

AD-A267 370



FINAL/30 SEP 91 TO 29 SEP 92

FAULT-TOLERANT ARCHITECTURES FOR
MULTIPROCESSOR & VLSI-BASED SYSTEMS (U)

Professor Dhiraj K. Pradhan

University of Massachusetts
Electrical/Computer Engineering
Amherst MA 01003

DTIC
ELECTE
AUG 4 1993
S c D

2304/A2
AFOSR-91-0403

AFOSR-91-0403 0561

AFOSR/NM
110 DUNCAN AVE, SUITE B115
BOLLING AFB DC 20332-0001

AFOSR-91-0403

12a DISTRIBUTION STATEMENT

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

12b DISTRIBUTION CODE

UL

The research investigated under this grant: (1) a virtual checkpointing scheme for recovery, (2) schemes for implementing reliable memory, (3) roll-forward recovery schemes for duplex systems, (4) REACT, a tool for reliable architecture characterization and its application to reliability evaluation of TMR systems, (5) a new approach for low-cost system level diagnosis and (6) the reliability safety trade-off in modular redundant systems.

93 8 3 002

93-17271



17 UNCLASSIFIED	18 UNCLASSIFIED	19 UNCLASSIFIED	20 UNCLASSIFIED
21 UNCLASSIFIED	22 UNCLASSIFIED	23 UNCLASSIFIED	24 UNCLASSIFIED
25 UNCLASSIFIED	26 UNCLASSIFIED	27 UNCLASSIFIED	28 UNCLASSIFIED
29 UNCLASSIFIED	30 UNCLASSIFIED	31 UNCLASSIFIED	32 UNCLASSIFIED
33 UNCLASSIFIED	34 UNCLASSIFIED	35 UNCLASSIFIED	36 UNCLASSIFIED
37 UNCLASSIFIED	38 UNCLASSIFIED	39 UNCLASSIFIED	40 UNCLASSIFIED
41 UNCLASSIFIED	42 UNCLASSIFIED	43 UNCLASSIFIED	44 UNCLASSIFIED
45 UNCLASSIFIED	46 UNCLASSIFIED	47 UNCLASSIFIED	48 UNCLASSIFIED
49 UNCLASSIFIED	50 UNCLASSIFIED	51 UNCLASSIFIED	52 UNCLASSIFIED
53 UNCLASSIFIED	54 UNCLASSIFIED	55 UNCLASSIFIED	56 UNCLASSIFIED
57 UNCLASSIFIED	58 UNCLASSIFIED	59 UNCLASSIFIED	60 UNCLASSIFIED
61 UNCLASSIFIED	62 UNCLASSIFIED	63 UNCLASSIFIED	64 UNCLASSIFIED
65 UNCLASSIFIED	66 UNCLASSIFIED	67 UNCLASSIFIED	68 UNCLASSIFIED
69 UNCLASSIFIED	70 UNCLASSIFIED	71 UNCLASSIFIED	72 UNCLASSIFIED
73 UNCLASSIFIED	74 UNCLASSIFIED	75 UNCLASSIFIED	76 UNCLASSIFIED
77 UNCLASSIFIED	78 UNCLASSIFIED	79 UNCLASSIFIED	80 UNCLASSIFIED
81 UNCLASSIFIED	82 UNCLASSIFIED	83 UNCLASSIFIED	84 UNCLASSIFIED
85 UNCLASSIFIED	86 UNCLASSIFIED	87 UNCLASSIFIED	88 UNCLASSIFIED
89 UNCLASSIFIED	90 UNCLASSIFIED	91 UNCLASSIFIED	92 UNCLASSIFIED
93 UNCLASSIFIED	94 UNCLASSIFIED	95 UNCLASSIFIED	96 UNCLASSIFIED
97 UNCLASSIFIED	98 UNCLASSIFIED	99 UNCLASSIFIED	100 UNCLASSIFIED

AFOSR 91-0403

Final Report for

**Fault-Tolerant Architectures for
Multiprocessor
and VLSI-Based Systems**

Dhiraj K. Pradhan
Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003

DTIC QUALITY INSPECTED 3

Accession For	
DTIC (CPA&I)	☒
DTIC (A&I)	☐
DTIC (S&I)	☐
Justification	
By	
Distribution	
Availability Codes	
Dist	Avail and/or Special
A-1	

Contents

1 Virtual Checkpointing	1
2 Reliable Memory Design	4
3 Roll-forward Checkpointing Schemes	6
4 Synthesis and Evaluation of Alternative Fault-Tolerant Architectures	8
4.1 The Reliable Architecture Characterization Tool	9
4.2 Reliability Analysis of Unidirectional Voting TMR Systems	11
5 Safe System Level Diagnosis	15
6 Safe Modular Redundant Systems	16
7 Papers Published Under AFOSR Grant 91-0403 and References	17
8 Curriculum Vitae	20
9 Biography	34

This report summarizes the research performed under AFOSR grant 91-0403. Section 1 presents a virtual checkpointing scheme for recovery. Section 2 presents schemes for implementing reliable memory. Roll-forward recovery schemes for duplex systems are discussed in Section 3. Section 4 discusses REACT, a tool for reliable architecture characterization and its application to reliability evaluation of TMR systems. Section 5 discusses a new approach for low-cost system level diagnosis. Section 6 presents results on the reliability-safety trade-off in modular redundant systems.

1 Virtual Checkpointing

Virtual Checkpoints combines concepts from two database recovery techniques of shadow paging and twin paging to support checkpoint and rollback recovery in the virtual memory translation hardware [1, 5, 6]. The concept of supporting the active data is implemented by dynamically allowing a second copy of the virtual page. The active pages can be identified by the use of a checkpoint counter associated with each page. In addition to detecting active pages in a rollback situation, the counters also allow the checkpoint processing to be deferred past the exact instance of the checkpoint (assuming a fault tolerant memory).

The technique supports two classes of data within the virtual memory system (i.e., active and checkpoint). Each class still supports the traditional two level store of virtual memory (i.e., real memory and paging disk). Similar to the other schemes, virtual checkpoints must be able to detect all active pages and make all active pages permanent at the checkpoint time. This is achieved by having a global checkpoint counter (V) covering all the data and local checkpoint counters (v) for individual pages. Essentially, the global checkpoint counter is copied to the local counter on every reference (note: this is the logical description and does not actually occur on every reference). Thus, active data is the pages with the local counter v equal to the global counter V . The global counter V is incremented when a checkpoint is taken. Thus, all active pages become checkpoint versions when the global counter V is incremented. Figure 1 illustrates the basic concepts. Virtual page k has not been referenced since the prior checkpoint. Page j has been accessed in the current interval and has both an active and checkpoint version. Note that in all cases the mapping

refers to both a real storage frame and a disk slot.

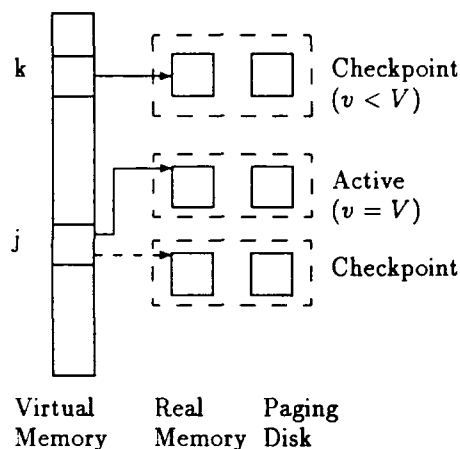


Figure 1: Virtual Checkpointing: Basic concept

The mappings for each virtual page are replicated and are referred to as m_0 and m_1 . A mapping m_i contains mappings for the real frame (r_i) and the disk copy (d_i). Each virtual page has a one bit field, l , which can be thought of as a switch that points to the most recently used mapping (i.e., m_0 or m_1). Thus, the notation m_l refers to the mapping that was used last. In addition, each page has a k -bit local checkpoint number (v) which contains a copy of the global checkpoint number (V) during the most recent reference. The checkpoint number, V , is a global value which is incremented on every checkpoint (the scope of V determines the scope of the checkpoint, e.g., the entire system, a single address space or portions of an address space).

An important aspect of the scheme is that the actions of taking a checkpoint are not concentrated at the actual time of the checkpoint but rather are distributed over the time following the checkpoint. This is because, under the assumption of fault tolerant memory, the only action required to perform a checkpoint is to increment the global checkpoint counter V . The processing for the individual pages is deferred until the first reference to the page after the checkpoint. In order to determine whether the deferred processing must occur,

the values V and v must be compared on every reference (using the translation look-aside buffer with the scheme avoids having to actually make this comparison on every reference). Thus, when a page is referenced it is either the case where the checkpoint processing must occur ($v \neq V$) or a normal access to the active page ($v = V$). Figure 2 shows a situation where checkpoints were taken at times t_{c1} and t_{c2} . Consider the events at time t_{R2} . The active page addressed by m_1 ($l = 1$) was last referenced at time t_{R1} (thus $v = 1$). The reference at time t_{R2} is the first reference after the checkpoint (because $v \neq V$) and the contents of page m_1 must be preserved as the checkpoint page. Furthermore, the contents of page m_1 must be used while the resources of the old checkpoint page m_0 (whose contents are no longer required) are used. Once the valid data has been copied to m_0 , the l -bit is inverted (to $l = 0$) so that m_0 becomes the active and m_1 becomes the checkpoint. Finally, the global checkpoint number V is copied to the local checkpoint number for this page so that on the next access in the checkpoint interval a normal translation occurs. Figure 3

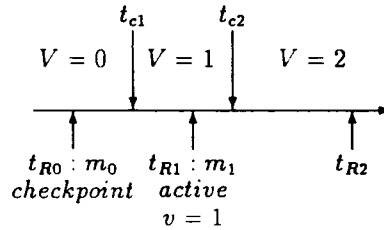


Figure 2: Case 1- first reference after checkpoint.

shows the situation at the next reference in this checkpoint interval. A reference at time t_{R3} proceeds normally to the active data at m_0 because V matches v .

A rollback requires discarding any data that has been modified since the prior checkpoint. If the page has not yet been referenced since the prior checkpoint then the page is essentially in a rolled back state and nothing needs to be done (e.g., Case 1 in Figure 2). If the page has been referenced since the prior checkpoint then there is an active page that must be discarded. For example, if a failure occurs at time t_{R3} in Figure 3, one wants to discard m_0 and restore m_1 . So for all pages with $V = v$, the v value is decremented and the

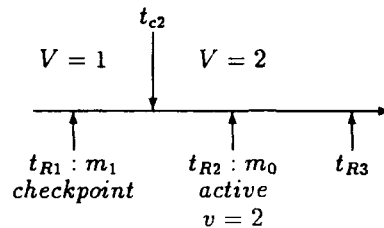


Figure 3: Case 2- page previously referenced.

1 bit is inverted. This forces the state to be like Figure 2 where m_1 contains the checkpoint and m_0 contains useless information.

2 Reliable Memory Design

The use of a hybrid memory structure consisting of both highly reliable and normal memory can further support persistent and recoverable memory [1]. Hybrid algorithms that manage the writable memory and read-only memory separately are proposed. The traditional measures of virtual memory algorithms (i.e., lifetime and space-time) have been extended to account for the dual nature of the policies. Several properties of the policies have been explored. It has been shown that the knee of a hybrid lifetime curve produces a near minimum space-time product as with the existing algorithms. Hybrid policies are more controllable with respect to highly reliable memory because they can constrain the amount of writable memory and gain performance by using additional read-only memory. The lifetime measure for the hybrid policies under constrained writable memory, when compared at equal amounts of highly reliable memory, is better than the single policy algorithm at a small cost of additional read-only memory. Furthermore, even at an unconstrained amount of writable memory, the hybrid policy produces approximately equal performance while the writable memory can be completely fixed in size. Theoretical results are also derived for a property which indicates the optimal performance for a hybrid reference stream based on two individual streams.

The ability to accurately predict the reliability of a system is very important. Two novel techniques have been developed which focus on dynamic aspects of memory [2, 3, 4, 7]. The first focuses on the memory reference patterns of a particular program while the second looks at memory behavior due to memory management actions.

The first novel technique evaluates the probability of correct execution of a program based on the program's memory access behavior. The approach is an analytical study using an existing model which characterizes an address trace with four parameters. Three cases are developed based on the storage allocation policy (i.e., pre-allocated, dynamically allocated, or constrained in allocation). The models are able to compare the traditional view that is taken in standard memory reliability analysis to that of a real world environment where a program uses a varying fraction of the memory at different instances. Using these models, it is shown that the reliability may be significantly better than the apparent reliability when the program behavior was not considered. It provides one explanation for the cause of unobserved faults along with an analytical basis for determining the extent of faults not being observed. Possibly the most important application of these models is to analytically quantify the observed phenomenon that failure rates increase with increased workload. A new explanation has been proposed for this phenomenon based on the notion that programs often have storage allocated which will never be referenced again and cannot cause a failure. Assuming a constant fault rate over increased workloads, the model shows that there could be a significant increase in observed failures. The model was validated with actual program traces and shown to be very accurate. Finally, several techniques have been shown for extracting the fractal parameters of a program trace.

The second novel technique for reliability analysis uses the memory space allocated to more accurately calculate the reliability. This can be used to understand the relationship between the amount of memory allocated and the reliability. This effect has been quantified based on the relative cost of a fault. Distinct effects have been measured depending on the relative speed of the paging device. For small reload times it is found that a decrease in the memory partition size leads to an increase in reliability at the cost of additional instruction overhead. For extremely long reload times it is found that larger amounts of memory lead to increased reliability. There also exists a middle reload time where the

optimal reliability corresponds to the optimal space-time performance. Other aspects of virtual memory algorithms such as small pages and different paging algorithms were studied. Furthermore, the methodology was applied to study the reliability of cache memories which have the characteristic of very small reload delays. The results show that the reliability improvement factor can change by several orders of magnitude based on the cache size. For small memory sizes it was found that a very small number of page durations contribute to a majority of the total unreliability. Two techniques have been suggested to remove these long durations, which then lead to even greater improvements in the reliability. One is an algorithm called *selective scrubbing* to break the long durations, which could either be implemented in software or hardware. A second technique showed that the addition of very small amounts of highly reliable memory can also lead to significant reliability improvements.

3 Roll-forward Checkpointing Schemes

A fault-tolerant multiprocessor environment wherein each task is executed simultaneously on two processing modules is considered. A pool of a small number of nondedicated spares or processing modules with spare processing capacity is assumed available (see Figure 4). Duplex fault-tolerant architectures that require no rollback for most faults are proposed.

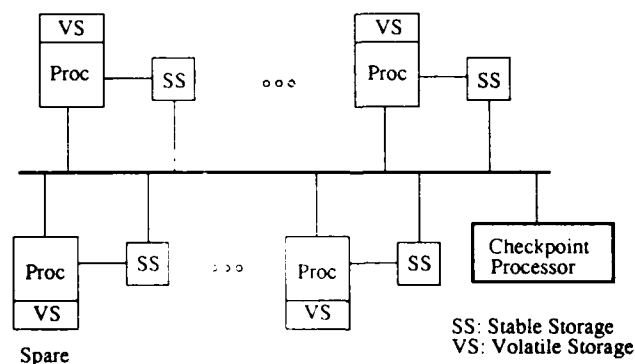
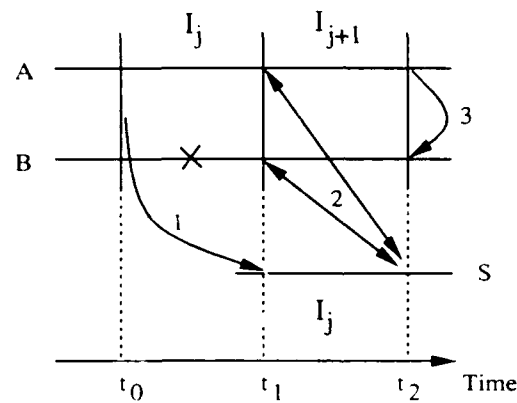


Figure 4: System architecture for roll-forward checkpointing schemes

In the proposed schemes, at each checkpoint the state of the two modules executing

the task is compared for detection of faults. If a fault is detected, instead of usual rollback, the following mechanism is used for identification of the faulty processing module [13, 14, 17]. The good state of the previous checkpoint is loaded into a spare module. The checkpoint interval in which the failure is detected is then "retried" on the spare module. Concurrently, the task continues execution on both processing modules in the duplex system. At the next checkpoint the state of the spare is compared with the state of the two processing modules at the previous checkpoint where disagreement occurred. This allows for the identification of the faulty module (see Figure 5). Once the faulty module is identified, the state of the faulty module is made consistent with the state of the fault-free module in the duplex system and the spare is released to the pool.



- 1 : Copy state to the spare
- 2 : Compare state of the spare with the state of A and B
- 3 : Copy state from A to B
- X : A fault

Figure 5: Roll-forward checkpointing scheme

These schemes are termed as Roll-Forward Checkpointing Schemes (RFCS). The proposed RFCS schemes provide a mechanism for identifying the faulty processing module and recovering it, in most cases, without the overhead of rollback. It is demonstrated that the proposed schemes have potential performance advantages over conventional duplex system

with rollback.

Specifically, the advantage of the proposed schemes is that they achieve a lower average execution time with a lower variance as compared to the rollback schemes. This is crucial for real-time systems with hard deadlines as lower variance enhances the predictability of the task completion time.

4 Synthesis and Evaluation of Alternative Fault-Tolerant Architectures

Another direction of our research was the study of alternative fault-tolerant computer systems. Our continuing goal is to synthesize and evaluate novel architectures which offer increased performance and/or require less hardware than traditional designs while providing nearly the same dependability. We are specifically interested in the class of architectures which can be represented by the generalized system model pictured in Figure 6. This multiprocessor abstraction consists of multiple, possibly redundant, processor (P) and memory (M) modules interconnected through some form of error control logic (such as voters, comparators, switches or error correcting codes). A wide variety of highly dependable architectures fit this model: static, dynamic and hybrid redundancy, systems with coding plus many non-fault-tolerant multiprocessors.

Reliability and availability are the metrics used to judge the efficacy of the performance/redundancy tradeoffs being investigated. Many hardware and software attributes influence the dependability of a system, including specific fault characteristics, error containment ability and variations in workload. We are particularly concerned with the effect program behavior has on reliability. Detailed system models which account for these factors are often very difficult to formulate through analytical techniques (such as combinatorial and Markov models) which are commonly used for dependability assessment. In order to facilitate our research, we have developed a simulated fault-injection testbed called REACT to experimentally analyze the dependability of these new computer architectures.

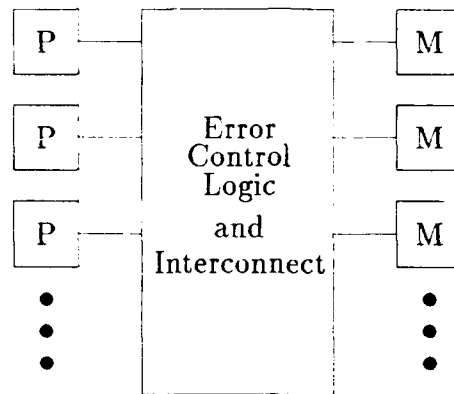


Figure 6: Generalized System Model

4.1 The Reliable Architecture Characterization Tool

The Reliable Architecture Characterization Tool (REACT) is a software testbed which performs automated life testing of many user-defined multiprocessor systems through simulated fault-injection [9, 12]. This involves emulating the high-level hardware and software components of a given system while concurrently injecting bit-level faults and errors into it. During a single *simulation run*, the code conducts a certain number of experiments or *trials* in which an initially fault-free system is operated until it fails or reaches a specified censoring time. The exact number of trials required is determined by the desired confidence intervals about the system dependability attribute being investigated. Extensive instrumentation has been included in the program in order to collect data from each trial which is later aggregated over the entire simulation run in order to generate the outputs. Graphs of reliability and availability, a comprehensive failure mode report and various statistical measurements are provided as output by the software. REACT consists of 8000 lines of C running under UNIX and completes a "typical" simulation run in less than 10 hours on a dedicated DECstation 5000/120.

REACT can analyze the class of architectures which was shown previously in Figure 6. Any number of processor and memory modules may be specified and each can be designated

as initially active or a hot or cold standby spare. *Groups* of processors or memories may also be defined in which all modules operate redundantly. The error control logic may be built from various combinations of components commonly found in fault-tolerant designs. Custom error control logic circuitry may also be specified by the user. Processors are simulated at the functional-level whereas a logical-level description is used for the memory modules and error control logic. Logic values 0 and 1 are not differentiated in the system model: only error-free and erroneous states exist for each bit. Memory depth is variable and a 16-bit word width for memory and all data paths has currently been implemented. Other word sizes may be realized with minor modifications in the code.

A synthetic workload is assumed in which processors continually perform *computation cycles* consisting of an instruction fetch, a possible operand read, a computation and a possible result write. Real code and data are not used by REACT, but errors are allowed to propagate throughout the system as if the application program was actually being executed. Behavior of the application workload is specified by a mean instruction execution rate, the probabilities of performing a data read and write per instruction plus a locality of reference model. Values for the mean number of data accesses made during the execution of an instruction may be obtained either through trace analysis or directly from the measurement of operational hardware. It is assumed that all memory references access one whole word. Which memory locations are accessed during a computation cycle are determined via the locality of reference model. The testbed implements a model based on Bradford-Zipf distributions which suggests that α % of all accesses go to β % of the memory under the condition $\alpha + \beta = 1$. Reference addresses are assumed to be uniformly distributed inside and outside of the locality and no attempt is made to separate code from data in memory with the model.

The fault/error model employed by REACT accounts for permanent, intermittent and transient faults in the processors plus permanent and transient faults in the memories as well as the error control logic. Faults with a Weibull distribution (of which the exponential distribution is a subset) for their inter-arrival times are injected into these modules only at the beginning of a computation cycle. Faults are assumed to always cause immediate errors, so their fault (but not error) latency is 0. Correlated failures are presently not considered.

Processor fault effects are assumed to be completely characterized by the rate at which errors appear on its memory bus. Three types of errors exist: transients lasting only one computation cycle, intermittents with a Weibull distributed duration and permanents which have an effect in every computation cycle. Errors may affect either addresses, (write) data or both addresses and data simultaneously. An erroneous address is assumed to access a random memory location while erroneous data take a random value. In addition, erroneous processor reads generate output errors in the same computation cycle.

Memory faults are divided among the bit-array and addressing-logic regions of a memory module. The fraction of faults which fall into each of these regions may be approximated by their relative chip areas. Bit-array faults are assumed to affect a single random bit in a word at a random address while a random location is referenced during an addressing-logic fault. A transient bit-array fault may be overwritten (changing it from the erroneous to error-free state) at any time, but a permanent can never be overwritten. Addressing-logic transients last one computation cycle and permanents will cause the memory module to endlessly access random words. An access to a random address reads or writes a value with randomly corrupted bits, representing the difference between the bit values of the word that was accessed and the word that should have been accessed. Finally, faults within one of the error control logic components are assumed to affect a single random bit either permanently or for one computation cycle in the case of transients.

4.2 Reliability Analysis of Unidirectional Voting TMR Systems

Computer systems used in aircraft and reactor control often require critically high reliability for moderately short mission times. Triple-modular redundant (TMR) hardware has been employed in many of these ultrahigh reliability applications. The three redundant processors of a TMR system concurrently execute identical tasks while the triplicated memories contain the same code and data. Majority voting is used to mask erroneous module outputs. As seen in Figure 7, the voter (V) is usually inserted into the redundant system buses between the processors (P) and memories (M). Bit-wise voting is typically performed on data, address and control lines during both read and write accesses to memory. Such a system will be

referred to as *bidirectional voting* (BDV) TMR.

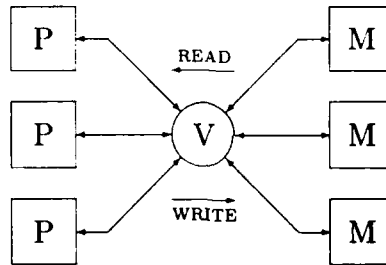


Figure 7: TMR System with Bidirectional Voting

Voting has a substantial performance penalty associated with it. This degradation can be attributed to two specific delays [8]. The **propagation delay** of signals through the voter logic is the more obvious contributor to increased memory access times. Less apparent is the **synchronization delay** incurred when clock skew requires modules to wait for a lagging signal before performing a vote. This penalty becomes even greater if a module fails in such a way that it does not respond, forcing a timeout period to be suffered on each memory reference. TMR systems used in hard real-time applications may not be able to tolerate the ensuing drop in throughput after this type of failure.

It is possible to significantly reduce the performance degradation of a BDV system by **voting only on one type of memory access**, either reads or writes. These *unidirectional voting* systems are expected to have lower reliability than the bidirectional design since a smaller fraction of errors will be masked, possibly allowing them to propagate and corrupt the state of non-faulty modules.

Because the voter may be by-passed on either memory read or write accesses to achieve higher performance, two different unidirectional voting systems exist. The *Read-*

Only Voting (ROV) TMR system removes the voting delays from the bus cycle on writes. Processor generated read addresses and memory outputs are voted upon and a single voted value is distributed to all three processors. Processor outputs are written straight into the corresponding memories without any error masking. The ROV TMR system therefore allows processor errors to propagate into the memories while all single errors from memory will be contained by the voter.

The dual of the ROV system is the *Write-Only Voting* (WOV) TMR system which eliminates the delay associated with voting on read accesses. It performs a vote only at the outputs of the processors and writes a single voted value into all three memories at a voted address. No masking of data or addressing errors takes place during reads, so erroneous memory outputs may propagate directly into the associated processors. Voting terminates any single processor error before it reaches the memories.

Both unidirectional voting TMR systems can realize better performance than the traditional bidirectional voting system. However, **WOV should have better performance than ROV** because it suffers the delays of voting less often since reads generally occur much more frequently than writes. In terms of fault-tolerance, one might **expect ROV to provide higher reliability than WO**V for similar reasons. When processors and memories experience faults at the same rate, the percentage of potentially fatal errors that will get masked will be larger with the ROV system. In addition, memory often has a higher fault rate than processors so the percentage of errors masked will be even greater when the voter is placed at the output of the less reliable component.

Two parametric analyses of the bidirectional and unidirectional voting TMR systems were carried out with REACT [9, 11]. Figure 8 shows a typical reliability plot from this investigation. The following observations were made:

- the tradeoff of reliability for performance made by the unidirectional voting systems becomes more effective as the difference between processor and memory module failure rates increases
- near ideal tradeoffs can be attained for some failure rate combinations, particularly when memory is more likely to fail than the processors

- the analytical model traditionally used to predict the reliability of TMR designs is indicative of some of the differences between the bidirectional and unidirectional voting systems, but is not always accurate
- reliability of the ROV system is generally better than the WOV system, except when processor failure rates are high relative to the memory failure rates
- system failure is caused by propagation of errors more often in the WOV system than in the ROV system
- workload has limited effects on reliability when memory error latency is low

Results demonstrated that in many cases, acceptably little reliability was sacrificed by the unidirectional voting TMR systems for a potentially large increase in performance.

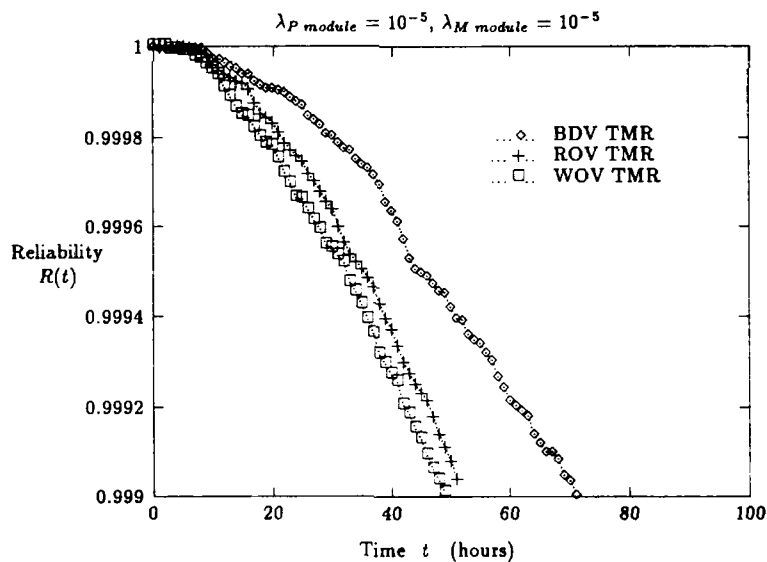


Figure 8: Example Reliability Plot from Analysis with REACT

5 Safe System Level Diagnosis

System level diagnosis has until now, focussed on location of faulty nodes in a system. A novel low-cost approach termed safe diagnosis has been developed. Diagnosing a large number of faulty nodes requires a large number of diagnostic tests. The proposed diagnosis approach alleviates the high cost of system level diagnosis by reducing the number of tests carried out periodically. By combining fault location with fault detection, this approach achieves high levels of diagnostic safety and recoverability. Systems which can guarantee correct diagnosis of up to t faults, and fault detection up to u faults, $u > t$, have been analyzed [15].

Systems that can perform safe system level diagnosis in the presence of permanent as well as intermittent faults have been characterized. The complexity of safe diagnosis algorithms is shown to be comparable with the diagnosis algorithms for systems performing only fault location. When only permanent faults are present, achieving a large fault detection capability in addition to an existing fault location capability requires only minimal additional test overhead. The testing overhead for intermittent fault detection is larger compared to permanent fault detection.

An adaptive diagnosis algorithm that performs fault location and detection is proposed for the permanent fault case. Compared to any adaptive algorithm for pure fault location, our algorithm requires just one additional test in the worst case. A distributed algorithm is also proposed for safe diagnosis of distributed multiprocessor systems. Repair of distributed systems requires that an external user be able to decide the status of all the system nodes or detect a fault situation beyond the fault location capability of the system. Algorithms for such user diagnosis of a distributed system have been developed.

The concept of safe diagnosis can be used for adaptive t -diagnosis on any t -diagnosable system, not necessarily with the traditionally-used fully connected testing graph. An adaptive algorithm for t -diagnosis on t -diagnosable testing graphs has been designed.

From the results obtained under the AFOSR grant, it is clear that the safe diagnosis approach results in low-cost. Thus, the proposed safe diagnosis approach is of significant interest from a practical viewpoint.

6 Safe Modular Redundant Systems

Dependability considerations warrant that in addition to reliability, a dependable system must have a high level of safety. Therefore, there is a need to ensure operation which is both error-free under adverse conditions, as well as safe under severely adverse conditions.

We have analyzed a technique for implementing systems requiring high reliability and safety [16]. These systems, named n -Safe modular redundant (n SMR) systems, achieve high reliability and safety using module replication and redundancy in module output.

An n SMR system consists of n identical modules and an arbiter. The arbiter uses outputs of all the n modules to decide the n SMR system output. Reliability and safety of the system are a function of the arbitration strategy used. When reliability is the only criterion, an optimal arbitration strategy that maximizes the reliability can be designed. With reliability and safety both of concern, usually no single arbitration strategy is optimal. We have presented an implementation of *maximal* arbitration strategies which achieve different maximal reliability and safety combinations. Maximal arbitration strategies are such that no arbitration strategy has better reliability and safety, compared to a maximal strategy.

The effect of increasing redundancy on the achievable reliability and safety has been analyzed for systems with and without redundant module outputs. Detailed results on binary SMR systems using binary arbiters have also been obtained. The results of this chapter are summarized below.

- It is shown that for modules without output redundancy, no arbitration strategy exists for $(n + 1)$ SMR which achieves better reliability and safety compared to certain arbitration strategies for n SMR. Further, given any arbitration strategy for n SMR, there always exists an arbitration strategy for $(n + 2)$ SMR that achieves higher reliability and safety.
- It is shown that if modules have output redundancy, given an arbitration strategy for n SMR, one can always find an arbitration strategy for $(n + 1)$ SMR that achieves better reliability and safety.

- A detailed analysis of binary n SMR systems with single bit output is presented. Whether binary $(n + 1)$ SMR dominates binary n SMR is shown to be dependent on the relation between the likelihood of a detected error (p_d) and the likelihood of an undetected error (p_u) in a binary module's output. It is shown that when $p_d = p_u$, binary $(n + 1)$ SMR does not dominate any of the plurality strategies for binary n SMR. Also, exact expressions for the reliability and safety of the maximal strategies for such systems have been presented.
- Design of a family of threshold-based maximal arbitration strategies which achieve different reliability and safety is presented. Design of a class of arbitration strategies easier to implement as compared to the threshold-based arbitration strategies is also presented. These arbitration strategies are obtained by generalizing the plurality strategies.

7 Papers Published Under AFOSR Grant 91-0403 and References

1. N. S. Bowen, *Fault-tolerant aspects of memory design*. PhD thesis, University of Massachusetts-Amherst, February 1992.
2. N. S. Bowen and D. K. Pradhan, "Program fault tolerance based on memory access behavior," in *21st Symp. on Fault-Tolerant Computing*, pp. 426-433, IEEE, June 1991.
3. N. S. Bowen and D. K. Pradhan, "Effect of memory management on reliability," Tech. Rep. TR-91-CSE-3, University of Massachusetts, Feb. 1991.
4. N. S. Bowen and D. K. Pradhan, "Reliability aspects of memory management policies," Tech. Rep. TR-91-CSE-16, University of Massachusetts, July 1991.
5. N. S. Bowen and D. K. Pradhan, "A virtual memory translation mechanism to support checkpoint and rollback recovery," in *Supercomputing '91*, pp. 890-899, Nov. 1991.

6. N. S. Bowen and D. K. Pradhan, "Virtual checkpoints: Architecture and performance," *IEEE Transactions on Computers*, May 1992.
7. N. S. Bowen and D. K. Pradhan, "Issues in fault tolerant memory management," Tech. Rep. TR-91-CSE-20, University of Massachusetts, Aug. 1991.
8. J. A. Clark and D. K. Pradhan, "Unidirectional voting TMR systems," Tech. Rep. TR-91-CSE-6, University of Massachusetts, Apr. 1991.
9. J. A. Clark and D. K. Pradhan, "REACT - the reliable architecture characterization tool," Tech. Rep. TR-92-CSE-22, University of Massachusetts, June 1992.
10. J. A. Clark and D. K. Pradhan, "Reliability analysis of unidirectional voting TMR systems through simulated fault-injection," in *Digest of Papers for the 1992 Workshop on Fault-Tolerant Parallel and Distributed Systems*, pp. 72-81, IEEE, July 1992.
11. J. A. Clark and D. K. Pradhan, "Reliability analysis of unidirectional voting TMR systems through simulated fault-injection," Tech. Rep. TR-92-CSE-9, University of Massachusetts, Mar. 1992.
12. J. A. Clark and D. K. Pradhan, "REACT - a synthesis and evaluation tool for fault-tolerant multiprocessor architectures," to appear in the *Annual Reliability and Maintainability Symposium*, Jan. 1993.
13. D. K. Pradhan and N. H. Vaidya, "Roll-forward checkpointing scheme: Concurrent retry with nondedicated spares," in *IEEE Workshop on Fault Tolerant Parallel and Distributed Systems*, July 1992.
14. D. K. Pradhan and N. H. Vaidya, "New roll-forward checkpointing schemes for modular redundant systems," in *Hardware and Software Fault Tolerance in Parallel Computing Systems* (D. R. Avresky, ed.), England: Ellis Horwood, 1992.
15. N. H. Vaidya and D. K. Pradhan, "System Level Diagnosis: Combining Detection and Location," in *21st Int. Symp. Fault Tolerant Comp.*, June 1991.

16. N. H. Vaidya and D. K. Pradhan, "Voting in fault-tolerant systems: Reliability and safety issues," Tech. Rep. TR-91-CSE-7, ECE Department, Univ. of Massachusetts, June 1991.
17. N. H. Vaidya, *Low-Cost Schemes for Fault Tolerance*. PhD thesis, University of Massachusetts-Amherst, August 1992.

8 Curriculum Vitae

Dhiraj K. Pradhan
H. R. Bright Building
Texas A&M University
College Station, TX 77843
Tel: (409) 862-2438
Fax: (409) 847-8578
Email: pradhan@cs.tamu.edu

Positions—Academic

- 1992 – present Endowed Chair Professor, Department of Computer Science,
Texas A&M University, College Station, Texas.
- 1983 – 1992 Professor and Coordinator of Computer Systems Engineering,
Department of Electrical and Computer Engineering,
University of Massachusetts, Amherst, Massachusetts.
- 1978 – 1982 Associate Professor, School of Engineering,
Oakland University, Rochester, Michigan.
- 1979 Research Associate Professor, Stanford University,
Stanford, California.
- 1973 – 1978 Associate Professor, Department of Computer Science,
University of Regina, Regina, Canada. (1973–1976,
Assistant Professor).

Positions—Industrial

- 1972 – 1973 Staff Engineer, IBM, Systems Development Laboratory,
Poughkeepsie, New York.

Honors

- 1990 *Humboldt Distinguished Senior Scientist Award, Germany*
- 1989 *Fellow, Japan Society for Promotion of Science*
- 1988 *Fellow, IEEE, "For contributions to techniques and
theory of designing fault-tolerant circuits and systems"*

Education

1972, Ph.D. (Electrical Engineering), University of Iowa
Iowa City, Iowa.

1970, M.S. (Electrical Engineering), Brown University,
Providence, Rhode Island.

Personal

Born on December 1, 1948, Married, Five Children, U.S. Citizen

Professional Activities

- 1992 - *Conference Chair*, 22nd International Symposium on Fault-Tolerant Computing, Boston, Massachusetts
- 1992 - *Program Chair*, 10th IEEE VLSI Test Symposium
- 1991 - *Editor*, *IEEE Transactions on Computers*
- 1990 - 1992 *ACM Lecturer*
- 1990 - 1993 *IEEE Distinguished Visitor*, Computer Society
- 1990 - *Editor*, *IEEE Computer Society Press*
- 1990 - *Keynote Speaker*, International Symposium on Fault-Tolerant Systems and Diagnostics, Varna, Bulgaria
- 1989 - *Associate Editor*, *Journal of Circuits, Systems and Computers*
World Scientific Publishing Co., New Jersey
- 1988 - *Editor*, *Journal of Electronic Testing, Theory and Applications*
Kluwer Academic Publishers, Boston
- 1987 *Co-Chairperson*, IEEE Workshop on Fault-Tolerant Distributed and Parallel Systems, San Diego, California
- 1986 *Guest Editor*, *IEEE Transactions on Computers*,
Special Issue on Fault-Tolerant Computing, April 1986
- 1986 - 1988 *Editor*, *Advances in VLSI Systems*, Computer Science Press,
Maryland
- 1982 - 1985 *IEEE Distinguished Visitor*, Computer Society
- 1982 - Consultant to Mitre, CDC, IBM, AT&T, DEC and Data General
- 1981 - 1988 *Editor*, *Journal of VLSI and Digital Systems*, Computer Science
Press, Maryland
- 1980 *Guest Editor*, Special Issue on Fault-Tolerant Computing,
IEEE Computer, March 1980
- Member of Program Committee* for Fault-Tolerant Computing Symposium,
Computer Architecture Conference and other conferences
- Chaired sessions and organized panel discussions at various international
conferences

Grants

1973 - present Multiple grants from NSF, AFOSR, ONR, SRC, Bendix, IBM and
NRC (Canada); supported continuously, \$50,000-\$200,000 per year.

Research Supervision

1978 - present Several Ph.D. Students, placed in IBM, AT&T as well as leading universities

1977 - Research Associates:

K.L. Kodandapani

T. Nanya

K. Matsui

I. Koren

D. Avresky

F. Meyer

Patents

"Easily Testable High Speed Architecture for Large RAMs", U.S. Patent No. 4,833,677,

May 23, 1989. "DeBruijn Graph Based VLSI Viterbi Decoder", Application No. 904341,

June 18, 1992.

List of Publications

Text Book

Fault-tolerant Computing: Theory and Techniques (Editor and Co-Author), Vol. I and Vol. II, Prentice-Hall, Inc., May 1986 (Second Edition to appear 1993).

In Journals

1. "Modeling Live and Dead Lines in Cache Memory Systems" (with D. Thiebault and A. Mendelson), *IEEE Transactions on Computers*, to appear.
2. "A New Algorithm for Rank-Order Filtering and Sorting" (with Barun Kar), *IEEE Transactions on ASSP*, to appear.
3. "Virtual Checkpoints: Architecture and Performance" (with N.S. Bowen), *IEEE Transactions on Computers*, to appear.
4. "Accelerated Dynamic Learning for Test Pattern Generation in Combinational Circuits" (with W. Kunz), *IEEE Transactions on Computer-Aided Design*, to appear.

5. "Survey of Checkpoint and Rollback Recovery Techniques", *Computer*, to appear.
6. "Yield Optimization in Large RAMs with Hierarchical Redundancy" (with K.N. Ganapathy and A.D. Singh), *IEEE Journal of Solid State*, Vol. 26, No. 9, pp. 1259-1264, September 1991.
7. "A New Framework for Designing and Analyzing BIST Techniques and Zero Aliasing Compression" (with S.K. Gupta), *IEEE Transactions on Computers*, Vol. 40, No. 6, pp. 743-763, June 1991.
8. "Consensus with Dual Mode Failures" (with F.J. Meyer), *IEEE Transactions on Parallel and Distributed Systems*, Vol. 2, No. 2, pp. 214-222, April 1991.
9. "Error Correcting Codes in Fault-Tolerant Computers" (with E. Fujiwara), *Computer*, Vol. 23, No. 7, pp. 63-72, July 1990.
10. "Aliasing Probability for a Multiple Input Signature Analyzer and a New Compression Technique" (with S. Gupta and M. Karpovsky), *IEEE Transactions on Computers*, Vol. 39, pp. 586-591, April 1990.
11. "Organization and Analysis of Gracefully-Degrading Inter-leaved Memory Systems" (with K. Saluja, G. Sohi and K. Cheung), *IEEE Transactions on Computers*, Vol. 39, No. 1, pp. 63-71, January 1990.
12. "Modeling Defect Spatial Distribution" (with F.J. Meyer), *IEEE Transactions on Computers*, Vol. 38, No. 4, pp. 538-546, April 1989.
13. "The DeBruijn Multiprocessor Networks: A Versatile Parallel Processing Network for VLSI" (with M. Samatham), *IEEE Transactions on Computers*, Vol. 38, No. 4, pp. 567-581, April 1989.
14. "Dynamic Testing Strategy for Distributed System" (with F.J. Meyer), *IEEE Transactions on Computers*, Vol. 38, No. 3, pp. 356-365, March 1989.
15. "TRAM: A Design Methodology for High Performance Testable Large RAMs" (with N. Jarwala), *IEEE Transactions on Computers*, Vol. C-37, No. 10, pp. 1235-1250, October 1988.

16. "Designing Interconnection Buses in VLSI and WSI for Maximum Yield and Minimum Delay" (with I. Koren and Z. Koren), *IEEE Journal of Solid State Circuits*, Vol. 23, pp. 859-866, June 1988.
17. "Flip Trees: A Fault-Tolerant Network with Wide Containers" (with F.J. Meyer), *IEEE Transactions on Computers*, Vol. 37, No. 4, pp. 472-478, April 1988.
18. "Modeling the Effect of Redundancy on Yield and Performance of VLSI Systems" (with I. Koren), *IEEE Transaction on Computers*, Vol. C-36, No. 3, pp. 344-355, April 1987.
19. "Yield and Performance Enhancement through Redundancy in VLSI and WSI Multiprocessor Systems" (with I. Koren), *IEEE Proceedings*, Vol. 74, No. 5, pp. 699-711, May 1986.
20. "Dynamically Restructurable Fault-tolerant Processor Network Architectures", *IEEE Transactions on Computers*, Vol. C-34, No. 5, pp. 434-447, May 1985.
21. "Fault-tolerant Multiprocessor Structures", *IEEE Transactions on Computers*, Vol. C-34, No. 1, pp. 33-45, January 1985.
22. "Synthesis of Directed Multi-Commodity Flow Problems" (with A. Itai), *Networks*, Vol. 14, pp. 213-224, 1984.
23. "Sequential Network Design Using Extra Inputs for Fault Detection", *IEEE Transactions on Computers*, Vol. C-32, No. 3, pp. 319-323, March 1983.
24. "A Fault-Tolerant Distributed Processor Communication Architecture" (with S. Reddy), *IEEE Transactions on Computers*, Vol. C-31, No. 9, pp. 863-870, September 1982.
25. "A Class of Unidirectional Error Correcting Codes", *IEEE Transactions on Computers*, Special Issue on Fault-Tolerant Computing, Vol. C-32, No. 6, pp. 564-568, June 1982.
26. "A Uniform Representation of Permutation Networks Used in Memory-Processor Interconnection" (with K.L. Kodandapani), *IEEE Transactions on Computers*, Special Issue on Parallel Processing, Vol. C-29, No. 9, pp. 777-791, September 1980.

27. "A New Class of Error Correcting-Detecting Codes for Fault-Tolerant Computer Applications", *IEEE Transactions on Computers*, Special Issue on Fault-Tolerant Computing, Vol. C-29, No. 6, pp. 471-481, June 1980.
28. "Error-Correcting Codes and Self-Checking Circuits" (with J.J. Stiffler), *IEEE Computer*, Special Issue on Fault-Tolerant Computing, Vol. 13, No. 3, pp. 27-38, March 1980.
29. "Undetectability of Bridging Faults and Validity of Stuck-at Fault Test Sets" (with K.L. Kodandapani), *IEEE Transactions on Computers*, Vol. C-29, No. 1, p. 55-59, January 1980.
30. "Fault-Tolerant Asynchronous Networks Using Read-Only Memories", *IEEE Transactions on Computers*, Vol. C-27, No. 7, pp. 674-679, July 1978.
31. "Fault Secure Asynchronous Networks", *IEEE Transactions on Computers*, Vol. C-27, No. 5, pp. 396-404, May 1978.
32. "A Theory of Galois Switching Functions", *IEEE Transactions on Computers*, Vol. C-27, No. 3, pp. 239-249, March 1978.
33. "Universal Test Sets for Multiple Fault Detection in AND-EXOR Arrays", *IEEE Transaction on Computers*, Vol. C-27, No. 2, pp. 181-187, February 1978.
34. "Store Address Generator with Built-In Fault Detection Capabilities" (with M.Y. Hsiao & A.M. Patel), *IEEE Transactions on Computers*, Vol. C-26, No. 11, pp. 1144-1147, November 1977.
35. "A Graph-Structural Approach for the Generalization of Data Management Systems", *Information Sciences*. American Elsevier Publishing Company, Inc., pp. 1-17, March 1977.
36. "Techniques to Construct (2,1) Separating Systems from Linear Codes" (with S.M. Reddy), *IEEE Transactions on Computers*, Vol. C-25, No. 9, pp. 945-949, September 1976.
37. "Reed-Muller Canonic Forms for Multivalued Functions" (with A.M. Patel), *IEEE Transactions on Computers*, Vol. C-24, No. 2, pp. 206-220, February 1975.

38. "Fault-Tolerant Carry Save Adders", *IEEE Transactions on Computers*, Vol. C-23, No. 11, pp. 1320-1322, November 1974.
39. "Design of Two-Level Fault-Tolerant Networks" (with S.M. Reddy), *IEEE Transactions on Computers*, Vol. C-23, No. 1, pp. 41-48, June 1974.
40. "Fault-Tolerant Asynchronous Networks" (with S.M. Reddy), *IEEE Transactions on Computers*, Vol. C-22, No. 7, pp. 662-669, July 1973.
41. "Error Correcting Techniques for Logic Processors" (with S.M. Reddy), *IEEE Transactions on Computers*, Vol. C-21, No. 12, pp. 1331-1335, December 1972.

In Conference Proceedings

1. "A Design for Testability Scheme to Reduce Test Application Time in Full Scan" (with Jayashree Saxena), *10th IEEE VLSI Symposium*, Atlantic City, New Jersey, April 1992.
2. "Signature Analysis under a Delay Fault Model" (with Jayashree Saxena), *European Conference on Design Automation*, pp. 285-290, Brussels, Belgium, March 1992.
3. "A Hierarchical Directory Scheme for Large-Scale Cache Coherent Multiprocessors" (with Y.-C. Maa and D. Thiebaut), *6th International Parallel Processing Symposium*, pp. 43-46, Beverly Hills, California, March 1992.
4. "Signature Analysis under a Delay Fault Model" (with Jayashree Saxena), *European Conference on Design Automation*, pp. 285-290, Brussels, Belgium, March 1992.
5. "Yield Optimization of Redundant Multimegabit RAM's using the Center-Satellite Model" (with D. Dassharma), *IEEE International Conference on Wafer Scale Integration*, San Francisco, California, January 1992.
6. "A Virtual Memory Translation Mechanism to Support Checkpoint and Rollback Recovery" (with N.S. Bowen), *Supercomputing '91*, pp. 890-899, November 1991.

7. "Two Economical Directory Schemes for Large-Scale Cache Coherent Multiprocessors" (with Y.-C. Maa and D. Thiebaut), *ACM SIGARCH Computer Architecture News*, pp. 10-18, September 1991.
8. "Technique for Virtual Memory Architecture to Support Checkpoint and Rollback Recovery" (with N.S. Bowen), *IBM Technical Disclosure Bulletin*, Vol. 34, pp. 451-457, September 1991.
9. "High Level Synthesis of Data Driven ASICs" (with B. Patel), *Proc. Fourth Annual IEEE International ASIC Conference & Exhibit — ASIC'91*, Rochester, NY, September 1991.
10. "Program Fault-tolerance Based on Memory Access Behavior" (with N. S. Bowen), *Proc. 1991 International Symposium on Fault-Tolerant Computers*, pp. 426-433, Montreal, Canada, June 1991.
11. "System Level Diagnosis: Combining Detection and Location" (with N. H. Vaidya), *Proc. 1991 International Symposium on Fault-Tolerant Computing*, pp. 488-495, Montreal, Canada, June 1991.
12. "A Methodology for Partial Scan Design" (with S. Nori and J. Swaminathan), *Proc. Second European Test Conference*, pp. 263-271, Munich, Germany, April 1991.
13. "Weight/Space Bounded Error Control", *Proc. 1990 International Conference on Information Theory and Its Applications*, pp. 31-34, Honolulu, Hawaii, November 1990.
14. "Application Specific VLSI Architectures Based on De Bruijn Graphs", *Application Specific Array Processors*, IEEE Computer Society Publications, pp. 628-640, November 1990.
15. "Modeling of Live Lines and Tree Sharing in Multi-Code Memory Systems", *International Conference on Parallel Processing*, Vol. I, pp. 326-330, August 1990.
16. "Zero Aliasing Compression", *Proc. 1990 International Symposium on Fault-Tolerant Computing*, Newcastle, U.K., pp. 254-263, July 1990.

17. "On Implementing Improved Access Control Protocol for Shared Data Systems" (with A. Mendelson and A.D. Singh), *Proc. of 1st Annual IEEE Symposium on Parallel and Distributed Computing*, Dallas, TX, pp. 389-396, May 1989.
18. "Yield Modeling and Optimization of Large Redundant RAMs" (with A.D. Singh and K. Ganapathy), *International Conference on Wafer Scale Integration*, San Francisco, CA, pp. 273-287, January 1989.
19. "RTRAM: Reconfigurable and Testable Multi-bit RAM Design", *International Test Conference*, Washington, DC, pp. 263-278, September 1988.
20. "A New Framework for Designing and Analyzing BIST Techniques: Computation of Exact Aliasing Probability", *International Test Conference*, Washington, DC, pp. 329-340, September 1988.
21. "An Easily Testable Architecture for Multimegabit RAMs" (with N. Jarwala), *Proc. of International Test Conference*, pp. 750-758, Washington, September 1987.
22. "Consensus with Dual Failure Modes" (with F.J. Meyer), *Proc. FTCS-17*, Pittsburgh, pp. 48-54, July 1987.
23. "Cost Analysis of OnChip Error Control Coding for Fault-Tolerant Dynamic RAMs" (with N. Jarwala), *Proc. FTCS-17*, Pittsburgh, pp. 278-283, July 1987.
24. "Organization and Analysis of Gracefully-Degrading Interleaved Memory Systems" (with K. Cheung, G. Sohi, K. Saluja), *Proc. 14th International Symposium on Computer Architecture*, pp. 224-231, Pittsburgh, June 1987.
25. "Wafer-Scale Integration of Multiprocessor Systems" (with I. Koren and Z. Koren), *Proc. of HICSS-20 Hawaii International Conference on System Sciences*, pp. 13-20, January 1987.
26. "Introducing Redundancy into VLSI Designs for Yield and Performance Enhancement" (with Israel Koren), *Proc. FTCS-15*, pp. 330-334, Ann Arbor, Michigan, June 1985.
27. "Dynamic Testing Strategy for Distributed Systems" (with F.J. Meyer), *Proc. FTCS-15*, pp. 84-90, Ann Arbor, Michigan, June 1985.

28. "A Versatile Sorting Network" (with M.R. Samatham), *Proc. 12th Annual Symposium on Computer Architecture*, pp. 360-367, June 1985.
29. "Fault-tolerant Multibus Architectures for Multiprocessors" (with M.L. Schlumberger and Z. Hanquan), *Proc. FTCS-14*, Kissimee, Florida, pp. 400-408, June 1984.
30. "A Multiprocessor Network Suitable for Single Chip VLSI Implementation", *Proc. 1984 IEEE 11th Annual International Symposium on Computer Architecture*, pp. 328-337, June 1984.
31. "Fault-Tolerant Network Architectures for Multiprocessors and VLSI Based Systems", *Proc. FTCS-13*, Milan, Italy, pp. 436-441, June 1983.
32. "On a Class of Multiprocessor Network Architectures", *Proc. of International Conference on Distributed Processing*, Miami, Florida, pp. 302-311, October 1982 (also reprinted in *Interconnection Networks for Parallel and Distributed Processing*, edited by C. Wu and T. Feng, August 1984).
33. "Testing for Delay Faults in a PLA" (with K. Son), *Proc. International Conference on Circuits and Computers*, pp. 346-349, September 1982.
34. "Interconnections Topologies for Fault-Tolerant Parallel and Distributed Architectures", *Proc. of 10th International Conference on Parallel Processing*, pp. 238-242, August 1981.
35. "Fault-Diagnosis of Parallel Processor Interconnection Networks" (with K.M. Falavarjani), *Proc. Eleventh Annual International Symposium on Fault-Tolerant Computing*, pp. 209-212, June 1981.
36. "A Fault-Tolerant Communication Architecture for Distributed Systems", *Proc. Eleventh International Conference on Parallel Processing*, pp. 214-220, June 1981.
37. "A Solution to Load-Balancing and Fault Recovery in Distributed Systems" (with K. Matsui), *Symposium on Reliability in Distributed Software and Database Systems*, pp. 89-94, July 1981.

38. "Completely Self-Checking Checkers" (with K. Son), *Digest of 1981 Test Conference*, pp. 231-237, October 1981.
39. "A Fault-Diagnosis Technique for Closed Flow Networks", *Proc. of 1980 Symposium on Fault-Tolerant Computing*, pp. 302-304, Kyoto, Japan, October 1980.
40. "Effect of Undetectable Faults on Testing PLAs" (with K. Son), *Digest of 1980 Test Conference*, pp. 359-367, November 1980.
41. "An Easily Testable Design of PLAs" (with K. Son), *Cherry Hill Test Conference*, Philadelphia, Pennsylvania, November 1980 (reprinted in *IEEE Tutorial on VLSI Testing*, edited by Rex Rice, 1981).
42. "A Generalization of Shuffle-Exchange Networks", *Proc. of Fourteenth Annual Conference on Information Sciences and Systems*, Princeton, New Jersey, March 1980.
43. "A Framework for the Study of Permutations and Applications to Memory Processor Interconnection Networks" (with K.L. Kodandapani), *Proc. 1979 International Conference on Parallel Processing*, pp. 148-158, August 1979.
44. "Shift Registers Designed for On-Line Fault Detection", *Proc. of 1978 International Symposium on Fault Tolerant Computing*, Toulouse, France, pp. 173-178, June 1978.
45. "A Synthesis Algorithm of Directed Two-Commodity Networks", *1978 IEEE International Symposium on Circuits and Systems*, New York, pp. 93-98, May 1978.
46. "Error Control Techniques for Array Processors", *1977 International Symposium on Information and Theory*, Ithaca, New York, October 1977.
47. "On Undetectability of Bridging Faults" (with K.L. Kodandapani), *Proceedings of 1977 International Symposium on Fault-Tolerant Computing*, p. 192, Los Angeles, California, June 1977.
48. "Fault-Tolerant Fail-Safe Logic Networks" (with S.M. Reddy), *Proceedings on IEEE Comcon*, pp. 363-366, March 1977.
49. "Further Results on m-RMC Forms" (with K.L. Kodandapani), *Proceedings of 1976 International Symposium on Multivalued Logic*, Logan, Utah, pp. 88-93, May 1976.

50. "A Graph Structural Approach to Data Management Systems" (with L.C. Chang), *Proc. Ninth Hawaii International Conference on System Sciences*, Western Periodicals, pp. 254-258, January 1976.
51. "Fault-Tolerant Asynchronous Networks Using (2,1)-Type Assignments", *Digest of Fifth International Symposium on Fault-Tolerant Computing*, Paris, France, June 1975.
52. "Construction on Error Correcting Codes with Run-Length Limited Properties", presented in *1974 International Symposium on Information and Theory*, Notre Dame, Indiana, November 1974.
53. "Synthesis of Arithmetic and Logic Processors by using Nonbinary Codes" (with L.C. Chang), *Digest of Papers, Fourth International Symposium on Fault-Tolerant Computing*, IEEE Computer Society Publications, pp. 4-22, June 1974.
54. "A Multi-Valued Switching Algebra Based on Finite Field", *Proc. 1974 International Symposium on Multiple Valued Logic*, IEEE Computer Society Publications, Vol. 3, pp. 95-113, May 1974.
55. "On Fault Diagnosis of Sequential Machines", *Proc. VI Hawaii Conference on System Sciences*, Western Periodicals, January 1973.
56. "A Design Technique for Synthesis of Fault-Tolerant Adders" (with S.M. Reddy), *Digest of Papers of 1972 International Symposium on Fault-Tolerant Computing*, IEEE Computer Society Publications, pp. 20-25, June 1972.

9 Biography

Dhiraj K. Pradhan
Computer Science Department
H. R. Bright Building
Texas A&M University
College Station, TX 77843
(409) 862-2438 (office)
(409) 847-8578 (fax)
pradhan@cs.tamu.edu

Dhiraj K. Pradhan is currently holder of the Endowed Chair Professorship in Computer Science at Texas A&M University, College Station, Texas. Prior to joining Texas A&M he served until 1992 as Professor and Coordinator of Computer Engineering at University of Massachusetts, Amherst. Funded by NSF, DoD and various corporations, he has been actively involved in VLSI testing, fault-tolerant computing and parallel processing research, presenting numerous papers, with extensive publication in journals over the last twenty years. Dr. Pradhan has served as guest editor of special issues on fault-tolerant computing of *IEEE Transactions on Computers* and *Computer*, published in April 1986 and March 1980 respectively. Currently, he is an editor for several journals, including *IEEE Transactions on Computers* and *JETTA*. He has also served as the General Chair of the 22nd Fault-Tolerant Computing Symposium and as Program Chair for the 10th IEEE VLSI Test Symposium. Also, Dr. Pradhan is a co-author and editor of *Fault-Tolerant Computing: Theory and Techniques*, Volumes I and II (Prentice Hall, 1986; 2nd ed., 1992).

Dr. Pradhan is a Fellow of the IEEE and is a recipient of the Humboldt Distinguished Senior Scientist Award.