(2)

AD-A244 034

...ION PAGE

Form Approved
OMB No. 0704-0188

...ge 1 hour per response, including the time for reviewing instructions, searching existing data sources, ...collection of information. Send comments regarding this burden estimate or any other aspect of this ...ashington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson ...agement and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. ...

3. REPORT TYPE AND DATES COVERED
FINAL 1 May 88 - 29 Sep 91

4. TITLE AND SUBTITLE
"FAULT TOLERANT ARCHITECTURES FOR MULTIPROCESSORS AND VLSI-BASED SYSTEMS" (U)

5. FUNDING NUMBERS
61102F
2301/A2

6. AUTHOR(S)
Dhira K Pradhan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Massachusetts
Electrical & Computer Engineering Dept
Amherst, MA 01003

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM
Building 410
Bolling AFB DC 20332-6448

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFOSR-88-0205

91-19205

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release;
Distribution unlimited

12b. DISTRIBUTION CODE
UL

DTIC
ELECTE
DEC 3 0 1991

13. ABSTRACT (Maximum 200 words)

A general framework for shift register-based test response compressors was developed based on algibraic coding theory. It has advantages over Markov modeling in allowing exact computation of aliasing probability and extension to other forms of built-in self-test. The use of deBruijn graphs was adopted to studies of VLSI-based multiprocessor networks. These allowed derivation of lower bounds on VLSI layout areas and provided a method to meet those bounds. The graphs were extended to hyper-de Bruijn networks. Finally a design was produced for fault-tolerant testable RAM's (TRAM's).

14. SUBJECT TERMS

15. NUMBER OF PAGES
30

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

Summary and Future Research Directions

Grant AFOSR 88-0205

Final Report

Principal Investigator:
Dhiraj K. Pradhan
Electrical and Computer Engineering Department
University of Massachusetts
Amherst, MA 01003
Telephone: 413/545-0160

Submitted to:
Air Force Office of Scientific Research/NM
Building 410
Bolling Air Force Base, DC 20332-6448

Table of Contents

1 Abstract

This final report summarizes research carried out under grant AFOSR 88–0205 and provides an overview of short-term and long-term research goals. The focus of our research has been primarily in the following four areas:

1. Built-in self-test
2. Development of VLSI-based multiprocessor networks
3. Design of large fault-tolerant testable RAM's
4. Error control coding for developing new fault-tolerant techniques

This report is organized into the following sections. Section 2 reviews key results developed under the grant in the above four areas. Section 3 lists the publications supported by AFOSR 88–0205. Section 4 outlines our short-term and long-term goals for research in fault-tolerant and VLSI-based systems and gives our perspective on the future of fault-tolerant computing.

2 Summary of Results

2.1 Built-In Self-Test

Built-in self-test (BIST) has become a standard industry-wide test technique. BIST provides a mechanism to simplify the process of testing chips to determine which ones survived the defects introduced in the manufacturing process. BIST also provides opportunities to periodically test systems meant for high reliability/availability/reconfigurability and to assist in the identification of field replaceable units (FRU) for high maintainability.

An important issue pertaining to BIST that we have considered is the development of a general framework for shift register-based test response compressors. In this research we developed precisely such a framework and a mathematical model based on algebraic coding theory for this general framework. A distinction of the formulation is that it not only allows a uniform model for analysis of shift register techniques, but also allows for the development of new techniques. Our research in BIST has evolved in the following stages:

1. Coding theory formulation/computation of aliasing probability

2. Anti-aliasing techniques

3. Extension to multiple-output circuits

4. Extension beyond symmetric error model

Coding Theory Formulation

A generic BIST structure, a multiple-input shift register (MISR) is depicted in Figure 1. The X marks represent logical AND operations with the values ϕ_0 through ϕ_{m-1}. The set of ϕ values reflects the feedback polynomial used. The + marks represent logical XOR operations. The vector i_0 through i_{m-1} represent the m outputs of the circuit for each test. For a single-output circuit, only i_0 exists and the BIST structure constitutes a linear feedback shift register (LFSR). A LFSR implements division of the circuit output sequence by the feedback polynomial. The remainder of the division remains in the shift registers D_0 through D_{m-1}. After applying the test sequence, the remainder is termed the signature of the circuit and is available for comparison against the signature of a fault-free circuit. The quotient of the division is represented by the bits shifted out from D_{m-1} after each test. Together, the remainder and quotient would completely represent the input sequence to a LFSR, but since the quotient is lost, a fault may yield a functionally different circuit with a different quotient but the same signature as a good circuit. Such instances are referred to as aliasing and a major problem of BIST is proving the extent of aliasing for particular circuits and BIST structures.

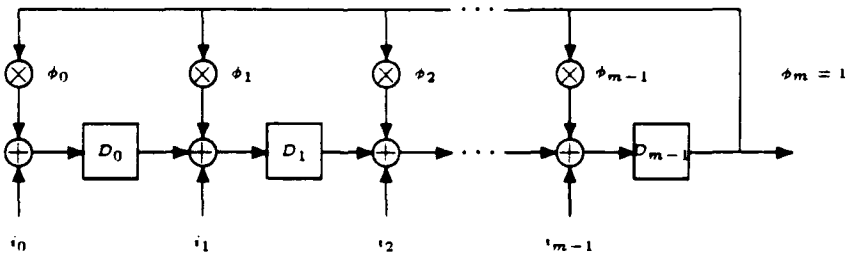

Figure 1: Conventional MISR Compressor

Conventional methods for determining the aliasing probability of a BIST structure use Markov models. Such models have the advantages of tractability for simple BIST structures. BIST structures, however, can be described in a natural way in terms of algebraic coding theory. We have developed such a formulation, which has the following advantages over Markov models:

1. Markov models predict the asymptotic aliasing probability as the length of the test sequence goes to infinity, whereas the coding theory formulation allows exact computation of the aliasing probability for any test sequence.

2. The coding theory formulation can be extended to consider MISR's and more complicated BIST structures, while extension of Markov models to more complicated structures causes them to lose their tractability.

3. Choices of the feedback polynomial and of the test pattern generator will yield differing aliasing probabilities, which the coding theory formulation will discern. Markov models can be modified to account for different feedback polynomials, but doing so causes them to lose their tractability.

Anti-Aliasing Techniques

Aliasing occurs because the quotient of the division is lost. The quotient is discarded because it is very nearly the same length as the circuit output sequence (quotient length equals test sequence length minus signature length). The question naturally arises whether it is possible to apply the quotient to a LFSR and produce a signature for it, thereby reducing aliasing potentially to zero. We have solved this problem. Given any circuit, any test sequence, and any LFSR, we can obtain a second LFSR that combined with the first produces a unique combination of signatures. The maximum number of shift registers needed for both divisions is about half the test sequence. There are two difficulties with this method:

1. Determining the feedback polynomial of the second LFSR is not computationally tractable, so it is generally possible only for small circuits and test sequences.

2. The maximum number of shift registers needed is still unacceptably large and we have proven that most circuits require near the maximum number of shift registers.

Despite these difficulties, researchers at United Technologies Corporation have reported that they have achieved zero aliasing for a chip that was particularly amenable to the technique.

Multiple-Output Circuits

Unlike conventional aliasing models, the coding theory formulation allows us to compute the exact aliasing probability for a wide variety of BIST structures. For circuits with many outputs, the cost of implementing BIST for each output is prohibitive. Accordingly, BIST

structures such as the MISR in Figure 1 are in common use since they incorporate entire output vectors into a single shift register.

Other testing paradigms that do not involve single outputs have been proposed. An example is the STUMPS paradigm as depicted in Figure 2. The STUMPS paradigm provides facilities to test a number of chips simultaneously. These chips may be expected to provide identical outputs (SO_1 through SO_m all the same) as might be the case for testing after manufacture. Alternatively, the chips may produce different outputs, as would be the case for board-level testing. Our general framework allows exact computation of aliasing probabilities in such settings and provides a research framework to determine good BIST structures and to determine good feedback polynomials within particular BIST structures.

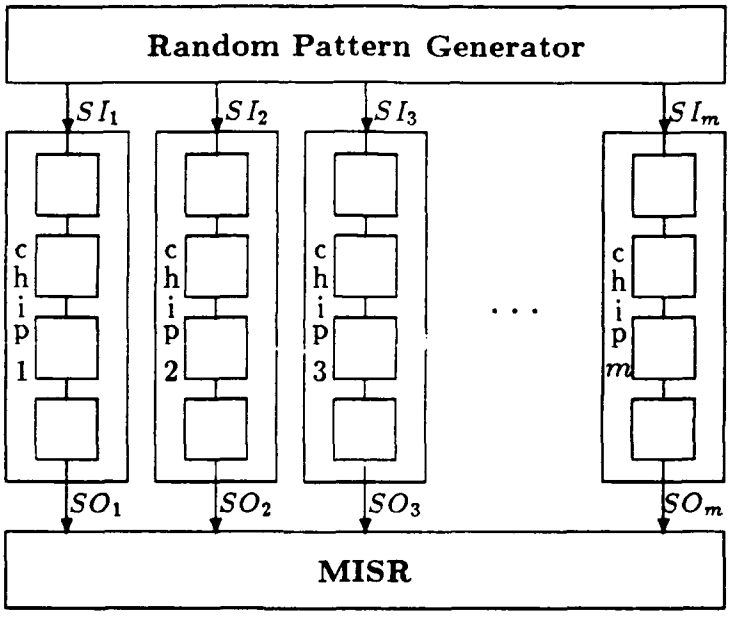

Figure 2: Global Test Using STUMPS

Various Error Models

Work described in previous sections have all assumed a symmetric error model (all outputs equally likely given there is a fault). But other error models may be more appropriate than the (2^m-ary) symmetric error model. For example, the independent error model (BSC) assumes that when an error exists each output is affected independently and with a given probability. We have developed a very general error model that subsumes these and other error models. The effects of different error models have been considered and our method has been applied to particular circuits. Figure 3 shows the aliasing probability computed

under different error models for various test lengths. The subject circuit is C432 of the MCNC combinational benchmark test circuits. Note that aliasing probability tends to a particular value as the test sequence length increases, as predicted by Markov model methods. The aliasing probability may significantly differ, however, under different error models when the test sequence is shorter.

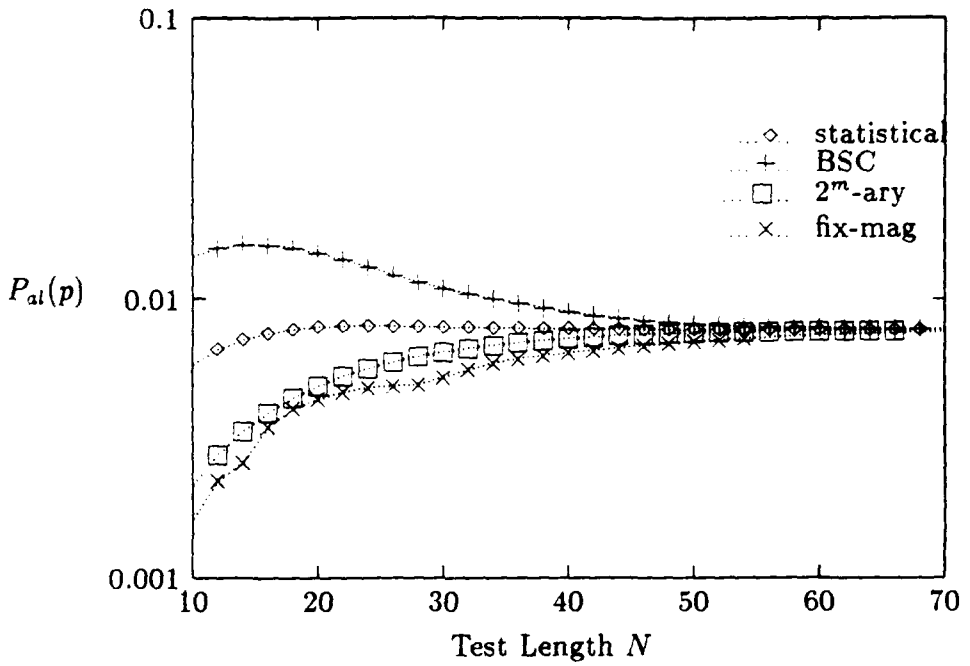

Figure 3: Aliasing Probability for C432

Our general error model has the following disadvantages versus the symmetric error model:

1. The coding theory formulation loses much of its computational tractability. This loss, however, is due entirely to the increased data that the general error model must consider. Under the most general model each fault must be considered for each test and the probability of a given output determined.

2. Under the symmetric error model, BIST methods could be analyzed and a good BIST method found very early in the development of the product. The symmetric error model only requires the circuit's functional specification. With a general error model, a more detailed circuit description is necessary (at least as low as the gate-level specification), because how the circuit's function is implemented dictates the effects of faults on the outputs and also the set of possible faults.

2.2 VLSI-Based Multiprocessor Networks

We have studied a number of topologies suitable for VLSI implementation. The primary criteria for evaluating VLSI topologies are (1) support for mapping common algorithms to the architecture for use as a parallel processing machine, (2) short distances between nodes (as reflected by the graph diameter), (3) ability to sustain diameter in the presence of faults, and (4) amenability to two-dimensional layout for VLSI.

De Bruijn Network

Under AFOSR support, we were first to discover the value of de Bruijn graphs for VLSI-based multiprocessor networks. A recent development that emphasizes the significance of de Bruijn networks is its use in the projected Galileo spacecraft for coding problems.

We have studied binary de Bruijn graphs (BDG) extensively. We derived a lower bound on the VLSI layout area of the BDG and obtained a layout method to meet the bound. We have shown that BDG's can be configured to match the data flow graph of a large class of algorithms. A careful comparison of BDG with the hypercube reveals that BDG's admit various important configurations such as complete binary trees and one-step shuffle-exchange networks (which are not admisable by hypercubes). Consequently, the BDG can support a wide variety of algorithms in addition to many algorithms suported by the binary cube. We have shown that the BDG is the only known network that can sort in all known categories of sorting. Also we have been able to show that the BDG can be a powerful technique for solving a wide variety of graph and linear algebra applications. We have shown that certain string comparison algorithms can run efficiently on the BDG.

Shuffle-exchange networks are useful for a variety of problems such as permutation and the fast Fourier transform. Trees are useful for problems of a divide-and-conquer nature such as sorting and parallel prefix operations. Many algorithmic paradigms exist that may be described as graphs. We have shown that BDG's support the most common paradigms and therefore form a quite useful basis for parallelizing algorithms.

Flip-Trees

Continued advances in VLSI technology hold the promise of very large distributed systems where each node in the system is fabricated on a single chip. Thousands or even millions of processors could be joined in such a system. Massive computational resources, however, imply that the communication effectiveness of the system may be the weakest link in the design. This holds both for the performance of the system (some parallel and (especially) distributed algorithms require communication that grows faster than the number of processors) and for reliability (having many processors may allow task/data replica-

tion without impacting performance, while increasing the reliance on the communication structure).

A *container* is a set of node-disjoint paths between a pair of nodes. The advantages of containers are briefly discussed below:

1. By sending a message along more than one of the node-disjoint paths, the message will arrive correctly at its destination if a majority of the paths has all nodes non-faulty (or, when all faults are site crashes, if any one path has all nodes nonfaulty).

2. A message can be sent along one path. The recipient can acknowledge receipt of the message by sending the acknowledgement along paths in the container. This is a *distributed handshake.* The acknowledgement is a brief message, so the expense of sending it along multiple paths can be justified. If the original message cannot be altered by a faulty node along its path, then the distributed handshake problem reduces to resolving whether the message was received (whether the path was intact). We have developed a very general solution to this problem.

3. Duplex communication between a pair of nodes can be achieved, without congestion, by assigning two different paths to the two different directions of communication.

4. Containers admit a simple fault-tolerant distributed routing strategy using table look-up. Each node can maintain, for every pair of nodes, the names of its two neighbors along the exclusive node-disjoint path it is along between those two nodes (if it is along one of the node-disjoint paths). This has the distinct advantage of reducing routing information in messages.

We describe a family of graphs called flip-trees that has two parameters—the degree, d, and the diameter, $2\ell - 1$. When $\ell = 1$, the graph degenerates into the complete graph on $d + 1$ nodes. d is constrained to be an integer at least equal to three. Let us consider a tree with the root having an extra son. Figure 4 shows such a basic tree of depth two to be used in constructing a flip-tree of degree three.

The figure reflects the labelings we use for the nodes of the graph. The root is labeled $*$, the sons of the root are labeled from $0*$ to $(d-1)*$. The sons of a node $ab_1b_2\ldots b_i*$ are $ab_1b_2\ldots b_i1*$ through $ab_1b_2\ldots b_i(d-1)*$. The leaves of the tree are labeled in the form $ab_1b_2\ldots b_{\ell-1}$, where a is an integer from 0 to $d-1$ and the b_i are integers from 1 to $d-1$. So the leaves are distinguished from the other nodes in the tree by labels not ending in $*$.

It remains to decide how to interconnect the leaves of the basic tree. Flip-trees are constructed from basic trees by connecting each leaf, $ab_1b_2\ldots b_{\ell-1}$ to all leaves $a'b_{\ell-1}\ldots b_2b_1$, where a' is any integer from 0 to $d-1$ other than a. Figure 5 shows the flip-tree for $d = 3$ and $\ell = 3$.

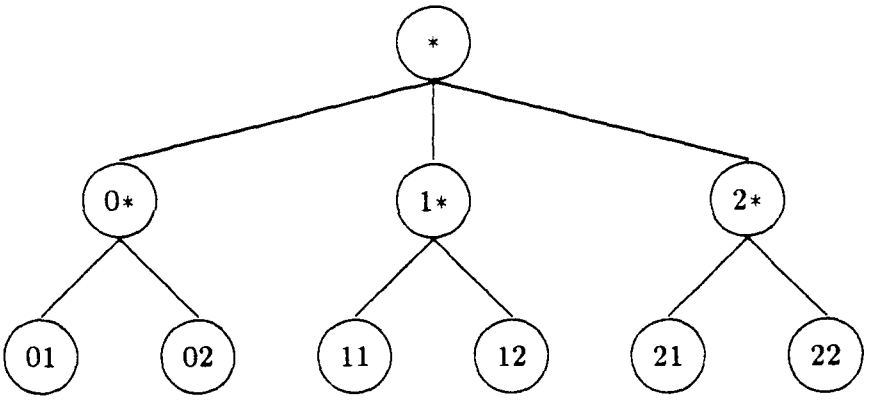

Figure 4: Basic Tree and Node Labelings

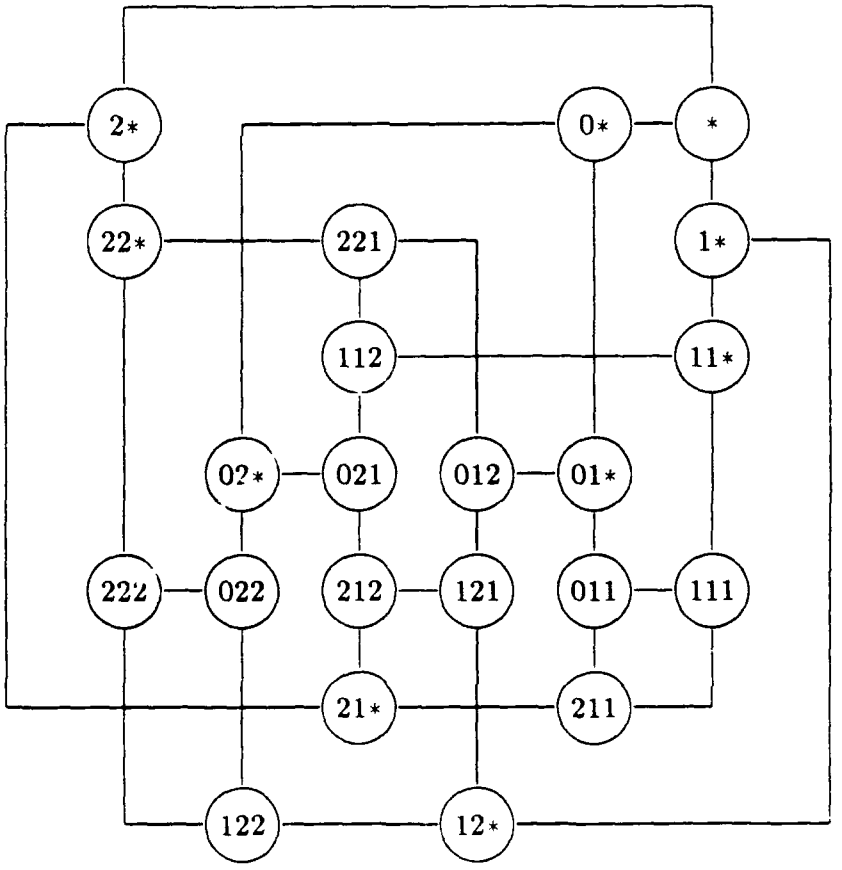

Figure 5: Example Flip-Tree

Our main result has been to show that flip-trees with parameters d and ℓ have a container of width d and length $\leq 2\ell + 1$ between every pair of nodes. Flip-trees have the best known containers.

We have shown that flip-trees are competitive with respect to many aspects of network topologies, such as diameter and fault-tolerant diameter, as well as having the best known containers. The primary areas of deficiency are: (1) traffic congestion and (2) distributed routing with localized routing information.

(1) Roughly $(d-2)/(d-1)$ of all messages must be routed through some node on level $\lceil \ell/2 \rceil$, but roughly $(d-1)^{-\lfloor \ell/2 \rfloor}$ of all nodes are at level $\lceil \ell/2 \rceil$. As the diameters of networks of interest increase, this imbalance is exacerbated. This is the same level of congestion that butterfly networks experience when conducting all-to-all communication.

(2) We have shown that topologies such as the de Bruijn graph and hypercube are amenable to a highly distributed routing approach, where each node need maintain only the faulty/nonfaulty status information of nearby nodes by detouring messages around faulty nodes. This approach is not practical for flip-trees, because many pairs of nodes at a distance of two from each other (for instance, nodes near the root) do not have a short detour if their common neighbor is faulty.

Hyper-de Bruijn Network

Hypercube and de Bruijn networks each possess certain desirable properties. But some of the attractive features of one network are not found in the other. We have developed an architecture, the hyper-de Bruijn (HDB) network, which is a Cartesian product of the hypercube and the de Bruijn network. Figure 6 depicts a 16-node binary de Bruijn network and Figure 7 depicts a 16-node HDB network obtained as a product of a 4-node hypercube and a 4-node de Bruijn network.

Like the hypercube and de Bruijn networks, HDB networks have logarithmic diameter. But while the de Bruijn network has a fixed degree (number of ports per node) of four and the hypercube has degree that grows with the number of nodes, the HDB allows the designer to select any node degree between four and the logarithm of the number of nodes. The fixed node degree of the de Bruijn network can be seen as a drawback when one considers the probability of a path existing between any two nodes in the presence of faults. As the number of nodes in a network increases while the reliability of each node remains constant, the degree necessary to maintain a prescribed level of path reliability would increase. For the hypercube, its logarithmic increase in degree exceeds what is necessary to maintain path reliability.

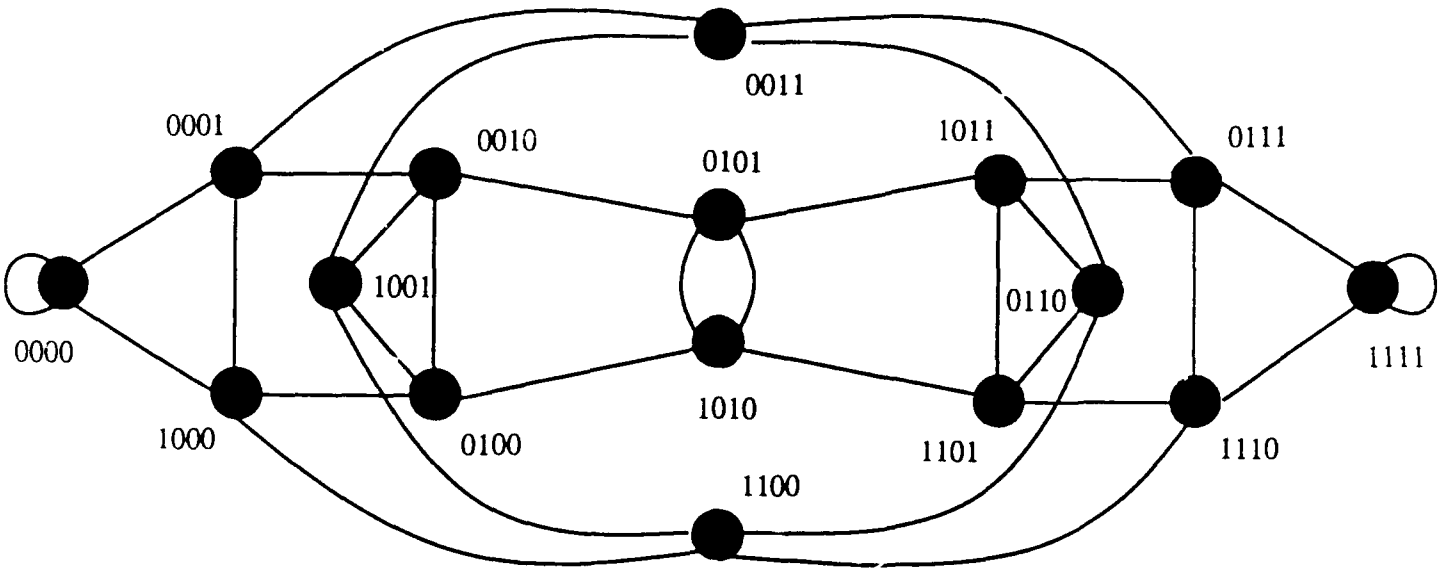

Figure 6: Example de Bruijn Network

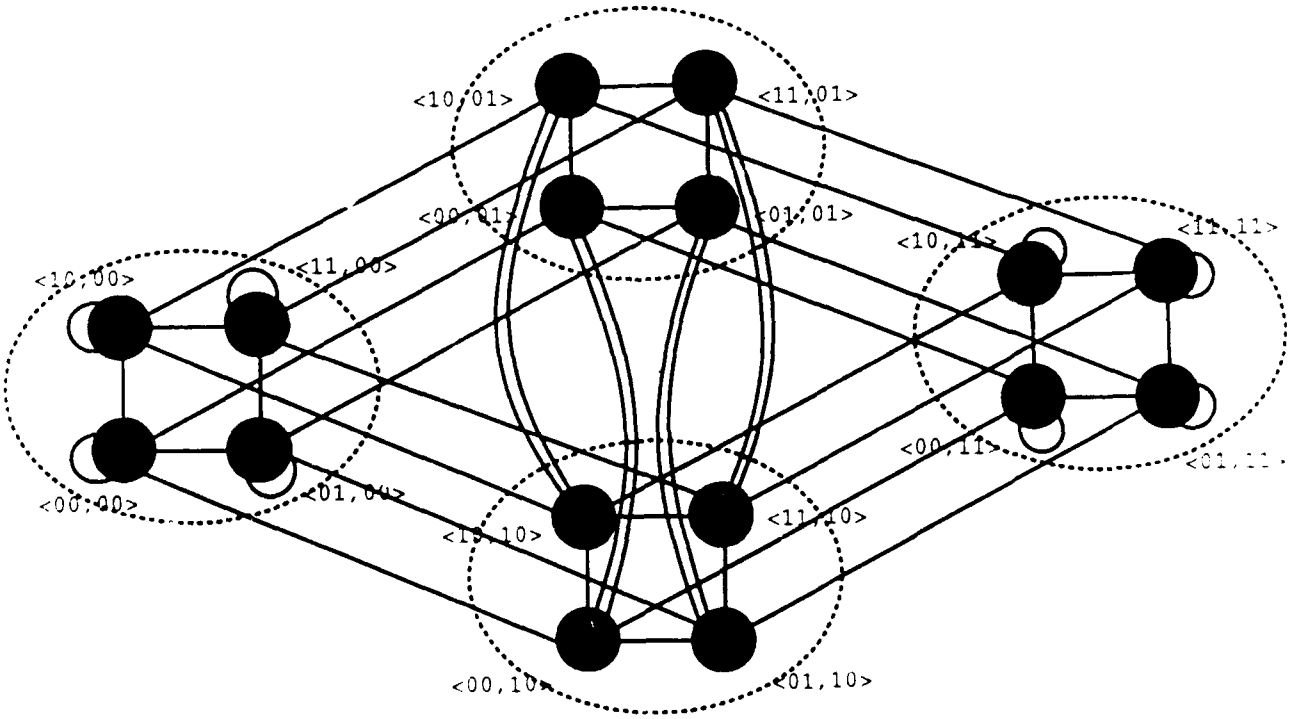

Figure 7: Example Hyper-de Bruijn Network

Because the HDB is a Cartesian product, the complexity of message routing on the HDB is no more complex than for the cube and the de Bruijn network (i.e., trivial). Further, being a Cartesian product, the HDB is quite resilient to faults. We have established facile routing algorithms for the HDB that route in the presence of faults. Further, these routing algorithms are distributed in nature — each node does not need to be aware of the good/faulty status of all nodes in the network; each node need only be aware of the status of its immediate neighbors.

The HDB network contains various computationally important networks as subgraphs: rings, multidimensional meshes, complete binary trees, meshes of trees, and others. The multidimensional meshes are important in a variety of algorithms such as the solution of partial differential equations. The meshes of trees are important to algorithms such as matrix multiplication.

2.3 Large Fault-Tolerant Testable RAM's

Description of TRAM

We have proposed a new design to implement large, fault-tolerant, testable RAM's in VLSI. This novel design has been patented by the USAF. The design (TRAM) implements the divide-and-conquer concept. A multimegabit RAM is implemented by dividing the RAM into a number of modules which are layed out in VLSI as the leaves of a tree. Figure 8 depicts an H-tree layout. H-tree is a two-dimensional tree layout that occupies about twice the area of the number of nodes and four times the area of the number of leaf nodes (under the Thompson grid layout model). An actual implementation of TRAM would not consume quadruple area, because only the leaf nodes are large and the width of busses connecting the nodes is likewise not large. TRAM has the following features:

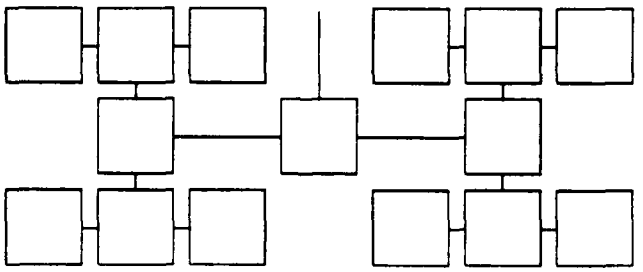

Figure 8: Example H-Tree

1. *Testability.* A major problem of testing RAM's is that the number of tests required, even under simple fault models, grows faster than the number of bits of memory implemented. By dividing the memory into a number of modules, the complexity of the testing problem is substantially reduced. In addition, each module may be supplied with an on-chip test mechanism thereby allowing the nodes to be tested in parallel, reducing test time further. TRAM is the first architecture that yields practical testing times for multimegabit RAM's.

2. *Performance.* For large RAM's, the TRAM architecture has the potential for reducing the access time by about 30 percent. The access time of a TRAM is dictated by the delay in using the tree to access the correct leaf node (logarithmic in the number of leaves) plus the delay to access the proper bit from the leaf node (grows as the square root of the number of bits in the leaf module) plus overhead delays. A conventional RAM does not experience a delay to traverse the tree, but the (square root) delay to access the proper bit is much larger, because TRAM has divided the problem into much smaller modules.

3. *Area Overhead.* The additional area overhead for the TRAM architecture is typically from 8 to 20 percent for a large RAM. The variation in overhead is due to the fundamental choice in the design of the TRAM — how many leaf modules to use. Choosing how large to grow the tree affects the area overhead, the access time, and the testing time for the TRAM.

4. *Partitionability.* The regular structure of the TRAM and its ability to test leaf modules independently allow the manufacturer to determine when a partially good product (e.g., half-size RAM) can be obtained. This improves the economic viability of manufacturing very large RAM's. The situation is similar to the manufacture of the Intel 80486DX and 80486SX processors. In manufacturing an 80486DX, if testing shows that the chip is functional except for the floating point unit, then the chip can still be shipped as an 80486SX.

Extended Yield Analysis of TRAM

The TRAM design has maximal benefits for very large RAM's. Sixteen megabit and larger RAM's are now in production and development. These memories will require large-area VLSI or even WSI to produce. Conventional IC fabrication yield models are not valid for large-area VLSI and beyond. We developed the center-satellite yield model to accommodate the necessities of ambitious designs. The center-satellite model provides different yield projections than conventional models for large-area VLSI designs incorporating redundancy. In addition to a fundamental rethinking of the defect process in IC fabrication, our yield model also directly incorporates well-known anomalies that become significant for WSI designs, such as the radial dependence of defect densities.

We have modelled the TRAM design for very large memories (e.g., 16 megabits to 1 gigabit). TRAM allows for testing of individual modules and reconfiguration to still yield a shippable product. Therefore it is not necessary to achieve near perfect yield of each module. The existence of *hardcore* in each module does not permit near perfect yield anyway. We have analyzed in depth the yield of individual modules with the following four redundancy schemes:

1. *Extra columns only.* This is the weakest scheme. Good yield requires each module to be substantially less than one megabit.
2. *Extra rows and columns.* This is marginally better than extra columns only. Coding is required for larger module sizes.
3. *Coding.* This has more hardcore, but with larger module sizes is worth it. Coding alone may be sufficient depending on the fabrication line-dependent parameters to the center-satellite model.
4. *Coding with extra rows.* This is the best scheme. With current fabrication line quality, this scheme can produce acceptable module yield for virtually any feasible module size. This scheme may be defeated if further decreases in feature sizes lead to much higher defect densities.

With acceptable module yields (e.g., better than 80 percent), it is possible to use the block substitution capabilities of TRAM. Our extended yield analysis has established the level of redundancy required at each module to optimize product yield. For example, modules near the center of the wafer may find coding alone to be most efficacious, while modules near the wafer periphery (and most susceptible to radial variation) may find coding with extra rows necessary. Current redundancy schemes for RAM's allow fine gradations in redundancy levels—e.g., extra rows may be added one at a time.

2.4 Error Control Coding

We are beginning work on the analysis of voting systems that employ coding. The principal emphasis of the work is to determine the reliability and safety issues involved and to characterize the nature of the tradeoff between reliability and safety. We describe the unit that determines the output of the system as the *arbitrator*. The arbitrator's purpose is to determine the most likely correct output and to also raise a safety flag when the doubt on the correctness of the output exceeds a selectable threshold.

Preliminary results have been obtained for n-modular redundant (nMR) systems. These results define and prove the optimal arbitration policies. We have shown that certain optimal arbitration policies for nMR cannot be exceeded (in terms of reliability and safety)

by any arbitration policy in an $(n+1)$MR system. This result holds when the n outputs to be arbitrated do not themselves contain redundancy. Similarly an $(n+2)$MR system always has arbitration policies strictly better than any nondegenerate arbitration policy for an nMR system. When *any* redundancy is incorporated into the n module outputs, $(n+1)$MR then is guaranteed to exceed nMR.

3 Patent and Publications Under AFOSR 88–0205

Patent:

"Easily Testable High Speed Architecture for Large RAMs," U.S. Patent Number 4,833,677. Date: May 23, 1989. Inventors: Najmi T. Jarwala and Dhiraj K. Pradhan. Assignee: U.S. Government represented by the Secretary of the Air Force, Washington, DC.

Publications:

"The hyper-deBruijn multiprocessor networks," *IEEE Trans. Parallel and Distr. Sys.*, (with E. Ganesan), submitted.

"Yield Optimization of Redundant Multimegabit RAM's Using the Center-Satellite Model," *Int. Conf, on Wafer Scale Integration*, (with D. Das Sharma and F. Meyer), submitted.

"A theorem on the fault-tolerance of a modified de Bruijn topology," *J. Discrete Math.*, (with S. Toida and F. Meyer), to appear.

"A Uniform Analysis of Aliasing in MISR compression for various error models," *Int. Test Conf.*, (with M. Karpovsky and S. Gupta), October 1991.

"A framework for designing and analyzing new BIST techniques and zero aliasing compression," *IEEE Trans. Comput.*, vol. 40, no. 6, (with S. Gupta), June 1991.

"System-Level Diagnosis: Combining Detection and Location," *Fault Tol. Comput. Symp.*, Montreal, pp. 488–495, (with N. Vaidya), June 1991.

"Program Fault Tolerance Based on Memory Access Behavior," *Fault Tol. Comput. Symp.*, Montreal, pp. 426–433, (with N. Bowen), June 1991.

"The Hyper-deBruijn Multiprocessor Networks," *Int. Conf. Distr. Comput. Sys.*, Arlington, TX, (with E. Ganesan), May 1991.

"Consensus with dual failure modes," *IEEE Trans. Parallel and Distr. Sys.*, vol. 2, no. 2.

pp. 214–222, (with F. Meyer), April 1991.

"Application Specific VLSI Architectures Based on de Bruijn Graphs," *Int. Conf. on Application Specific Array Processors*, Princeton, NJ, September 1990.

"Zero Aliasing Compression," *Fault Tol. Comput. Symp.*, Newcastle upon Tyne, UK, pp. 254–263, (with S. Gupta and S. Reddy), June 1990.

"Aliasing probability for a multiple input signature analyzer and a new compression technique," *IEEE Trans. Comput.*, vol. 39, no. 4, pp. 586–591, (with S. Gupta and M. Karpovsky), April 1990.

"Organization and analysis of gracefully-degrading inter-leaved memory systems," *IEEE Trans. Comput.*, (with K. Saluja, G. Sohi, and K. Cheung), 1989.

"On Implementing Improved Access Control Protocol for Shared Data Systems," *IEEE Symp. on Parallel and Distr. Comput.*, Dallas, TX, pp. 389–396, (with A. Mendelson and A. Singh), May 1989.

"The de Bruijn multiprocessor networks: A versatile parallel processing network for VLSI," *IEEE Trans. Comput.*, vol. 38, no. 4, pp. 567–581, (with M. Samatham), April 1989.

"Modeling defect spatial distribution," *IEEE Trans. Comput.*, vol. 38, no. 4, pp. 538–546, (with F. Meyer), April 1989.

"Dynamic testing strategy for distributed systems," *IEEE Trans. Comput.*, vol. 38, no. 3, pp. 356–365, (with F. Meyer), March 1989.

"Yield Modeling and Optimization of Large Redundant RAMs," *Int. Conf. on Wafer Scale Integration*, San Francisco, CA, pp. 273–287, (with A. Singh and K. Ganapathy), January 1989.

"TRAM: A design methodology for high performance testable large RAMs," *IEEE Trans. Comput.*, vol. 37, no. 10, pp. 1235–1250, (with N. Jarwala), October 1988.

"RTRAM: Reconfigurable and Testable Multi-Megabit RAM Design," *Int. Test Conf.*, Washington, DC, pp. 263–278, (with N. Kamath), September 1988.

"Designing interconnection buses in VLSI and WSI for maximum yield and minimum delay," *IEEE J. Sol. State Circ.*, pp. 859–866, (with I. Koren and Z. Koren), June 1988.

"Flip-trees: Fault-tolerant graphs with wide containers," *IEEE Trans. Comput.*, vol. 37, no. 4, pp. 472–478, (with F. Meyer), April 1988.

"Cost Analysis of On Chip Error Control Coding for Fault Tolerant Dynamic RAMs," *Fault Tol. Comput. Symp.*, Pittsburgh, PA, pp. 278–283, (with N. Jarwala), July 1987.

4 Short-Term and Long-Term Research Goals

4.1 Area-Specific Research Goals

This section describes opportunities that remain for further research in the four areas that have been the subject of AFOSR 88–0205. The next section describes our general perspective on the future of fault-tolerant computing and research opportunities.

BIST

The most important area for progress is in applying our methods to sequential circuits. All results so far have assumed combinational circuits. The testing problem itself is extremely difficult for sequential circuits, but important methodologies currently in practice such as boundary scan have significantly eased the testing problem—although not to acceptable levels. All models to determine aliasing in BIST for sequential circuits are intractable. The coding theory formulation, however, holds some promise. To exploit the coding theory formulation, however, may require totally new BIST structures.

Circuits with multiple outputs generally have the effect of their outputs distributed across the BIST structure as in Figure 1. Multiple outputs, however, may also be compressed first with a combinational circuit to produce a single output to feed the BIST structure. Output compression has been avoided, though, because the affect on aliasing was hard to predict. With our general error model, however, we can accurately calculate the aliasing probability. Therefore, output compression should be reconsidered.

The coding theory formulation provides a framework to evaluate BIST structures, but much work remains to be done to apply our results. Procedures should be developed for popular BIST structures to determine good parameters (e.g., feedback polynomial) for them. Also there are many opportunities for novel BIST structures, while for the first time we have the tools to properly evaluate them.

VLSI-Based Multiprocessor Networks

We continue to seek network topologies with excellent diameters, especially in the presence of faults. We have discovered the VARSEA topology. When its properties are fully characterized, it is expected to have the best known diameter in the presence of faults. The VARSEA topology is already known to have very facile routing in its fault-free state. The VARSEA topology is node-symmetric and therefore will provide congestion-free communication.

Large Fault-Tolerant Testable RAM's

Our analysis of the yield of modules using the classical four-quadrant architecture is largely complete. But large modern RAM's, such as IBM's new 16 megabit design do not use the four-quadrant architecture. The yield analysis of TRAM modules should be extended to eight- and sixteen-quadrant architectures in order to analyze the most ambitious TRAM designs.

In developing TRAM yield projections we have assumed that faulty modules are logically isolated through global block substitution, but global substitution has a larger hardcore than local substitution methods. Unfortunately, local block substitution is less powerful and makes the design more susceptible to the spatial autocorrelation of defects that the center-satellite model reflects. This tradeoff warrants study. There are a wide variety of local substitution methods, such as interstitial redundancy, etc. We intend to analyze the merits of various local block substitution methods.

Error Control Coding

Most of our progress to date on arbitration policies has considered modules without redundancy. Memory modules in modern systems would clearly include redundancy and it is also possible to have modest levels of redundancy in arithmetic/logical modules. Our results need to be extended to apply to modules incorporating redundancy. Different types of modules will have different constraints imposed on them. Arithmetic/logical modules have practical limits on the within-module redundancy feasible, so modular redundant systems for such modules would depend heavily on high replication of the modules. Memory modules, however, can very efficiently incorporate redundancy; further, the redundancy within a module is more valuable than the replication of modules, so memory systems would tend to rely heavily on within-module redundancy with module replication limited to 2MR (i.e., a mirroring system). It is possible that our research at this juncture will branch to allow for an in-depth analysis of 2MR.

4.2 Future Research Directions

Under AFOSR 88-0205 we began to broaden our fault-tolerant computing emphasis to explore reliability while keeping safety issues in mind. This trend will continue. Ever larger systems are being built and fault-tolerant computing techniques are being applied to ever larger systems. As a result, greater attention must be paid to a wide variety of possible failure modes. These failure modes may result in different levels of safety violations and may also lead to degraded systems that provide different levels of mission effectiveness.

In addition to reliability and safety, security is an issue for many systems. Decisions made

to enhance the reliability/safety tradeoff may have consequences on the security of the design (and vice versa). We plan to develop an integrated framework that allows for the evaluation of designs in terms of reliability, safety, and security criteria. A major part of this effort will be to expand our models of reliability to allow degraded modes of operation. In the following we briefly describe two examples to illustrate the diversity of systems we plan for our integrated framework to accommodate. The second example system also discusses how designing for reliability, safety, and security are interrelated and motivates the need for an integrated framework to analyze such systems.

DACP Example

A data access control protocol (DACP) is a set of rules that specify who or what is allowed access to sensitive data and under what circumstances. Gigantic databases can only be usefully implemented via computer systems. Evaluating the effectiveness of DACP's is particularly difficult for computer systems. We list a few of the pertinent issues:

1. The integrated framework must allow the user to specify the meaning of common terms such as the sensitivity of data and the integrity of computer systems and human operators. As an example of the difficulties involved, consider that a computer system may provide programs to manipulate data. Such programs may be of a general nature, such as text processing. The sensitivity of the manipulated data has four components: (1) part of the sensitivity of the original data, (2) part of the sensitivity of the manipulating program, (3) the sensitivity of the knowledge that applying the manipulating program to the data was useful (and how the program was applied), and (4) a component that reflects security restrictions on allowing the program to manipulate the data.

2. The common paradigm for human access to data is: clearance + need to know (C+NTK) = access. The sheer volume of sensitive data in human-readable form makes such a simple and vague paradigm necessary. C+NTK is not plausible for sensitive data in computer systems. The good news is that for a major part of the access problem (deciding which programs are allowed to manipulate which data), C is not a relevant part of C+NTK. The bad news is that NTK is too vague to be implemented by computer DACP's. A very flexible DACP would allow accesses that a human would consider invalid. An inflexible DACP would need frequent updating to allow clearly needed accesses. An entirely new paradigm may be necessary to both meet security objectives while permitting the computer system to fulfill its mission.

C^3I Example

Consider a system with a C^3I mission. Our concerns for such a system when it operates in an adverse environment are: (1) (reliability) how effectively it accomplishes its mission, (2) (safety) whether it causes acts that adversely affect friendly/neutral forces, and (3) security. Figure 9 shows the communication connections for an example C^3I system. The nodes labelled CP are higher level command posts.

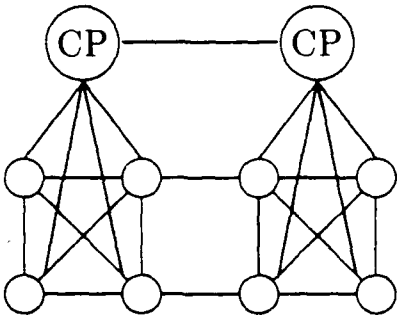

Figure 9: Example C^3I Structure

1. C^3I systems are amenable to graphical representations. Unit capabilities can be represented by node labellings and the relationships between units can be represented by edge labellings. The integrated framework should allow the user to specify the nature of the relationships and to define the criteria for the system effectiveness, safety, and security.

2. The C^3I system depicted is somewhat robust. In an adverse environment, even if one of the CP units is incapacitated, the system may be able to partially fulfill its mission. To provide workarounds for such contingencies, however, may mean disseminating information widely in the system. If designed to operate only when not impaired, then information can be centralized at the CP units, which can dynamically decide what information other units need. If designed to operate even when impaired, additional information may be needed *a priori* at the subordinate units; this could have adverse security consequences. The integrated framework should reflect such tradeoffs and support their analysis.

3. A C^3I system needs to be able to initiate conflict. For instance, it may be desired that all units determine that the command (initiate, B) has been given, where B is a possible battle plan. We have done some work along these lines under AFOSR support. The interactive consistency problem has the objective of ensuring that all units agree on the commands issued. The consequences of failing to achieve interactive consistency range from units not carrying out their correct orders to

units mistakenly initiating conflict. Our work on interactive consistency applies to fully distributed systems. But C^3I systems are not fully distributed; they tend to be at least somewhat hierarchical. Common interactive consistency protocols involve substantial communication across the system; this may lead to increased risk of message interception.

The integrated framework should provide a set of common methods that permit evaluation of each of reliability, safety, and security in depth. The integrated framework should also enhance efforts to study when decisions to augment one objective may impact others. The ability to unify the analysis of reliability, safety, and security can point out when design decisions need broader evaluation (because collateral impact is negative) and when design decisions bear additional merit (because collateral impact is positive).

Dhiraj K. Pradhan

CURRICULUM VITAE

Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003

(413) 545-0160

Positions–Academic

1/83 - present	Professor and Coordinator of Computer Systems Engineering, Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, Massachusetts.
9/78 - 12/82	Associate Professor, School of Engineering, Oakland University, Rochester, Michigan.
6/79 - 8/79	Research Associate Professor, Stanford University, Stanford, California.
9/73 - 7/78	Associate Professor, Department of Computer Science, University of Regina, Regina, Canada. (9/73–7/76, Assistant Professor).

Positions–Industrial

10/72 - 8/73	Staff Engineer, IBM, Systems Development Lab., Poughkeepsie, New York.

Honors

1990	*Humboldt Senior Scientist Award*, Germany
1989	*Fellow*, Japan Society of 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. Thesis area: Fault-Tolerant Computing
1970, M.S.	(Electrical Engineering), Brown University, Providence, Rhode Island. Thesis area: Complexity Theory

Personal

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

Professional Activities

1992 -	*Conference Chair*, International Symposium on Fault-Tolerant Computing, Boston, Mass.
1991 -	*Editor, IEEE Transactions on Computers*
1990 - 1991	*ACM Lecturer*
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
	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
1980	*Guest Editor*; Special Issue on Fault-Tolerant Computing; *IEEE Computer*, March 1980

Grants

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

Research Supervision

1978 - present	Several Ph.D. Students, placed in IBM, AT&T as well as leading universities.
1977 - 1985	Research Associates K.L. Kodandapani T. Nanya K. Matsui I. Koren

Patent

"Easily Testabie High Speed Architecture for Large RAMs", U.S. Patent No. 4,833,677, May 23, 1989.

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 1991).

In Journals:

1. "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, September 1991.
2. "Consensus with Dual Mode Failures", *IEEE Transactions on Parallel and Distributed Computers*, April 1991.
3. "A New Framework for Designing and Analyzing BIST Techniques", *IEEE Transactions on Computers*, (accepted for publication).
4. "Error Correcting Codes in Fault-Tolerant Computers" (with E. Fujiwara), *Computer*, pp. 63-72, July 1990.
5. "Aliasing Probability for a Multiple Input Signature Analyzer and a New Compression Technique" (with S. Gupta and M. Karpovsky), *IEEE Transactions on Computers*, pp. 586-591, April, 1990.
6. "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.
7. "Modeling Defect Spatial Distribution" (with F. Meyer), *IEEE Transactions on Computers*, Vol. 38, No. 4, pp. 538-546, April 1989.
8. "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.
9. "Dynamic Testing Strategy for Distributed System" (with F. Meyer), *IEEE Transactions on Computers*, Vol. 38, No. 3, pp. 356-365, March 1989.
10. "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.
11. "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*, pp. 859-866, June 1988.
12. "Flip Trees: A Fault-Tolerant Network with wide Containers" (with Fred Meyer), *IEEE Transactions on Computers*, pp. 472-478, April 1988.
13. "Modeling the Effect of Redundancy on Yield and Performance of VLSI Systems" (with I. Koren), *IEEE Transaction on Computers*, pp. 344–355, Vol. C-36, No. 3, April 1987
14. "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.
15. "Dynamically Restructurable Fault-tolerant Processor Network Architectures", *IEEE Transactions on Computers*, Vol. C–34, No. 5, pp. 434–447, May 1985.
16. "Fault-tolerant Multiprocessor Structures", *IEEE Transactions on Computers*, Vol. C-34, No. 1, pp. 33–45, January 1985.

17. "Synthesis of Directed Multi-Commodity Flow Problems" (with A. Itai), *Networks*, Vol. 14, 1984, pp. 213-224.
18. "Sequential Network Design Using Extra Inputs for Fault Detection", *IEEE Transactions on Computers*, Vol. C-32, No.3, pp. 319–323, March 1983.
19. "A Fault-Tolerant Distributed Processor Communication Architecture" (with S. Reddy), *IEEE Transactions on Computers*, Vol. C-31, No. 9, pp. 863-870, September 1982.
20. "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.
21. "A Uniform Representation of Permutation Networks Used in Memory-Processor Interconnection" (with K.L. Kodandapani), *IEEE Transactions on Computers*, Special Issue on Parallel Processing, pp. 777-791, Vol. C–29, No. 9, September 1980.
22. "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.
23. "Error-Correcting Codes and Self-Checking Circuits" (with J.J. Stiffler), *IEEE Computer*, Vol. 13, Number 3, pp. 27-38, Special Issue on Fault-Tolerant Computing, March 1980.
24. "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.
25. "Fault-Tolerant Asynchronous Networks Using Read-Only Memories", *IEEE Transactions on Computers*, Vol. C–27, No. 7, pp. 674–679, July 1978.
26. "Fault Secure Asynchronous Networks", *IEEE Transactions on Computers*, Vol. C–27, No. 5, pp. 396–404, May 1978.
27. "A Theory of Galois Switching Functions", *IEEE Transactions on Computers*, Vol. C–27, No. 3, pp. 239–249, March 1978.
28. "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.
29. "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.
30. "A Graph-Structural Approach for the Generalization of Data Management Systems", *Information Sciences*, American Elesevier Publishing Company, Inc., pp. 1–17, March 1977.
31. "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.
32. "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.
33. "Fault-Tolerant Carry Save Adders", *IEEE Transactions on Computers*, Vol. C–23, No. 11, pp. 1320–1322, November 1974.
34. "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.
35. "Fault-Tolerant Asynchronous Networks" (with S.M. Reddy), *IEEE Transactions on Computers*, Vol. C–22, No. 7, pp. 662–669, July 1973.
36. "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

37. "Program Fault-tolerance Based on Memory Access Behavior" (with Nicholas S. Bowen), *Proc. 1991 International Symposium on Fault-Tolerant Computers, pp. 426-433*, Montreal, Canada, June 1991.
38. "System Level Diagnosis: Combining Detection and Location" (with Nitin H. Vaidya), *Proc. 1991 International Symposium on Fault-Tolerant Computing, pp. 488-495*, Montreal, Canada, June 1991.
39. "A Methodology for Partial Scan Design", *Proc. Second European Test Conference*, Munich, Germany, April 1991.
40. "Byte Error Locating Codes", *IEEE USA/Japan Information Theory Symposium*, Honolulu, Hawaii, November 1990.
41. "Application Specific VLSI Architectures Based on De Bruijn Graphs", *Proc. ASAP 90*, November 1990.
42. "Modeling of Live Lines and Tree Sharing in Multi-Code Memory Systems", *Int. Conf. on Parallel Processing*, August 1990.
43. "Zero Aliasing Compression", *Proc. 1990 Int'l Symp. on Fault-Tolerant Computing*, Newcastle, U.K., July 1990.
44. "Yield Modeling and Optimization of Large Redundant RAMs" (with A.D. Singh and K. Ganapathy), *Int. Conf. on Wafer Scale Integration*, San Francisco, CA, pp. 273-287, January 1989.
45. "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 20-24, 1989.
46. "RTRAM: Reconfigurable and Testable Multi-bit RAM Design", *International Test Conference*, Washington, DC, pp. 263-278, Sept. 1988.
47. "A New Framework for Designing and Analyzing BIST Techniques: Computation of Exact Aliasing Probability", *International Test Conference*, Washington, DC, pp. 329-340, Sept. 1988.
48. "Consensus with Dual Failure Modes" (with F.J. Meyer), *Proc. FTCS-17*, Pittsburgh, pp. 48-54, July 1987.
49. "Cost Analysis of OnChip Error Control Coding for Fault-Tolerant Dynamic RAMs" (with N. Jarwala), *Proc. FTCS-17*, Pittsburgh, pp. 278-283, July 1987.
50. "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.
51. "Organization and Analysis of Gracefully-Degrading Interleaved Memory Systems" (with K. Cheung, G. Sohi, K. Saluja), *Proc. 14th International Symposium on Computer Architecture*, Pittsburgh, June 1987.
52. "An Easily Testable Architecture for Multimegabit RAMs" (with N. Jarwala), *Proc. of International Test Conference*, Washington, September 1987.
53. "Introducing Redundancy into VLSI Designs for Yield and Performance Enhancement" (with Israel Koren), *Proc. FTCS-15*, pp. 330–334, Ann Arbor, Michigan, June 1985.
54. "Dynamic Testing Strategy for Distributed Systems" (with Fred Meyer), *Proc. FTCS-15*, Ann Arbor, Michigan, June 1985.
55. "A Versatile Sorting Network" (with M.R. Samatham), *Proc. 12th Annual Symposium on ComputerArchitecture*, pp. 360–367, June 1985.
56. "Fault-tolerant Multibus Architectures for Multiprocessors" (with M.L. Schlumberger and Z.Hanquan), *Proc.FTCS-14*, Kissimee, Florida, pp. 400-408, June 1984.
57. "A Multiprocessor Network Suitable for Single Chip VLSI Implementation", *Proc. 1984 IEEE 11th Annual Int. Symp. on Computer Architecture*, June 1984, pp. 328-337.

58. "Fault-Tolerant Network Architectures for Multiprocessors and VLSI Based Systems", *Proc. FTCS-13*, Milan, Italy, pp. 436–441, June 1983.
59. "On a Class of Multiprocessor Network Architectures", *Proc. of International Conference on Distributed Processing*, Miami, Florida, October 1982, pp. 302-311, (Also reprinted in Interconnection Networks for Parallel and Distributed Processing edited by C. Wu and T. Feng, Aug. 1984).
60. "Interconnections Topologies for Fault-Tolerant Parallel and Distributed Architectures", *Proc. of 10th International Conference on Parallel Processing*, pp. 238-242, August 1981.
61. "Testing for Delay Faults in a PLA" (with K. Son), *Proc. International Conference on Circuits and Computers*, pp. 346-349, September 1982.
62. "Fault-Diagnosis of Parallel Processor Interconnection Networks" (with K.M. Falavarajani), *Proc. Eleventh Annual International Symposium on Fault-Tolerant Computing*, pp. 209-212, June 1981.
63. "A Fault-Tolerant Communication Architecture for Distributed Systems", *Proc. Eleventh International Conference on Parallel Processing*, pp. 214-220, June 1981.
64. "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.
65. "A Fault–Diagnosis Technique for Closed Flow Networks", *Proc. of 1980 Symposium on Fault-Tolerant Computing*, Kyoto, Japan, October 1980.
66. "Completely Self–Checking Checkers" (with K. Son), *Digest of 1981 Test Conf.*, pp. 231–237, October 1981.
67. "Effect of Undetectable Faults on Testing PLAs" (with K. Son), *Digest of 1980 Test Conf.*, pp. 359–367, November 1980.
68. "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).
69. "A Generalization of Shuffle-Exchange Networks", *Proc. of Fouteenth Annual Conference on Information Sciences and Systems*, Princeton, New Jersey, March 1980.
70. "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.
71. "Shift Registers Designed for On-Line Fault Detection", *Proc. of 1978 International Symposium on Fault Tolerant Computing*, Toulouse, France, pp. 173–178, June 1978.
72. "A Synthesis Algorithm of Directed Two-Commodity Networks", *1978 IEEE International Symposium on Circuits and Systems*, New York, pp. 93–98, May 17–19, 1978.
73. "Error Control Techniques for Array Processors", *1977 International Symposium on Information and Theory*, Ithaca, New York, October 1977.
74. "Fault-Tolerant Fail-Safe Logic Networks" (with S.M. Reddy), *Proceedings on IEEE Compcon*, pp. 363–366, March 1977.
75. "On Undetectability of Bridging Faults" (with K.L. Kodandapani), *Proceedings of 1977 International Symposium on Fault-Tolerant Computing*, Los Angeles, California, June 1977.
76. "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.
77. "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.
78. "Fault-Tolerant Asynchronous Networks Using (2,1)-Type Assignments", *Digest of Fifth International Symposium on Fault-Tolerant Computing*, Paris, France, June 1975.

79. "Construction on Error Correcting Codes with Run-Length Limited Properties", presented in *1974 International Symposium on Information and Theory*, Notre Dame, Indiana, November 1974.
80. "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 .
81. "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.
82. "On Fault-Diagnosis of Sequential Machines", *Proc. VI Hawaii Conference on System Sciences*, Western Periodicals, January 1973.
83. "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.