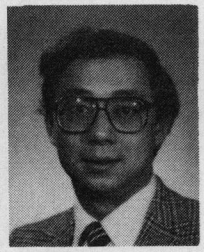
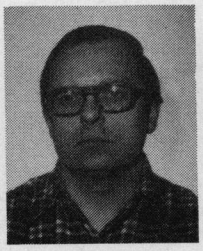
[13] J. B. Dennis, "Notes on computation structure," M.I.T., Cambridge, MA, 1969.
[14] H. A. Sholl and T. L. Booth, "Software performance modeling using computation structures," *IEEE Trans. Software Eng.*, pp. 414–420, Dec. 1977.
[15] T. L. Booth, *Sequential Machines and Automata Theory*. New York: Wiley, 1967, ch. VI.
[16] R. Graham, "Bounds on multiprocessor timing anomalies," *SIAM J. Appl. Math.*, vol. 17, no. 2, pp. 416–429.
[17] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*. Woodland Hills, CA: Comput. Sci. Press, 1978, pp. 567–572.
[18] Special Issue on Supercomputers, *IEEE Comput. Mag.*, Nov. 1980.



**Martin C. Wei** (S'69–A'73–S'75–M'77) was born in Canton, China. He received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from the University of Connecticut, Storrs, in 1971, 1973, and 1978, respectively.

From 1971 to 1973 he was a Research Assistant at the University of Connecticut. From 1973 to 1974 he was an Associate Engineer with the I.B.M. System Product Division, E. Fishkill, NY. From 1974 to 1978 he was an Instructor and Research Fellow at the University of Connecticut. Since 1978 he has been with the Customer Systems Laboratory, Bell Laboratories, Holmdel, NJ, where he is now a Technical Supervisor. His research interests are in the areas of computer architecture, distributed and multiprocessing systems, computer networking, distributed operating systems, and software reliability.

Dr. Wei is serving in the Communication Software Technology Committee of IEEE ComSoc.



**Howard A. Sholl** (M'55–M'68) was born in Northampton, MA, on October 14, 1938. He received the B.S. and M.S. degrees in electrical engineering from Worcester Polytechnic Institute, Worcester, MA in 1960 and 1963, respectively, and the Ph.D. degree in computer science from the University of Connecticut, Storrs, in 1970.

From 1961 to 1963 he was a Graduate Teaching Assistant at Worcester Polytechnic Institue. He worked as a Senior Engineer in computer and logic design for Sylvania Electric Company, Needham, MA, from 1963 to 1966. Since 1966 he has been with the Department of Electrical Engineering and Computer Science at the University of Connecticut, where he is now Associate Professor. He was a Leverhulme Visiting Fellow at the University of Edinburgh in 1973–1974, and a Fulbright Senior Research Fellow at the Technical University of Munich in 1981–1982. His research interests are in the areas of digital system design and software engineering.

Dr. Scholl is a member of Eta Kappa Nu, Sigma Xi, and Tau Beta Pi.

# A Fault-Tolerant Communication Architecture for Distributed Systems

DHIRAJ K. PRADHAN, SENIOR MEMBER, IEEE, AND SUDHAKAR M. REDDY, MEMBER, IEEE

***Abstract*—A communication architecture for distributed processors is presented here. This architecture is based on a new topology we have developed, one which interconnects $n$ nodes by using $rn$ links where the maximum internode distance is $\log_r n$, and where each node has, at most, $2r$, I/O ports. It is also shown that this network is fault-tolerant, being able to tolerate up to $(r-1)$ node failures.**

**One of the particularly attractive features of this network is that it allows for simple routing as well as for easy distributed fault-diagnosis. Algorithms are also developed here for the purpose of routing messages from node to node; these are useful both with and without the presence of faults in the network. A procedure is developed, too, whereby each fault-free node can diagnose the faulty nodes independently, without the use of any central observer.**

***Index Terms*—Distributed architecture, distributed fault diagnosis, fault-tolerant communication networks, graph connectivity, interconnection network graph diameters, self-diagnosis, store and forward networks.**

## INTRODUCTION

THE increased emphasis on fault-tolerance and reliability has made distributed processor architecure particularly attractive. Such an occurrence has provided the impetus for the development of certain distributed fault-tolerant computers which have recently been surveyed by Rennels [17]. Since the processing, control, and database are distributed in these systems, they do enjoy certain natural advantages over the centralized systems from the reliability viewpoint. Simple, graceful degradation is thus allowed for, as well as the elimination of certain critical components from the system. Furthermore, the potential exists for incorporating distributed diagnosis and recovery directly into the system. If the processors in the distributed system are provided with the capacity to test and reconfigure faulty processors, then it is possible to design diagnosis and recovery algorithms which are distributed.

An important component of a distributed system is the system topology. The system topology defines the interprocessor communication architecture. Also, there are well-defined relationships between the system topology and the message delay, the routing algorithms, fault-tolerance, and fault-diagnosis. Specifically, the message delay may be directly proportional to the internode distance [2]. The complexity of a routing algorithm may be determined by the regularity of the topology. A highly regular structure may allow simple routing; on the other hand, a highly irregular structure may require extensive hardware/software support. The fault-tol-

Manuscript received December 30, 1980; revised December 1, 1981. This work was supported by U.S. Air Force Office of Scientific Research Grants 80-0217 and 78-3482.

D. K. Pradhan is with the School of Engineering, Oakland University, Rochester, MN 55901.

S. M. Reddy is with the Department of Electrical and Computer Engineering, University of Iowa, Iowa City, IA 52240.

0018-9340/82/0900-0863$00.75 © 1982 IEEE

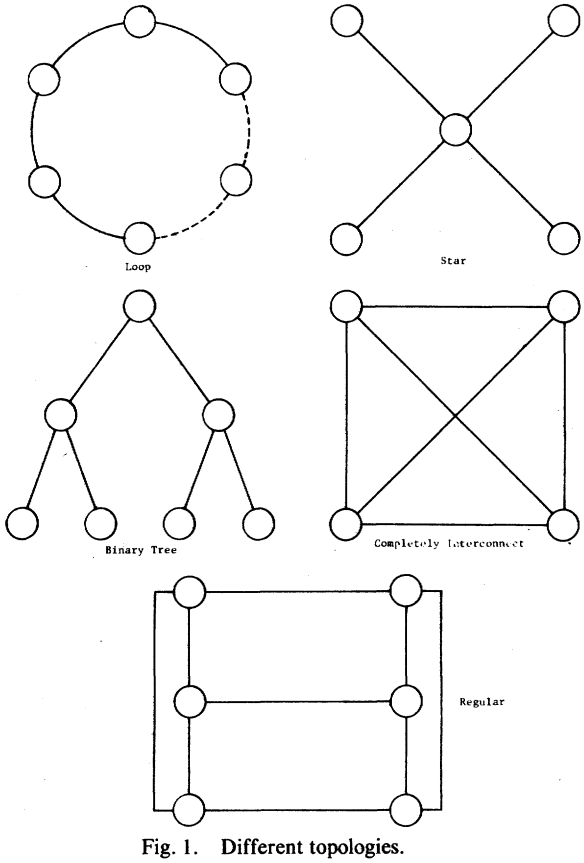



Fig. 1. Different topologies.

erance and fault-diagnosis capabilities may also be directly related to the node connectivity [2]-[5], [7].

Several different topologies have been proposed in the literature and some of them have already been implemented. Noted among them is the loop structure [8], the star structure [1], and the binary tree structure [16], complete interconnect system [1], regular networks [5], etc., shown in Fig. 1.

All of these structures have certain attractive features, as well as certain inherent inadequacies. For example, the loop system allows for very simple routing, but may have too large an internode distance. On the other hand, the star and the binary tree structures possess small internode distances but poor fault-tolerance. Or, the complete interconnect system has good fault-tolerance, but lacks cost effectiveness. Also, the regular structure can be cost effective, but may have a large internode distance.

This paper proposes a new topology. The objective here is to have a system with the following properties.

*1) Fault-Tolerance:* A system which will provide any desired degree of fault-tolerance, without excessive cost.

*2) Fault-Diagnosis:* A system which will provide a simple way to implement distributed fault-diagnosis without the use of a central observer.

*3) Routing:* A system which will provide simple routing of messages (packets) without extensive hardware/software support.

This paper is organized into four main sections. In Section II, we develop certain notations and definitions. Section III then presents the proposed topology and the related results.

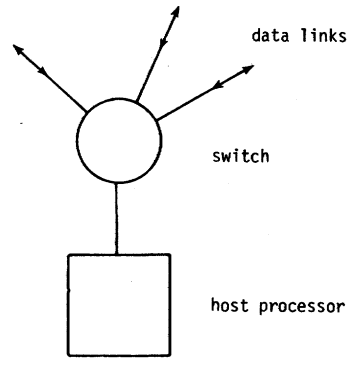



Fig. 2. A node in a system.

Following this in Section IV, two different distributed routing algorithms are developed—the first one, for use in a fault-free network; the second, for use in a faulty network. Using this second algorithm in Section V, we formulate a fault-diagnosis procedure. This procedure is truly distributed and does not require the use of a central observer.

## I. Notations and Definitions

In this section, we present certain notations and definitions to be used for developing the system topology. The system topology will be described in terms of an undirected graph, $G = \langle V, E \rangle$ where $V$ represents the set of nodes and where $E$ represents the set of edges. Let $0, 1, 2, \cdots, (n-1)$ denote the set of $n$ nodes in $G$. The node $i$ in the graph represents the $i$th processor. Each processor may be connected to the system through a switch (communication processor), as shown in Fig. 2. An edge in $G$ will be denoted as $ij$, representing a bidirectional data link between nodes $i$ and $j$.

The degree of node $i$ represents the number of edges incident on it; this will be denoted as $d_i$. Thus, $d_i$ represents the number of nodes connected directly to the $i$-th node through a dedicated data link. So, $d_i$ is limited by the number of I/O ports available.

Let $k_{ij}$ represent the minimum number of hops (a hop represents transmission through one data link) that are required to send a message from $i$ to $j$. (Note that since the links are bidirectional, $k_{ij} = k_{ji}$.)

*Definition:* Let $d_{\max} = \max \{d_i | 0 \le i \le n-1\}$ be the maximum of the degrees of nodes in $G$.

Thus, $d_{\max}$ may not exceed the maximum number of I/O ports available per node.

*Definition:* Let $d_{\min} = \min \{d_i | 0 \le i \le n-1\}$ be the minimum of the degrees of the nodes in $G$.

*Definition:* Let $k = \max \{k_{ij} | \text{ all } i, j, 0 \le i, j \le n-1\}$ be the maximum of the number of hops required to send messages between all pairs of nodes.

Thus, the value of $k$ may be related to the maximum delay experienced by any message.

*Definition:* Let $k_{av} = 2 \left( \sum_{j=0}^{n-1} \sum_{i=j+1}^{n-1} k_{ij} \right) \Big/ (n^2 - n)$ represent the average of the number of hops required to send messages between all pairs of nodes.

The value of $k_{av}$ may be related to the average delay expe-

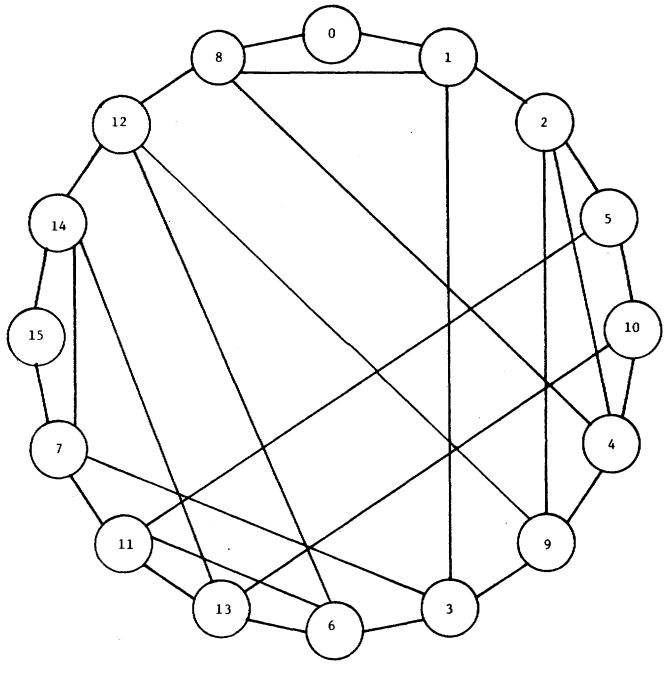



Fig. 3. A 16-node topology.

rienced by any message. This assumption may not be valid when the likelihood of communication between a pair of nodes that is closer is higher than a pair of nodes that is farther apart.

*Definition:* Let $c_{ij}$ represent the number of node disjoint paths between $i$ and $j$ (two paths are node disjoint if they do not have any node in common).

Note $c_{ij} = c_{ji}$.

*Definition:* Let $c$ represent the node connectivity of graph $G$ defined as $c = \min \{c_{ij} |$ all $i, j, 0 \leq i, j \leq n-1\}$.

Note that $c \leq d_{\min}$.

## II. System Topology

This section presents a topology of the proposed distributed communication architecture.

The following defines the proposed interconnection topology in terms of a parameter $r$. The number of nodes $n$ will be assumed to be equal to $r^m$. Given $n$, one can use different values of $r$ to construct different interconnections. The value of $r$ may be selected based on the number of I/O ports available and/or the maximum allowable transmission delays for messages (packets).

Let $(i_{m-1}, i_{m-2} \cdots, i_0)$ be the radix-$r$ representation of $i$, $0 \leq i \leq (n-1)$.

Let $(j_{m-1}, j_{-2}, \cdots, j_0)$ be the radix-$r$ representation of $j$, $0 \leq j \leq (n-1)$.

*System Topology G:* Every node $i$ is connected to a node $j$ if

$$i_w = j_{w-1} \qquad 1 \leq w \leq m-1$$

or

$$i_w = j_{w+1} \qquad 0 \leq w \leq m-2.$$

*Example 1:* Fig. 3 illustrates the system topology for $n = 16$ and $r = 2$. It may be noted that in this particular topology, there are twelve nodes of degree 4, two nodes of degree 3 and two nodes of degree 2.

Also here, $k = 4$ and $c = 2$. The node connectivity follows from the observation that there is a loop structure imbedded in this graph, as shown in Fig. 3. Given any two nodes, one can construct two node disjoint paths—one clockwise, the other counterclockwise.

*Definition:* A node $j$ will be called a neighbor of $i$ if it is connected directly to $i$.

*Definition:* A node $j$ will be called a type-$L$ neighbor of $i$ if $j$ in radix-$r$ is equal to

$$(i_{m-2}, \cdots, i_0, x), \qquad 0 \leq x \leq r-1.$$

*Definition:* A node $j$ will be called a type-$R$ neighbor of $i$ if $j$ in radix-$r$ is equal to

$$(x, i_{m-1}, \cdots, i_1), \qquad 0 \leq x \leq r-1.$$

*Theorem 1:* There are exactly

1) $n - r^2$ nodes of degree $2r$,
2) $r$ nodes of degree $2r-2$, and
3) $r^2 - r$ nodes of degree $2r-1$.

*Proof:* There are, at most, $r$ type-$L$ neighbors and $r$ type-$R$ neighbors of $i$.

Let a type-$L$ neighbor be also a type-$R$ neighbor. This implies that

$$(i_{m-2}, \cdots, i_0, h) = (g, i_{m-1}, \cdots, i_1) \tag{1}$$

Thus, one has, from (1),

$$i_{m-1} = i_{m-3} = i_{m-5} = \cdots \tag{2}$$

and

$$i_{m-2} = i_{m-4} = i_{m-6} = \cdots \tag{3}$$

It can be easily seen that there are exactly $r^2$ nodes that satisfy (2) and (3), hence, (1). All other $(n - r^2)$ nodes *must* have degree $2r$.

Of these $r^2$ nodes, there are exactly $r$ nodes of the type for which

$$i_{m-1} = i_{m-2} = \cdots = i_1 = i_0.$$

The degree of these $r$ nodes can be readily seen to be $(2r-2)$. The remaining $(r^2 - r)$ nodes must be of the type

$$(h, g, h, g, \cdots) \text{ in radix } r, \qquad \text{where } h \neq g.$$

These nodes can be readily seen to have degree $(2r-1)$.

Q.E.D.

*Corollary 1:* There are exactly $nr - (r^2 + r)/2$ data links in the system.

*Proof:* Proof is a direct consequence of Theorem 1.

Q.E.D.

*Theorem 2:* For any $i, j, i \neq j$, $k_{ij} \leq \min(f1, f2)$ where $f1$ and $f2$ are the least numbers for which

$$j_e = i_{e-f1}, \qquad \text{for all } e, f1 \leq e \leq m-1 \tag{3}$$

and

$$j_e = i_{e+f2}, \qquad \text{for all } e, 0 \leq e \leq m-f2-1 \tag{4}$$

*Proof:* We will prove that (3) and (4) imply the existence of two paths by using $f1$ and $f2$ hops, respectively.

Using (3), one has

$$\begin{aligned} j_{f1} &= i_0 \\ j_{f1+1} &= i_1 \\ &\vdots \\ j_{m-1} &= i_{m-1-f1} \end{aligned} \tag{5}$$

Now, consider the following paths from $i$ (the nodes are shown in radix-$r$):

$$\begin{aligned} i = &(i_{m-1} \cdots\cdots\cdots\cdots i_1 i_0) \\ &(i_{m-2} \cdots\cdots\cdots i_0 j_{f1-1}) \\ &(i_{m-3} \cdots\cdots i_0 j_{f1-1} j_{f1-2}) \\ &(i_{m-f1} \cdots i_0 j_{f1-1} \cdots j_1) \\ &(i_{m-f1-1} \cdots i_0 j_{f1-1} \cdots j_0) = j. \end{aligned}$$

From (5) it is clear that the last node in the above path is node $j$. Since there are $(f1 - 1)$ intermediate nodes, there are exactly $f1$ hops from $i$ to $j$. Similarly, one can see that (4) implies that there is a path between $i$ and $j$ which requires $f2$ hops. Q.E.D.

It may be noted that for a certain $i, j$, $k_{ij} < \min(f1, f2)$. As an example, consider $i = 0$ and $j = r^{m-1} + 1$. Here, $f1 = f2 = m$, but $k_{ij} = 2$.

*Corollary 2:* $k = m$.

*Proof:* Let $i = 0$ and $j = (n - 1)$. Thus $i = (0, 0, \cdots, 0)$ and $j = (r - 1, r - 1, \cdots, r - 1)$ in radix-$r$. Since one hop can introduce one new digit, we require $m$ hops. Q.E.D.

*Corollary 3:* $k_{av} \leq (m - 1)$.

*Proof:* Proof is obvious from Theorem 2. Q.E.D.

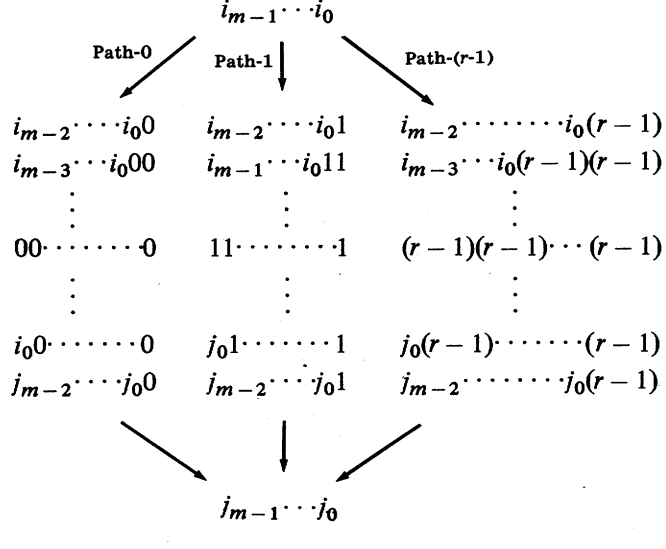
*Theorem 3:* A loop structure is imbedded in $G$.

*Proof:* The proof follows from the observation that there is a Euler digraph [6] for maximum-length sequence of length $r^m$ for every $r$ and $m$. Q.E.D.

Thus Theorem 3 implies the node connectivity $c \geq 2$. In the following, we show that $c \geq r$.

*Theorem 4:* The node connectivity $c$ of $G$ is at least $r$.

*Proof:* Consider $r$ paths between nodes $i$ and $j$, shown below; the nodes are represented in radix-$r$:



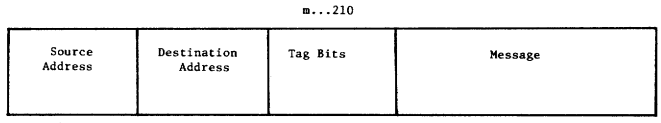




Fig. 4. Message format.

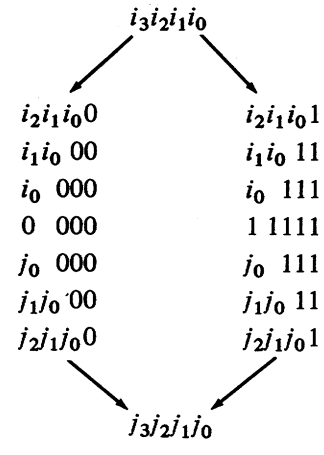
These $r$ paths are clearly node disjoint since an intermediate node in the $e$th path has $e$ as the least significant digit. Q.E.D.

The above theorem implies that the removal of any $(r - 1)$ nodes from $G$ does not disconnect $G$. Let $\overline{G}$ be the graph resulting from the removal of any $(r - 1)$ nodes. The following is a direct consequence of the above theorem.

*Corollary 3:* For any $i, j$ in $\overline{G}$,

$$k_{ij} \leq 2m.$$

*Example 2:* Consider the $G$ shown in Fig. 3. Given any two nodes $i = (i_3, i_2, i_1 i_0)$ and $j = (j_3, j_2, j_1, j_0)$ we have the following two node disjoint paths:




Thus even after removal of any node from $G$ there exists a path between any pair of nodes $i, j$ where $k_{ij} \leq 8$.

## III. Routing Algorithms

In this section routing algorithms are developed that transmit messages (packets) from a source to a destination node in the topology described in the last section. The first one will be referred to as *primary* algorithm, used by the network when there are no node failures in the network. The second one will be referred to as an *alternate* algorithm, used in the presence of node failures. Both of these algorithms are simple and require a minimal amount of hardware/software support.

To begin with, each message is formatted, as shown in Fig. 4. The message carries certain tag bits (digits), in addition to the usual source and destination addresses. These tag bits are generated *at* the source and go through certain minor modifications at each intermediate node.

The primary algorithm routes the message in min $(f1, f2)$ hops, where $f1$ and $f2$ are as given in Theorem 2. The alternate algorithm routes the message in, at most, $2m$ hops and also allows for up to $(r - 1)$ node failures.

First, it may be noted that when a message arrives at node

$p$, there are two possible actions that may be taken by $p$. If the message is destined for $p$, itself, the switch at $p$ must remove the message and send it to the host processor. On the other hand, if the message is destined for some other node, then $p$ must forward the message to one of its neighbors, the next node in the path. (The selection of a particular neighbor is based on certain tag bits contained in the message.)

*Primary Algorithm*

For a given source–destination pair, this routing algorithm routes the message through one of two possible paths, depending on the values of $f1$ and $f2$; if $f1 \leq f2$, the primary algorithm selects a path which uses only type-$L$ neighbors to route the message. On the other hand, if $f2 < f1$, then it selects a path which uses only type-$R$ neighbors to route the message. The particular path and hence, the particular type of neighbors that are to be used are decided at the source node. Thus, at most $(m + 1)$ tag digits are needed. One digit to indicate what type of neighbors that are to be used and at most $m$ digits to provide the selection of the appropriate neighbors at each node. Let these tag digits be numbered 0 through $m$, as shown in Fig. 4. The digit 0 has a binary value (0 or 1) and indicates to each intermediate nodes what type of neighbors are to be used for routing the message. The digits $m$ through 1 carry information regarding which particular neighbor is to be selected at each hop. The source generates these digits, which are computed in the following way. The $(m - e)$th tag digit is obtained from the address of the particular neighbor to which the message is forwarded during the $(e + 1)$th hop. If this neighbor is a type-$L(R)$ neighbor then the least (most) significant digit of its address constitute the $(m - i)$th tag digit.

When the message is received at node $p$, the following sequence of steps is carried out (by the switch).

*Step 1:* If the destination address is $p$, then the message is removed; otherwise, steps 2 and 3 are performed.

*Step 2:* The 0th digit is used for selecting either type-$L$ or type-$R$ neighbors and the $m$th digit is used for selecting the particular type-$L$ (type-$R$) neighbor for forwarding the message.

If the 0th tag digit is 0(1) and the $m$th tag digit is $t$ then the message is forwarded to that type-$L(R)$ neighbor which has $t$ in the least (most) significant digit.

*Step 3:* After the next node in the path has been determined in step 2, the tag digits 1 through $(m - 1)$ are shifted one digit to the left; then the message is forwarded, which is shown in Fig. 5.

The tag digits are shifted before each hop so that the $(m - i)$th tag digit is at the $m$th position when the message arrives at the $i$th intermediate node. This is important because it relieves the burden (for each intermediate node) of determining how many hops the message has gone through before arriving at the intermediate node.

In order to complete the algorithm we develop a procedure for the source node to compute the tag digits in the following.

*Definition:* Let $*$ represent a special digit.

*Definition:* Given $x = (x_{m-1}, \cdots, x_0)$, let $*x = (*, x_{m-1} \cdots x_1)$ and $x* = (x_{m-2}, \cdots, x_0, *)$.

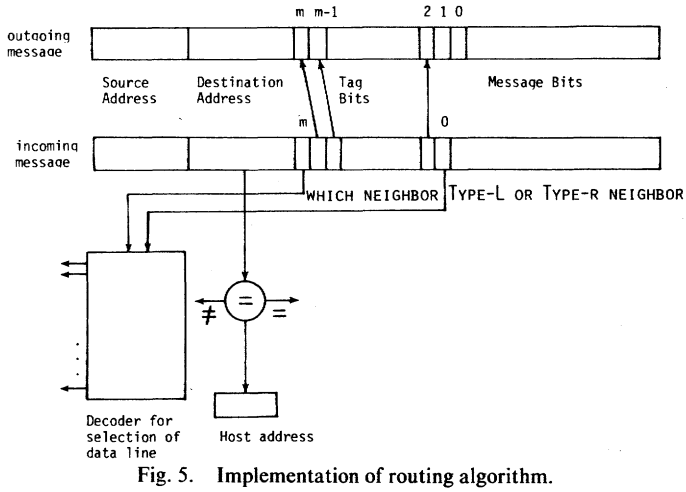



Fig. 5. Implementation of routing algorithm.

*Definition:* If $x = (x_{m-1}, \cdots, x_0)$ and $y = (y_{m-1} \cdots, y_0)$, then $x = y$ if $x_j = y_j$, for all $j$, where $x_j \neq *$ and $y_j \neq *$.

Given the source node, $i = (i_{m-1}, \cdots, i_0)$, and the destination node, $j = (j_{m-1}, \cdots, j_0)$, the following generates the tag digits $T$.

*Step 1:* Let $a = b = i$ and $e = 1$.

*Step 2:* Let $a = *a$ and $b = b*$.

*Step 3:* If $j = a$, then $T = j_{m-1} \cdots j_{m-e}\, 11 \cdots 1$.

*Step 4:* If $j = b$, then $T = j_{e-1} \cdots j_0\, 00 \cdots 0$.

*Step 5:* $e = e + 1$; go to step 2.

Given $i = 1011$ and $j = 1100$; then the above procedure generates 0000. (Note that $i$ to $j$ requires two hops. Both of these hops use type-$L$ neighbors and the intermediate node is 0110.)

*Alternate Algorithm*

This algorithm is used when one or more node failures occur. Let there be $e$ node failures, $0 \leq e \leq (r - 1)$. Let the least significant digits of the $e$ failed nodes be $u_1, u_2, \cdots, u_e$. Out of the $r$ disjoint paths shown in the proof of Theorem 4, no more than $e$ paths can contain the faulty nodes; these paths are precisely path-$u_1$, path-$u_2$, $\cdots$, path-$u_e$. Each node can send the message through any one of the other $(r\text{-}e)$ paths.

Let the source and destination nodes be $i$ and $j$, respectively. Let one of the $(r\text{-}e)$ fault-free paths be

$$i \rightarrow \cdots \rightarrow v\left(\frac{r^m - 1}{r - 1}\right) = (v, v, \cdots, v) \rightarrow \cdots \rightarrow j$$

where $v \neq u_b$, $1 \leq b \leq e$.

The node, $i$, can use the tag digits, $vv \cdots v0$, to send the message to node $vv \cdots v$, via $(i_{m-2} \cdots i_0 v)$, $(i_{m-3} \cdots i_0, v, v)$, $\cdots$, $(i_0 vv \cdots v)$.

Then, the node $v(r^m - 1/r - 1)$ can replace the tag digits by a new set of tag digits, $j_0, j_1, \cdots, j_{m-1}1$, and forward the message. This will take the message through nodes $(j_0, v, v, \cdots, v)$, $(j_1, j_0, v, v, \cdots, v)$, $\cdots$, $(j_{m-2}, \cdots, j_0, v)$ to $j$.

It may be observed that in the event of $(r - 1)$ failures, there is the potential for bottleneck, since there is only a single path available from $i$ to $j$ for every $i, j$ and this passes through some node, $v(r^m - 1/r - 1)$. This node lies in the path of all the pairs of nodes.

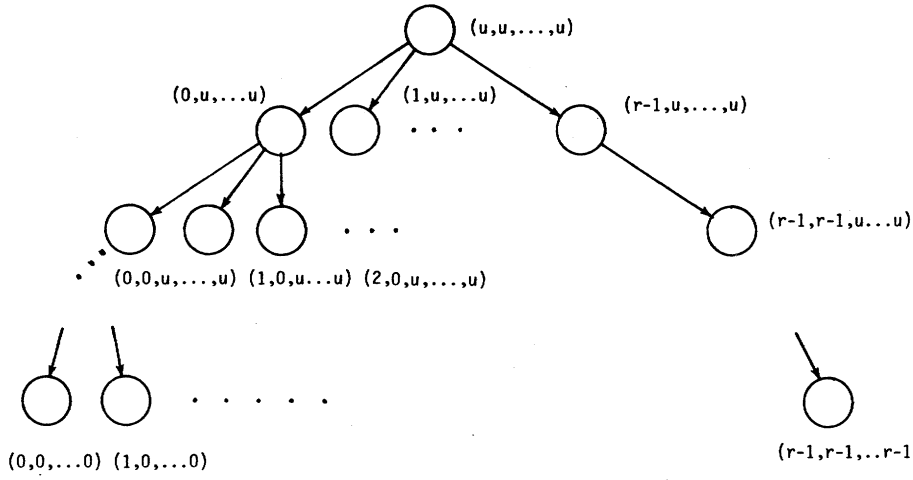



Fig. 6. Diagnostic graph for root node $u$ $\left(\frac{r^{m-1}}{r-1}\right)$.

However, for $r = 2$, there is a solution to this problem. In the event of a single node failure, one can use the loop connection to communicate. This will overcome the bottleneck problem.

*Example 3:* Let $n = 16, r = 2, i = 6$ and $j = 7$. The primary algorithm will generate the tag bits (11100) which will constitute the path $6 \rightarrow 13 \rightarrow 9 \rightarrow 3 \rightarrow 7$. Assume the faulty node is 13 then the alternate algorithm will generate the tag bits (00000) at the source. This will route the message from 6 to node 0 via the nodes 12 and 8. Then the node 0 will generate the tag bits (11101). This will route the message to node 7 via 8, 12, and 14.

It may be noted that both of these algorithms are truly distributed, there is no central control.

## V. Distributed Fault Diagnosis

In this section, a distributed algorithm is developed for fault-diagnosis using the routing algorithm described in Section IV. The important characteristic of our algorithm, unlike that used in the PMC [11] model, is that there is no central observer node monitoring the test result. The assumption of central observer is unrealistic in the distributed environment [7], [18]. The algorithm given here is designed particularly for our topology.

We make the following usual assumption.

A node, $u$, can test a node, $v$, only if $v$ is a neighbor of $u$.

The testing graph (which is a directed graph) thus will be a subgraph of $G$.

Consider the $r$ testing graphs, shown in Fig. 6; these are directed tree structures. The top node here will be referred to as a *root* node, the bottom nodes as *leaf* nodes. Any node other than the leaf nodes will be referred to as a *nonleaf* node.

There are $r$ trees. Each tree corresponds to the root node, $u(r^m - 1/r - 1)$, $0 \leq u \leq r - 1$. The nodes in level $e$ are type-$R$ neighbors of nodes in level $(e - 1)$. The following properties can be readily proved and will be useful for formulating the following diagnosis of the algorithm.

Consider any one of the $r$ trees. Let the root node be $u(r^m - 1/r - 1)$ for the $u$th tree.

1) All the nodes appear in each testing graph.

2) There are exactly $(r^{m-1})$ nonleaf nodes where each nonleaf node has $u$ in the least significant digit.

3) There are exactly $(r - 1)\, r^{m-1}$ leaf nodes which constitute all the nodes that do not have $u$ as the least significant digit.

4) Any combination of $(r - 1)$ nodes appear as the leaf nodes of at least one of the trees.

### *Diagnosis Algorithm*

*D1:* Each root node initiates testing top down.

*D2:* The test results are transmitted bottom up to the root node.

*D3:* A node in the tree at level $(e - 1)$ accepts information from a node in level $e$ regarding the results of tests conducted by level $e$ nodes, only if it has itself tested the level $e$ node and found it to be fault free.

*D4:* A root node examines the test results to see if all the faulty nodes found are leaf nodes. If they are, then the root node broadcasts a message to all of the nodes using the alternate routing algorithm, thereby avoiding faulty nodes. If all the faulty nodes are *not* leaf nodes, then no action is taken by the root node.

*D5:* Upon receipt of the message that identifies the faulty nodes, the immediate neighbors of the root node take the following action: each neighbor tests the root node; if they find the root node fault-free, then they broadcast a message to all of the nodes informing them of the fault free status of the root node. Here, the neighbors again use the *alternate* routing algorithm to avoid the faulty nodes.

*D6:* Each node accepts the information from the root node (identifying faulty nodes) *only* if it also receives a message from at least $(r - 1)$ neighbors of the root node, validating the fault-free status of the root node.

*Theorem 5:* The above algorithm diagnoses up to $(r - 1)$ faulty nodes.

*Proof:* The proof can be deduced from the following: First it may be observed that if the root node is fault-free, then either it will correctly identify all of the faulty nodes, or it will not identify any faulty node to other nodes. This follows from P4 and D4.

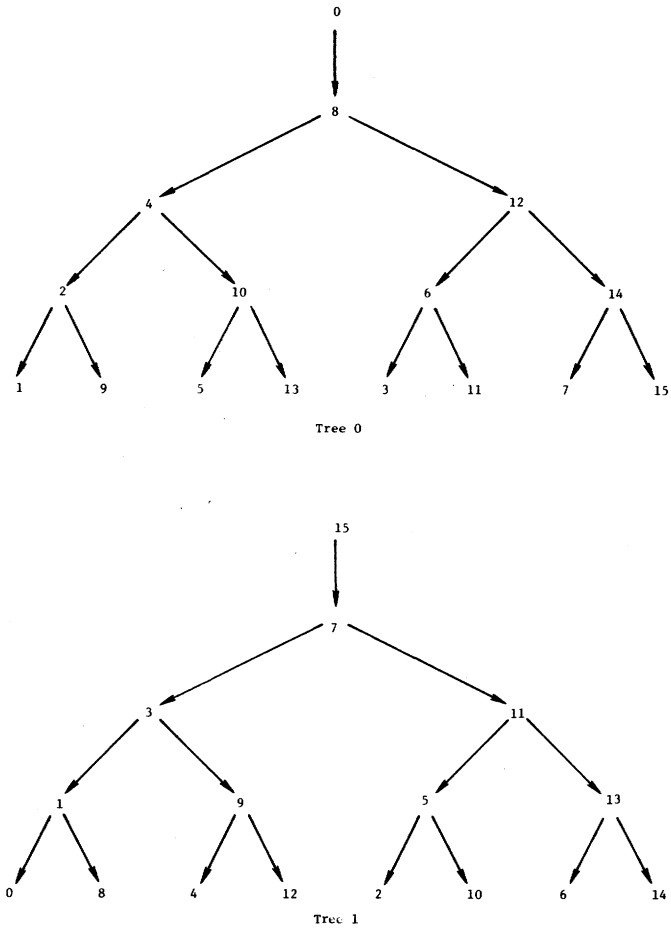



Fig. 7. Testing graph for Example 4.

Furthermore, since there are at most $(r-1)$ nodes that can be faulty there are at least $(r-1)$ of the $(2r-2)$ neighbors which are fault-free.

On the other hand if the root node is faulty then there is the possibility of an erroneous message from the root node identifying some good nodes as faulty. But this message will not be accepted by other nodes. The reason being that for the message to be acceptable there must be at least $(r-1)$ neighbors of the root node which must attest that the root node is fault-free. But this is impossible since for a neighbor to do this it must be itself faulty and there are at most only $(r-2)$ nodes other than the root node which are faulty. Q.E.D.

*Example 4:* Consider the topology shown in Example 1. The two root nodes for the purpose of testing are 0 and 15. The testing graphs are shown in Fig. 7.

Note that in tree-0 all the leaf nodes are odd and the nonleaf nodes are even. In tree 1 the reverse is the case. Let us assume that node 7 is faulty, only node 0 will broadcast the message in D4 since 7 is a leaf node for tree-0. Nodes 8 and 1 will then validate the fault-free status of node 0.

*Corollary:* The total time that is required to complete the testing is lower bounded by $0(m)$ and upper bounded by $0(nr)$.

*Proof:* The lower bound follows from the depth of the testing graph and the upper bound follows from the total number of tests. Q.E.D.

## VII. Conclusion

A system topology for a distributed fault-tolerant communication architecture is presented. It was shown that this architecture enjoys the advantages of the low complexity, short routing distance and ease of diagnosis. Furthermore, it was also shown that it is fairly easy to tolerate faults in the network. A distributed fault-diagnosis algorithm was presented which allows for diagnosis of faults without the use of a central observer.

Currently under investigation are several other related networks, as well as generalizations of those that were presented here. Preliminary results of these studies to date have been reported in [19]. Additionally, of interest is work that has been done on DeBruijn graph [20] by the authors [21] and also by others independently [22], [23], which has recently been brought to our attention.

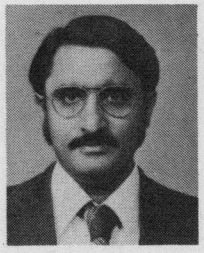
## References

[1] K. J. Thurber and G. M. Masson, *Distributed Processor Communication Architecture.* Lexington, MA: Lexington Books, 1980.
[2] D. W. Davies *et al.*, *Computer Networks and Their Protocols.* New York: Wiley, 1979.
[3] H. Frank and I. Frisch, "Analysis and design of survivable networks," *IEEE Trans. Commun. Technol.*, vol. COM-18, pp. 501–519, Oct. 1970.
[4] R. S. Wilkov, "Analysis and design of reliable computer networks," *IEEE Trans. Commun.*, vol. COM-20, pp. 660–678, June 1972.
[5] W. J. Bouknight *et al.*, "The Illiac IV system," *Proc. IEEE*, vol. 60, pp. 369–388, Apr. 1972.
[6] N. Deo, *Graph Theory with Applications to Engineering and Computer Science.* Englewood Cliffs, NJ: Prentice-Hall, 1974.
[7] J. G. Kuhl and S. M. Reddy, "Distributed fault-tolerance for large multiprocessor systems," in *Proc. 1980 Comput. Arch. Conf.*, LaBaule, France, May 1980.
[8] M. T. Liu *et al.*, "System design of the distributed double-loop computer network," in *Proc. 1st Int. Conf. Distributed Comput. Syst.*, Huntsville, AL, Oct. 1979, pp. 95–105.
[9] B. W. Arden and H. Lee, "Analysis of chordal ring network," *IEEE Trans. Comput.*, vol. C-30, pp. 291–296, Apr. 1981.
[10] J. Losq, "Fault-tolerant communication networks for computer networks," Tech. Note 127, Cent. Reliable Comput., Stanford Univ., Stanford, CA, Mar. 1978.
[11] F. P. Preparata, G. Metze, and R. T. Chien, "On the connection assignment problem of diagnosable systems," *IEEE Trans. Electron. Comput.*, vol. EC-16, pp. 848–854, Dec. 1967.
[12] A. D. Friedman and L. Simoncini, "System-level fault diagnosis," *Computer*, vol. 13, Mar. 1980.
[13] J. D. Russel and C. R. Kime, "System fault diagnosis: Closure and diagnosability with repair," *IEEE Trans. Comput.* vol. C-24, pp. 1078–1088, Nov. 1975.
[14] B. Arden and H. Lee, "A multitree structured network," in *Proc. 1978 Fall COMPCON*, Sept. 1978, pp. 201–210.
[15] A. Granov, L. Kleinrock, and M. Gerla, "A highly reliable distributed loop network architecture," in *Proc FTCS-10*, Kyoto, Japan, Oct. 1980.
[16] A. M. Despain and D. A. Patterson, "X-tree: A tree structured multiprocessor computer architecture," in *Proc. 5th Annu. Symp. Comput. Arch.*, Apr. 1980, pp. 144–152.
[17] D. A. Rennels, "Distributed fault-tolerant computer systems," *Computer*, vol. 13, pp. 55–66, Mar. 1980.
[18] R. K. Nair, "Diagnosis, self-diagnosis and roving diagnosis in distributed digital systems," Univ. Illinois, Urbana, CSL Rep. R-823, 1976.
[19] D. K. Pradhan, "Interconnection topologies for fault-tolerant parallel and distributed architectures," in *Proc. 1981 Int. Conf. Parallel Processing*, Aug. 1981, pp. 238–242, IEEE Cat. 8ICHI634-5.
[20] N. G. DeBruijn, "A combinatorial problem," in *Proc. Akademe Van Wetenschappen*, vol. 49, part 2, 1946, pp. 758–764.
[21] S. M. Reddy, D. K. Pradhan, and J. Kuhl, "Directed graphs with minimal diameter and maximal connectivity," School of Engineering, Oakland Univ., Tech. Rep., July 1980.

[22] M. A. Schlumberger, "Connectivity of DeBruijn graphs," IMATE, Univ. of Grenoble, Grenoble, France, Res. Rep. 146, Oct. 1978.
[23] M. Imase and M. Itoh, "Design to minimize a diameter on building block network," *IEEE Trans. Comput.*, vol. C-30, pp. 439–443, June 1981.

**Dhiraj K. Pradhan** (S'70–M'72–SM'80) received the M.S. degree from Brown University, Providence, RI, and the Ph.D. degree from the University of Iowa, Iowa City.

Since 1978, he has been an Associate Professor in the School of Engineering, Oakland University, Rochester, MI. Previously, he was with the Department of Computer Science, University of Regina, Regina, Canada. He has also worked as a Staff Engineer for the IBM Systems Development Lab, Poughkeepsie, NY, and has held Visiting Professor positions at Stanford University, Stanford CA, and Wayne State University, Detroit, MI. His research interests include fault-tolerant computing and parallel processing and networks, areas in which he has published extensively. He was the Guest Editor of a Special Issue on Fault-Tolerant Computing published in the IEEE COMPUTER magazine and is the editor of a forthcoming book entitled *"Fault-tolerant Computing: Theory and Techniques"* to be published by Prentice-Hall. He is also an IEEE Computer Society Distinguished Visitor.



**Sudhakar M. Reddy** (S'68–M'68) was born in Gadwal, Andhra Pradesh, India. He received the B.S.C.E. degree in 1962 from Osmania University, the M.S.E.C.E. degree in 1963 from the Indian Institute of Science, and the Ph.D. degree in electrical engineering from the University of Iowa, Iowa City.

Currently, he is a Professor of Electrical and Computer Engineering at the University of Iowa. He was the Guest Editor for the June 1978 issue of the IEEE TRANSACTIONS ON COMPUTERS and is currently the Associate Editor of the TRANSACTIONS for Fault-Tolerant Computing.

# Logic Networks of Carry–Save Adders

HUNG CHI LAI, MEMBER, IEEE, AND SABURO MUROGA, FELLOW, IEEE

***Abstract*—Logic networks of carry–save adders such as high-speed multipliers, multioperand adders, and double-rail input parallel adders are designed based on the parallel adders with a minimum number of NOR gates discussed in [1]. After a discussion of the derivation of carry–save adder modules (CSAM's) by the integer programming logic design method, general design procedures are illustrated with example networks. Compared to conventional networks of carry–save adders, the derived networks of carry–save adder modules (NOCSAM's) have the advantages of fewer gates, fewer connections, and faster operation. In particular, the parallel adder of NOR gates in double-rail input logic obtained has six gates and $15\frac{1}{2}$ connections per stage, whereas the previously known best design under the same condition requires six gates and 17 connections per stage with the same carry propagation delay.**

***Index Terms*—Carry–save adders, full adders, input bundles, logic design, multipliers, multioperand adders, NAND gates, NOR gates, output bundles, parallel adder in double-rail input logic.**

## I. INTRODUCTION

ONE-bit full adders can be used as basic building blocks for realizing high-speed multipliers and multioperand adders. Because of the nature of this type of networks, a one-bit full adder is often called a *carry–save adder* (*CSA*) in this type of application. We will use the term *network of carry–save adders* (*NOCSA's*) to mean this type of network. In this paper, networks and carry–save adders such as high-speed multipliers and multioperand adders are designed based on the *G*-minimum parallel adders with NOR (NAND) gates discussed in [1]. Here, a *G-minimum* network means a network with a minimum number of gates.

Fig. 1 shows a typical NOCSA for a 6-bit × 6-bit multiplier (adopted from [2]) which consists of 21 CSA's and an 11-bit parallel binary adder. Each CSA is a one-bit full adder which has three inputs of the same weight and two outputs with different weights, i.e., the sum output and the carry output. This multiplier operates as follows. When a 6-bit unsigned number $A = \sum_{i=0}^{5} a_i 2^i$ is multiplied by another 6-bit unsigned number $B = \sum_{i=0}^{5} b_i 2^i$, the product $P$ is given by

$$P \equiv \sum_{i=0}^{11} p_i 2^i = A \cdot B = \left(\sum_{i=0}^{5} a_i 2^i\right)\left(\sum_{i=0}^{5} b_i 2^i\right) = \sum_{i=0}^{5}\sum_{j=0}^{5} a_i b_j 2^{i+j}.$$

Since both $A$ and $B$ are binary numbers, $a_i b_j$ has a binary value and can be easily obtained by an AND gate with inputs from $a_i$ and $b_j$. Therefore the inputs of the NOCSA shown in Fig. 1, $a_i b_j$ for $i, j = 0, \cdots, 5$, can be obtained by an "AND matrix" which is not shown in the figure. In order to obtain the product

Manuscript received March 24, 1980; revised January 14, 1981 and April 19, 1982. This work was supported in part by the National Science Foundation under Grants MCS77-09744 and GJ-40221 and was part of the Ph.D. dissertation of [7].

H. C. Lai was with the Department of Computer Science, University of Illinois, Urbana, IL 61801. He is now with the Microtechnology Corporation, Sunnyvale, CA 94086.

S. Muroga is with the Department of Computer Science, University of Illinois, Urbana, IL 61801.

0018-9340/82/0900-0870$00.75 © 1982 IEEE