Other applications of the above algorithms are possible. For example, if we are required to enumerate all elements in the strongly-connected component of a given vertex S, we need only generate all possible vertex labels, and run each such generated vertex label through the above algorithm. This would expend $O(N \log^2 N)$ time to perform this enumeration (because each such test takes $O(\log^2 N)$ time). In fact, it is possible to perform the enumeration much more efficiently, using regular expression representation; the details are omitted.

It is also possible, using the above finite-state automata for representing fault sets, to handle varying fault sets. More precisely, suppose (for example) that some currently nonfaulty processor $v \notin F$ becomes faulty, and the remaining nonfaulty processors need to update their data structures to reflect this fact. This amounts to adding v to F, and augmenting the Aho–Corasick machine by adding the new string v to the machine. This can be done in time $O(n|F|)$, and this time bound is optimal [24]. The inverse problem of removing a faulty vertex from the fault set (when a faulty vertex becomes nonfaulty) is also solvable in the same time bound.

V. Conclusions and Future Work

We have presented a rich new class of fault-tolerant network topologies, along with very efficient routing algorithms in these topologies. We have shown that these networks have connectivity $k-1$, and can degrade gracefully when there are multiple node failures. Our methods have drawn upon the theory of regular languages, which is likely to be applicable to many other such problems. We are currently studying the undirected version of the derived de Bruijn graphs, and have obtained some encouraging results.

It is interesting that, in the networks we have presented, the situation of two canonical walks sharing a vertex (as depicted in Lemma 4) does not occur when $n \leq 4$. Also, these networks can be modified slightly, analogous to [21], to obtain networks with $(k-1)$-fault tolerance.

References

[1] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," *Commun. ACM*, pp. 333–340, June 1975.

[2] S. B. Akers, D. Harel, and B. Krishnamurthy, "The star graph: An attractive alternative to the n-cube," in *Proc. Int. Conf. Parallel Processing*, Aug. 1987, pp. 393–400.

[3] S. B. Akers and B. Krishnamurthy, "Group graphs as interconnection networks," in *Proc. 14th Int. Conf. Fault Tolerant Comput. Syst.*, June 1984, pp. 422–427.

[4] A.-H. Esfahanian and S. L. Hakimi, "Fault-tolerant routing in de Bruijn communications networks," *IEEE Trans. Comput.*, vol. C-34, pp. 777–788, Sept. 1985.

[5] M. A. Fiol, J. L. A. Yebra, and I. Miquel, "Line digraph iterations and the (d,k)-digraph problem," *IEEE Trans. Comput.*, vol. C-33, pp. 400–403, May 1984.

[6] H. Fredricksen, "A survey of full length nonlinear shift-register cycle algorithms," *SIAM Rev.*, vol. 24, pp. 195–221, Apr. 1982.

[7] R. L. Hemminger and L. W. Beineke, "Line graphs and line digraphs," in *Selected Topics in Graph Theory*, L. W. Beineke, Ed. New York: Academic, 1978.

[8] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation.* Reading, MA: Addison-Wesley, 1979.

[9] M. Imase and M. Itoh, "A design for directed graphs with minimum diameter," *IEEE Trans. Comput.*, vol. C-32, pp. 782–784, Sept. 1983.

[10] ——, "Design to minimize diameter on building-block network," *IEEE Trans. Comput.*, vol. C-30, pp. 439–443, June 1981.

[11] M. Imase, T. Soneoka, and K. Okada, "Connectivity of regular directed graphs with small diameter," *IEEE Trans. Comput.*, vol. C-34, pp. 267–273, Mar. 1985.

[12] D. E. Knuth, J. H. Morris, and V. R. Pratt, "Fast pattern matching in strings," *SIAM J. Comput.*, vol. 6, pp. 323–350, June 1977.

[13] C. W. H. Lam and J. H. van Lint, "Directed graphs with unique paths of fixed lengths," *J. Combinatorial Theory*, vol. B-24, pp. 331–337, 1978.

[14] A. Lempel, "On a homomorphism of the de Bruijn graph and its applications to the design of feedback shift registers," *IEEE Trans. Comput.*, vol. C-12, pp. 1204–1209, Dec. 1970.

[15] N. S. Mendelsohn, "Directed graphs with the unique path property," in *Combinatorial Mathematics and its Applications*, P. Erdos, A. Renyi and V. T. Sos, Eds., 1969.

[16] R. Y. Pinter, "Efficient string matching with don't-care patterns," in *Combinatorial Algorithms on Words*, A. Apostolico and Z. Galil, Eds., NATO ASI Series, Springer-Verlag, 1985, pp. 11–29.

[17] D. K. Pradhan, "Fault-tolerant multiprocessor link and bus network architectures," *IEEE Trans. Comput.*, vol. C-34, pp. 33–45, Jan. 1985.

[18] D. K. Pradhan and S. M. Reddy, "A fault-tolerant communication architecture for distributed systems," *IEEE Trans. Comput.*, vol. C-31, pp. 863–870, Sept. 1982.

[19] F. P. Preparata and J. Vuillemin, "The cube-connected cycles: A versatile network for parallel computation," *Commun. ACM*, vol. 24, pp. 300–309, May 1981.

[20] A. Sengupta, A. Sen, and S. Bandyopadhyay, "Fault-tolerant distributed system design," *IEEE Trans. Circuits Syst.*, vol. CAS-35, pp. 168–172, Feb. 1988.

[21] ——, "On an optimally fault-tolerant multiprocessor network architecture," *IEEE Trans. Comput.*, vol. C-36, pp. 619–623, May 1987.

[22] M. A. Sridhar, "On the connectivity of the de Bruijn graph," *Inform. Proc. Lett.*, vol. 27, pp. 315–318, May 1988.

[23] M. A. Sridhar and C. S. Raghavendra, "Uniform minimal full-access networks," *J. Parallel Distributed Comput.*, vol. 5, pp. 383–403, 1988.

[24] M. A. Sridhar, "Updating the Aho-Corasick pattern matching machine," *Int. J. Comput. Math.*, to be published.

Information Dissemination in Trees with Nonuniform Edge Transmission Times

Jae-moon Koh and Dong-wan Tcha

***Abstract*—This paper is concerned with the information dissemination process in a tree. It is assumed that a vertex can either transmit or receive a message and an informed vertex can transmit it to only one of its neighbors at a time. For the case of uniform edge transmission times, Slater *et al.* [4] showed that a simple ordering policy achieves the minimum broadcast time and presented an algorithm which determines a set of broadcast centers. We show that a simple ordering policy is still optimal for the case of nonuniform edge transmission times and suggest modifications of the algorithm which enables it to solve this general problem within $O(N \log N)$ time.**

***Index Terms*— Algorithm, broadcasting, communication, information dissemination, time-complexity.**

I. Introduction

"Broadcasting" in a tree refers to the information dissemination process whereby a message at a vertex is transmitted to all the vertices of the tree. It is assumed that a vertex can only transmit a message to an adjacent vertex by making a call. Restricting our attention to the case where a vertex is scheduled to participate as

Manuscript received November 15, 1988; revised April 22, 1989.

J.-m. Koh is with the Department of Industrial Engineering, College of Engineering, University of Ulsan, Kyungnam, Korea.

D.-w. Tcha is with the Department of Management Science, Korea Advanced Institute of Science and Technology, Seoul, Korea.

IEEE Log Number 9100998.

a sender or a receiver in at most one call at any time, there are three types of broadcasting—local broadcasting, line broadcasting, and path broadcasting [1], [2]. In this paper, we consider local broadcasting in a tree-type network.

The constraints in local broadcasting are as follows: 1) at any time, a vertex can participate in at most one call; 2) for each call, the sender is either the message originator or has been the receiver of a previous call; 3) an informed vertex can only call an adjacent one. Under the constraints, information is transmitted from a vertex to its neighbors one after another.

Under the simplifying assumption that every call requires one unit of time (i.e., uniformly weighted time), Slater *et al.* [4] showed that a simple ordering policy achieves the minimum broadcast time and based on this, they presented an algorithm which determines the amount of time required to broadcast a unit of information from an arbitrary vertex to every other vertex in a tree. But they left open the problem with nonuniformly weighted times.

In this paper, we consider the extended case which allows nonuniform edge transmission times. We show that the algorithm of Slater, Cockayne, and Hedetniemi [4] (which will be denoted by SCHA for brevity), by modifying the rules on assigning values for vertices and on identifying broadcast centers, can be used to solve this general problem and that a simple ordering policy is still optimal in this case. We also show that the modified algorithm has time complexity of $O(N \log N)$, where N is the number of vertices of the tree.

II. Model Description and Algorithm

A. Notations and Terminologies

Let $T = T(V, E)$ be a free tree consisting of a set of vertices V and a set of edges E with edge transmission times c_{ij}. Assume that $c_{ij} = c_{ji}$ for all $(i, j) \in E$. The terminologies and notations by Slater *et al.* will be used. Some of them, however, are listed below to make the presentation self-complete.

Broadcasting from vertex u: the process of passing information from u to every other vertex in a connected graph $G = (V, E)$ by local broadcasting.

The broadcasting number of u in G, $b(u, G)$ or $b(u)$: the minimum time required to broadcast from u.

The broadcast number of a connected graph G, $b(G)$: the minimum broadcast number in G, i.e.,

$$b(G) = \min_{u \in V(G)} \{b(u)\}.$$

The *broadcast center of G, $BC(G)$:* the set of all vertices having the minimum broadcast number, i.e.,

$$BC(G) = \{u | b(u) = b(G)\}.$$

$T(u, v)$: the subtree of T containing u among the two subtrees of $T - (u, v)$.

With these we consider the problem of finding the set $BC(T)$ of broadcast centers of a given tree T.

Since our algorithm is a direct generalization of SCHA to the case of nonuniform edge transmission times, most portions of their algorithm remain the same. For exposition brevity, their algorithm is first reviewed in moderate detail, and then only the parts of our algorithm which are different from those in their algorithm are spotlighted.

B. Outline of SCHA

The algorithm assigns a value to each vertex v of T. It begins with endvertices of T by assigning their values zero. Then it determines the values to be assigned to the other vertices iteratively from endvertices toward the "center" of T.

At each stage of the algorithm, a vertex w is chosen which has the smallest value among value-assigned vertices of the current tree T', and w is removed from T'. Consider the unassigned neighbor v of w in the inward direction. If v becomes an endvertex in T', all but one of whose neighbors, $u_1, u_2, \cdots, u_k$, have already been assigned values such that $t(u_1) \geq t(u_2) \geq \cdots \geq t(u_k)$, then a value $t(v)$ is assigned to v as follows.

$$t(v) = \max_{1 \leq i \leq k} \{t(u_i) + i\}. \tag{1}$$

If $t(u_i)$ is the minimum time required to broadcast from u_i to all vertices of $T(u_i, v)$, then $t(v)$ obtained by (1) can be shown to equal $b(v, T(v, x))$ where x is the unassigned neighbor of v (cf. Theorem 1 of [4]).

This process is repeated until T' has only one vertex, which is a broadcast center of T. Neighbors $u_1, u_2, \cdots, u_p$ of the broadcast center v are also broadcast centers if p is the smallest integer such that

$$t(u_p) + p = \max_{1 \leq i \leq k} \{t(u_i) + i\} \tag{2}$$

where $u_1, u_2, \cdots, u_k$ are neighbors of v and $t(u_1) \geq t(u_2) \geq \cdots \geq t(u_k)$.

C. Modifications of SCHA

Modifications of the two key formulas in the above algorithm are indeed all that is required for solving the more general case with nonuniform edge transmission times.

1) Modification I of value assignment rule (1): For the nonuniform case, the formula (1) is modified as follows.

$$t(v) = \max_{1 \leq i \leq k} \left\{ t(u_i) + \sum_{j=1}^{i} c_{v,u_j} \right\} \tag{1'}$$

where c_{v,u_j} is the edge transmission time from v to u_j.

2) Modification II of $BC(T)$ identification rule (2): At the last step of the algorithm, let v be the only one vertex left in T' and $u_1, u_2, \cdots, u_k$ be neighbors of v such that $t(u_1) \geq t(u_2) \geq \cdots \geq t(u_k)$. Then any neighbor u_q of v is a broadcast center if it satisfies

$$c_{v,u_q} + M_{q-1} \leq t(v) \tag{2'}$$

where

$$M_r = \max_{1 \leq i \leq r} \left\{ t(u_i) + \sum_{j=1}^{i} c_{v,u_j} \right\},$$
$$r = 1, 2, \cdots, k \quad \text{and} \quad M_0 = 0.$$

Note that

$$t(v) = \max_{1 \leq i \leq k} \left\{ t(u_i) + \sum_{j=1}^{i} c_{v,u_j} \right\} = M_k.$$

It is worth noting that the formulas (1) and (2) of SCHA are specializations of (1') and (2'), respectively, to the case of uniform edge transmission times. If $c_{ij} = 1$ for all (i, j), then clearly (1') is reduced to (1), and all vertices, u_q's, satisfying (2') are $u_1, u_2, \cdots, u_p$ where p is the smallest integer satisfying (2).

III. Proof of Correctness

In this section we prove that the SCHA algorithm with the revised formulas (1') and (2') solves the problem with nonuniform edge transmission times. Only the results that are directly related to the correctness of the revised formulas for the nonuniform case will be presented. For the remaining details, refer to [4].

A. Proof of Correctness of (1')

Lemma 1: Let $f(i)$ and $d(i)$ be *mappings* from the set $\{1, 2, \cdots, k\}$ to the set of nonnegative numbers. Let

$$Z(\pi) = \max_{1 \le i \le k} \left\{ f(\pi_i) + \sum_{j=1}^{i} d(\pi_j) \right\}$$

where π is a permutation of $\{1, 2, \cdots, k\}$. Suppose that for a permutation π of $\{1, 2, \cdots, k\}$, there exists m such that

$$f(\pi_m) < f(\pi_{m+1}). \tag{3}$$

Suppose further that π' is another permutation which is obtained from π by interchanging its mth and $(m+1)$st elements, i.e.,

$$\pi_i' = \begin{cases} \pi_i & i \neq m, m+1, \\ \pi_{m+1} & i = m, \\ \pi_m & i = m+1. \end{cases} \tag{4}$$

Then $Z(\pi) \ge Z(\pi')$.

Proof: $Z(\pi) = \max\{Z_0(\pi), Z_1(\pi), Z_2(\pi)\}$ where

$$Z_0(\pi) = \max\left\{ f(\pi_i) + \sum_{j=1}^{i} d(\pi_j) | 1 \le i \le k, i \neq m, i \neq m+1 \right\},$$

$$Z_1(\pi) = f(\pi_m) + \sum_{j=1}^{m} d(\pi_j),$$

$$Z_2(\pi) = f(\pi_{m+1}) + \sum_{j=1}^{m+1} d(\pi_j).$$

From this and (3), we have

$$Z_2(\pi) - Z_1(\pi) = d(\pi_{m+1}) + f(\pi_{m+1}) - f(\pi_m) > 0. \tag{5}$$

Also from (3) and (4),

$$\begin{aligned} Z_0(\pi) &= Z_0(\pi'), \\ Z_2(\pi) - Z_1(\pi') &= d(\pi_m) \ge 0, \\ Z_2(\pi) - Z_2(\pi') &= f(\pi_{m+1}) - f(\pi_m) > 0. \end{aligned} \tag{6}$$

Therefore,

$$\begin{aligned} Z(\pi) &= \max\{Z_0(\pi), Z_1(\pi), Z_2(\pi)\} \\ &= \max\{Z_0(\pi), Z_2(\pi)\} \\ &\ge \max\{Z_0(\pi'), \max\{Z_1(\pi'), Z_2(\pi')\}\} \\ &= \max\{Z_0(\pi'), Z_1(\pi'), Z_2(\pi')\} = Z(\pi') \end{aligned}$$

where the second equality is due to (5) and the inequality follows from (6).

Proposition 2: Let π^0 be a permutation of $\{1, 2, \cdots, k\}$ satisfying

$$f(\pi_1^0) \ge f(\pi_2^0) \ge \cdots \ge f(\pi_k^0). \tag{7}$$

Then $Z(\pi^0) \le Z(\pi)$, where π is any other permutation of $\{1, 2, \cdots, k\}$.

Proof: Let π be any permutation which does not satisfy (7), i.e., there exists m such that $f(\pi_m) < f(\pi_{m+1})$. Let π' be the permutation obtained from π by interchanging its mth and $(m+1)$st elements. Then by Lemma 1, $Z(\pi') \le Z(\pi)$. This interchange operation which does not increase the Z value and may actually reduce it can be repeated until we obtain π^0 satisfying (7). Therefore, $Z(\pi^0) \le Z(\pi)$.

From this proposition, we obtain the following corollary which is a generalized version of Theorem 1 of [4].

Corollary 3: Let $u_1, u_2, \cdots, u_k$ be the neighbors of a vertex v in a tree T, and let $T_i = T(u_i, v)$, for $i = 1, 2, \cdots, k$. Suppose further that $b(u_i, T_i) \ge b(u_{i+1}, T_{i+1})$, for $i = 1, 2, \cdots, k-1$. Then

$$b(v, T) = \max_{1 \le i \le k} \left\{ b(u_i, T_i) + \sum_{j=1}^{i} c_{v,u_j} \right\}.$$

Proof: For convenience, let $f(i) = b(u_i, T_i)$ and $d(i) = c_{v,u_i}$. Suppose that u_i receives the information from v at time $\sum_{j=1}^{i} d(\pi_j)$, $i = 1, 2, \cdots, k$, where π is a permutation of $\{1, 2, \cdots, k\}$. Then every vertex in T_i is informed by the time $f(\pi_i) + \sum_{j=1}^{i} d(\pi_j)$, and we deduce that

$$b(v, T) = \min_{\pi} \left\{ \max_{1 \le i \le k} \left\{ f(\pi_i) + \sum_{j=1}^{i} d(\pi_j) \right\} \right\}.$$

By Proposition 2, the permutation $\pi_i = i$ is the one attaining the above minimum.

This corollary states that our algorithm, which is the modified version of SCHA wherein the formulas (1) and (2) are replaced by (1') and (2'), respectively, is correct for the general case of nonuniform edge transmission times. If u is an endvertex of T whose inward neighbor is w, $t(u) = 0 = b(u, T(u, w))$. Then the algorithm proceeds to move inward and assigns the value $t(v)$ to each vertex v recursively by (1'). By induction and from Corollary 3, it can easily be seen that $t(v) = b(v, T(v, x))$, where x is the neighbor of v in T'.

B. Proof of Correctness of (2')

It will be shown that (2') is a sufficient condition for u_q to be a broadcast center. After a certain number of iterations, let v be the only one vertex left in T' and $u_1, u_2, \cdots, u_k$ be all the neighbors of v satisfying $t(u_1) \ge t(u_2) \ge \cdots \ge t(u_k)$. Note that

$$b(T) = b(v, T) = t(v) = \max_{1 \le i \le k} \left\{ t(u_i) + \sum_{j=1}^{i} c_{v,u_j} \right\}.$$

Let p be an index such that

$$t(u_i) + \sum_{j=1}^{i} c_{v,u_j} < b(T) \quad \text{for all } i < p \quad \text{and}$$

$$t(u_p) + \sum_{j=1}^{p} c_{v,u_j} = b(T).$$

Focusing on the above situation where there is only one vertex v left in T', we have the following lemma and proposition.

Lemma 4: $b(u_r, T) > b(T)$, for any $r > p$.

Proof: Since $b(v, T(v, u_r)) \ge t(u_1) \ge t(u_r) = b(u_r, T(u_r, v))$, it is optimal to transmit information first to v in order to find $b(u_r, T)$ and thus

$$b(u_r, T) = c_{v,u_r} + b(v, T(v, u_r)).$$

But by Corollary 3,

$$\begin{aligned} b(v, T(v, u_r)) &\ge \max_{1 \le i \le p} \left\{ t(u_i) + \sum_{j=1}^{i} c_{v,u_j} \right\} \\ &\ge t(u_p) + \sum_{j=1}^{p} c_{v,u_j} = b(T). \end{aligned}$$

Therefore, $b(u_r, T) > b(T)$.

This lemma indicates that u_r cannot be a broadcast center if $r > p$. We now list a condition for a vertex to be a broadcast center.

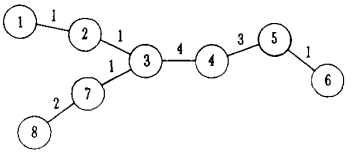

Fig. 1. Example of a tree with nonuniform edge transmission times.

Proposition 5: Let q be an index less than or equal to p such that

$$c_{v,u_q} + t(u_i) + \sum_{j=1}^{i} c_{v,u_j} \le b(T) \quad \text{for all } i < q. \tag{8}$$

Then $b(u_q, T) = b(T)$.

Proof: Since v is the vertex assigned a value last under our algorithm, we have $b(u_q, T(u_q, v)) = t(u_q) \le b(v, T(v, u_q))$. It then follows that

$$b(u_q, T) = c_{v,u_q} + b(v, T(v, u_q)) \ge b(v, T) = b(T). \tag{9}$$

But by Corollary 3, $c_{v,u_q} + b(v, T(v, u_q)) = \max\{b_1, b_2\}$ where $b_1 = \max_{1 \le i < q} \left\{ c_{v,u_q} + t(u_i) + \sum_{j=1}^{i} c_{v,u_j} \right\}$ and $b_2 = \max_{q < i \le k} \left\{ t(u_i) + \sum_{j=1}^{i} c_{v,u_j} \right\}$. Now by (8), $b_1 \le b(T)$ and $b_2 \le \max_{1 \le i \le k} \{ t(u_i) + \sum_{j=1}^{i} c_{v,u_j} \} = t(v) = b(T)$. From these and (9), $b(T) \ge \max\{b_1, b_2\} = b(u_q, T) \ge b(T)$. Hence, $b(u_q, T) = b(T)$.

For computational efficiency, condition (8) can be written equivalently as (2′).

Remark: It is worth noting that u_1 always satisfies (2′) and thus at least two centers exist for any tree, which is compatible with Corollary 8 in [4]. But there is no parallel theorem to Theorem 9 in [4], which is illustrated in Fig. 1. It is easy to see that $BC(T) = \{3, 4\}$ and $b(T) = 8$. Choose a vertex, say 5. Then we have $b(5) = 10$. But this value is not equal to $b(T) + c_{5,4}$.

IV. Time Complexity of the Algorithm

This section shows that the worst case time complexity of our algorithm is $O(N \log N)$, where N is the number of vertices of the tree. The most time-consuming parts are 1) finding the minimum $t(w)$ of the assigned values in the current tree T', 2) calculating $t(v)$ when v becomes an endvertex of T' after deleting w from T', and 3) checking the condition (2′) at the final step.

First, using a heap sort [3], it is simple to show that the time complexity is $O(N \log N)$ for the part 1). Second, the process of calculating $t(v)$ is the same as the one in [4] except that the updated value of $\mathrm{MAX}(v)$ is determined by $\mathrm{MAX}(v) \leftarrow \max\{\mathrm{MAX}(v), t(w)\} + c_{vw}$. But this modification does not increase the time complexity. Finally, for checking the condition (2′), let $C_i = t(u_i) + \sum_{j=1}^{i} c_{v,u_j}$. Then we have $M_r = \max_{1 \le i \le r} \{C_i\}$, $C_{i+1} = C_i - t(u_i) + t(u_{i+1}) + c_{v,u_i}$, and $M_{r+1} = \max\{M_r, C_{r+1}\}$. These values can be obtained recursively in $O(N)$ time, which means that the time complexity for the final step is $O(N)$.

References

[1] A. M. Farley, "Minimal broadcast networks," *Networks,* vol. 9, pp. 313–332, 1979.

[2] ——, "Minimum-time line broadcast networks," *Networks,* vol. 10, pp. 59–70, 1980.

[3] D. E. Knuth, *The Art of Computer Programming: Sorting and Searching, Vol. 3.* Reading, MA:, CA: Addison-Wesley, 1973.

[4] P. Slater, E. Cockayne, and S. Hedetniemi, "Information dissemination in trees," *SIAM J. Comput.,* vol. 10, pp. 692–701, Nov. 1981.

Testing for Coupled Cells in Random-Access Memories

J. Savir, W. H. McAnney, and S. R. Vecchio

***Abstract*—Two test strategies for memory testing are compared for their ability to detect coupled-cell faults in an n word by 1 bit random access memory. In both strategies the data-in line is randomly driven. One of the two strategies uses random selection of both the address lines and the read/write control. The other strategy sequentially cycles through the address space with deterministic setting of the read/write control.**

The relative merit of the two strategies is measured by the average number of accesses per address needed to meet a standard test quality level.

***Index Terms*— Escape probability, memory testing, pattern-sensitive fault, random testing, signal probability.**

I. Introduction

In this paper, we compare two different test strategies for their ability to detect coupled-cell faults in a random access memory. A coupled-cell fault is a transition-creating influence from one storage cell i to another storage cell j, under various conditions relating to the values of the neighboring cells.

Section II describes the memory used in the study. Section III provides a description of the fault model, and Section IV lists and describes the two test strategies that are considered. Section V compares the two strategies and draws some conclusions.

Fuentes *et al.* [1], and David *et al.* [2], show numerical results on several types of pattern sensitive faults for the random test strategy. No analytical solution to either the test length or the test quality is given in [1] and [2]. Also, the Markov chain used in [1] and [2] to describe the detection process can be greatly simplified [3].

II. The Memory Model

Since we are interested in a comparative analysis between two different test strategies, we assume a simple single-port n word by 1 bit random access memory.

When the read/write (R/W) control line is set to write, the bit on the data-in-line is written to the selected address. When the control line is at a read, the stored bit at the selected address is read to the data-out line.

III. The Fault Hypothesis

The fault model assumes a coupling between a pair of cells such that a $0 \rightarrow 1$ transition in one cell causes a $0 \rightarrow 1$ transition in the other cell only for a fixed value of other cells G in the neighborhood [1], [2].

Manuscript received January 19, 1989; revised September 30, 1990.

J. Savir and W. H. McAnney are with IBM Data Systems Division, Poughkeepsie, NY 12602.

S. R. Vecchio is with IBM General Technology Division, Hopewell Junction, NY 12533.

IEEE Log Number 9101693.