broadcast Protocols in Packet Switched Networks," Tech. Rpt. no. 128, April 1977.

Garcia-Molina, H. and G. Wiederhold, "Application of the Contract Net Protocol to Distributed Databases," Heuristic Programming Project Report HPP-77-21, Stanford University, April 1977.

Warren-Angelucci, W., "The Optimal Placement of Dynamic Recovery Checkpoints in Recoverable Computer Systems," Tech. Rpt. no. 126, December 1976.

Parallel Computer Systems

Owicki, S. S., "Specifications and Proofs for Abstract Data Types in Concurrent Programs," Tech. Rpt. no. 133, March 1977.

Computer Reliability

Beaudry, M. D., "Performance Related Reliability Measures for Computing Systems," Tech. Note no. 101, December 1976.

Losq, J., "Effects of Failures on Performance of Gracefully Degradable Systems," Tech. Note no. 103, December 1976.

Losq, J., "Efficiency of Compact Testing for Sequential Circuits," Tech. Note no. 104, December 1976.

McCluskey, E. J., "A Survey of Research at the Center for Reliable Computing," Tech. Note no. 96, October 1976.

Savir, J., "Optimal Random Testing of Single Intermittent Failures in Combinational Circuits," Tech. Note no. 105, November 1976.

Thompson, P. A., "A Simulator for the Evaluation of Reliability," Tech. Note no. 106, December 1976.

Verdillon, A., "Symmetry, Automorphism, and Test," Tech. Note no. 87, June 1976.

Computer Architecture

Flynn, M. J., "Studies in the Organization of Computer Systems - 1976 Progress Summary," Tech. Note no. 100, November 1976.

Hoevel, L. and W. A. Wallach, "Emulation Oriented Software First Development," Tech. Note no. 95, August 1976.

Hoevel, L., "DELtran Principles of Operation: A Directly Executed Language for FORTRAN- II," Tech. Note no. 108, March 1977.

Iliffe, J. K., "Computing in Store," Tech. Note no. 117, June 1977.

TECHNICAL NOTES (continued)

Computer Architecture (continued)

Neuhauser, C., "An EMMY Based PDP-11/20 Emulation," Tech. Note no. 110, March 1977.

Neuhauser, C., "DEBUG User's Guide (Version E)," Tech. Note no. 113, March 1977.

Neuhauser, C., "EMMY System Processor - Principles of Operation," Tech. Note no. 114, May 1977.

Roush, E. T., "An EMMY Based Emulation of the CDC 6000 Series CPU," Tech. Note no. 120, July 1977.

Shih, M., "EMMY/UNIBUS Interface Design Specification," Tech. Note no. 109, January 1977.

Design Automation

Rau, B. R., "An Experiment in Wire Routing," Tech. Note no. 98, November 1976.

vanCleemput, W. M., "Topological Circuit Layout," Tech. Note no. 99, October 1976.

vanCleemput, W. M., "On the Planarity of Hypergraphs," Tech. Note no. 115, June 1977.

vanCleemput, W. M., "An Algorithm for Testing the Planarity of Partially Oriented Graphs," Tech. Note no. 116, June 1977.

Computer Networks

Crane, R., "Bell 303 Modem Replacement, Tech. Note no. 91, August 1976.

PAPERS ACCEPTED FOR PUBLICATION

McCluskey, E. J., K. P. Parker, and J. J. Shedletsky, "Boolean Network Probabilities and Network Design," IEEE Trans. on Computers, to appear Spring 1978.

Parker, K. P. and E. J. McCluskey, "Sequential Circuit Output Probabilities from Regular Expressions," IEEE Trans. on Computers, to appear Spring 1978.

Wakerly, J. F., Error-Detecting Codes, Self-Checking Circuits, and Applications, to be published by American Elsevier, in preparation.

Wiederhold, G. and I. Kuhn, "Effective Services of Automated Ambulatory Medical Record Systems," Int. J. on Policy Analysis and Information Systems, to appear January 1978.

Wiederhold, G. and R. El-Masri, "A Structured Model for Database Systems," to be published in DATABASES: Improving Usability and Responsiveness, Academic Press, New York, 1978.

PAPERS ACCEPTED FOR PRESENTATION

Owicki, S. S., "Verifying Concurrent Programs with Shared Data Classes," Working Conference on Formal Description of Programming Concepts, New Brunswick, Canada, August 1-5, 1977.

vanCleemput, W. M., "An Algorithm for Testing the Planarity of Partially Oriented Graphs," Midwest Symposium on Circuits and Systems, Lubbock, Texas, August 15-19, 1977.

Yu, P. and M. J. Flynn, "Performance Analysis of Distributed Computer Systems," Sixth Texas Conference on Computing Systems, Austin, Texas, November 14-14, 1977.