2

# REPORT DOCUMENTATION PAGE

DULE

1b. RESTRICTIVE MARKINGS

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for public release;
distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)
AFOSR-TR- 89-0782

6a. NAME OF PERFORMING ORGANIZATION
University of Massachusetts

6b. OFFICE SYMBOL (If applicable)

7a. NAME OF MONITORING ORGANIZATION
AFOSR/NM

6c. ADDRESS (City, State, and ZIP Code)
Amherst, MA 01003

7b. ADDRESS (City, State, and ZIP Code)
AFOSR/NM
Bldg 410
Bolling AFB DC 20332-6448

8a. NAME OF FUNDING/SPONSORING ORGANIZATION
AFOSR Grant No: 87-0161

8b. OFFICE SYMBOL (If applicable)
NM

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
AFOSR-87-0161

8c. ADDRESS (City, State, and ZIP Code)
AFOSR/NM
Bldg 410
Bolling AFB DC 20332-6448

10. SOURCE OF FUNDING NUMBERS

| PROGRAM ELEMENT NO. | PROJECT NO. | TASK NO. | WORK UNIT ACCESSION NO. |
|---|---|---|---|
| 61102F | 2304 | A2 | |

11. TITLE (Include Security Classification)
Fault Tolerant Multiprocessors and VLSI-Based Systems

12. PERSONAL AUTHOR(S)
Dr. Dhiraj Pradhan

13a. TYPE OF REPORT
Final

13b. TIME COVERED
FROM 2/15/87 TO 2/15/88

14. DATE OF REPORT (Year, Month, Day)
3/16/88

15. PAGE COUNT
18

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

| FIELD | GROUP | SUB-GROUP |
|---|---|---|
| | | |
| | | |

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Two significant aspects of fault-tolerant computing were the focus of this project. Concurrent research was carried out as well in the areas of fault-tolerant testable VLSI system design and fault-tolerant multiprocessor design. A novel concept for testable RAM designs was developed, too, allowing for the design of large RAMs with built-in test capabilities. Such a testability feature is, in fact, an integral part of the design, not added on adhoc, and as such, is the subject of a patent application filed by the U.S. Air Force.

The second major focus of research concentrated on the development of fault-tolerant multiprocessor topologies. It was demonstrated that DeBruijn multiprocessor networks provide a naturally fault-tolerant robust interconnection network. The attractive feature of these networks includes their ability to provide fault-tolerance in a wide variety of applications. Also developed was a new topology, termed Flip Trees, which provides certain optimal fault-tolerant properties. Finally, a practical perspective on distributed agreement algorithms was formulated, which can admit a large variety of faults.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
□ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. □ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL
Dr. Abraham Waksman

22b. TELEPHONE (Include Area Code)
(202) 767-5027

22c. OFFICE SYMBOL
NM

83 APR edition may be used until exhausted.
All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

# Table of Contents

AFOSR-TR- 89 - 0782

## Abstract

Two significant aspects of fault-tolerant computing were the focus of this project. Concurrent research was carried out as well in the areas of fault-tolerant testable VLSI system design and fault-tolerant multiprocessor design. A novel concept for testable RAM designs was developed, too, allowing for the design of large RAMs with built-in test capabilities. Such a testability feature is, in fact, an integral part of the design, not added on adhoc, and as such, is the subject of a patent application filed by the U.S. Air Force.

The second major focus of research concentrated on the development of fault-tolerant multiprocessor topologies. It was demonstrated that DeBruijn multiprocessor networks provide a naturally fault-tolerant robust interconnection network. The attractive feature of these networks includes their ability to provide fault-tolerance in a wide variety of applications. Also developed was a new topology, termed Flip Trees, which provides certain optimal fault-tolerant properties. Finally, a practical perspective on distributed agreement algorithms was formulated, which can admit a large variety of faults.

# 1 Introduction

This report summarizes the research carried out under AFOSR 87-0161. Three distinct and major achievements occurred under this sponsorship. First, it was demonstrated that the DeBruijn multiprocessor network provides an efficient fault-tolerant architecture that can be useful for a wide variety of application. This result is the thrust of a paper to appear shortly in *IEEE Transactions on Computers.* Second, a new concept in the area of RAM design was developed which allows for the design of large RAMs that are both testable and defect tolerant (Another serendipitous result observed is that the proposed design also achieves higher levels of performance than the traditional design). The design is the basis of both a patent application, as well as a pending publication.

The problem of achieving consensus in a distributed system was also explored. Investigated were the systems where two types of faults can occur: benign (omission and timing faults) and malicious (exhibiting arbitrary behavior). A continuum was established between the previous results when no malicious faults are present and when at most one third of the nodes are faulty. Additional research was conducted in the area of VLSI yield models.

The following section elaborates all of the above research carried out. Section III lists all resulting publications, and all student supported. Section IV briefly overviews future research directions and plans.

# 2 Summary of Research Results

## 2.1 Research in Fault-Tolerant Multiprocessor Architectures

The search for computationally efficient multiprocessor architectures that are suitable for VLSI has spawned an increasingly important research area. Several parallel architectures which solve a wide variety of problems have been proposed. These include the linear array [1][1], the ring [1], the complete binary tree (CBT), the tree machine (TM), the shuffle-exchange (SE), the cube-connected cycles (CCC), the two-dimensional mesh, the even double-exchange, the orthogonal trees and the doubly-twisted torus.

Certain of the real life problems that must be accomodated have been successfully

[1] The numbers in parenthesis refers to publication numbers in Section 3

grouped into various classes. These classifications are important because statements such as, "all problems in this class have this complexity" inform much more than statements like "this problem is of this complexity". Included, among these classifications are the pipeline class, the multiplex class , the NP-complete class, the ASCEND and DESCEND classes, as well as the decomposable searching class.

Firstly, problems in the pipeline class can be efficiently solved in a pipe (linear array). Depending on the problem, data may flow in one direction or in both directions simultaneously. Matrix-vector multiplication is a typical example of those problems that can be solved by one-way pipeline algorithms. Band matrix-vector multiplication, recurrence evaluation and priority queues are good representations of problems that can be solved by two-way pipeline algorithms.

The multiplex class covers a range of problems characterized by [1]: (1) Operation on N data operands to produce a single result; (2) Evaluation that can be described by a tree. Evaluation of general arithmetic expressions, polynomial evaluation, etc. is included in this category. The natural computation graph for this paradigm is a tree whose nodes correspond to operations and whose edges correspond to data flow between operations. The CBT (complete binary tree) can be used to solve the problems inherent in this class.

Another important class of problems is the NP-complete class [24]. For this class, the CBT can efficiently implement exhaustive search algorithms [1]. Here, time complexity still is exponential.

ASCEND and DESCEND classes are comprised of highly parallel algorithms [15]. Here, the paradigm of the algorithms is the iterative rendition of a divide-and-conquer scheme. The input and output are each a vector of $N(= 2^k)$ data items; 'divide' refers to two subproblems of equal size, where the "marry step" combines the results of two subproblems consisting of the execution of a single operation on the corresponding pairs of data items. That is: assume that input data $D_0, D_1, ...D_{N-1}$ are stored, respectively, in storage location $T[0], T[1], ..., T[N-1]$. An algorithm in the DESCEND class performs a sequence of basic operations on pairs of data successively $2^{k-1}, 2^{k-2}, ..., 2^1, 2^0$ locations apart. In terms of the above divide-and-conquer model, the marry step involves pairs of $2^0$ locations apart. On the other hand, in the dual class (the ASCEND class), basic operations are performed on the data that are successively $2^0, 2^1, ..., 2^{k-1}$ locations apart;

the marry step involves pairs of $2^{k-1}$ locations apart, problems which can be solved in the SE and the CCC.

Problems in the decomposable searching class can be described as illustrated below [1]. Preprocess a set, F, of N objects into a data structure, D, such that certain kinds of queries about F can be answered quickly. A searching problem is decomposable if the response to a query, Q, asking the relation of an object, z, to the set, F, can be written as $Q(z, F) = \Delta q(z, f)$, for all f in F, where f is an element in F; $\Delta$ is a binary operator which is associative, commutative, and has an identity; and where q is the query asking the relation of the object z to the element f. The TM described in [1] solves this large class of searching problems. We demonstrate in [1] that multiprocessor networks based on binary de Bruijn graphs referred to as binary de Bruijn multiprocessors (BDM) and general de Bruijn graphs referred to as de Bruijn multiprocessors (DM) can solve all of the above classes of problems efficiently. Additionally, it is shown that these multiprocessor networks can be used as versatile sorting networks.

Sorting is a theoretically interesting problem with a great deal of practical significance. The sorting problem as defined is described below. We are given $N$ items;

$$D_1, D_2, D_3, ..., D_{N-1}, D_N$$

to be sorted; we shall call them data items (or records). Each data item, $D_j$, has a key, $K_j$, which governs the sorting process. The object of such sorting is to determine a permutation $p(1)p(2) \cdots p(N)$ of the data items, which puts the keys in nondecreasing order;

$$D_{p(1)} \leq D_{p(2)} \leq \cdots \leq D_{p(N)}$$

Usually, we output the sorted sequence or the $i$th smallest item is placed in the $i$th processor.

A classification method for sorting architectures was presented by Winslow and Chow. The sorters have been classified into the following categories:

(A) Sequential Input/Sequential Output (SI/SO)

(B) Parallel Input/Sequential Output (PI/SO)

(C) Parallel Input/Parallel Output (PI/PO)

(D) Sequential Input/Parallel Output (SI/PO)

(E) Hybrid Input/Hybrid Output (HI/HO)

Note that the classification is based not only on the I/O method, but also on the interconnection network, the sorting algorithm and the type of keys used.

Our paper [1] demonstrates that the de Bruijn multiprocessor networks can be used to sort elements in all of the five categories. The main advantages are four-fold in having an interconnection network which can sort data items in all of the categories, as the following situations can be handled:

1. Although it is theoretically possible to load the data items in parallel, the number of ports available for I/O may be limited.

2. Even though the I/O ports are available, it may not be possible to load the data items, from the secondary storage, in parallel.

3. Different sets of data may have different types of keys.

4. In practice, there may be faults in the network.

For each of these categories, time complexity and size complexity is given in [1], *time complexity* being the worst case time required to sort the data items, *size complexity*, the number of data items that can be sorted.

## 2.2 Research in VLSI Systems

During this past year, we have addressed the problems of developing methodologies for the design VLSI systems that are defect/fault-tolerant and testable, and of evaluating their cost/performance [2,3]. Today's complex VLSI systems place additional burdens on the designer - demanding that designs not only meet specifications, but are also testable, reliable, and can be fabricated with reasonable yields. Design methodologies are therefore essential which enhance both the testability and yield of VLSI systems. Also, it is important to be able to provide the designer with certain feedback, early in the design cycle, ensuring that correct decisions regarding testability and yield enhancement can be made.

Finally, we must be able to compare the different techniques available for enhancing these metrics so that, for a given application, a suitable technique can be chosen.

The proposed design methodology for future multi-megabit DRAMs, has the following properties [2]:

- **Easily Testable:** Dividing the nodes into modules reduces the size of the problem and reduces the test time. Testing these nodes in parallel, as well as having on-chip test evaluation further reduces the test time. Therefore, what this architecture results in is practical testing times for multi-megabit RAMs.

- **Low Area Overhead:** The additional area required for large RAMs is typically 8–20%. Depending on the total area allowed, the designer can select the appropriate node granularity, taking into consideration the other tradeoffs involved.

- **Improved Performance:**

- For large RAMs, this architecture can be faster a potential reduction in access time is about 30%. Because performance enhancement also depends upon node granularity, it can be selected by the designer.

- Refreshing the nodes in parallel substantially reduces the amount of time the RAM is not available. Such a reduction in refresh time has to be traded off against an increase in power dissipation that results from all nodes being refreshed simultaneously.

- **Partitionable and Restructurable:** These two properties make it possible to salvage defective chips as partially good chips; enhances effective wafer yield, making fabrication of very large memory systems economically viable. Because restructuring involves address remapping, it can be used as part of a highly reliable, self-test, self-repair system, where the tester can program the address map in real time.

Those VLSI models developed to evaluate the cost/performance of the TRAM architecture were used to evaluate the cost/performance when error control codes were used to protect DRAMs against soft errors, and to enhance defect tolerance. Two codes - the product code with full code word correction and the odd-weight-column code, were analyzed for DRAMs. The important advantage these codes possess us that they are free of high error latency and large time to scrub the RAM of proposals. The code analysis demonstrated that in spite of an increase in area, the yield is, in fact, enhanced. Also, there is only a moderate performance penalty for implementing either the odd weight code or the full product code, for large RAMs. However, the odd-weight code has better yield and lower performance cost that the product code [2].

The TRAM architecture can be used as a basic module in a wafer scale memory system, discussed in detail in the section that follows. The TRAM VLSI models can be used as benchmarks to evaluate and compare the cost/performance of other defect/fault-tolerant, testable systems. The methodology can then be used to develop benchmarks for other architectures. Also, the VLSI modeling technique can be developed into a design tool that can aid researchers and designers to estimate the cost/performance of VLSI systems, as well as integrating knowledge about testability techniques.

## 2.3 Research in Fault-Tolerant Multiprocessor Distributed Systems

In distributed systems, our work is strongly dependent on the graph model used to describe the system. The nodes (communication links) of the system are reflected by the vertices (edges) of a graph. We have enumerated several desirable properties of a system graph (low internode distances, high connectivity, easy routing, etc.). Mostly researchers seek graphs that are best for only one desirable property, but those graphs obtained are usually of no practical use because they lack relation to the other properties. This very problem we addressed this problem in [4]. We also, gave a very general method for constructing graphs, termed color product construction which can generate many well-known graphs (such as hypercube, cube-connected cycles, and generalized Petersen). We were able to show that colored product graphs are, indeed, a fertile field for graphs, good for every one of the desirable graph properties we have enumerated.

In addition, we have examined some other particular graphs, developing a family of graphs called flip-trees, obtained by interconnecting the leaves of Moore trees. Flip-trees are regular and have optimal connectivity; if a graph is c-connected, then between every pair of nodes there exists a set of c node-disjoint paths. We refer to such a set of node-disjoint paths as a container. The length of a container is dictated by the path in it of worst length. The best containers (shortest) of width c are of interest, especially when c is the connectivity. In [5], we demonstrated that flip-trees have containers of maximal width, with length one more than twice the Moore bound. Flip-trees are the graphs with the best-known containers.

A practical perspective on distributed agreement algorithms is given in [5]. There, we showed that more faults can be accommodated (than in prior work) if some faults are of a less serious nature. The algorithm given does not require a priori knowledge of less severe faults. An interesting aspect of this work are the requirements to establish communication between any two nodes. Messages are sent in a container so that each fault may only destroy/corrupt one message. A simple protocol, along with relaying rules for intermediate nodes establish a virtual link between any two nodes. Separate protocols are given for the case where the receiver knows a message will be sent, and for the case where the receiver does not know when the next message will be sent. The protocols achieve communication wherever the faults in the system are few enough to allow it. We intend to further break down faults according to severity, so as to be able to assess whether further gains is feasible. Also we intend to analyze and compare different methods of establishing virtual links (various coding schemes, sending multiple messages, etc.).

Work we have previously reported on, on System-level diagnosis will appear in [7]. We are interested in evaluating the usefulness of containers in distributed diagnosis. A probabilistic approach will allow us to ascribe behavior to faulty nodes, instead of assuming they can/cannot find other nodes faulty. An interesting question is how much is lost if only the tests within a container are performed instead of all possible tests. Also, a metric is needed to consistently interpret the results.

## 2.4 Other Research Designing Buses for Maximum Yield and Minimum Delay

It has been common in recent publications concerned with fault-tolerance in VLSI and WSI to assume that interconnection buses can be design to be almost defect-free by enlarging the width of the lines and spacing between lines. Although this assumption may often be valid, the cost-effectiveness of this proposed "robust" bus layout is questionable, particularly in the case of wide buses (e.g. 32 bit wide).

In our paper [8], we derive exact expressions for the yield of an interconnection bus, as a function of its physical dimensions and the parameters and distribution of the possible open-circuit and short-circuit defects. Also examined is the effect of introducing redundancy into the bus, as well as obtaining the optimal layout of a given bus (with and without redundancy).

Any change in the layout of a bus may affect the propagation delay of the bus and as a consequence, the performance of the VLSI chip. Hence, in addition to its yield, the delay of the designed bus must be taken into account when determining the final layout of the bus. Both yield and delay are discussed in this paper through several examples.

# 3 Publications and Students Supported

## 3.1 Publications

1. D.K. Pradhan and M.R. Samatham, "The DeBruijn Multiprocessor Network: A Versatile Parallel Processing and Sorting Network for VLSI", *IEEE Transactions on Computers*, (to appear).

2. N. Jarwala and D.K. Pradhan, "TRAM: A Design Methodology for Testable Fault-Tolerant Large RAMs", *IEEE Transactions on Computers*, Vol. C-37, Oct. 1988, U.S. Patent Pending.

3. N. Jarwala and D.K. Pradhan, "Cost Analysis of On-Chip Error Control Coding for Fault-Tolerant Dynamic RAMs", *Proc. of 17th International Symposium on Fault-Tolerant Computing*, Pittsburgh, PA, July 6-8, 1987, pp. 278-282.

4. D.K. Pradhan and F.J. Meyer, "Communication Structures in Distributed Systems," *Proc. 10th Fault-Tolerant Systems and Diagnostics Conf.*, Varna, Bulgaria, pp. 193-202, September 1987.

5. F.J. Meyer and D.K. Pradhan, "Flip-Trees: Fault Tolerant Graphs with Wide Containers," *IEEE Transactions on Computers*, Vol. C-37, No. 4, (to appear), April, 1988.

6. F.J. Meyer and D.K. Pradhan, "Consensus with Dual Failure Modes," *Proc. 17th Int. Symp. on Fault-Tolerant Comput.*, Pittsburgh, PA, pp. 48-54, July 1987.

7. F.J. Meyer and D.K. Pradhan, "Dynamic Testing Strategy for Distributed Systems," *IEEE Transactions on Computers*, (to appear).

8. I. Koren, Z. Koren and D.K. Pradhan, "Designing Interconnection Buses in VLSI and WSI for Maximum Yield and Minimum Delay", *IEEE Journal of Solid State Circuits*, (to appear), May 1988.

### 3.2 Students Supported

- Fred Meyer, Ph.D. Student

- Najmi Jarwala, Ph.D. Student

## 4 Future Research

The following elaborates on tractable future research directions. The following research is being carried out in the area of defect/fault-tolerant VLSI system design.

- Studying the problem of designing defect/fault tolerant, testable VLSI systems, a study emphasizes DRAMs.

- Developing an architecture for Multi-Megabit DRAMs that is easily testable, partitionable, and restructurable.

- Developing a fault model and test algorithms.

- Studying the issues involved in developing a VLSI macro-modeling technique, so as to analyze and compare the cost/performance of VLSI structures. Developing VLSI models that compute the cost/performance tradeoffs of the memory architecture.

- Analyzing the yield potential of the above architecture.

- Studying the suitability of the above architecture for Wafer Scale Integration.

- Exploring the cost/performance tradeoff in using error correcting codes to protect DRAMs against soft errors by using the modeling technique earlier developed. Analyzing the effect on yield. Exploring codes other than the commonly-used product codes for this application.

With respect to reliable broadcast, the reliability achieved with the mixed algorithm is always superior to both benign and malicious algorithms [6]. A mixed-sum algorithm was introduced in [6] which achieves the provably maximal reliability (under the dual failure mode model), but the message complexity of the algorithm is substantial. Further research should be devoted to finding a more feasible mixed-sum algorithm, as substantial:

We assumed that the probability of a failure and that the probability of failure malicious are independent of the algorithm used. Actually, an electrical fault would appear malicious to a *benign algorithm* more often than to a *malicious algorithm*, because a benign algorithm is simpler. For purposes of comparison, though, this probability, $P(m)$, was taken to be the same for each algorithm. Also, the probability of failure has two components—due to permanents and due to transients. The impact of permanent faults can be expected to remain largely the same for any algorithm, but the impact of transients is related to the amount of message traffic. This decomposition of faults has yet to be modelled.

With respect to communication topologies, a big advance cannot be made by finding individual graphs and sequences of graphs; general construction methods are needed that ensure reasonable graphs with respect to all important parameters. A construction method has been suggested here that could fill this need, as could other similar methods. Further efforts should concentrate on such methods. Many good graphs are available via colored product construction, but these need to be developed:

(1) A thorough enumeration of the useful colored product graphs is in progress.

(2) The fault-tolerant diameters of many colored product graphs are known, but general results establishing the fault-tolerant diameters are needed.

(3) Of interest are colored graphs using a palette with colors that do not commute.

Flip-trees were shown in [5] to be competitive with respect to many aspects of network topologies, such as diameter and fault-tolerant diameter; as they possess the best-known containers. The primary areas of deficiency are: (1) traffic congestion and (2) distributed routing with localized routing information. Further research is underway to address such deficiencies.

While it is understood that undetectable faults can cause difficulties, the extent of that problem is not known. Also, it is not clear whether countermeasures (test set extension, addition of observation points, etc.) are worthwhile. We see this area of research developing along the following lines:

(1) provide *analytic* function to assess the impact of undetectable faults

It will be necessary to model the impact of undetectable faults that are known as well as those that are unknown. Known undetectable faults are important, because: (1) products may already be deployed in the field when redundant design is discovered and (2) considerations such as circuit delay might dictate a redundant design. Undetectable faults that are not known are important because the problem of deciding whether a fault is detectable or intractable. Knowing the location of redundant leads can be used in two direct ways: (a) test generation can include a check for double faults involving one of the suspect leads, and also include a search for additional tests, if necessary and (b) FRUs with the most undetectable faults are the most likely source of failure when diagnostics do not isolate the failed FRU.

(2) extend analysis to predict reliability of chip at run time

Many faults that occur in the field elude detection. We conjecture that some of these faults are second faults (with the first fault being an undetectable fault). The first (undetectable) fault might occur in the field or during manufacture. If field service encounters such a circumstance, then it may not be able to isolate the FRU. If the chip has a BIST feature, then its test set may be invalid due to the manufacturing defect that was not detectable.

(3) provide tools to predict the effectiveness of countermeasures

One key consideration is the accuracy of attempts to locate redundant leads. Without a good knowledge of the untestable and/or difficult-to-test portions of a circuit, we see no effective way to prevent test invalidation. One of the more promising countermeasures is to aid observability by adding observation points. This would be especially frugal for a BIST methodology, because no extra pins would be needed for the package.

# Biography

## Dhiraj Pradhan

Dr. Dhiraj Pradhan is currently a Professor in the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst. Previously, he has held positions with the University of Regina, Sask, Canada, Oakland University, Rochester, Michigan, and IBM Corporation. He also was a visiting research professor at Stanford University, California and has served as a consultant to various industries.

Since receiving his Ph.D. from the University of Iowa in 1972, he has been actively involved with research in fault-tolerant computing, testing, computer architecture and parallel processing, publishing numerous papers in these areas. He is the editor and co-author of the book entitled, *Fault-Tolerant Computing: Theory and Techniques*, Vol. I and II, published by Prentice-Hall (1986).

Dr. Pradhan edited the special issue on Fault–Tolerant Computing, published in *IEEE Transactions on Computers* (April 1986) and *IEEE Computer*, (March 1980). Also he has served as Session Chairman and Program Committee Member for various conferences. Recently he was the Co-chairman for the 1987 IEEE Workshop on Fault-Tolerance in Parallel and Distributed Computing.

Dr. Pradhan has served as a Distinguished Visitor for the IEEE Computer Society and was elected in 1987 to be a Fellow of the IEEE for contributions to the design of fault-tolerant circuits and systems.

# Dhiraj K. Pradhan

*CURRICULUM VITAE*

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

(413) 545-0160

## Personal

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

## 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

## Positions–Academic

1/83 – present Professor and Coordinator of Computer Systems Engineering; Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, Massachusetts.

6/79 – 8/79 Research Associate Professor; Stanford University; Computer Systems Lab.; Stanford, California.

9/78 – 12/82 Associate Professor; School of Engineering, Oakland University, Rochester, Michigan.

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

1988 *Fellow*, IEEE, "For contributions to techniques and theory of designing fault-tolerant circuits and systems"

1982 – 1985 *IEEE Distinguished Visitor,* Computer Society

## Professional Activities

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 – present *Editor, Advances in VLSI Systems*, Computer Science Press, Maryland

1981 – 1987 *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.

1980 *Guest Editor*; Special Issue on Fault-Tolerant Computing; *IEEE Computer*, March 1980.

1982 – *Consultant* to Mitre, CDC, AT&T, DEC and Data General

## Grants

1973-present Multiple Grants from NSF, AFOSR, SRC and NRC (Canada)

## Research Supervision

1978 – present Several Ph.D. and M.S. Students.

1977 – 1985 Research Associates
K.L. Kodandapani
T. Nanya
K. Matsui
I. Koren

**Patent:** "A Testable RAM Design", Patent Office Serial No: 60,882, June 1987.

## List of Publications

### Text Book

*Fault-tolerant Computing: Theory and Techniques*, Vol. I and Vol. II, Prentice-Hall, Inc., May 1986. (Editor and Co–Author).

### In Journals:

1. "Flip Trees: A Fault-Tolerant Network with wide Containers", *IEEE Transactions on Computers*, (with Fred Meyer), April 1988.
2. "Designing Interconnection Buses in VLSI and WSI for Maximum Yield and Minimum Delay", *IEEE Journal of Solid State Circuits*, (with I. Koren and Z. Koren), in press.

3. "Dynamic Testing Strategy for Distributed System", *IEEE Transactions on Computers*, (to appear).
4. "A Design Methodology for High Performance Testable Large RAMs", *IEEE Transactions on Computers*, (with N. Jarwala), October 1988.
5. "The DeBruijn Multiprocessor Networks: A Versatile Parallel Processing Network for VLSI", *IEEE Transactions on Computers*, (with M. Samatham) (to appear).
6. "Modeling the Effect of Redundancy on Yield and Performance of VLSI Systems," *IEEE Transaction on Computers*, pp. 344–355, Vol. C-36, No. 3, March 1987 (with I. Koren).
7. "Yield and Performance Enhancement through Redundancy in VLSI and WSI Multiprocessor Systems," *IEEE Proceedings*, Vol. 74, No. 5, pp. 699–711, (with I. Koren) May 1986.
8. "Dynamically Restructurable Fault-tolerant Processor Network Architectures," *IEEE Transactions on Computers*, Vol. C–34, No. 5, pp. 434–447, May 1985.
9. "Fault-tolerant Multiprocessor Structures," *IEEE Transactions on Computers*, Vol. C-34, No. 1, pp. 33–45, January 1985.
10. "Synthesis of Directed Multi-Commodity Flow Problems," *Networks*, (with A. Itai) Vol. 14, 1984, pp. 213-224.
11. "Sequential Network Design Using Extra Inputs for Fault Detection," *IEEE Transactions on Computers*, Vol. C-32, No.3, pp. 319–323, March 1983.
12. "A Fault-Tolerant Distributed Processor Communication Architecture," *IEEE Transactions on Computers*, Vol. C-31, No. 9, pp. 863-870 (with S. Reddy), September 1982.
13. "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 (with B. Bose), June 1982.
14. "A Uniform Representation of Permutation Networks Used in Memory- Processor Interconnection," *IEEE Transactions on Computers*, Special Issue on Parallel Processing, (with K.L. Kodandapani), pp. 777-791, Vol. C–29, No. 9, September 1980.
15. "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.
16. "Error-Correcting Codes and Self-Checking Circuits," *IEEE Computer*, Vol. 13, Number 3, pp. 27-38, Special Issue on Fault–Tolerant Computing, (with J.J. Stiffler), March 1980.
17. "Undetectability of Bridging Faults and Vaildity of Stuck-at Fault Test Sets," *IEEE Transactions on Computers*, Vol. C-29, No. 1, (with K.L. Kodandapani) p. 55-59, January 1980.
18. "Fault–Tolerant Asynchronous Networks Using Read–Only Memories," *IEEE Transactions on Computers*, Vol. C–27, No. 7, pp. 674–679, July 1978.
19. "Fault Secure Asynchronous Networks," *IEEE Transactions on Computers*, Vol. C–27, No. 5, pp. 396–404, May 1978.
20. "A Theory of Galois Switching Functions," *IEEE Transactions on Computers*, Vol. C–27, No. 3, pp. 239–249, March 1978.
21. "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.
22. "Store Address Generator with Built-In Fault Detection Capabilities," *IEEE Transactions on Computers*, Vol. C 26. No. 11, pp. 1144–1147, (with M.Y. Hsiao & A.M. Patel), November 1977.
23. "A Graph–Structural Approach for the Generalization of Data Management Systems," *Information Sciences*, American Elesevier Publishing Company, Inc., pp. 1–17, March 1977.
24. "Techniques to Construct (2,1) Seperating Systems from Linear Codes," *IEEE Trans-*

*actions on Computers*, (with S.M. Reddy), Vol. C–25, No. 9, pp. 945–949, September 1976.

25. "Reed–Muller Canonic Forms for Multivalued Functions," *IEEE Transactions on Computers*, (with A.M. Patel), Vol. C–24, No. 2, pp. 206– 220, February 1975.
26. "Fault–Tolerant Carry Save Adders," *IEEE Transactions on Computers*, Vol. C–23, No. 11, pp. 1320–1322, November 1974.
27. "Design of Two–Level Fault–Tolerant Networks," *IEEE Transactions on Computers*, Vol. C–23, No. 1, pp. 41–48, (with S.M. Reddy), June 1974.
28. "Fault–Tolerant Asynchronous Networks," *IEEE Transactions on Computers*, Vol. C–22, No. 7, pp. 662–669, (with S.M. Reddy), July 1973.
29. "Error Correcting Techniques for Logic Processors", *IEEE Transactions on Computers*, Vol. C–21, No. 12, pp. 1331–1335, (with S.M. Reddy), December 1972.

**In Conference Proceedings**

30. "Wafer-Scale Integration of Multiprocessor Systems", *Proc. HICSS-20*, pp. 13-20, January 1987, (with I. Koren and Z. Koren).
31. "Consensus with Dual Failure Modes," *Proc. FTCS–17*, Pittsburgh, (with F.J. Meyer), pp. 48-54, July 1987.
32. "Cost Analysis of OnChip Error Control Coding for Fault–Tolerant Dynamic RAMs", *Proc. FTCS–17*, Pittsburgh, (with N. Jarwala), pp. 278-283, July 1987.
33. "Wafer–Scale Integration of Multiprocessor Systems", *Proc. of HICSS–20 Hawaii International Conference on System Sciences*, pp. 13–20, January 1987, (with I. Koren and Z. Koren).
34. "Organization and Analysis of Gracefully–Degrading Interleaved Memory Systems", *Proc. 14th International Symposium on Computer Architecture*, (with K. Cheung, G. Sohi, K. Saluja), Pittsburgh, June 1987.
35. "An Easily Testable Architecture for Multimegabit RAMs", *Proc. of International Test Conference*, Washington, (with N. Jarwala), September 1987.
36. "Introducing Redundancy into VLSI Designs for Yield and Performance Enhancement," *Proc. FTCS–15*, pp. 330–334, Ann Arbor, Michigan (with Israel Koren), June 1985.
37. "Dynamic Testing Strategy for Distributed Systems," *Proc. FTCS-15*, Ann Arbor, Michigan, (with Fred Meyer), June 1985.
38. "A Versatile Sorting Network," *Proc. 12th Annual Symposium on Computer Architecture*, (with M.R. Samatham), pp. 360–367, June 1985.
39. "Fault-tolerant Multibus Architectures for Multiprocessors," *Proc. FTCS-14*, Kissime, Florida, (with M.L. Schlumberger and Z. Hanquan) pp. 400-408, June 1984.
40. "A Multiprocessor Network Suitable for Single Chip VLSI Implementation," *Proc. 1984 IEEE 11th Annual Int. Symp. on Computer Architecture*, June 1984, pp. 328-337.
41. "Fault-Tolerant Network Architectures for Multiprocessors and VLSI Based Systems," *Proc. FTCS-13*, Milan, Italy, pp. 436–441, June 1983.
42. "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).
43. "Interconnections Topologies for Fault-Tolerant Parallel and Distributed Architectures," *Proc. of 10th International Conference on Parallel Processing*, pp. 238-242, August 1981.
44. "Testing for Delay Faults in a PLA," *Proc. International Conference on Circuits and Computers*, (with K. Son) pp. 346-349, September 1982.
45. "Fault-Diagnosis of Parallel Processor Interconnection Networks," *Proc. Eleventh An-*

*nual International Symposium on Fault-Tolerant Computing*, pp. 209-212, (with K.M. Falavarajani), June 1981.

46. "A Fault-Tolerant Communication Architecture for Distributed Systems," *Proc. Eleventh International Conference on Parallel Processing*, pp. 214-220, June 1981.
47. "A Solution to Load-Balancing and Fault Recovery in Distributed Systems," *Symposium on Reliability in Distributed Software and Database Systems*, pp. 89-94, (with K. Matsui), July 1981.
48. "A Fault–Diagnosis Technique for Closed Flow Networks," *Proc. of 1980 Symposium on Fault–Tolerant Computing*, Kyoto, Japan, October 1980.
49. "Completely Self–Checking Checkers," *Digest of 1981 Test Conf.*, (with K. Son), pp. 231–237, October 1981.
50. "Effect of Undetectable Faults on Testing PLAs," *Digest of 1980 Test Conf.*, pp. 359–367, (with K. Son), November 1980.
51. "An Easily Testable Design of PLAs," *Cherry Hill Test Conference*, Philidelphia, Pennsylvania, (with K. Son), November 1980. (Reprinted in IEEE Tutorial on VLSI Testing edited by Rex Rice, 1981).
52. "A Generalization of Shuffle–Exchange Networks," *Proc. of Fouteenth Annual Conference on Information Sciences and Systems*, Princeton, New Jersey, March 1980.
53. "A Framework for the Study of Permutations and Applications to Memory Processor Interconnection Networks," *Proc. 1979 International Conference on Parallel Processing*, (with K.L. Kodandapani), pp. 148–158, August 1979.
54. "Shift Registers Designed for On–Line Fault Detection," *Proc. of 1978 International Symposium on Fault Tolerant Computing*, Toulouse, France, pp. 173–178, June 1978.
55. "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.
56. "Error Control Techniques for Array Processors," *1977 International Symposium on Information and Theory*, Ithaca, New York, October 1977.
57. "Fault–Tolerant Fail–Safe Logic Networks," *Proceedings on IEEE Compcon*, (with S.M. Reddy), pp. 363–366, March 1977.
58. "On Undetectability of Bridging Faults," *Proceedings of 1977 International Symposium on Fault–Tolerant Computing*, Los Angeles, California, (with K.L. Kodandapani), June 1977.
59. "Further Results on m–RMC Forms," *Proceedings of 1976 International Symposium on Multivalued Logic*, Logan, Utah, pp. 88–93, (with K.L. Kodandapani), May 1976.
60. "A Graph Structual Approach to Data Management Systems," *Proc. Ninth Hawaii International Conference on System Sciences*, (with L.C. Chang), Western Periodicals, pp. 254–258, January 1976.
61. "Fault–Tolerant Asynchronous Networks Using (2,1)–Type Assignments," *Digest of Fifth International Symposium on Fault–Tolerant Computing*, Paris, France, June 1975.
62. "Construction on Error Correcting Codes with Run–Length Limited Properties," presented in *1974 International Symposium on Information and Theory*, Notre Dame, Indiana, November 1974.
63. "Synthesis of Arithmetic and Logic Processors by using Nonbinary Codes ," Digest of Papers, *Fourth International Symposium on Fault–Tolerant Computing*, IEEE Computer Society Publications, (with L.C. Chang), pp. 4–22, June 1974 .
64. "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.
65. "On Fault–Diagnosis of Sequential Machines," *Proc. VI Hawaii Conference on System Sciences*, Western Periodicals, (with S.M. Reddy), Januray 1973.
66. "A Design Technique for Synthesis of Fault–Tolerant Adders," *Digest of Papers of 1972 International Symposium on Fault–Tolerant Computing*, IEEE Computer Society Publications, (with S.M. Reddy), pp. 20–25, June 1972.