The procedure of hazard detection is performed in two phases:

Phase 1—Determining all the changing $Y$ signals: Line 1 of the analysis table corresponds to the initial stable state of the sequential machine. The first step is to determine which of the $Y$ signals may be changing as a result of the specified $X$ variable change. By substituting $X_1 = \bar{X}_1 = 1$ (line 2) we detect a possible change in $Y_2 = 1$ while $Y_1 = 0$ remains unchanged. We proceed by determining whether any additional change in the $Y$ signals may be detected as a result of both the $X$ variable and the $Y$ variable change. In line 3 we detect a change in $Y_1$ as a result of a possible change in $X$ and $Y_2$. Evidently, the additional change in the $Y_1$ signal cannot eliminate the previous change in $Y_2$ because a hazard that exists for a change of some variables will not disappear if additional variables are changed during the same transition. Therefore, any $Y$ signal for which a change was detected in this phase need not be further considered. This phase will end after all possible changes in the $Y$ signals have been detected.

Phase 2—Determining which $Y$ signals stabilize: Line 4, the beginning of this phase, corresponds to the expected final stable state of the machine. With regards to this state, the stabilization of the machine is analyzed. Therefore both $Y_1$, $Y_2$ depend on $y_1$ as well as $y_2$ and the test for 0-hazard in $Y_1(y_1 = 11, y_2 = 11)$ differs from the 1-hazard test in $Y_2(y_1 = 00, y_2 = 00)$; therefore, a 0-hazard for $Y_1$ is analyzed in line 5a and a 1-hazard for $Y_2$ is analyzed in line 5b. The result is that $Y_2$ will stabilize on “1” level while the $Y_1$ signal a hazard is still expected.

The process will be continued by examining additional $Y$ signals which may be stabilized. In line six it is shown that the $Y_1$ signal did not stabilize due to stabilization of the $Y_2$ signal.

Clearly, a stabilized $Y$ signal will not change again due to stabilization of additional $Y$ signals. Therefore, in this phase stabilized $Y$ signals need not further be considered. This phase will end when no additional $Y$ signal will stabilize $Y_1$, $Y_2 = (1,1)$, $(0,1)$.

In using this procedure, oscillations, hazards, and races are automatically detected. For $n$ feedback lines, only $2n$ evaluations are required at most.

VI. Conclusions

A simplified binary technique has been described for detecting multiple logic and function hazards. It has been shown that the conditions for hazard appearance can be studied by analyzing the network behavior during extreme transient conditions. These extreme conditions may be simulated for forcing a 0 or 1 value simultaneously to all the changing signals. The same technique was applied to detect hazards in sequential circuits.

ACKNOWLEDGMENT

The author wishes to thank Prof. M. Yolli and Prof. A. Lempel for their helpful suggestions and contributions to the research and Prof. A. Kandel for his advice in preparation of the article.

REFERENCES


Combinatorial Merging

MARTIN C. GOLUMBIC

Abstract—Three algorithms are given to construct a weighted $r$-ary tree with a given set of integer weighted leaves in which the weight of a parent node is $1 + max$ of its sons. Two goals are of interest; to minimize

1) the weight of the root of the tree, and
2) the number of internal nodes.

Formulas for the optimum values of 1) and 2) are presented. We show that the first algorithm simultaneously optimizes both 1) and 2), the second only 2) while nearly optimizing 1), and the third opt-

Manuscript received November 17, 1974; revised May 11, 1975 and November 18, 1975. This work was supported by the National Science Foundation under Grant C4-34738.

The author was with the Department of Mathematics, Columbia University, New York, NY 10027. He is now with the Department of Computer Science, New York University, New York, NY.
timizes only 1). However, the first algorithm can be used in dynamic situations only when the number of leaves is known at the outset; thus, suboptimal procedures are sometimes required.

The results are applicable to parallel processing problems and to switching circuit theory.

Index Terms—Fan-in, integer weighted nodes, number of internal nodes, parallel processing, r-ary tree, switching circuit theory, weight of the root of the tree.

I. INTRODUCTION AND MOTIVATION

The level of a signal in a combinational logic circuit is defined inductively. Input signals are of level zero, and the output of a gate has level one plus the maximum of the levels of its inputs. We can reduce delay in a circuit by minimizing the level of its output.

A procedure realizing the disjunction of single output circuits having varying levels using limited fan-in OR gates, which minimizes the resulting circuit, is given in [2] and [4]. Unfortunately, this uses more OR gates than necessary. In this correspondence we will present a method which both minimizes the level of the resulting circuit and uses exactly the minimum number of such gates (Procedure I). A numerical formula is also given for the minimum level.

This combining operation is actually only a combinatorial merging operation and is not dependent upon switching circuit theory. We will therefore formulate our model for a general r-ary commutative and associative merging operation with a level structure analogous to that described above.

II. DEFINITIONS AND NOTATION

A rooted tree is a system \( \langle N, n_0, f \rangle \), where \( N \) is called the set of nodes, \( n_0 \in N \) is called the root of the tree, and \( f \) is a function from \( N - \{ n_0 \} \) to \( N \) such that \( f^h(x) \neq x \), for all \( h > 0 \) and \( x \in N \) - \( \{ n_0 \} \). [Recursively, \( f^0(x) = f(f^{h-1}(x)) \).]

The image of \( f \) is called the set of internal nodes while the remaining nodes are called leaves of the tree. An r-ary tree is a rooted tree with the additional property that \( f^{-1}(x) \leq r \), for all \( x \in N \), (where we denote by \( |S| \) the cardinality of a set \( S \)). The number \( |f^{-1}(x)| \) is called the fan-in of \( x \). The following lemma is immediate.

Lemma 1: An r-ary tree with \( n \) leaves must have at least \( \lceil (n-1)/(r-1) \rceil \) internal nodes.

The symbol \( \lceil z \rceil \) denotes the smallest integer greater than or equal to \( z \).

To complete the model a weight \( w(x) \) is assigned to each node \( x \) of a rooted tree according to the following rules.

Rule 1: Leaves are assigned arbitrary integer values.

Rule 2: An internal node \( y \) is assigned the value

\[
w(y) = 1 + \max\{w(x) \mid x \in f^{-1}(y)\}.
\]

Thus, the sequence of values assigned to the nodes along any branch of the tree is strictly increasing in the direction of the root, and there is at least one branch whose increments are exactly one at each successive node from leaf to root.

Our problem can now be formulated mathematically. Given a set of \( n \) integer weighted nodes, construct an r-ary tree with these \( n \) nodes as leaves such that the following two values are both minimized:

1) the value assigned to the root, and
2) the number of internal nodes.

III. THE ALGORITHMS

Let \( C \) be a collection of \( n \) integer weighted nodes and \( R \) be the set of nodes for which the successor function \( f \) has not yet been defined. Each algorithm works inductively beginning with \( R = C \) and continuing until \( |R| = 1 \). The expression merge a set of nodes \( S \subseteq R \) means creating a set of \( \lceil |S|/r \rceil \) new nodes \( S' \) extending \( f \) to \( S' \) such that for all \( s \in S \) and \( s' \in S' \) we have \( f(s) \in S' \) and \( |f^{-1}(s')| \leq r \). Thus, \( R \) is updated by replacing \( S \) by \( S' \).

Examples of the algorithms are given in Fig. 1.

Procedure I:

Write \( |C| = q + k(r - 1) \) with \( 2 \leq q \leq r, k > 0 \); \( q, k \) integers. Let

\[
t = \left\{ \begin{array}{ll}
q, & \text{on first iteration,} \\
r, & \text{otherwise.}
\end{array} \right.
\]

a) Merge \( t \) nodes from \( R \) having the smallest weights in \( R \).

Procedure II:

b) Merge \( \min\{r, |R|\} \) nodes from \( R \) having the smallest weights in \( R \).

Procedure III:

c) Merge

\[
|c \in R| w(c) = \min_{d \in R} w(d).
\]

Note that if \( |C| \) is of the form \( m(r - 1) + 1 \), then a) and b) coincide, and this is true in particular for \( r = 2 \).

IV. THE MAIN RESULTS

Theorem 2: Let \( C \) be a collection of nodes and \( w(c) \) be the integer weight assigned to node \( c \in C \). The minimum value that can be assigned to the root of an r-ary tree with \( C \) as its leaves is

\[
L = \left\lceil \log_r \sum_{c \in C} r^{w(c)} \right\rceil.
\]

Thus, Theorem 2 and Lemma 1 give lower bounds for 1) and 2), respectively.

Theorem 3:

a) Procedure I yields root weighted \( L \).

b) Procedure II yields root weighted at most \( 1 + L \).

c) Procedure III yields root weighted \( L \).

d) Procedures I and II use \( (n-1)/(r-1) \) internal nodes.

This theorem shows that Procedure I is optimal w.r.t 1) and 2), Procedure II is optimal w.r.t 2) and at worst one greater than optimal w.r.t 1), while Procedure III is optimal w.r.t 1).

V. THE PROOFS

Theorem 3d): Immediate.

Theorem 3a): Procedure I yields root weighted \( L \).

The theorem is trivial for \( |C| > 1 \). Continuing inductively, let \( |C| = q + k(r - 1) \) with \( 2 \leq q \leq r \) and \( k \geq 0 \). Let \( Q \) be a subcollection of \( C \) containing precisely \( q \) nodes of smallest possible weights, \( D = C - Q \) and \( m = \max\{w(c) \mid c \in Q\} \). Since \( 2 \leq q \leq r \),

\[
r^m + 1 \leq \sum_{c \in Q} r^{w(c)} \leq r^{m+1}.
\]

Notice that \( |D| = k(r - 1) \); this will be important later.

Merge the \( q \) nodes of \( Q \) getting a new node weighted \( m + 1 \). We are left with a new collection \( C' \) of \( 1 + |D| \) nodes. By induction, Procedure I applied to \( C' \) yields root with weight given by the right side of (3), where

\[
d = \sum_{c \in D} r^{w(c)}.
\]

The left side of (3) equals \( L \), and the inequality follows from (2):

\[
\left\lceil \log_r \left( d + \sum_{c \in Q} r^{w(c)} \right) \right\rceil \leq \left\lceil \log_r (d + r^{m+1}) \right\rceil.
\]
Since \( w(c) \geq m \) for all \( c \in D \) we can write

\[
d = a_M a_{M-1} \cdots a_m 00 \cdots 0 \text{(base } r)\text{.}
\]

Moreover, \( r - 1 \) divides \( a_M + \cdots + a_m \) since \( r - 1 \) divides \(|D|\).

(\text{Remark: Earth dwellers may find this proof most satisfying by recalling many special properties of numbers divisible by 9 in base 10.) If \( d + r^{m+1} \leq r^{M+1} \), then equality holds in (3) and equals \( M + 1 \). Otherwise,

\[
d + r^{m+1} = a_M r^M + \cdots + (a_{m+1} + 1)r^{m+1} + a_m r^m > r^{M+1}
\]

implying \( a_{m+1} = a_{m+2} = \cdots = a_M = r - 1 \), and since \( r - 1 \) divides

\[
\sum_{i=m}^{M} a_i
\]

the strict inequality of (4) implies \( a_m = r - 1 \). Hence, (2) yields

\[
r^{M+1} - d + \sum_{c \in C} w(c) \leq d + r^{m+1} = a_m r^m + r^{M+1} < r^{M+2}
\]

Thus equality holds in (3) and equals \( M + 2 \).

Theorem 3b): Procedure II yields root weighted at most \( 1 + L \).

Given two copies of \( C \), let \( A_i \) and \( B_i \) be the remainders after \( i \) iterations of Procedures I and II, respectively, (i.e., \( A_0 = B_0 = C \) while \( A_N = [a] \) and \( B_N = [b] \), where \( N = [(n - 1)/(r - 1)] \) and \( w(a) = L \). We will show that \( w(b) = 1 + w(a) \).

Now clearly \( B_{i-1} \leq \max A_i \), and by simultaneous iterations of a) and b) we have

\[
\max B_{i-1} \leq \max A_i, \quad \text{for } 1 \leq i \leq N
\]

(max \( B_i \) denotes \( \max \{w(c) : c \in B_i \} \)).

Thus equality holds in (3) and equals \( M + 2 \).

Theorem 3c): Procedure III yields root weighted \( L \). Let

\[
m = \min \{w(c) : c \in C \}, \quad D = C - C_m,
\]

\[
M = \max \{w(c) : c \in C \}, \quad k = |C_m|,
\]

\[
C_m = \{c \in C : w(c) = m\}, \quad d = \sum_{c \in D} w(c).
\]

Notice that \( r^{m+1} \) divides \( d \). Proof will be by induction on the triple \((|C|, M - m, k)\) under the lexicographic ordering:

\[
(|C'|, j', k') < (|C|, j, k), \quad \text{if and only if}
\]

\[
|C'| < |C|\text{ or } |C'| = |C| \text{ and } j' < j, \text{ or } |C'| = |C| \text{ and } j' = j \text{ and } k' < k.
\]

We assume \(|C| > 1\) and that Procedure III attains the lower bound of Theorem 2 for all triples less than \((|C|, M - m, k)\).

Merging \( C_m \) will leave a collection \( C' \) consisting of \( D \) plus \( [k/r] \) new nodes each weighted \( m + 1 \).

Thus \( |C'| = |D| + [k/r] \leq |D| + k = |C| \) with equality if and only if \( k = 1 \), in which case \( M > m \) and \( M - (m + 1) < M - m \).

Therefore, by induction, Procedure III applied to \( C' \) yields an \( r \)-ary tree whose root is weighted

\[
L' = \left\lfloor \log_r \left(d + \frac{k}{r}\right)^{r^{m+1}} \right\rfloor.
\]

Certainly

\[
L = \left\lfloor \log_r \left(d + \sum_{c \in C_m} w(c)\right)^{r^{m+1}} \right\rfloor \leq L',
\]

since \( kr^m \leq \lfloor k/r \rfloor r^{m+1} \). Suppose there is strict inequality in (5), then

\[
d + kr^m \leq r^L < d + \frac{k}{r} r^{m+1}.
\]

But the right most entry of (6) is divisible by \( r^{m+1} \), and since \( r^L + r^{m+1} \) is the smallest integer greater than \( r^L \) which is divisible by \( r^{m+1} \), it follows that

\[
d + kr^m \leq r^L < r^L + r^{m+1} \leq d + \frac{k}{r} r^{m+1}.
\]

Thus from (7), \( kr^m + r^{m+1} \leq \lfloor k/r \rfloor r^{m+1} \), however, \( [k/r] r^{m+1} < (k/r + 1) r^{m+1} = kr^m + r^{m+1} \), contradiction. Therefore equality must hold in (5).

Proof of Theorem 2: Continue under the assumption that Procedure III is optimal for all triples less than \((|C|, M - m, k)\). Any other procedure would have to join a node \( c_m \) of minimal weight \( m \) with a node of higher weight \( p > m \). Then we could just as well replace the weight of \( c_m \) by \( p \) without affecting the result of this new procedure. This replacement would keep \(|C|\) fixed and either reduce \( M - m \) if \( k = 1 \), or reduce \( k \) leaving \( M - m \) fixed if \( k > 1 \). So by induction, the best possible weight assignable to the root under this procedure is

\[
\left\lfloor \log_r \left(r^p + \sum_{c \in C \setminus C_m} w(c)\right)^{r^{m+1}} \right\rfloor
\]

which is greater than or equal to \( L \).

VI. CONCLUSION

In applying our results to parallel processing problems a tradeoff from optimality may be necessary. Consider the following dynamic situation.

A new batch of data items arrives at a processing station at unit intervals. Beginning immediately, the set \( C \) of items received before a deadline is to be merged into a single data item using machines in parallel which can process up to \( r \) items per unit time merging their results. Since \(|C|\) is not known in advance of the deadline, Procedure I cannot be used. Procedure III yields optimal completion time at the expense of extra machine costs, whereas Procedure II minimizes these costs but may cause a delay of one time unit.
Huffman's algorithm [3] for optimal prefix coding in an r-letter alphabet is essentially our Procedure I, while Procedure III is similar to that of Hicks and Bernstein [2], [4] for switching circuits.

REFERENCES

System Diagnosis and Redundant Tests
S. TOIDA

Abstract—If the addition of a test does not improve the diagnosability of a system the test is called redundant. It is shown that if at least one test is nonredundant, then the previous algorithms to determine the diagnosability [2], [3] can be further improved. It is also shown how and when the diagnosability of a system can be improved by the addition of an extra test.

Index Terms—Fault diagnosis, redundant tests, self-checking, test connections.

I. INTRODUCTION

Let us consider a system of operating units such as computers, people, circuits, etc. Let us assume that each unit of the system is capable of testing the correctness of others. One way of diagnosing the system then is as follows: first a sequence of tests is performed. In each test a unit tests another unit to find if the tested unit is faulty or faulty free. After testing each unit of the system in some sequence we obtain a set of test outcomes. From this set of test outcomes a central unit determines what units are faulty.

One question one might ask concerning the diagnosis of a system is, given a set of test outcomes, can one identify all faulty units in the system?

Let us call a system t-diagnosable iff all the faulty units can be uniquely identified provided there are no more than t faulty units in the system [1].

Preparata et al. [1] first showed that if a system of n units is t-diagnosable then n ≥ 2t + 1 and each unit must be tested by at least t other units in the system. Hakimi and Amin [2] showed that these conditions are in fact sufficient if no two units test each other in the system. They also give a necessary and sufficient condition for a general system to be t-diagnosable. Allan et al. [3] proposed a new unified approach to the problem of system diagnosis and presented a more efficient algorithm to determine the diagnosability of a system.

In this paper we introduce the concept of redundant tests in relation to the diagnosability of a system. The study of redundant edges reveals that there is a chance of improving the efficiency of the previous algorithms [2], [3] to determine the diagnosability. Furthermore the study suggests a way to improve the diagnosability of a system by adding an extra test.

II. PRELIMINARIES

A. Representation by Digraph

As Preparata et al. mention [1], a set of test outcomes can be represented by a digraph with binary weights. In this graph each unit of a system will be a vertex of the graph. The presence of an edge eij with direction from i to j signifies the fact that there is a test in which vi tests vj. The weight associated with eij is 0 or 1. It is 0 if under the hypothesis that vi is fault free, vj is fault free. It is 1 if under the same hypothesis vi is faulty. If weights are ignored, a digraph clearly represents testing arrangements of a system.

B. Notation

A digraph is denoted by \( G = (V, E) \), where \( V \) is the set of vertices and \( E \) is the set of edges. A simple "graph" means a digraph unless otherwise specified.

Following are the symbols used frequently in this paper:

\[ x, y, e \in E \text{ if } x \text{ tests } y, \text{ where } x, y \in V; \]
\[ E(Y \times Z) = \{(y, z) \mid y \in Y \text{ and } z \in Z\}; \]
\[ |X| = \text{ the cardinality of a set } X; \]
\[ \Gamma X = \{v \in V \mid \exists x \in X, \text{ such that } (x, v) \in E \} \setminus X \]
\[ \Delta X = \{v \in V \mid \exists z \in X, \text{ such that } (z, v) \in E \} \setminus X \]
\[ p = \text{ the smallest integer not less than } p. \]

C. Partition of V

The following partition of \( V \) was first introduced by Allan [4]. It plays an important role in this paper.

Definition: \( [X, Y, Z] \) is a partition of \( V \) if \( X, Y, Z \subset V \mid Z \mid \geq 1, \]
\[ \Gamma X = Y, \text{ and } \Delta Z = Y. \]

Allan et al. [3] showed that a system is t-diagnosable iff

\[ t \leq \min \left( |V| + \left\lceil \frac{|Z|}{2} \right\rceil - 1 \right); \]

\[ \text{min is over all partitions.} \]

The right side of this inequality is defined to be \( \tau \), i.e.,

\[ \tau = \min \left( |V| + \left\lceil \frac{|Z|}{2} \right\rceil - 1 \right). \]

To indicate a partition or \( \tau \) is that of a graph \( G \), we use a subscript such as \( [X_0, Y_0, Z_0] \) or \( \tau_G \).

D. How to Obtain a Partition of V

According to Allan [4] a partition of \( V \) can be obtained by the following procedure.

1. Start with an arbitrary nonempty set \( Z', Z' \subset V \), as an initial \( Z \).
2. Form \( X \) as \( X = \{x \in V \mid \forall z \in Z', (x, z) \notin E\} \).
3. Form \( Y \) as \( Y = \{y \in V - X - Z' \mid \exists x \in X, \text{ such that } (x, y) \in E\} \).
4. Form \( Z \) as \( Z = V - X - Y \).
5. \( [X, Y, Z] \) is a partition of \( V \).

III. REDUNDANT EDGES

As one might expect, in general, some edges in a graph do not contribute to the magnitude of \( \tau \); that is, removing them does not