1972

Extensions to the e-algorithm for fault detection in combinational logic cicuits

Siu-Chong Si
Lehigh University

EXTENSIONS TO THE E-ALGORITHM FOR FAULT DETECTION IN COMBINATIONAL LOGIC CIRCUITS

by Siu-Chong Si

ABSTRACT

This thesis deals with the detection of stuck-at type faults in combinational circuits. Two different ways are presented of modifying the "E-algorithm" for generating fault detection test sets of a combinational network.

We show information about the topological structure of a combinational network can be embodied within its "E-expression". Under the single fault assumption, the generation of test sets using the proposed modifications to the original E-algorithm is likely to require somewhat less effort, since it needs only one pass in the E-algorithm in generating a sufficient test set. It is expected that the principal application of this approach will be in fault location problems even though this approach is guaranteed to be as good as existing fault detection methods.

We also show how to apply the E-algorithm to a large combinational network, and how to bring the D-algorithm and the E-algorithm together. How a "D-cover" of a sub-network can be built efficiently will also be described.

EXTENSIONS TO THE E-ALGORITHM FOR FAULT DETECTION IN COMBINATIONAL LOGIC CIRCUITS

by

SIU-CHONG SI

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Electrical Engineering

Lehigh University

1972

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

May 18, 1972
(date)

A K Susskind
Professor in Charge

A K Susskind
Professor A. K. Susskind
Chairman of Department

ACKNOWLEDGEMENT

I wish to thank my advisor Professor A. K. Susskind for his patience and guidance during the preparation of this thesis. Appreciation is also expressed to Miss Patti Talbot for her helpful discussions and to Henrik Schutz for his help with the English language.

The financial support provided by the Department of Electrical Engineering through a Byllesby Fellowship and a William Esty Memorial Research Fund Fellowship is gratefully acknowledged.

TABLE OF CONTENTS

ABSTRACT

This thesis deals with the detection of stuck-at type faults in combinational circuits. Two different ways are presented of modifying the "E-algorithm" for generating fault detection test sets of a combinational network.

We show information about the topological structure of a combinational network can be embodied within its "E-expression". Under the single fault assumption, the generation of test sets using the proposed modifications to the original E-algorithm is likely to require somewhat less effort, since it needs only one pass in the E-algorithm in generation a sufficient test set. It is expected that the principal application of this approach will be in fault location problems even though this approach is guaranteed to be as good as existing fault detection methods.

We also show how to apply the E-algorithm to a large combinational network, and how to bring the D-algorithm and the E-algorithm together. How a "D-cover" of a sub-network can be built efficiently will also be described.

I. Introduction

This thesis deals with the detection of stuck-at type faults in combinational circuits. A switching circuit is combinational if its outputs at time t are independent of past inputs. A fault of a combinational circuit is a physical defect of one or more components which can cause the circuit to malfunction. Faults that are permanently stuck-at-high or permanently stuck-at-low are called stuck-at faults (logical faults). Marginal, intermittent faults are excluded from this class. (Ref. 1)

The approaches that we present here are all based on the so-called E-algorithm. (Refs. 2, 3) The E-algorithm was originally designed for handling multiple-faults in combinational circuits. In order for this thesis to be more self-contained, a description of the original E-algorithm is presented at the end of this chapter. In Chapter II, we show how information about the topological structure of a combinational network can be imbedded into its E-expression. In Chapter III, we show how to apply the E-algorithm to a large combinational network, and how to bring the D-algorithm and the E-algorithm together.

The proof of the sufficiency of the E-algorithm may be found in Ref. 1 and is omitted here. Below we only present the rules for finding an E-expression and show how to compute the test set from an E-expression.

An <u>E-expression</u> is a somewhat unconventional sum-of-products Boolean expression for the output of a combinational

circuit. It is obtained by following the ordinary rules except for

(i) Literal conflicts (like $x\bar{x}$) are preserved (not to zero) and denoted by the corresponding capital (here X).

(ii) Simplification due to proper containment is not allowed, i.e., $a + ab \not\Rightarrow a$.

The only simplification rules permitted are:

1. $p \cdot 1 = 1 \cdot p = p$; $p + 1 = 1 + p = 1$
2. $p \cdot 0 = 0 \cdot p = 0$; $p + 0 = 0 + p = 0$
3. $p \cdot p = p$; $p + p = p$

Created terms are those that arise because one of the two literals that form a conflict is eliminated (i.e., changed to value 1 because of some error). For example, given a conflict term like $a\bar{b}cD$, due to some fault, terms like $a\bar{b}cd$ or $a\bar{b}c\bar{d}$ may arise. The concepts of "growth" and "shrink" of the "true body" need to be clarified. For any cube defined by a product in a Boolean expression (meaning E-expression here), due to some fault (or faults) one or more of the literals in that product may change to have the permanent value 1. This results in a growth (enlargement) of the true body. For example, in Fig. 1.1, the normal output response of the circuit is $ab\bar{c} + d$. If the literal b is changed to have permanent value 1, the resulting output response of the circuit will be $a\bar{c} + d$. Because the true vertices in $ab\bar{c} + d$ are contained in true vertices of $a\bar{c} + d$, we say that the logical response of the network grows from the true body of $ab\bar{c} + d$ to $a\bar{c} + d$. We say

that a function shrinks if due to some fault(s), one or more of the literals in a product changes to have permanent value zero. This results in a "shrinkage" ("drop") of the true body. For example, if in Fig. 1.1 the literal a is changed to

$F = ab\bar{c} + d$

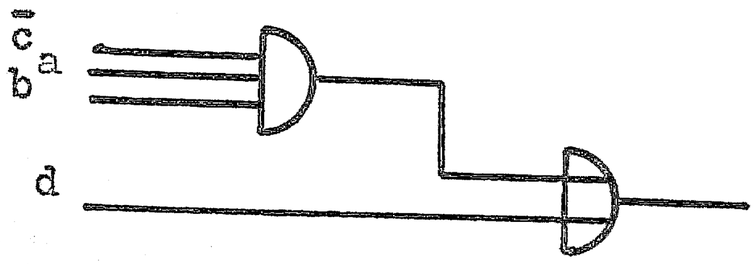


Fig. 1.1

have permanent value zero, the resulting output response of the circuit will become d. Because the true body of d is in the true body of $ab\bar{c} + d$, we say that the response of the network shrinks from the true body of $ab\bar{c} + d$. A test checking for any growth (shrink) of a network is called a growth-test (existence-test) for that network. The following three algorithms are copied verbatim from Ref. 2. In following chapters, the $\{T_i\}$ set is equivalent to the $\{R_k\}$ set in the algorithms described here.

Growth Test Generation Algorithm (GTGA)

Let P_i^j be the literal product which equals $P_i = x_1^* x_2^* \ldots x_g^*$ except that the jth literal is complemented, and let P_i' be the product which equals P_i except that the jth literal is omitted.

1. Set k = 0.

2. For each conflict term in the network output E-expression E_N, derive all the created terms which can be obtained by resolving all literal conflicts in the term in all possible ways, i.e., replace each capital letter X in turn by x and $\bar{x}$ and then apply DeMorgan's law and Rule (ii) as may be appropriate. Literal conflicts arising during the reduction of a conflict term are not set to zero, but are treated as any other conflict term. Label the distinct created products obtained as P_a through P_q. (Example: $\bar{A}bc$ yields $\bar{a}c$, ac, and $\bar{b}c$).
3. Set all literal conflicts in E_N to zero, i.e., set all capital letters to zero, and, if applicable, use only DeMorgan's law along with the idempotent laws of Rule (ii) to obtain the sum-of-products expression E_F for fault-free operation.
4. Place each created term P_d in R_k if $P_d \subset E_F$; otherwise put in set L all vertices for which P_d is true and E_F false.
5. Number the products in E_F from 1 to p and call them P_1 to P_p. Thus $E_F = \sum_{i=1}^{p} P_i$.
6. Set i to 1.
7. Set j to 1.
8. Is $P_i^j \subset E_F$? If YES, place P_i^j in R_k. If NO, place in set L all vertices for which P_i^j is

true and E_F false.

9. Increment j and return to step 8 until all p_i^j have been processed.

10. Increment i and return to step 7 until all p_i^j have been processed.

11. Obtain R_k' from R_k by removing from R_k all repeated elements, or elements that are copies of products in E_F or R_j', for $j < k$.

12. If R_k' is empty, go to step 14. If not, continue.

13. Return to step 5 with products in E_F replaced by the products in R_k', and in step 8 the index k is incremented.

14. Obtain L' from L by removing all duplications from L and enter the Growth Test Minimization Algorithm (GTMA).

Set L' contains all the nominally false vertices of E_F which could be used in testing for all possible combinations of enlargements, creations, and enlargements of creations. In general, not all the vertices in set L' need be included in the growth test set because, as pointed out before, it is both proper and highly desirable to test simultaneously for more than one growth. The problem then is to determine a minimum set of vertices from set L' which will detect each of the created terms, P_d, and enlargements, P_i^j, (those of E_F and the R_k' sets) which were not covered by E_F. Determining a

minimum size growth test set from L' is a form of the classical prime implicant covering problem.

Growth Test Minimization Algorithm (GTMA)

1. Form a matrix in which the row coordinates are the vertices in the set L', and the column coordinates are the growths detected by the elements in L'. Place a check mark in element (i, j) of the matrix if the coordinate (vertex) of row i is covered by (detects) the coordinate (growth term) of column j.
2. Select a minimum subset of the vertex labels such that there is at least one check in each column. (This is the classical prime implicant covering problem. Methods for obtaining solutions can be found in almost any basic text on switching theory.)

Existence Test Generation Algorithm (ETGA)

1. Construct a matrix in which the row coordinates are the true vertices of the fault-free function E_F, and the column coordinates are all the products of E_F, and also the elements of all R'_k sets. Place a check mark in element (i, j) of the matrix if the vertex label of row i is covered by the product label of column j.

2. Eliminate rows from the matrix using the following two rules:

   a. Eliminate any row which is a duplicate of any other row.

   b. If row k has a check in every column that row i does, and also in at least one additional column, then row k may be eliminated. Row k is said to dominate row i.

3. The surviving rows specify the set of vertices, L_E, that assures a complete existence (true-body) test.

In general, the test set obtained by the ETGA need not be unique in content, but only in size. This is because of the choice permitted by the first matrix simplification rule in step 2. The justification for the second simplification rule is as follows. Suppose r_k dominates r_i and vertices v_k and v_i are the row coordinates of the patterns of checks, respectively. Clearly, if the network being tested responds correctly (is 1) for v_i, then it must also respond correctly for v_k because r_k "contains" all the products (column headings) that r_i does. The existence matrix does not lead to the "classical covering problem" as the growth matrix does. No columns may be eliminated from the existence matrix (as can be done in the growth matrix) because which column headings (terms) actually exist for the network on hand is unknown.

II. Adding Path Information to E-Expression and Single-Fault Tests

A. Introduction

The E-algorithm is a purely algebraic procedure, originally designed for handling multiple stuck-type faults in combinational networks. Tests are found with little regard for details of network structure and without explicitly building up sensitized paths. Undetectable faults are automatically taken care of by the algorithm, as are problems of multiple reconvergent fan-out. Most of the equivalent faults (or undistinguishable faults) are taken care of by the minimization algorithm. That the E-algorithm ignores some structural information is clearly demonstrated from the fact that given an E-expression of a network, the network cannot be uniquely found.

There are cases where the failure to fully consider topological structure leads to serious drawbacks. For example, it is well-known that the parity tree shown in Fig. 2.1 has a complete test set of only four inputs. However, if we use the

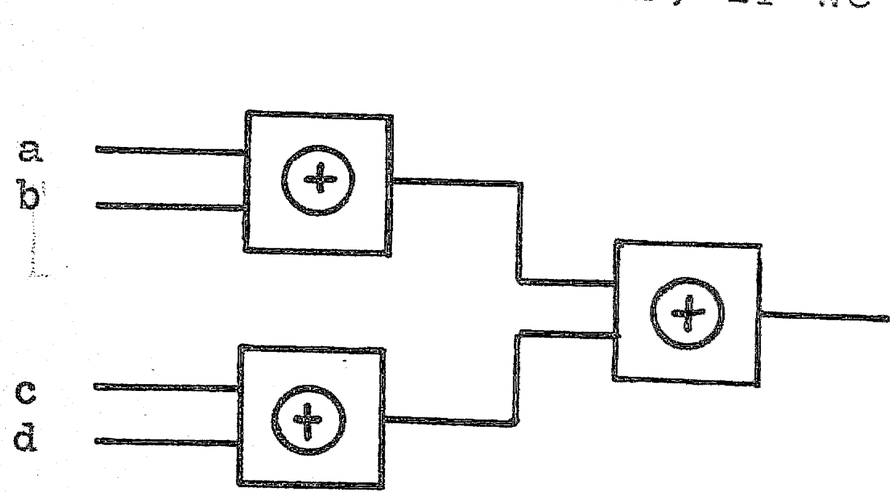


Fig. 2.1

E-algorithm to find the tests, a list of 16 tests is found.

Furthermore, the larger the parity tree, the more unfavorable the results of the E-algorithm become.

In this chapter we show a way of incorporating information about the network structure into the E-expression by modifying our previous procedure. Specifically, we will use subscripts on the variables to keep track of the signal flow. It will turn out that our modification leads to a formulation somewhat like Armstrong's ENF (Refs. 5, 6, 7).

The modified E-algorithm to be presented is designed to generate test sets under the single-fault assumption. Its nature is such that it can be very easily extended to the multiple-fault assumption and fault location. How these extensions can be done is discussed at the end of this chapter.

Several key words will be used frequently. "Forward" means from input to output, i.e., in the direction of signal flow. "Backward" is just the opposite of forward. The "label" of a lead means the number (or some other identifier) used to name it. A "complemented label" is a label over which a bar has been placed. Example: $x_{1,2}$ means the signal due to x on lead 2 after it has passed over lead 1.

B. Subscripted E-Expressions

An E-expression with subscripts is denoted by E_s. It is formed by starting at the primary input leads and working forward toward the output gate by gate. Subscripts are added in a natural manner. As the signal flow is traced, each time a new lead number is encountered a subscript is added. For

an example, see Fig. 2.2.

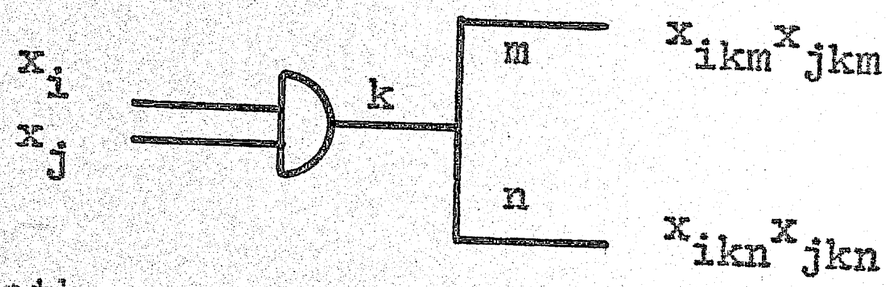


Fig. 2.2
Subscripting
Convention

In other words, the label of a lead at the output of a gate is appended to the right of the subscripts of each literal in the expression for the input leads. Fan-outs are treated as "identity" gates, i.e., if lead k fans out into m and n, then the path through lead m has the subscript k m. Subscripts are complemented in accordance with the following rule:

Rule 2.1

1) Lead i carries a primary input signal:

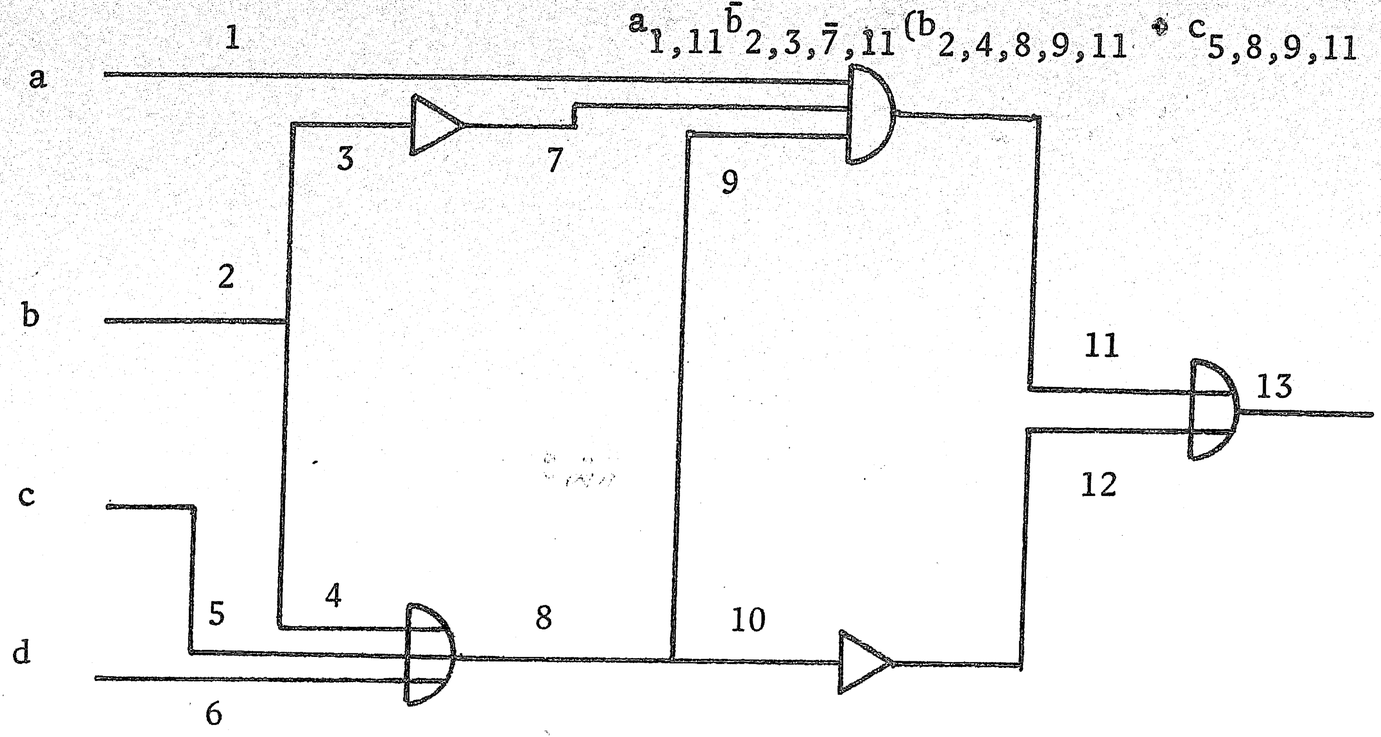
   If the input variable is complemented (uncomplemented), then the subscript of the input variable is $\bar{i}$ (i).

2) Lead i is the output lead (or connected to the output lead) of either an AND or OR gate:

   The subscript corresponding to lead i is of the same type as the right-most subscript of each literal in its Boolean expression.

3) Lead i is the output lead (or connected to the output lead) of a NOT, NAND, or NOR gate:

   The subscript corresponding to lead i will be complemented (uncomplemented) if the right-most subscript of the literal in its Boolean expression is uncomplemented (complemented).



$$E_s = a_{1,11,13}\bar{b}_{2,3,\bar{7},\overline{11},\overline{13}}b_{2,4,8,9,11,13} + a_{1,11,13}\bar{b}_{2,3,\bar{7},\overline{11},\overline{13}}c_{5,8,9,11,13}$$
$$+ a_{1,11,13}\bar{b}_{2,3,\bar{7},\overline{11},\overline{13}}d_{6,8,9,11,13} + \bar{b}_{2,4,8,10,\overline{12},\overline{13}}\bar{c}_{5,8,10,\overline{12},\overline{13}}\bar{d}_{6,8,10,\overline{12},\overline{13}}$$

Figure 2.3

For illustration, consider the example in Fig. 2.3, which shows the development of its E_s expression. Suppose input $\bar{a}\bar{b}cd$ is applied to this network. The normal output response of the network is zero and will be so unless one of the leads on the path 1,11,13 is stuck in one. This is revealed by the second term in E_s, which is $a_{1,11,13}\bar{b}_{2,3,\bar{7},\overline{11},\overline{13}}c_{5,8,9,11,13}$. Conversely, by considering the term $p=a_{1,11,13}\bar{b}_{2,3,\bar{7},\overline{11},\overline{13}}c_{5,8,9,11,13}$ we know from the basic E-algorithm that the vertices in $\bar{a}\bar{b}c$ are a growth test for $a_{1,11,13}$, providing the remaining products in E_s equal zero, which is so for the vertex $\bar{a}\bar{b}cd$. Now consider input $abc\bar{d}$. Again the response should be zero, but because the second literal in the cube $a_{1,11,13}\bar{b}_{2,3,\bar{7},\overline{11},\overline{13}}c_{5,9,11,13}$, the response will be one if leads 2 or 3 are stuck at zero (sa0) or one of the leads 7,11, or 13 is sa1. Thus input $abc\bar{d}$ also represents a growth test. With these examples in mind, it is not hard to see that the following lemmas are valid.

Lemma 2.1:

When testing a <u>complemented</u> letter for growth, e.g., $\bar{a}$, then leads corresponding to barred (unbarred) subscripts of the letter are tested for sa1 (sa0). If the growth test is for an <u>uncomplemented</u> letter x, then leads corresponding to barred (unbarred) subscripts of x are tested for sa0 (sa1).

Lemma 2.2:

For an existence-test, interchange sa1 and sa0 in Lemma 2.1.

The proofs of these two lemmas are quite straightforward and follow directly from the rules for subscripting.

Theorem 2.3:

Under the single-fault assumption, in applying the E-algorithm to a subscripted E-expression, a sufficient test-set can be found without going through more than one pass in the test generation process, i.e., it is not necessary to "process" the terms in the sets $\{T_i\}$, if checking is based on the response of each lead.

Proof:

The proof is based on the observation that testing information provided by processing the sets $\{T_i\}$ is already available from other sources after one passes through the test generation process.

The products in $\{T_i\}$ originate from two sources:

i) From resolving a conflict term
   a) A literal in the product generates a conflict which has been resolved.
   b) A literal that is not involved in a conflict (although it is in a product containing conflicts).

In case i-a, the processing of such literals yields information of interest only in the multiple-faults case, and is redundant under the single-fault assumption. This is a consequence of the way in which conflicts are resolved.

In case i-b, these literals yield no new information because the leads checked by those tests from $\{T_i\}$ must have been previously checked by tests from some other Pi's associated with these leads. The same situation is obtained in case ii.

Q.E.D.

We are now ready to give the complete algorithm for finding single-fault test sets from a subscripted E-expression. A complete example will be given later.

Algorithm 2.1:

1) Label all the leads of the given network. It is convenient to use numbers such that the signal on lead n comes only from leads with lower numbers.

2) Using Rule 2.1 (this chapter) and Rules 1.1 and 1.2 (Chapter I), obtain a subscripted E-expression for the given network. Note that "conflict terms" are allowed to be broken as is implied by Ref. 8. This is recommended in order to simplify the computation.

3) Build a two-dimensional table. The row labels are the tests that will be obtained later on. (Thus the size of this table cannot be fixed beforehand, but is upper-bounded by 2^n where n is the number of independent inputs). The column labels are the fault conditions, so that the number of columns is twice the total number of lead labels.

4) By letting all subproducts of the form $x\bar{x}$ equal zero and using DeMorgan's law, obtain an expression in sum-of-products

form for E_s. Call the resulting expression F_{nr} (normal response).

5) Number the products in F_{nr} from 1 to q, so that F_{nr} may be represented as $\sum_{i=1}^{q} P_i$.

6) Set i equal to 1.

7) Set j equal to 1.

8) If P_i^j is contained in F_{nr}, then go to step 12.

9) (Existence test) We find all those true vertices of F_{nr} that would be dropped if the jth literal in P_i were stuck in zero, i.e., if the jth literal were effectively zero. The cube defined by P_i when the jth literal is complemented is given by P_i^j. Thus the vertices in the intersection of the vertices in F_{nr} and P_i^j, denoted by $\{P_i^j \cap F_{nr}\}$, are the ones that result in a correct (true) output even when the jth literal is stuck in zero. Hence the <u>input</u> vertices that result in incorrect (false) output when the jth literal is stuck are those in the set $S = \{P_i - \{P_i^j \cap F_{nr}\}^j\}$, where the second superscript specifies that in $\{P_i^j \cap F_{nr}\}$ the coordinate corresponding to the jth variable in P_i is to be complemented. Those elements in S which are not yet in the table are made row coordinates, and for each element in S columns are checked in accordance with Lemma 2.2.

10) (Growth test) Find all those vertices which are contained in the set $G = \{P_i^j - \{P_i^j \cap F_{nr}\}\}$ and add all of those vertices which are not yet in the table as row coordinates. Then for each element in G, check the columns that can be tested in accordance with Lemma 2.1.

11) If all the literals in P_i have been tested, then go to step 13. Otherwise, increase j by 1, then go to step 8.

12) If i equals q, i.e., if all the terms in F_{nr} have been processed, then go to step 13. Otherwise, increase index i by 1 and go to step 7.

13) (Conflict resolution) For each term of the original E_s-expression that involves a conflict-pair, resolve each conflict-pair (i.e., $y_k\bar{x}_i x_j$ is resolved to $y_k\bar{x}_i$ and $y_k x_j$). DeMorgan's law may be applied when necessary. Because the single-fault assumption is made, only certain enlargements due to conflict terms need be considered: those that are due to a single signal path. Thus $a_x\bar{b}_{ijk}b_{mno}c_{mnp}\bar{c}_{iqs}$ can lead to created terms

$$a_x\,\bar{b}_{ij}\,\bar{c}_{iqs}$$

$$a_x\,b_{mno}\,c_{mnp}$$

but not to created terms $a\bar{b}c$ or $ab\bar{c}$, because b and $\bar{c}$ have no common path, nor do $\bar{b}$ and c. The possible conflict terms are labelled PC_1 through PC_n.

14) Set j equal to 1.

15) If PC_j is contained in F_{nr}, i.e., if $PC_j \subset F_{nr}$, then go to step 17.

16) If PC_j is not entirely contained in F_{nr}, add the vertices in $\left\{PC_j - (PC_j \cap F_{nr})\right\}$ to the table. Check columns according to the common subscripts on the literals that were set to 1 to resolve the conflict, making use of Lemma 2.1.

17) If j is equal to n, i.e., if all the resolved conflict terms have been processed, then go to step 18. Otherwise, increase

index j by 1, then go to step 15.

18) Find a minimum cover of the resulting table.

We illustrate by means of the previous example.

From Fig. 2.3, we have

$$F_s = a_{1,11,13}\bar{b}_{2,3,\bar{7},\bar{11},\bar{13}}b_{2,4,8,9,11,13} + a_{1,11,13}$$
$$\bar{b}_{2,3,\bar{7},\bar{11},\bar{13}}c_{5,8,9,11,13} + a_{1,11,13}\bar{b}_{2,3,\bar{7},\bar{11},\bar{13}}$$
$$d_{6,8,9,11,13} + \bar{b}_{2,4,8,10,\bar{12},\bar{13}}\bar{c}_{5,8,10,\bar{12},13}$$
$$\bar{d}_{6,8,10,\bar{12},\bar{13}}$$

$$F_{nr} = a_{1,11,13}\bar{b}_{2,3,\bar{7},\bar{11},\bar{13}}c_{5,8,9,11,13} + a_{1,11,13}$$
$$\bar{b}_{2,3,\bar{7},\bar{11},\bar{13}}d_{6,8,9,11,13} + \bar{b}_{2,4,8,10,\bar{12},\bar{13}}$$
$$\bar{c}_{5,8,10,\bar{12},\bar{13}}\bar{d}_{6,8,10,\bar{12},\bar{13}}$$

Now we set up the columns in the table of Fig. 2.4.

$$P_1 = a_{1,11,13}\bar{b}_{2,3,\bar{7},\bar{11},\bar{13}}c_{5,8,9,11,13}$$

$$P_2 = a_{1,11,13}\bar{b}_{2,3,\bar{7},\bar{11},\bar{13}}d_{6,8,9,11,13}$$

$$P_3 = \bar{b}_{2,4,8,10,\bar{12},\bar{13}}\bar{c}_{5,8,10,\bar{12},\bar{13}}\bar{d}_{6,8,10,\bar{12},\bar{13}}$$

P_1^1: $\bar{a}\bar{b}c \subset F_{nr}$? No.

Existence tests obtained are $a\bar{b}cd$, $a\bar{b}c\bar{d}$ (i.e., from $\{P_1 - (P_1^1 \cap F_{nr})^1\} = \{P_1 - \emptyset\}$). Check columns 1, 11, 13 for sa0 (by Lemma 2.2).

Growth tests obtained are $\bar{a}\bar{b}cd$, $\bar{a}\bar{b}c\bar{d}$ (i.e., from $\{P_1^1 - (P_1^1 \cap F_{nr})\}$). Check columns 1, 11, 13 for sa1 (by

TESTS	1		2		3		4		5		6		7		8		9		10		11		12		13	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
$a\bar{b}cd$		✓	✓		✓								✓								✓				✓	
$a\bar{b}c\bar{d}$		✓	✓		✓								✓								✓				✓	
$\bar{a}\bar{b}cd$	✓																				✓				✓	
$\bar{a}\bar{b}c\bar{d}$	✓								✓						✓				✓		✓		✓		✓	
$abc\bar{d}$				✓		✓							✓								✓				✓	
$abcd$				✓		✓							✓								✓				✓	
$a\bar{b}\bar{c}d$		✓	✓		✓									✓								✓				✓
$\bar{a}\bar{b}\bar{c}d$	✓											✓			✓				✓		✓		✓		✓	
$ab\bar{c}d$				✓		✓							✓								✓				✓	
$a\bar{b}\bar{c}\bar{d}$			✓				✓							✓									✓			✓
$\bar{a}\bar{b}\bar{c}\bar{d}$			✓				✓		✓		✓			✓			✓						✓			✓
$\bar{a}bc\bar{d}$				✓				✓								✓								✓		✓
$ab\bar{c}\bar{d}$				✓		✓		✓						✓		✓				✓				✓		✓

Figure 2.4

Lemma 2.1).

P_1^2: $abc \subset F_{nr}$? No.

Existence tests obtained are $a\bar{b}cd$, $a\bar{b}c\bar{d}$. Both vertices are already listed. Check columns 2, 3 for sa1, columns 7, 11, 13 for sa0.

Growth tests obtained are $abc\bar{d}$, $abcd$. Add them to the table, and check columns 2, 3 for sa0, columns 7, 11, 13 for sa1.

P_1^3: $a\bar{b}\bar{c} \subset F_{nr}$? Yes.

Increase index i by 1, so that we consider P_2.

P_2^1: $\bar{a}\bar{b}d \subset F_{nr}$? No.

Existence tests obtained are $a\bar{b}cd$, $a\bar{b}\bar{c}d$. Add $a\bar{b}\bar{c}d$ to the table ($a\bar{b}cd$ is already in the table). In rows $a\bar{b}cd$ and $a\bar{b}\bar{c}d$ place check marks in columns 1, 11, 13 under sa0.

Growth tests obtained are $\bar{a}\bar{b}\bar{c}d$ and $\bar{a}\bar{b}cd$. Add $\bar{a}\bar{b}\bar{c}d$ to the table. Then check in rows $\bar{a}\bar{b}\bar{c}d$ and $\bar{a}\bar{b}cd$ columns 1, 11, 13 under sa1.

P_2^2: $abd \subset F_{nr}$? No.

Existence tests obtained are $a\bar{b}cd$, $a\bar{b}\bar{c}d$. These are already in the table. Check their columns 2, 3 for sa1, columns 7, 11, 13 for sa0.

Growth tests obtained are $ab\bar{c}d$, $abcd$. Add $ab\bar{c}d$ to the table. Then check in rows $ab\bar{c}d$, $abcd$ columns 2,3 for sa0, columns 7, 11, 13 for sa1.

P_2^3: $a\bar{b}\bar{d} \subset F_{nr}$? Yes.

Increase i by 1, so that we consider P_3.

P_3^1: $b\bar{c}\bar{d} \subset F_{nr}$? No.

Existence tests obtained are $a\bar{b}\bar{c}\bar{d}$, $\bar{a}\bar{b}\bar{c}\bar{d}$. So add them to the table and check columns 2, 4, 8, 10 for sa1, columns 12, 13 for sa0.

Growth tests obtained are $\bar{a}b\bar{c}\bar{d}$, $ab\bar{c}\bar{d}$. Add them to the table. Check columns 2, 4, 8, 10 for sa0, columns 12, 13 for sa1.

P_3^2: $\bar{b}c\bar{d} \subset F_{nr}$? No.

Existence test obtained is $\bar{a}\bar{b}c\bar{d}$. In row $\bar{a}\bar{b}c\bar{d}$, check columns 5, 8, 10 for sa1, columns 12, 13 for sa0.

Growth test obtained is $\bar{a}\bar{b}cd$. In row $\bar{a}\bar{b}c\bar{d}$, check columns 5, 8, 10 for sa0, columns 12, 13 for sa1.

P_3^3: $\bar{b}\bar{c}d \subset F_{nr}$? No.

Existence test obtained is $\bar{a}\bar{b}\bar{c}\bar{d}$, which checks columns 6, 8, 10 for sa1, columns 12, 13 for sa0.

Growth test obtained is $\bar{a}\bar{b}\bar{c}d$, which checks columns 6, 8, 10 for sa0, columns 12, 13 for sa1.

Now all terms in F_{nr} have been processed. We turn to the conflict terms. (step 13).

Conflict terms:

$a_{1,11,13}\bar{b}_{2,3,\bar{7},\bar{11},\bar{13}} b_{2,4,8,9,11,13}$

$PC_1 = a_{1,11,13}\bar{b}_{2,3,\bar{7},\bar{11},\bar{13}}$

$PC_2 = a_{1,11,13} b_{2,4,8,9,11,13}$

$PC_1 < F_{nr}$? Yes.

$PC_2 < F_{nr}$? No.

Grpwth tests obtained are $ab\bar{c}\bar{d}$, $ab\bar{c}d$, $abc\bar{d}$, $abcd$. All of these are possible if $\bar{b}_{2,3,\bar{7},\bar{11},\bar{13}} = 1$. Therefore, columns 2, 3 are checked for sa0, columns 7, 11, 13 for sa1.

A minimum cover for the table can be obtained by selecting $\bar{a}\bar{b}cd$, $\bar{a}\bar{b}\bar{c}\bar{d}$, $ab\bar{c}\bar{d}$, $\bar{a}\bar{b}\bar{c}\bar{d}$ and ($a\bar{b}cd$, or $a\bar{b}c\bar{d}$, or $a\bar{b}\bar{c}d$).

C. Discussion

Several points can be made about the above algorithms:

1) The test table built in the algorithm may be divided into two parts, one listing all growth tests and the other, all existence tests. It is then easier to check for duplication of tests, since only one part of the table need be scanned, and so human effort (or computer time) is saved. Furthermore, one can sometimes simplify the table as he goes along. If a lead has been tested for sa1 (sa0) by a test of one type (e.g., growth test), then it will probably have to be tested for sa0 (sa1) by the other type of test. This is true for most of the networks we encountered. But counterexamples do exist, e.g., EX-OR trees. Thus caution will have to be exercised.

2) Instead of adding tests to the table immediately after they are found, it is more efficient to wait until <u>all</u> tests obtained from a given P_i have been found. Then the table need be scanned only once for each P_i.

3) To minimize the work in keeping track of subscripts in resolving conflicts, it is recommended that in forming the E_s-expression, conflicts be expanded whenever possible. With the aid of a computer in finding the E-expression, the work of

expanding a Boolean expression is not as costly as one might expect intuitively. (Note the difference between "mind-cycle" and "machine-cycle").

4) The amount of subscripting handled can be reduced in building up the E_s-expression by initially using the same label for all leads on a given gate. After the complete E_s-expression has been formed, it is easy to transform to complete labels, using distinct numbers for each lead.

It is also quite obvious that the proposed algorithm can be easily extended to an algorithm that generates a multiple-fault detection test-set, in which case the set $\{ T_i \}$ cannot be discarded. Set $\{ T_i \}$ can be obtained from steps 8 and 15 in Algorithm 2.1. Then we follow the original E-algorithm, and a set of tests will then be obtained from processing $\{ T_i \}$. Add any additional tests to those obtained from Algorithm 2.1. The number of new tests that will be added is not big (relatively), because we have followed the strategy that conflicts are expanded when possible, and highly redundant networks are only rarely encountered.

It is believed that subscripting an E-expression will be particularly helpful in error-location problems. This matter will be further investigated at a later time.

III. Finding Tests for Decomposed Networks

A. Motivation for Decomposing Large Networks.

From Chapter I, we recall that the task of generating a test-set for a given combinational network by the E-algorithm consists of three main parts: 1. generation of the E-expression, 2. finding all possible tests, and 3. selecting a sufficient (possibly minimum) test-set from the results of step 2. Unfortunately, we suspect that the relationship between the amount of work required to obtain a set of tests and the size of a network (i.e., the number of gates and levels) is worse than linear.

We fear that above some network size, execution of the E-algorithm becomes excessively costly. While we do not yet have the experience to suggest a bound at which difficulties arise, we believe that much can be gained in efficiency by decomposing large networks of practical significance. Moreover, it is true that we are not usually required to find a test-set that detects <u>every</u> possible combination of multiple-faults in a very big network. Rather, we suspect that it will often suffice to find a test-set that detects every single-fault and detects only those multiple-faults where the faulty leads happen to be near each other.

These ideas led us to try to modify the E-algorithm and make it more practical for very big networks.

B. Procedure for Finding Tests in Decomposed Networks.

The overall process consists of three main parts:

1) Partitioning the given network into disjoint subnetworks.

2) Generating test-sets for each subnetwork.

3) Combining the test-sets found in step 2 to obtain a complete test-set that detects any one faulty subnetwork.

We emphasize that the procedure to be given below is certain to detect only a single faulty subnetwork; if more than one subnetwork is faulty, the tests do not necessarily work.

1) Partitioning a Given Network:

In addition to considering the physical structure and gate placements, the following nine points seem useful as guidelines in partitioning a network:

i) Group those gates that are likely to fail simultaneously together into one subnetwork. E.g., if one likely physical fault may influence more than one lead, place all leads involved into one subnetwork.

ii) Leads that are accessible from the outside world should be used as the input and/or output leads of subnetworks. In other words, "cut" the network at places where it is accessible.

iii) Make the number of interconnections between subnetworks small.

iv) Try to reduce the number of subnetworks at which a given primary input appears. This will make subnetworks as independent from each other as possible.

v) Keep in mind that it is not necessary to cut all the branches at the same level of the overall network.

vi) Try to cut a network before an "N-gate" (NAND, NOR, NOT) instead of after it. It is also wise where possible to make cuts after an <u>even</u> number of negations in the signal path. These suggestions are intended to reduce the number of complementations required in generating an E-expression.

vii) Divide the network into nearly equal parts, i.e., try to cut the network in such a way that each individual subnetwork requires about the same number of tests. This will probably reduce the total number of tests.

viii) Try to cut at a place following a reconvergence point. This will reveal the redundancy created by the reconvergence directly in the E-expression of the subnetwork and will decrease the number of interconnections between subnetworks.

ix) Try to partition into identical subnetworks. This will reduce the test-generation efforts by allowing repeated use of subnetwork tests.

Of course, one should also consider the network size for which the E-algorithm has its highest performance capability (relatively). At this time, however, we do not know yet what this optimum size is. We expect that the way in which a network is partitioned will have great influence over the time spent in combining the test-sets from each subnetwork (step 3), but this will remain the subject of future study.

After a network is partitioned, we may treat each subnetwork as a block or supergate, and arrange these blocks in level-organized form. In labelling leads, we shall use the lowest numbers for leads entering level 1, next we number those entering level 2, etc. The blocks, however, are numbered in just the opposite way: those nearest the network output are numbered first. These conventions are illustrated in Figs. 3.1 and 3.2.

2) Test Generation for Each Subnetwork (Block):

To find a test-set for each block one must modify the original E-algorithm only very little. However, four new terms need to be introduced. The first two are CL and PCL, which have the purpose of providing information for "merging" tests between blocks. The second two are the PTI and OLAI, which will be defined later on in the explanation of the merging process.

Definition: Connectivity List, CL.

A connectivity list is a set of ordered pairs

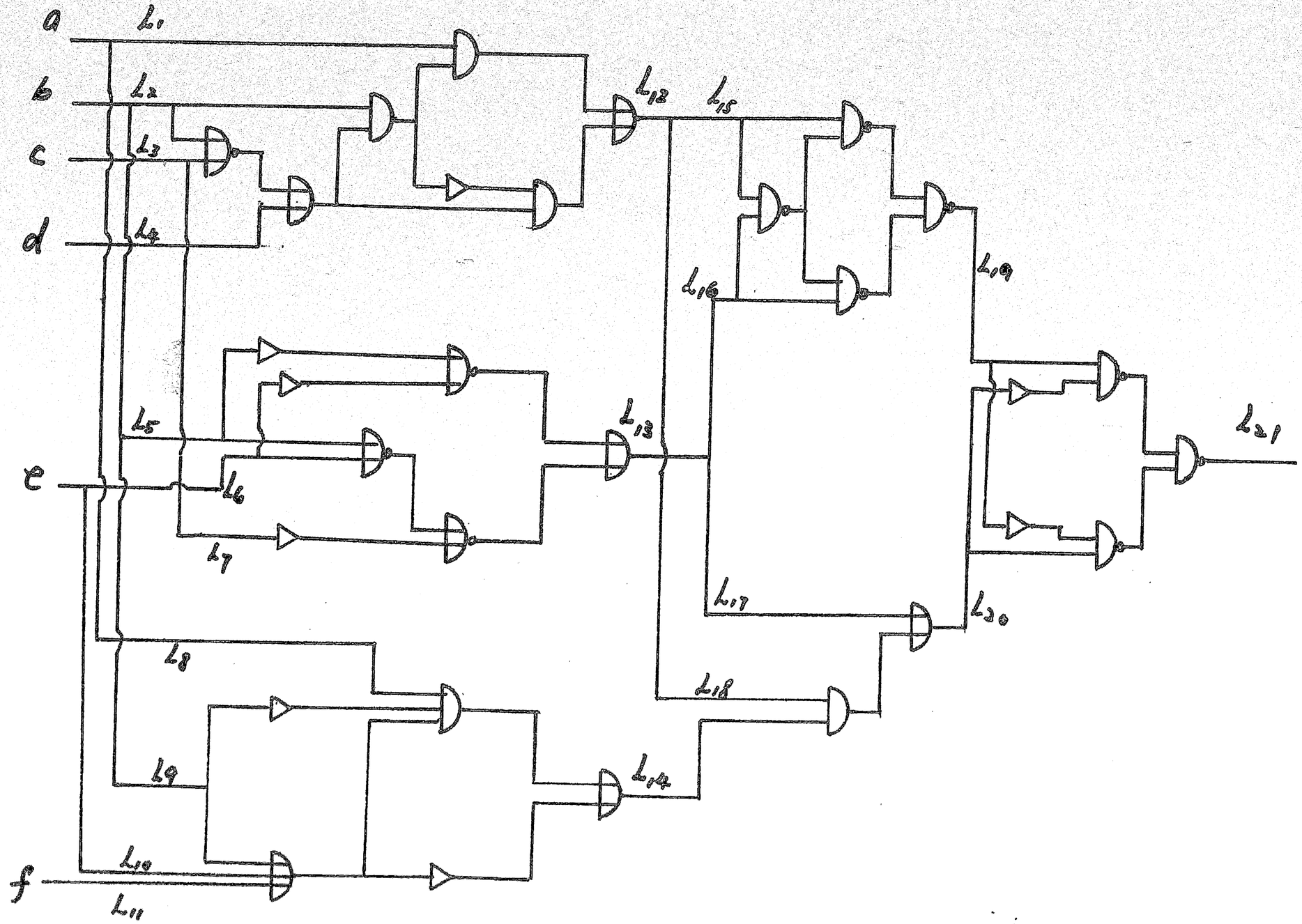

a
b
c
d
e
f
L1
L2
L3
L4
L5
L6
L7
L8
L9
L10
L11
L12
L13
L14
L15
L16
L17
L18
L19
L20
L21

Figure 3.1

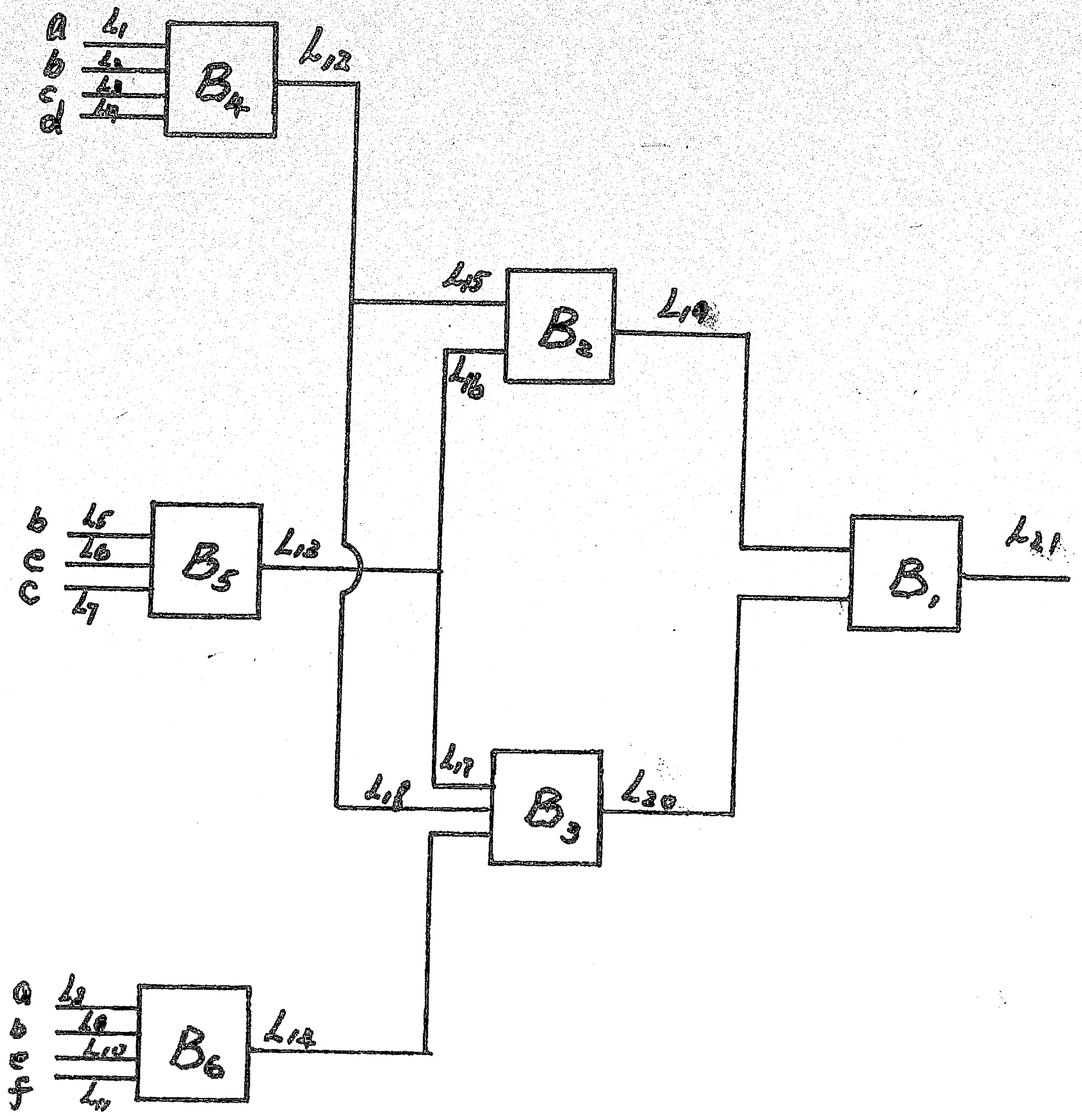

a
b
c
d
L_1
L_2
L_3
L_4
B_4
L_{12}
L_{15}
B_2
L_{19}
L_{16}
b
c
c
L_5
L_6
L_7
B_5
L_{13}
B_1
L_{21}
L_{17}
L_{18}
B_3
L_{20}
a
b
c
f
L_8
L_9
L_{10}
L_{11}
B_6
L_{14}

Figure 3.2

$\{(\{I\}, O)\}$ associated with each input vertex to a block, where the cardinality of the set is equal to the number of block outputs. Here $\{I\}$ is a subset of the block input leads element. The second element in each pair, O, stands for a block output lead number and is restricted to a single element.

The ordered pair $\{(\{I\}, O)\}$ shows the conditions under which, when one of the leads specified in the I component changes its value from that specified by the associated input vertex, the output lead in the O component also changes value.

We shall use superscripts 0 and 1 on the lead literals. By i^0 (i^1) we mean that the normal (fault-free) value of lead i is 1(0).

Definition: Path Connectivity List, PCL

A <u>path connectivity list</u> has the same format as connectivity list, but it is not necessary for the I component and O component to involve leads on the same block. The PCL is obtained by <u>chaining</u> several CL's together. A CL $\{(\{I_1\}, O_1)\}$ can be chained to another CL $\{(\{I_2\}, O_2)\}$ iff O_1 is contained in $\{I_2\}$. The resulting PCL is obtained by replacing O_2 in $\{I_2\}$ by $\{I_1\}$.

Example:

Refer to block B4 in Figs. 3.1, 3.2. For this block, The E-expression is $x_{12} = x_1x_2x_4 + \bar{x}_2\bar{x}_3 + \bar{x}_2\bar{x}_4 + x_1x_2\bar{x}_3 + x_2\bar{x}_3\bar{x}_4 + \bar{x}_2x_3\bar{x}_4 + x_2x_4 + x_3x_4$. Consider the input vertex

$x_1 x_2 \bar{x}_3 \bar{x}_4$. Its associated CL is $\{(2^0, 4^1; 12^1)\}$. This means that whenever lead L2 changes from 1 to 0 or lead L4 changes from 0 to 1, the output lead changes its value from 0 to 1.

Now consider block B2, which has the E-expression $x_{19} = X_{15} + x_{15}\bar{x}_{16} + \bar{x}_{15}x_{16} + X_{16}$. One of its possible input vertices is $\bar{x}_{15}x_{16} = \bar{x}_{12}x_{16}$, which has the CL $\{(12^1, 16^0; 19^0)\}$. We can chain the above two CL's together and obtain $x_1 x_2 \bar{x}_3 \bar{x}_4 x_{16}$, which has the associated PCL $\{(2^0, 4^1, 16^0; 19^0)\}$.

One of the differences in dealing with partitioned networks is the need for associating a connectivity list with each test. Another difference is that we do not perform the test minimization step for each block.

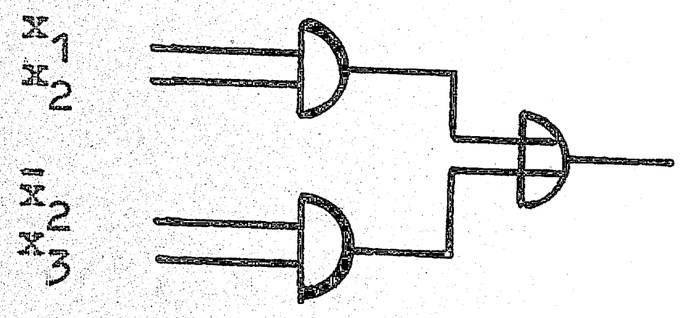
It is not difficult to compute the CL associated with a vertex. First, we consider growth tests. When we perform the intersection of the true vertices in P_i^j with the false vertices of F, if $\{P_i^j \cap \bar{F}\}$ is not empty, we can obtain some growth test(s). Growth tests, it will be recalled from Chapter II, are the vertices in $G = \{P_i^j - \{P_i^j \cap F\}\}$. All the elements in G are sensitive to the jth literal of P_i, and so we associate with the I component of the CL of each test in G the lead number corresponding to the literal that has been complemented, i.e., the lead number of literal j in P_i. The lead number added is superscripted with 1 or 0 according as the original literal is uncomplemented or complemented.

With regard to existence tests, we can compute the sensitivity of the jth literal in P_i by finding the set

$\{P_i - \{P_i^j \cap F\}^j\}$, which contains all the vertices in P_i that are sensitive to the jth literal. With each of these we write in its CL the lead number corresponding to the jth variable. We use superscript 0 if the jth variable is not complemented. Otherwise, we use the superscript 1.

The 0 component of the CL is superscripted with a 1 (0) if it is a growth (existence) test.

Example:



$$F = x_1x_2 + \bar{x}_2x_3$$

Existence Test

$$\{P_1 - \{P_1^2 \cap F\}^2\} = \{x_1x_2\} - \{x_1x_2x_3\} = \{x_1x_2\bar{x}_3\}$$

T: $x_1x_2\bar{x}_3$ $(2^0;6^0)$

Growth Test

$P_1 = x_1x_2$

$P_1^2 = x_1\bar{x}_2$

$$\{P_1^2 \cap F\} = \{x_1\bar{x}_2x_3\}$$

$$G = \{P_1^2 - \{\{P_1^2 \cap F\}\}\} = \{x_1\bar{x}_2\bar{x}_3\}$$

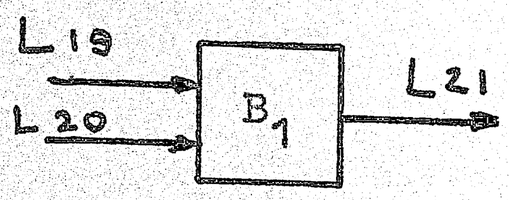
L: $x_1\bar{x}_2\bar{x}_3$ $(2^0;6^0)$

After the entries for the GT/ET tables of each block are computed, we arrange the rows of each table according to the following procedure:

The rows of the GT table are arranged according to the number of check marks in decreasing order.

In the ET table, we put at the top the set of required tests (i.e., tests that are necessary to detect every possible drop) and place vertices with empty CL's at the bottom. The remaining test vertices are arranged in increasing order of check marks between the required tests and those vertices with empty CL's.

We are now ready to handle the example in Figs. 3.1 and 3.2. The test-sets for each block are shown in Fig. 3.3.



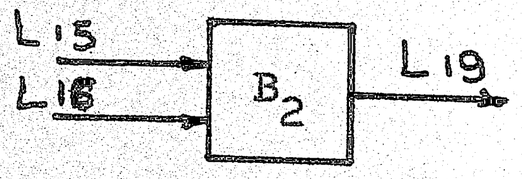
$$x_{21} = \overline{\overline{(x_{19}\bar{x}_{20})}\,\overline{(\bar{x}_{19}x_{20})}}$$
$$= x_{19}\bar{x}_{20} + \bar{x}_{19}x_{20}$$

GT:

	$x_{19}x_{20}$	$\bar{x}_{19}\bar{x}_{20}$	
$x_{19}x_{20}$	✓		$(19^0, 20^0; 21^1)$
$\bar{x}_{19}\bar{x}_{20}$		✓	$(19^1, 20^1; 21^1)$

ET:

	$x_{19}\bar{x}_{20}$	$\bar{x}_{19}x_{20}$	
$x_{19}\bar{x}_{20}$	✓		$(19^0, 20^1; 21^0)$
$\bar{x}_{19}x_{20}$		✓	$(19^1, 20^0; 21^0)$



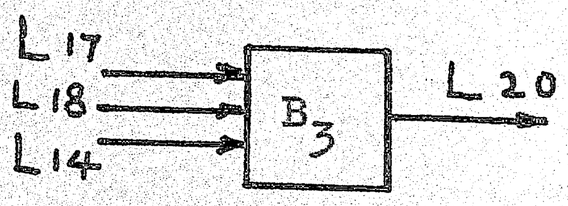
$$x_{19} = \overline{\overline{\bar{x}_{15}\bar{x}_{16}}\,x_{15}}\;\overline{\overline{\bar{x}_{15}\bar{x}_{16}}\,x_{16}}$$
$$= \overline{\bar{x}_{15}\bar{x}_{16}}\,x_{15} + \overline{\bar{x}_{15}\bar{x}_{16}}\,x_{16}$$
$$= (\bar{x}_{15} + \bar{x}_{16})x_{15} + (\bar{x}_{15} + \bar{x}_{16})x_{16}$$
$$= x_{15} + x_{15}\bar{x}_{16} + \bar{x}_{15}x_{16} + x_{16}$$

GT:

	$x_{15}x_{16}$	$\bar{x}_{15}\bar{x}_{16}$	x_{15}	$\bar{x}_{15}$	x_{16}	$\bar{x}_{16}$	
$x_{15}x_{16}$	✓		✓		✓		$(15^0, 16^0; 19^1)$
$\bar{x}_{15}\bar{x}_{16}$		✓		✓		✓	$(15^1, 16^1; 19^1)$

ET:

	$x_{15}\bar{x}_{16}$	$\bar{x}_{15}x_{16}$	
$x_{15}\bar{x}_{16}$	✓	✓	$(15^0, 16^1; 19^0)$
$\bar{x}_{15}x_{16}$		✓	$(15^1, 16^0; 19^0)$



$$x_{20} = x_{17} + x_{14}x_{18}$$

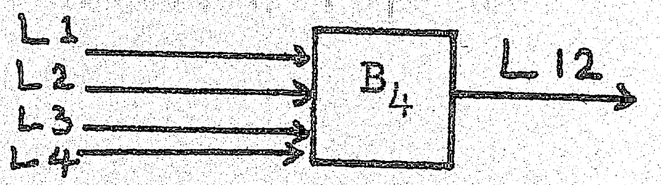
Figure 3.3

GT:

	$\bar{x}_{17}$	$\bar{x}_{14}x_{18}$	$x_{14}\bar{x}_{18}$	
$x_{14}\bar{x}_{17}\bar{x}_{18}$	✓	✓		$(17^1, 18^1; 20^1)$
$\bar{x}_{14}\bar{x}_{17}x_{18}$	✓		✓	$(14^1, 17^1; 20^1)$
$\bar{x}_{14}\bar{x}_{17}\bar{x}_{18}$	✓			$(17^1; 20^1)$

ET:

	x_{17}	$x_{14}x_{18}$	
$x_{14}\bar{x}_{17}x_{18}$		✓	$(14^0, 18^0; 20^0)$
$\bar{x}_{14}x_{17}x_{18}$	✓		$(17^0; 20^0)$
$\bar{x}_{14}x_{17}\bar{x}_{18}$	✓		$(17^0; 20^0)$
$x_{14}x_{17}\bar{x}_{18}$	✓		$(17^0; 20^0)$
$x_{14}x_{17}x_{18}$	✓	✓	



$$x_{12} = x_1x_2x_4 + \bar{x}_2\bar{x}_3 + \bar{x}_2x_4 + x_1x_2\bar{x}_3 + x_2\bar{x}_3\bar{x}_4 + \bar{x}_2x_3\bar{x}_4 + x_2x_4 + x_3x_4$$

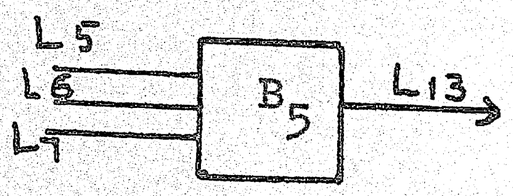
ET:

	$x_1x_2x_4$	$\bar{x}_2\bar{x}_3$	$\bar{x}_2\bar{x}_3\bar{x}_4$	$\bar{x}_2x_4$	x_1x_4	$x_1\bar{x}_2\bar{x}_3$	
$\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$		✓	✓				$(2^1, 3^1; 12^0)$
$\bar{x}_1\bar{x}_2x_3x_4$				✓			$(2^1, 4^0; 12^0)$
$x_1\bar{x}_2\bar{x}_3x_4$	✓				✓		$(1^0, 4^0; 12^0)$
$\bar{x}_1\bar{x}_2x_3x_4$		✓		✓			$(2^1; 12^0)$
$x_1x_2x_3x_4$	✓				✓		$(1^0, 4^0; 12^0)$
$x_1\bar{x}_2\bar{x}_3\bar{x}_4$		✓	✓			✓	$(2^1, 3^1; 12^0)$
$x_1\bar{x}_2x_3x_4$				✓	✓		$(4^0; 12^0)$
$x_1\bar{x}_2\bar{x}_3x_4$		✓		✓	✓	✓	

Figure 3.3 cont.

GT:	$\bar{x}_8\bar{x}_9x_{10}$	$x_8x_9x_{10}$	$\bar{x}_8\bar{x}_9x_{11}$	$x_8x_9x_{11}$	$x_9\bar{x}_{10}\bar{x}_{11}$	$\bar{x}_9x_{10}\bar{x}_{11}$	$\bar{x}_9\bar{x}_{10}x_{11}$	x_8x_9	$\bar{x}_8\bar{x}_9$	
$\bar{x}_8\bar{x}_9x_{10}\bar{x}_{11}$	✓					✓			✓	$(8^1,10^0;14^1)$
$\bar{x}_8\bar{x}_9\bar{x}_{10}x_{11}$			✓				✓		✓	$(8^1,11^0;14^1)$
$\bar{x}_8\bar{x}_9x_{10}x_{11}$	✓		✓						✓	$(8^1;14^1)$
$x_8x_9x_{10}x_{11}$		✓		✓				✓		$(9^0;14^1)$
$x_8x_9\bar{x}_{10}\bar{x}_{11}$					✓			✓		$(9^0;14^1)$
$x_8x_9x_{10}\bar{x}_{11}$		✓						✓		$(9^0;14^1)$
$x_8x_9\bar{x}_{10}x_{11}$				✓				✓		$(9^0;14^1)$
$\bar{x}_8x_9\bar{x}_{10}\bar{x}_{11}$					✓					$(9^0;14^1)$

Fig. 3,3 cont.



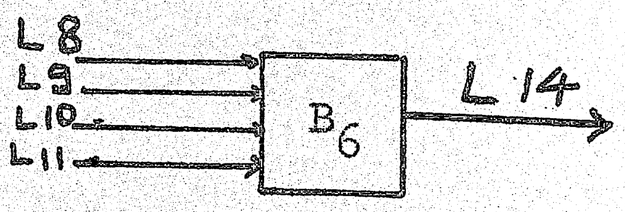
$$x_{13} = \overline{\bar{x}_7 + \overline{x_5 + x_6}} + \overline{\bar{x}_5 + \bar{x}_6}$$
$$= x_7(x_5 + x_6) + x_5x_6$$
$$= x_5x_6 + x_6x_7 + x_7x_5$$

GT:

	$\bar{x}_5x_6$	$x_5\bar{x}_6$	$\bar{x}_6x_7$	$x_6\bar{x}_7$	$\bar{x}_7x_5$	$x_7\bar{x}_5$	
$\bar{x}_5x_6x_7$	✓			✓			$(5^1, 7^1; 13^1)$
$x_5\bar{x}_6x_7$		✓			✓		$(6^1, 7^1; 13^1)$
$x_5x_6\bar{x}_7$			✓			✓	$(6^1, 5^1; 13^1)$

ET:

	x_5x_6	x_6x_7	x_7x_5	
$\bar{x}_5x_6x_7$		✓		$(6^0, 7^0; 13^0)$
$x_5x_6\bar{x}_7$	✓			$(5^0, 6^0; 13^0)$
$x_5\bar{x}_6x_7$			✓	$(5^0, 7^0; 13^0)$
$x_5x_6x_7$	✓	✓	✓	



$$x_{14} = x_8\bar{x}_9x_{10} + x_8\bar{x}_9x_{11} + \bar{x}_9\bar{x}_{10}\bar{x}_{11} + x_8x_9$$

ET:

	$x_8\bar{x}_9x_{10}$	$x_8\bar{x}_9x_{11}$	$\bar{x}_9\bar{x}_{10}\bar{x}_{11}$	$x_8\bar{x}_9$	
$\bar{x}_8\bar{x}_9\bar{x}_{10}\bar{x}_{11}$			✓		$(9^1, 10^1, 11^1; 14^0)$
$x_8\bar{x}_9x_{10}x_{11}$		✓		✓	$(8^0, 9^1; 14^0)$
$x_8\bar{x}_9x_{10}\bar{x}_{11}$	✓			✓	$(8^0, 9^1; 14^0)$
$\bar{x}_8\bar{x}_9x_{10}\bar{x}_{11}$			✓	✓	$(9^1; 14^0)$
$x_8\bar{x}_9x_{10}x_{11}$	✓	✓		✓	$(8^0, 9^1; 14^0)$

Figure 3.3 cont

GT:	$\bar{x}_1x_2x_4$	$x_1x_2\bar{x}_4$	$x_2\bar{x}_3$	$\bar{x}_2x_3$	x_2x_4	$\bar{x}_2\bar{x}_4$	$x_1x_2\bar{x}_3$	$x_2\bar{x}_3\bar{x}_4$	$x_2\bar{x}_4$	$x_3\bar{x}_4$	x_3x_4	$\bar{x}_1x_4$	$x_1\bar{x}_4$	$x_1\bar{x}_2x_3$	$\bar{x}_2x_3\bar{x}_4$	
$x_1x_2x_3x_4$		✓	✓				✓	✓	✓				✓			$(2^0,4^1;12^1)$
$x_1\bar{x}_2x_3\bar{x}_4$				✓		✓				✓			✓	✓	✓	$(3^0,4^1;12^1)$
$\bar{x}_1x_2x_3x_4$	✓				✓						✓	✓				$(1^1,2^0;12^1)$
$\bar{x}_1x_2\bar{x}_3x_4$	✓		✓		✓							✓				$(1^1,2^0;12^1)$
$x_1x_2x_3\bar{x}_4$		✓							✓	✓			✓			$(4^1 \quad ;12^1)$
$\bar{x}_1\bar{x}_2x_3\bar{x}_4$				✓		✓				✓					✓	$(3^0,4^1;12^1)$
$\bar{x}_1x_2\bar{x}_3\bar{x}_4$		✓						✓	✓							$(2^0 \quad ;12^1)$
$\bar{x}_1x_2x_3\bar{x}_4$									✓	✓						

Fig. 3.3 cont.

3) Combining Test-Sets:

Before going into the details of each individual step, we will describe the process briefly with the aid of Fig. 3.4.

The overall strategy is this: We combine the tests for the various blocks by starting from the output level and working backward toward the first level of gates in the complete network.

First, we select a sufficient test-set for the first block (Block 1) which becomes test-set T_1. Then we merge T_1 with elements in the G/E tables of the adjacent block of the same level. (In the running example, we have only one block in the output level, so we skip this step). After all of the blocks of the same level have been merged, we check to determine if the resultant test-set is sufficient for all the blocks that have been handled so far. If so, we start merging the present test-set with tests for blocks in the previous level. If not, we try to "catch up" by adding enough tests to obtain sufficiency. Thereupon, we begin to consider blocks in the previous level.

Because we do not intend to exhaust all possible choices in the merging process (in order to reduce unnecessary work), the normal merging process considers only entries in the G/E tables (i.e., tests) for the next block, instead of merging with all possible inputs to the next block. Consequently, we may fail to find all the tests which are needed. This is why we have to perform the so-called catch-up process.

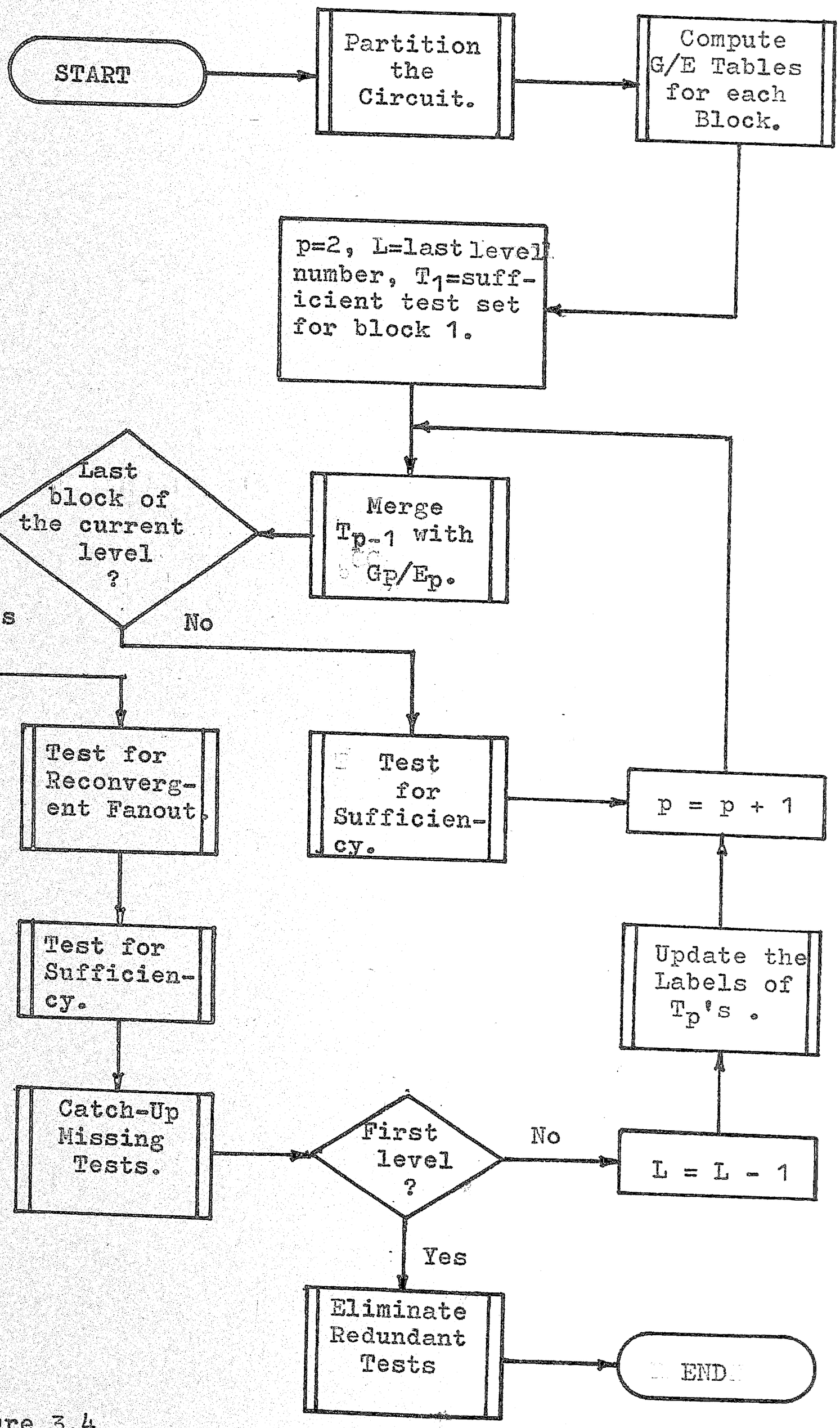

START
Partition the Circuit.
Compute G/E Tables for each Block.
p=2, L=last level number, T_1=sufficient test set for block 1.
Last block of the current level ?
Yes
No
Merge T_{p-1} with G_p/E_p.
Test for Reconvergent Fanout.
Test for Sufficiency.
p = p + 1
Update the Labels of T_p's .
Test for Sufficiency.
Catch-Up Missing Tests.
First level ?
No
L = L - 1
Yes
Eliminate Redundant Tests
END

Figure 3.4

Because of the way we perform the merging process, it is more efficient to find out immediately upon completion of each block which tests are still needed but have not yet been provided, rather than wait until the end of the merging process. This will become clearer after we discuss the merging process in detail.

C. Merging Process.

In general, there are three possible approaches that one could follow in the merging process: 1. Merger with all possible inputs to the next block, an exhaustive procedure. 2. Merger with every element in the G/E tables of the next block. 3. Merger with a cover of the G table and the required tests for the E table.

We believe that approach 1 should be avoided because 2 and 3 make tests more powerful, since they will check more than one block. If the partitioned network is relatively big, we recommend approach 3. Otherwise, we recommend approach 2. At this time we do not have sufficient information for giving a precise criterion for choosing between 2 and 3.

In the process of merging, we must make sure that the specific test t_i for block i is such that there exists a sensitive path between block i and the network output. The information about the sensitive path is provided by the PCL. An element in T_i (the working test-set after merging with block i) is a test for block i if it is derived from an element in T_{i-1} which has in its PCL the output lead number of block i;

otherwise, it is not a test for block i (but it may be a test for other blocks).

We give the merging algorithm as a flowchart in Fig. 3.5 and explain each step in a rather informal way. In Fig. 3.5, T_{p-1} stands for the working test-set before merging with block p. The ith element of the set T_{p-1} is denoted by $T_{p-1}(i)$, $i = 1,2,\ldots,N_{p-1}$. G_p and E_p stand for the growth table and existence table, respectively, of block p. $G_p(j)$ ($E_p(k)$) stands for the jth (kth) element of the G table (E table) of block p.

Before explaining the merging algorithm, we must define two new terms, PTI and OLAI.

Definition: Parents Test Indicator, PTI.

A PTI is an ordered pair associated with each element in T_p (the working test-set in the merging process). The first component indicates which element in T_{p-1} was used in the construction of the current element in T_p. If the element is also a test for block p, the second component of PTI indicates which test it is, i.e., which element in G_p/E_p. If the current element in T_p is not a test for block p, the second component is empty, denoted by $\emptyset$.

The current element in T_p is a test for block p if and only if the output lead number of block p appears in the I component of the PCL of the element of T_{p-1} that it was merged with.

The PTI of an element of T_p will be used to determine

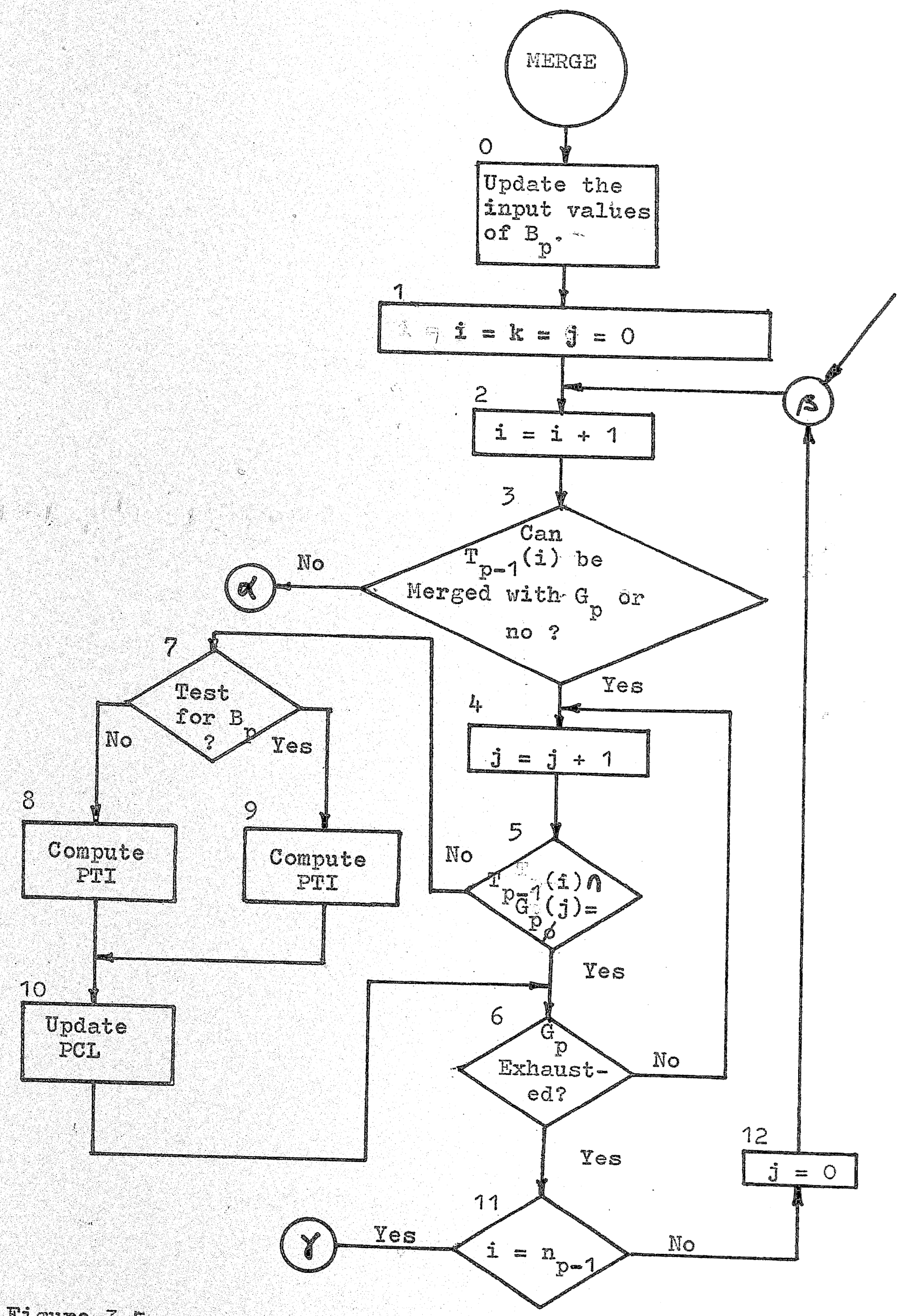

MERGE
0
Update the input values of B_p.
1
i = k = j = 0
2
i = i + 1
3
Can $T_{p-1}(i)$ be Merged with G_p or no ?
No
α
Yes
4
j = j + 1
5
No
$T_{p-1}(i) \cap G_p(j) = \phi$
Yes
6
G_p Exhaust-ed?
No
Yes
7
Test for B_p ?
No
Yes
8
Compute PTI
9
Compute PTI
10
Update PCL
11
i = n_{p-1}
Yes
No
β
γ
12
j = 0

Figure 3.5

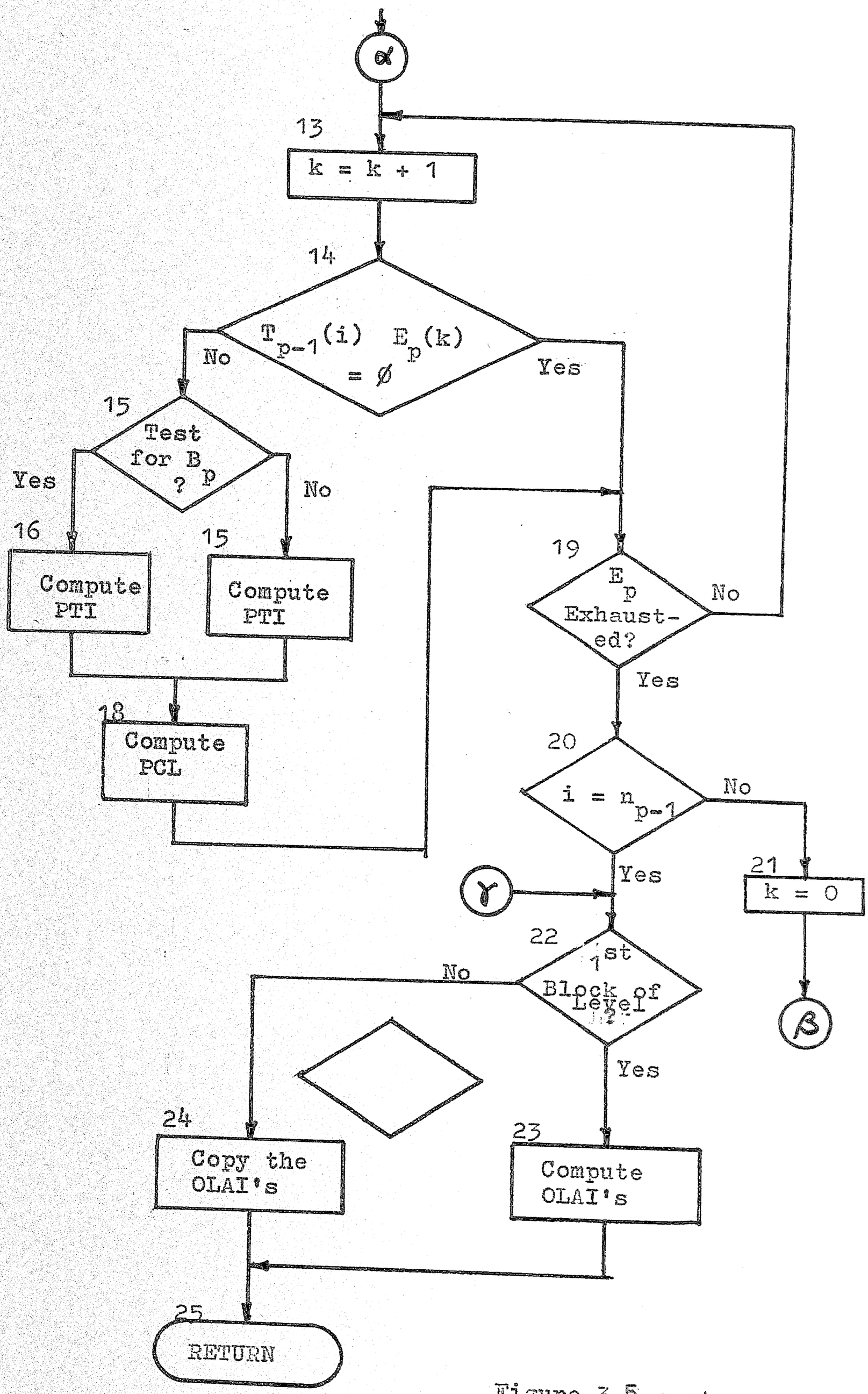


Figure 3.5 cont.

the sufficiency of the test-set T_p for block 1 through p.

Definition: One Level Ancestor Indicator, OLAI.

An OLAI is an ordered pair of numbers (L, i) associated with each element in T_p. L is the number of the level downstream from block p; i indicates which element of T_{p-1} was used in merging.

The OLAI is used to trace conveniently the pattern of signal values from network input to output. This will help greatly in checking for the sufficiency of the working test-set T_p and in solving the reconvergent fanout problem for certain test candidates in T_p.

Algorithm Merge:

Where applicable, express the inputs to B_p in terms of the lead numbers already used.

1) Initialize indexes i, j, k to 0.
2) Increase the index i of the working test-set $T_{p-1}(i)$ by 1.
3) If in $T_{p-1}(i)$ the literal corresponding to the output lead of block p is uncomplemented, then go to step 13. [This step decides whether we use the G_p table or the E_p table for merging].
4) Increase index j of table G_p by 1.
5) If the intersection of $T_{p-1}(i)$ with $G_p(j)$ is not empty, then go to step 7.
6) If all the elements in table G_p have been used, then go to step 11; otherwise, go to step 4.

7) If the result of the intersection of $T_{p-1}(i)$ and $G_p(j)$ is a test for block p, then go to step 9.

[If the output lead number of block p appears in the PCL of $T_{p-1}(i)$ the result of the intersection is a test for block p.]

8) Associate with the intersection of $T_{p-1}(i)$ and $G_p(j)$ the PTI $(i, \emptyset)$. Then go to step 6.

9) Associate with the intersection of $T_{p-1}(i)$ and $G_p(j)$ the PTI $(i, G_p(j))$.

10) Replace the lead number of block p in the PCL of $T_{p-1}(i)$ by the I component of the CL of $G_p(j)$. Then go to step 6. [This step updates the PCL of the result of the intersection.]

11) If all the elements in T_{p-1} have been used, then go to step 26.

12) Reset the index for table G_p to 0. Then go to step 2.

13) Increase the index k of table E_p by 1.

14) If the intersection of $T_{p-1}(i)$ and $E_p(k)$ is empty, then go to step 19.

15) If the result of the intersection of $T_{p-1}(i)$ and $E_p(k)$ is not a test for block p, then go to step 17.

[See step 7.]

16) Associate with the intersection of $T_{p-1}(i)$ and

$E_p(k)$ an ordered pair (i, $E_p(k)$). Then go to step 18.

17) Associate with the intersection of $T_{p-1}(i)$ and $E_p(k)$ the ordered pair $(i,\emptyset)$.

18) Replace the lead number of block p in the PCL of $T_{p-1}(i)$ by the I component of the CL of $E_p(k)$.

19) If all the elements in the E table of block p have been used, then go to step 20; otherwise go to step 13.

20) If all the elements in T_{p-1} have been used, then go to step 22.

21) Reset the index k of the E table to 0. Go to step 2.

22) If the current working block p is the first block in a level, then go to step 23; otherwise go to step 24.

23) Associate with each non-empty intersection in T_p the OLAI (L,i). Go to step 25.

24) Copy the OLAI from $T_{p-1}(i)$ and associate it with the current element in T_p.

25) Exit from the merging process.

D. Sufficiency of the Working Test-Set T_p.

We say that a working test-set T_p is sufficient if all the detectable faults in blocks 1 through p can be detected by elements in T_p. We use the PTI for the purpose

of establishing sufficiency. If the numbers in the second component of the set of PTI's of T_p forms a cover of table G_p and contains the set of required tests for table E_p, then T_p is a sufficient test-set for block p. If all the elements of T_{p-1} have been used in forming T_p, i.e., if the numbers in the first component of the set of PTI's of T_p include every element of T_{p-1}, then T_p is also a sufficient test-set for blocks 1 through p-1. If T_p satisfies both of the above two conditions, we say that T_p is sufficient and is a complete test for all blocks handled so far, providing there is no reconvergent fan-out. The additional considerations that have to be undertaken where reconvergent fan-out is involved will be discussed later.

If the sufficient conditions are <u>not</u> satisfied, we list those tests which would complete fault detection for block p and eliminate from T_{p-1} those tests that have not been used in forming T_p. Then we check for the sufficiency of the surviving elements in T_{p-1}. This process may repeat (reaching into T_{p-2}, T_{p-3}, etc.) as many times as needed. It will stop when sufficiency for some T_p is found or when p = 1. (i.e., the starting block has been reached.)

Fig. 3.6 is the flow-chart of the algorithm for checking the sufficiency of T_p, which is as follows:

1) If the second components of the PTI's of T_p form a cover of table G_p, then go to step 3.

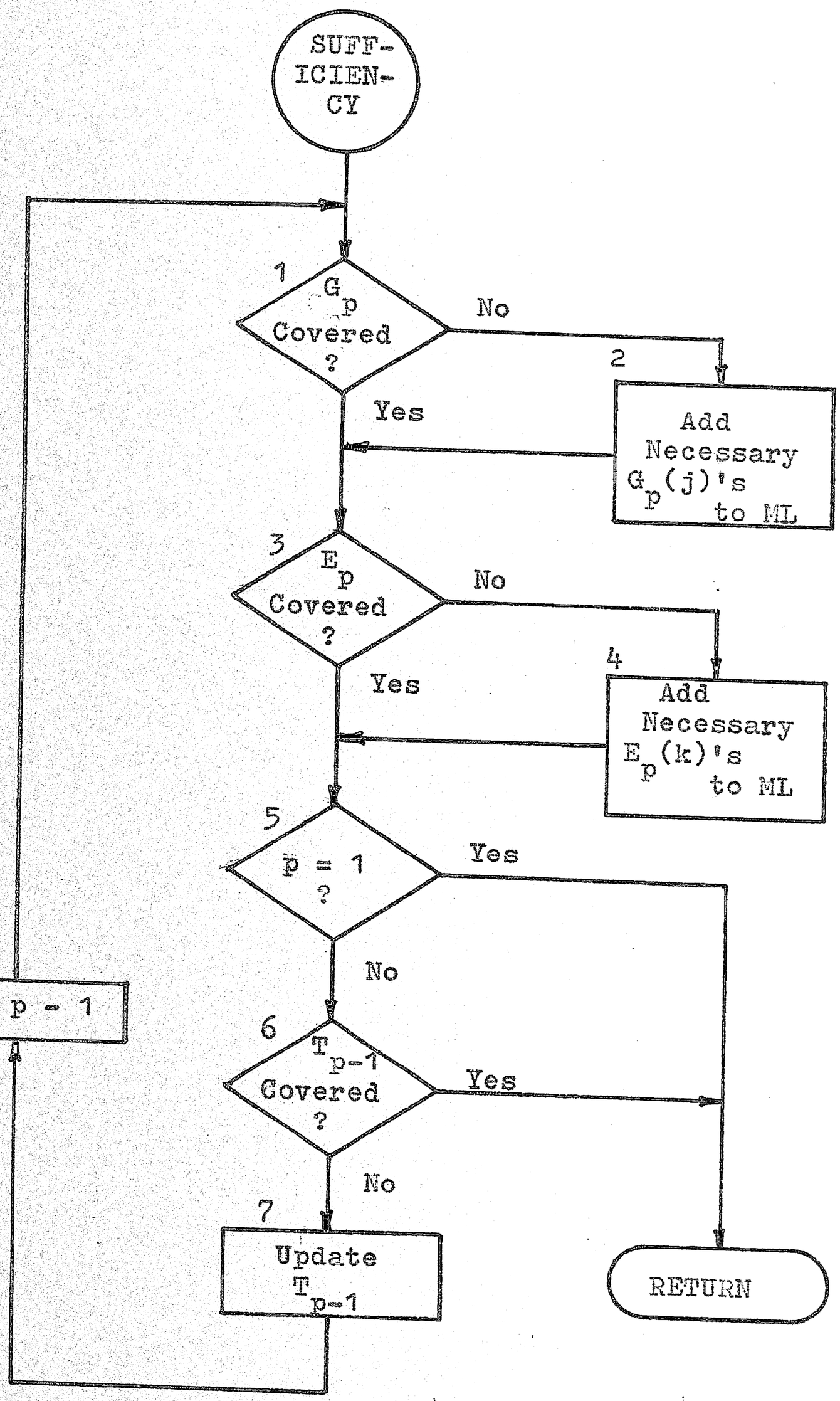

SUFF-
ICIEN-
CY
1
G_p
Covered
?
No
2
Add
Necessary
$G_p(j)$'s
to ML
Yes
3
E_p
Covered
?
No
4
Add
Necessary
$E_p(k)$'s
to ML
Yes
5
p = 1
?
Yes
No
6
T_{p-1}
Covered
?
Yes
No
7
Update
T_{p-1}
8
p = p - 1
RETURN

Figure 3.6

2) Add to the "Missing list" all the elements of G_p that together with T_p will give a complete cover of G_p.

3) If all the required tests of E_p (i.e., the first part in the ordered E-table) have been used in forming T_p, i.e., the second components of the PTI's of T_p include all the required tests in E_p, then go to step 5.

4) Add to the "missing list" the numbers corresponding to those required tests that do not appear in the PTI's of T_p.

[In steps 3 and/or 4, we may use the notation (p, G(j)) or (p, E(k)) to denote which tests are still needed.]

5) If the current block is the first block of the merging process then exit from the process.

6) If all the elements of T_{p-1} are contained in the first components of the PTI's of T_p, then exit from this sufficiency checking process.

[If T_p is a sufficient test-set for blocks 1 to (p-1), then we may stop the process.]

7) Eliminate the elements in T_{p-1} that do not appear in the first components of the PTI's of T_p.

[Note that if one wants to relabel the elements in T_{p-1} after deletions have occurred, all the indicators of the succeeding blocks will have to be revised.]

8) Decrease the index p of the block number by 1.

Go to step 1.

E. Reconvergent Fan-Out.

As mentioned before, T_p is surely sufficient only if there is no reconvergent fan-out between blocks. If reconvergent fan-out does exist, we must investigate whether or not it prevents propagation of signal changes due to faults all the way to the output. For this purpose, we use PCL.

Observe that the rule for forming the new PCL upon merging is one of pure substitution <u>without</u> simplification. Consequently, in case of reconvergence, the PCL will have a repeated entry. For example, when we merge both $(2^1;5^0)$ and $(2^1;6^0)$ into $(5^0,6^0;7^1)$, there results the PCL $(2^1,2^1;7^1)$. This shows that there is reconvergent fan-out from lead 2. Thus we see that reconvergent fan-out can be detected by inspection of the PCL. Whereas the PCL shows sensitivity to all <u>single</u> changes in the set of symbols on the left, (i.e., the I part) it does not assure sensitivitity when more than one change occurs. Consequently, whenever reconvergent fan-out is detected, additional steps must be taken, as indicated in the overall test-generation algorithm. (Fig. 3.4)

A flow-chart for testing for reconvergent fan-out is presented in Fig. 3.7, where again i stands for the ith elements in the set T_p, which has a total number of n members.

We denote by O_1 the normal response to $T_p(i)$ and by $\{O_2\}$ the set of responses to $T_p(i)$ as the value of each lead that has reconvergent fanout is changed one at a time.

An informal description of the Fanout-Algorithm is as follows:

1) Initialize i to 1, so that the first test to be considered is $T_p(1)$.
2) If there is any reconvergent fanout in $T_p(i)$, then go to step 5.
3) If all the elements in T_p have been tested, exit from the process.
4) Increase the value of index i by 1. Go to step 2.
5) Store the output due to $T_p(i)$ in O_1.
[It is very easy to find the output response to $T_p(i)$. Since we have associated with each element in T_p a path connectivity list (PCL), we can obtain the normal (fault-free) output response by looking at the last component of its PCL. If the superscript on the output lead is a 1(0), its normal value is 0(1).]
6) Store the output responses to $\{T_p^j(i)\}$ in $\{O_2\}$, where each $T_p^j(i)$ represents the vertex derived from $T_p(i)$ by changing the value of the duplicated lead j in the PCL of $T_p(i)$. An element T_p^j is obtained by changing only the value of the repeated lead j.
[For example, let $T_p(i) = \bar{x}_{11}\bar{x}_{12}x_{13}x_{14}\bar{x}_{15}\bar{x}_{16}$ and its PCL be $(12^1, 15^1, 12^1, 15^1, 16^1; 29^1)$. Then $\{T_p^j(i)\}$ =

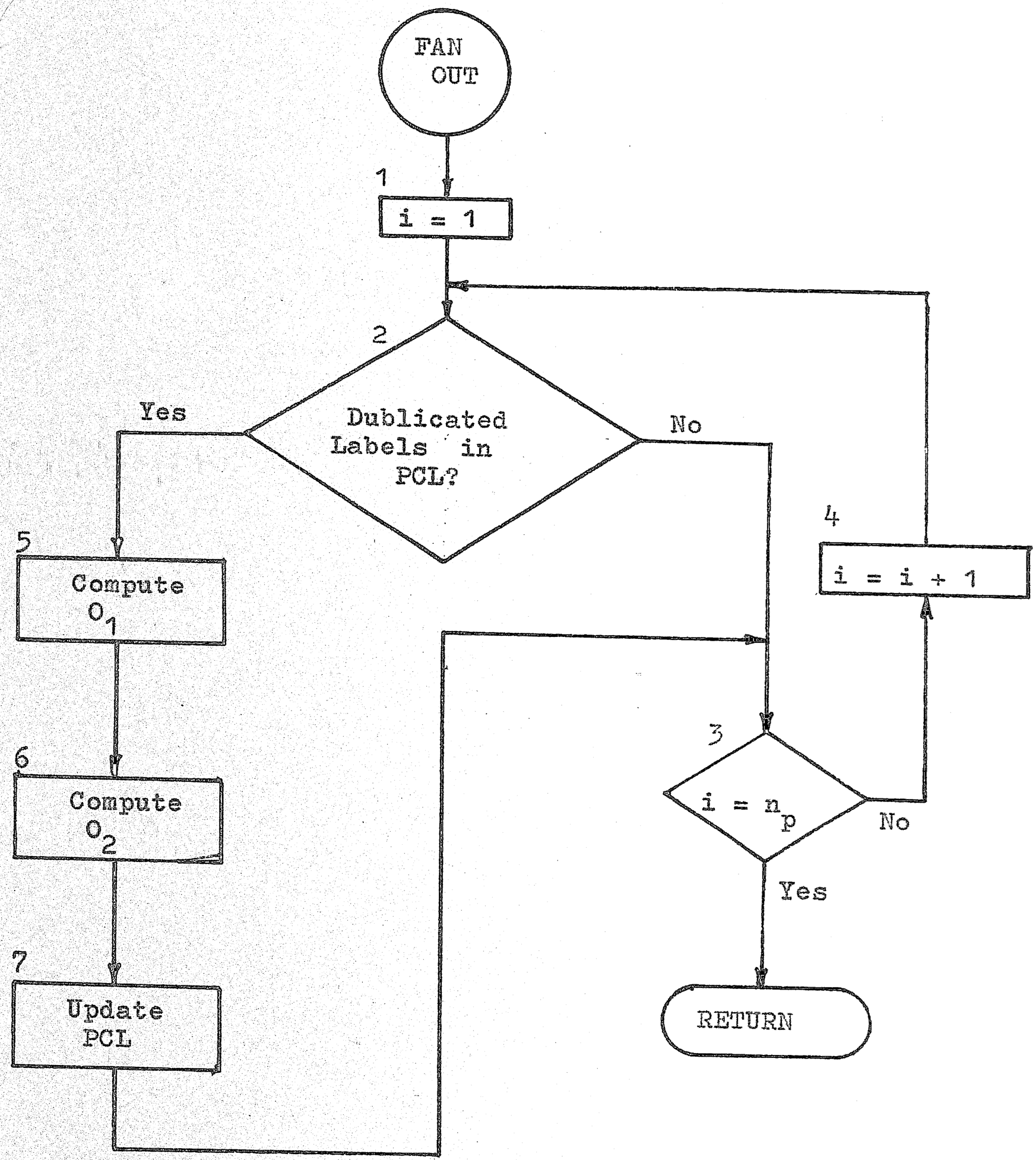

FAN OUT
1
i = 1
2
Dublicated Labels in PCL?
Yes
No
5
Compute O_1
6
Compute O_2
7
Update PCL
4
i = i + 1
3
i = n_p
No
Yes
RETURN

Figure 3.7

$\{\bar{x}_{11}x_{12}x_{13}x_{14}\bar{x}_{15}\bar{x}_{16}, \bar{x}_{11}\bar{x}_{12}x_{13}x_{14}x_{15}\bar{x}_{16}\}$. The way to obtain the responses to $\{T_p^j(i)\}$ is a bit more tedious. If an element in $\{T_p^j(i)\}$ is contained in $T_p(k)$ for some k, then we may look up the response as in step 5 by using the corresponding PCL. Otherwise, we either have to trace the responses of $\{T_p^j(i)\}$ level-by-level to the output or trace it to the next level and then use some other set, T_k. Here the OLAI may be helpful. We have presented here only one of several possibilities for doing the fanout-checking process. Other approaches are indeed possible. For example, we could check the responses to $T_p(i)$ and $\{T_p^j(i)\}$ level-by-level instead of all the way to the observable output. If the responses happen to be identical at some level before the observable output is reached, then we can quit the tracing process.]

7) Eliminate the lead numbers in the PCL of $T_p(i)$ that are insensitive to error (i.e., O_1 and O_2 are identical). Eliminate all but one occurrence of each sensitive lead number in the PCL of $T_p(i)$. Go to step 3.

F. At this point, we continue the running example of Fig. 3.1.

First, we assign to T_1 the test-sets. (G/E tables) of block 1 with non-empty CL's to T_1. So we have

$T_1(1) = x_{19}x_{20}$ $\qquad\qquad$ $(19^0, 20^0; 21^1)$

$T_1(2) = \bar{x}_{19}\bar{x}_{20}$ $(19^1, 20^1; 21^1)$

$T_1(3) = x_{19}\bar{x}_{20}$ $(19^0, 20^1; 21^0)$

$T_1(4) = \bar{x}_{19}x_{20}$ $(19^1, 20^0; 21^0)$

Next we merge T_1 with growth and existence tests for B_2. We replace the input lead values of B_2 by the appropriate output lead values of the previous level, i.e., we let $X_{15} = x_{12}$, $x_{16} = x_{13}$.

	PCL	PTI	OLAI
$T_2(1) = x_{12}\bar{x}_{13}x_{20}$	$(12^0, 13^1, 20^0; 21^1)$,	$(1, E1)$,	$(3,1)$
$T_2(2) = \bar{x}_{12}x_{13}x_{20}$	$(12^1, 13^0, 20^0; 21^1)$,	$(1, E2)$,	$(3,1)$
$T_2(3) = x_{12}x_{13}\bar{x}_{20}$	$(12^0, 13^0, 20^1; 21^1)$,	$(2, G1)$,	$(3,2)$
$T_2(4) = \bar{x}_{12}\bar{x}_{13}\bar{x}_{20}$	$(12^1, 13^1, 20^1; 21^1)$,	$(2, G2)$,	$(3,2)$
$T_2(5) = x_{12}\bar{x}_{13}\bar{x}_{20}$	$(12^0, 13^1, 20^1; 21^0)$,	$(3, E1)$,	$(3,3)$
$T_2(6) = \bar{x}_{12}x_{13}\bar{x}_{20}$	$(12^1, 13^0, 20^1; 21^0)$,	$(3, E2)$,	$(3,3)$
$T_2(7) = x_{12}x_{13}x_{20}$	$(12^0, 13^0, 20^0; 21^0)$,	$(4, G1)$,	$(3,4)$
$T_2(8) = \bar{x}_{12}\bar{x}_{13}x_{20}$	$(12^1, 13^1, 20^0; 21^0)$,	$(4, G2)$,	$(3,4)$

Sufficiency? Yes, all the entries in the G/E tables of block 2 have been used in the mergers, and every element of T_1 has been used in forming T_2.

Now merge T_2 with the tests for B_3. First, make the substitutions

$x_{17} = x_{13}$, $x_{18} = x_{12}$.

	PCL	PTI	OLAI
$T_3(1) = x_{12}\bar{x}_{13}x_{14}$	$(12^0, 13^1, 14^0, 12^0; 21^1)$,	$(1, E1)$,	$(3,1)$
$T_3(2) = \bar{x}_{12}x_{13}\bar{x}_{14}$	$(12^1, 13^0, 13^0, \quad ; 21^1)$,	$(2, E3)$,	$(3,1)$
$T_3(3) = \bar{x}_{12}x_{13}x_{14}$	$(12^1, 13^0, 13^0, \quad ; 21^1)$,	$(2, E4)$,	$(3,1)$
$T_3(4) = \bar{x}_{12}\bar{x}_{13}x_{14}$	$(12^1, 13^1, 13^1, 12^1; 21^1)$,	$(4, G1)$,	$(3,2)$
$T_3(5) = \bar{x}_{12}\bar{x}_{13}\bar{x}_{14}$	$(12^1, 13^1, 13^1, \quad ; 21^1)$,	$(4, G3)$,	$(3,2)$
$T_3(6) = x_{12}\bar{x}_{13}\bar{x}_{14}$	$(12^0, 13^1, 14^1, 13^1; 21^0)$,	$(5, G2)$,	$(3,2)$
$T_3(7) = x_{12}x_{13}\bar{x}_{14}$	$(12^0, 13^0, 13^0, \quad ; 21^0)$,	$(7, E2)$,	$(3,4)$
$T_3(8) = x_{12}x_{13}x_{14}$	$(12^0, 13^0, 13^0, \quad ; 21^0)$,	$(7, E5)$,	$(3,4)$

This is at the end of the second level, so we have to test for reconvergent-fanout before checking the sufficiency of T_3. We choose to work with the OLAI to compute the appropriate outputs.

$$T_3(1) \rightarrow x_{12}\bar{x}_{13}x_{14} \rightarrow (3,1) \rightarrow x_{19}x_{20} \rightarrow 0$$

$$T_3^1(1) \rightarrow \bar{x}_{12}\bar{x}_{13}x_{14} \rightarrow T_3(4) \rightarrow (3,2) \rightarrow \bar{x}_{19}\bar{x}_{20} \rightarrow 0$$

Test $T_3(1)$ is <u>insensitive</u> to lead 12. Hence the PCL of $T_3(1)$ is changed to $(13^1, 14^0; 21^1)$.

$$T_3(2) \rightarrow \bar{x}_{12}x_{13}\bar{x}_{14} \rightarrow (1,1) \rightarrow 0$$

$$T_3^1(2) \rightarrow \bar{x}_{12}\bar{x}_{13}\bar{x}_{14} \rightarrow T_3(5) \rightarrow (1,2) \rightarrow 0$$

Test $T_3(2)$ is <u>insensitive</u> to lead 13. Hence the PCL of $T_3(2)$ is changed to $(12^1; 21^1)$.

$$T_3(3) \rightarrow \bar{x}_{12}x_{13}x_{14} \rightarrow (1,1) \rightarrow 0$$

$$T_3^{13}(3) \rightarrow \bar{x}_{12}\bar{x}_{13}x_{14} \rightarrow (1,2) \rightarrow 0$$

This is at the end of the second level, se we have to test for reconvergent-fanout, before checking the sufficiency of T_3. We choose to work with the OLAI to compute the appropriate outputs.

$$T_3(1) = x_{12}\bar{x}_{13}x_{14} \rightarrow (3,1) \rightarrow x_{19}x_{20} \rightarrow 0$$

$$T_3^{12}(1) = \bar{x}_{12}\bar{x}_{13}x_{14} = T_3(4) \rightarrow (3,2) \rightarrow \bar{x}_{19}\bar{x}_{20} \rightarrow 0$$

Test $T_3(1)$ is <u>insensitive</u> to lead 12. Hence the PCL of $T_3(1)$ is changed to $(13^0, 14^0; 21^1)$.

$$T_3(2) = \bar{x}_{12}x_{13}\bar{x}_{14} \rightarrow (3,1) \rightarrow 0$$

$$T_3^{13}(2) = \bar{x}_{12}\bar{x}_{13}\bar{x}_{14} = T_3(5) \rightarrow (3,2) \rightarrow 0$$

Test $T_3(2)$ is <u>insensitive</u> to lead 13. Hence the PCL of $T_3(2)$ is changed to $(12^1; 21^1)$.

$$T_3(3) = \bar{x}_{12}x_{13}x_{14} \rightarrow (3,1) \rightarrow 0$$

$$T_3^{13}(3) = \bar{x}_{12}\bar{x}_{13}x_{14} \rightarrow (3,2) \rightarrow 0$$

$T_3(3)$ is insensitive to lead 13. The PCL of $T_3(3)$ is changed to $(12^1; 21^1)$.

$$T_3(4) = \bar{x}_{12}\bar{x}_{13}x_{14} \rightarrow (1,2) \rightarrow 0$$

$$T_3^{12}(4) = x_{12}\bar{x}_{13}x_{14} = T_3(1) \rightarrow (3,1) \rightarrow 0$$

$$T_3^{13}(4) = \bar{x}_{12}x_{13}x_{14} = T_3(3) \rightarrow (3,1) \rightarrow 0$$

The PCL of $T_3(4)$ is changed to $(\emptyset;\ 21^1)$

$$T_3(5) = \bar{x}_{12}\bar{x}_{13}\bar{x}_{14} \rightarrow (3,2) \rightarrow 0$$

$$T_3^{13}(5) = \bar{x}_{12}x_{13}\bar{x}_{14} \rightarrow T_3(2) \rightarrow (3,1) \rightarrow 0$$

The PCL of $T_3(5)$ is changed to $(12^1;\ 21^1)$

$$T_3(6) = x_{12}\bar{x}_{13}\bar{x}_{14} \rightarrow (3,3) \rightarrow x_{19}\bar{x}_{20} \rightarrow 1$$

$$T_3^{13}(6) = x_{12}x_{13}x_{14} = T_3(7) \rightarrow (3,4) \rightarrow \bar{x}_{19}x_{20} \rightarrow 1$$

The PCL of $T_3(6)$ is changed to $(12^0,\ 14^1;\ 21^0)$

$$T_3(7) = x_{12}x_{13}\bar{x}_{14} \rightarrow (3,4) \rightarrow 1$$

$$T_3^{13}(7) = x_{12}\bar{x}_{13}\bar{x}_{14} = T_3(6) \rightarrow (3,3) \rightarrow 1$$

The PCL of $T_3(7)$ is changed to $(12^0;\ 21^0)$

$$T_3(8) = x_{12}x_{13}x_{14} \rightarrow (3,4) \rightarrow 1$$

$$T_3^{13}(8) = x_{12}\bar{x}_{13}x_{14} = T_3(1) \rightarrow (3,1) \rightarrow 0$$

Test $T_3(8)$ <u>is</u> sensitive to lead 13, so that the PCL of $T_3(8)$ remains $(12^0,\ 13^0;\ 21^0)$.

Now we check the sufficiency of T_3. First, check the

PTI of each element. Because E_1 and E_3 of B_3 are sufficient for existence tests and all three growth tests of B_3 are merged, block B_3 is fully tested.

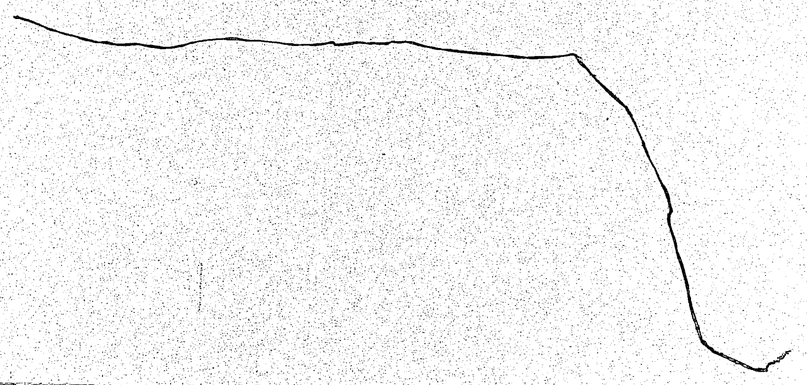
Missing list: $\{\phi\}$

Next we find that $\{T_2(3), T_2(6), T_2(8)\}$ have not been used in forming T_3. We eliminate $\{T_2(3), T_2(6), T_2(8)\}$ from T_2. Then by checking the PTI's of the remaining elements of T_2, we find that B_2 is "covered" by the remaining tests in the new T_2. Also, all elements of T_1 have been used in forming the surviving elements in T_2. Thus our missing list at this time is again empty, i.e., T_3 is a sufficient test-set for blocks 1 to 3.

Note the special role played by $T_3(4)$ in this example. The first component of its PCL is empty, which implies that no "sensitive-path" can be "connected" to $T_3(4)$. But from the PTI of $T_3(4)$, we see that $T_3(4)$ is a necessary test for B_3, B_2 and B_1.

Because all necessary tests are included, there is no need to enter into the "catch-up" procedure, so we may continue our merging process.

The elements of T_2 that were not merged and eliminated cause a revision in T_2 and updating of the notation in the PTI's. T_1 has not been changed, however, so we do not have to relabel T_1. (Note, this corresponds to the step of "updating labels of $\{Tp's\}$ " in Fig. 3.4.)



We relabel the elements of T_2 as:

$T_2(1) = x_{12}\bar{x}_{13}x_{20}$ $(12^0, 13^1, 20^0; 21^1)$, (1,E1), (3,1)

$T_2(2) = \bar{x}_{12}x_{13}x_{20}$ $(12^1, 13^0, 20^0; 21^1)$, (1,E2), (3,1)

$T_2(3) = \bar{x}_{12}\bar{x}_{13}\bar{x}_{20}$ $(12^1, 13^1, 20^1; 21^1)$, (2,G2), (3,2)

$T_2(4) = x_{12}\bar{x}_{13}\bar{x}_{20}$ $(12^0, 13^1, 21^1; 21^0)$, (3,E1), (3,3)

$T_2(5) = x_{12}x_{13}x_{20}$ $(12^0, 13^0, 20^0; 21^0)$, (4,G2), (3,4)

Relabel T_3 as follows, where the revised PCL's are used.

$T_3(1) = x_{12}\bar{x}_{13}x_{14}$ $(13^1, 14^0, \quad ; 21^1)$, (1,E2), (3,1)

$T_3(2) = \bar{x}_{12}x_{13}\bar{x}_{14}$ $(12^1, \quad ; 21^1)$, (2,E3), (3,1)

$T_3(3) = \bar{x}_{12}x_{13}x_{14}$ $(12^1, \quad ; 21^1)$, (2,E4), (3,1)

$T_3(4) = \bar{x}_{12}\bar{x}_{13}\bar{x}_{14}$ $(12^1, \quad ; 21^1)$, (3,G3), (3,2)

$T_3(5) = x_{12}\bar{x}_{13}\bar{x}_{14}$ $(12^0, 14^1, \quad ; 21^0)$, (4,G2), (3,3)

$T_3(6) = x_{12}x_{13}\bar{x}_{14}$ $(12^0, \quad ; 21^0)$, (5,E2), (3,4)

$T_3(7) = x_{12}x_{13}x_{14}$ $(12^0, 13^0, \quad ; 21^0)$, (5,E2), (3,4)

$T_3(8) = \bar{x}_{12}\bar{x}_{13}x_{14}$ $(\emptyset \quad ; 21^1)$, (3,G1), (3,2)

Now we are ready to merge T_3 with the tests for B_4. First, we make the substitutions

$x_1 = a$, $x_2 = b$, $x_3 = c$, $x_4 = d$

$\times$ $T_4(1) = \bar{a}\bar{b}\bar{c}\bar{d}\ \bar{x}_{13}x_{14}$ $(13^1, 14^0,$; $21^1), (1,\emptyset), (2,1)$

$T_4(2) = \bar{a}\bar{b}cd\ \bar{x}_{13}x_{14}$ $(13^1, 14^0,$; $21^1), (1,\emptyset), (2,1)$

$\times$ $T_4(3) = ab\bar{c}d\ \bar{x}_{13}x_{14}$ $(13^1, 14^0,$; $21^1), (1,\emptyset), (2,1)$

$\times$ $T_4(4) = \bar{a}\bar{b}\bar{c}d\ \bar{x}_{13}x_{14}$ $(13^1, 14^0,$; $21^1), (1,\emptyset), (2,1)$

$*$ $T_4(5) = abcd\ \bar{x}_{13}x_{14}$ $(13^1, 14^0,$; $21^1), (1,\emptyset), (2,1)$

$T_4(6) = a\bar{b}\bar{c}\bar{d}\ \bar{x}_{13}x_{14}$ $(13^1, 14^0,$; $21^1), (1,\emptyset), (2,1)$

$T_4(7) = a\bar{b}cd\ \bar{x}_{13}x_{14}$ $(13^1, 14^0,$; $21^1), (1,\emptyset), (2,1)$

$T_4(8) = a\bar{b}\bar{c}d\ \bar{x}_{13}x_{14}$ $(13^1, 14^0,$; $21^1), (1,\emptyset), (2,1)$

$T_4(9) = ab\bar{c}\bar{d}\ \bar{x}_{13}\bar{x}_{14}$ $(b^0, d^1,$; $21^1), (2,G1), (2,2)$

$\times$ $T_4(10) = a\bar{b}c\bar{d}\ x_{13}\bar{x}_{14}$ $(c^0, d^1,$; $21^1), (2,G2), (2,2)$

$T_4(11) = \bar{a}bcd\ x_{13}\bar{x}_{14}$ $(a^1, b^0,$; $21^1), (2,G3), (2,2)$

$T_4(12) = \bar{a}b\bar{c}d\ x_{13}\bar{x}_{14}$ $(a^1, b^0,$; $21^1), (2,G4), (2,2)$

$T_4(13) = abc\bar{d}\ x_{13}\bar{x}_{14}$ $(d^1,$; $21^1), (2,G5), (2,2)$

$T_4(14) = \bar{a}\bar{b}cd\ x_{13}\bar{x}_{14}$ $(c^0, d^1,$; $21^1), (2,G6), (2,2)$

$\times$ $T_4(15) = \bar{a}b\bar{c}\bar{d}\ x_{13}\bar{x}_{14}$ $(b^0,$; $21^1), (2,G7), (2,2)$

$T_4(16) = \bar{a}bcd\ x_{13}\bar{x}_{14}$ $(\emptyset,$; $21^1), (2,G8), (2,2)$

$\times$ $T_4(17) = ab\bar{c}\bar{d}\ x_{13}x_{14}$ $(b^0, d^1,$; $21^1), (3,G1), (2,3)$

$T_4(18) = a\bar{b}c\bar{d}\ x_{13}x_{14}\ (c^0, d^1,$; $21^1), (3,G2), (2,3)$

$\times T_4(19) = \bar{a}bcd\ x_{13}x_{14}\ (a^1, b^0,$; $21^1), (3,G3), (2,3)$

$\times T_4(20) = \bar{a}b\bar{c}d\ x_{13}x_{14}\ (a^1, b^0,$; $21^1), (3,G4), (2,3)$

$\times T_4(21) = abc\bar{d}\ x_{13}x_{14}\ (d^1,$; $21^1), (3,G5), (2,3)$

$\times T_4(22) = \bar{a}\bar{b}c\bar{d}\ x_{13}x_{14}\ (c^0, d^1,$; $21^1), (3,G6), (2,3)$

$\times T_4(23) = \bar{a}b\bar{c}\bar{d}\ x_{13}x_{14}\ (b^0,$; $21^1), (3,G7), (2,3)$

$\times T_4(24) = \bar{a}bc\bar{d}\ x_{13}x_{14}\ (\emptyset,$; $21^1), (3,G8), (2,3)$

$T_4(25) = ab\bar{c}\bar{d}\ \bar{x}_{13}\bar{x}_{14}\ (b^0, d^1,$; $21^1), (4,G1), (2,4)$

$\times T_4(26) = a\bar{b}c\bar{d}\ \bar{x}_{13}\bar{x}_{14}\ (c^0, d^1,$; $21^1), (4,G2), (2,4)$

$* T_4(27) = \bar{a}bcd\ \bar{x}_{13}\bar{x}_{14}\ (a^1, b^0,$; $21^1), (4,G3), (2,4)$

$T_4(28) = \bar{a}b\bar{c}d\ \bar{x}_{13}\bar{x}_{14}\ (a^1, b^0,$; $21^1), (4,G4), (2,4)$

$* T_4(29) = abc\bar{d}\ \bar{x}_{13}\bar{x}_{14}\ (d^1,$; $21^1), (4,G5), (2,4)$

$T_4(30) = \bar{a}\bar{b}cd\ \bar{x}_{13}\bar{x}_{14}\ (c^0, d^1,$; $21^1), (4,G6), (2,4)$

$\times T_4(31) = \bar{a}b\bar{c}\bar{d}\ \bar{x}_{13}\bar{x}_{14}\ (b^0,$; $21^1), (4,G7), (2,4)$

$* T_4(32) = \bar{a}bc\bar{d}\ \bar{x}_{13}\bar{x}_{14}\ (\emptyset$; $21^1), (4,G8), (2,4)$

$T_4(33) = \bar{a}\bar{b}c\bar{d}\ \bar{x}_{13}\bar{x}_{14}\ (b^1, c^1, 14^1;\ 21^0), (5,E1), (2,5)$

$T_4(34) = \bar{a}\bar{b}cd\ \bar{x}_{13}\bar{x}_{14}\ (b^1, d^0, 14^1;\ 21^0), (5,E2), (2,5)$

$T_4(35) = ab\bar{c}d\ \bar{x}_{13}\bar{x}_{14}\ (a^0, d^0, 14^1; 21^0), (5,E3), (2,5)$

$T_4(36) = \bar{a}\bar{b}\bar{c}d\ \bar{x}_{13}\bar{x}_{14}\ (b^1, 14^1, ; 21^0), (5,E4), (2,5)$

* $T_4(37) = abcd\ \bar{x}_{13}\bar{x}_{14}\ (a^0, d^0, 14^1; 21^0), (5,E5), (2,5)$

X $T_4(38) = a\bar{b}\bar{c}\bar{d}\ \bar{x}_{13}\bar{x}_{14}\ (b^1, c^1, 14^1; 21^0), (5,E6), (2,5)$

X $T_4(39) = a\bar{b}cd\ \bar{x}_{13}\bar{x}_{14}\ (d^0, 14^1, ; 21^0), (5,E7), (2,5)$

X $T_4(40) = a\bar{b}\bar{c}\bar{d}\ \bar{x}_{13}\bar{x}_{14}\ (14^1, ; 21^0), (5,E8), (2,5)$

* $T_4(41) = \bar{a}b\bar{c}\bar{d}\ x_{13}\bar{x}_{14}\ (b^1, c^1, ; 21^0), (6,E1), (2,6)$

$T_4(42) = \bar{a}\bar{b}cd\ x_{13}\bar{x}_{14}\ (b^1, d^0, ; 21^0), (6,E2), (2,6)$

$T_4(43) = ab\bar{c}d\ x_{13}\bar{x}_{14}\ (a^0, d^0, ; 21^0), (6,E3), (2,6)$

* $T_4(44) = \bar{a}\bar{b}\bar{c}d\ x_{13}\bar{x}_{14}\ (b^1, ; 21^0), (6,E4), (2,6)$

$T_4(45) = abcd\ x_{13}\bar{x}_{14}\ (a^0, d^0, ; 21^0), (6,E5), (2,6)$

* $T_4(46) = a\bar{b}\bar{c}\bar{d}\ x_{13}\bar{x}_{14}\ (b^1, c^1, ; 21^0), (6,E6), (2,6)$

X $T_4(47) = a\bar{b}cd\ x_{13}\bar{x}_{14}\ (d^0, ; 21^0), (6,E7), (2,6)$

* $T_4(48) = a\bar{b}\bar{c}\bar{d}\ x_{13}\bar{x}_{14}\ (\emptyset, ; 21^0), (6,E8), (2,6)$

* $T_4(49) = \bar{a}\bar{b}c\bar{d}\ x_{13}x_{14}\ (b^1, c^1, 13^0; 21^0), (7,E1), (2,7)$

X $T_4(50) = \bar{a}\bar{b}cd\ x_{13}x_{14}\ (b^1, d^0, 13^0; 21^0), (7,E2), (2,7)$

X $T_4(51) = ab\bar{c}d\ x_{13}x_{14}\ (a^0, d^0, 13^0; 21^0), (7,E3), (2,7)$

* $T_4(52) = \bar{a}\bar{b}cd\ x_{13}x_{14}\ (b^1, 13^0, ; 21^0), (7,E4), (2,7)$

$T_4(53) = abcd\ x_{13}x_{14}$ (a^0, d^0, 13^0; 21^0), (7,E5), (2,7)

$T_4(54) = a\bar{b}\bar{c}d\ x_{13}x_{14}$ (b^1, c^1, 13^0; 21^0), (7,E6), (2,7)

$T_4(55) = a\bar{b}cd\ x_{13}x_{14}$ (d^0, 13^0, ; 21^0), (7,E7), (2,7)

$T_4(56) = a\bar{b}\bar{c}d\ x_{13}x_{14}$ (13^0, ; 21^0), (7,E8), (2,7)

$T_4(57) = ab\bar{c}\bar{d}\ \bar{x}_{13}x_{14}$ (Ø, ; 21^1), (8,Ø), (2,8)

$T_4(58) = a\bar{b}cd\ \bar{x}_{13}x_{14}$ (Ø, ; 21^1), (8,Ø), (2,8)

$T_4(59) = \bar{a}bcd\ \bar{x}_{13}x_{14}$ (Ø, ; 21^1), (8,Ø), (2,8)

$T_4(60) = \bar{a}b\bar{c}d\ \bar{x}_{13}x_{14}$ (Ø, ; 21^1), (8,Ø), (2,8)

$T_4(61) = abc\bar{d}\ \bar{x}_{13}x_{14}$ (Ø, ; 21^1), (8,Ø), (2,8)

$T_4(62) = \bar{a}\bar{b}cd\ \bar{x}_{13}x_{14}$ (Ø, ; 21^1), (8,Ø), (2,8)

$T_4(63) = \bar{a}b\bar{c}\bar{d}\ \bar{x}_{13}x_{14}$ (Ø, ; 21^1), (8,Ø), (2,8)

$T_4(64) = \bar{a}bc\bar{d}\ \bar{x}_{13}x_{14}$ (Ø, ; 21^1), (8,Ø), (2,8)

Sufficiency is met. Turning now to B5, we let $x_5 = b$, $x_6 = e$, $x_7 = c$ and merge T_4 with the G/E tables of B_5.

$T_5(1) = \bar{a}\bar{b}\bar{c}\bar{d}e\ x_{14}$ (b^1, c^1, 14^0, ; 21^1), (1,G1), (2,1)

$T_5(2) = \bar{a}\bar{b}cd\bar{e}\ x_{14}$ (b^1, e^1, 14^0, ; 21^1), (2,G3), (2,1)

$T_5(3) = ab\bar{c}d\bar{e}\ x_{14}$ (c^1, e^1, 14^0, ; 21^1), (3,G2), (2,1)

X $T_5(4) = \bar{a}\bar{b}\bar{c}de\ x_{14}$ (b^1, c^1, 14^0, ; 21^1), (4,G1), (2,1)

$T_5(5) = a\bar{b}\bar{c}\bar{d}e\ x_{14}$ (b^1, c^1, 14^0, ; 21^1), (6,G1), (2,1)

$T_5(6) = a\bar{b}cd\bar{e}\ x_{14}$ (b^1, e^1, 14^0, ; 21^1), (7,G3), (2,1)

$T_5(7) = a\bar{b}\bar{c}de\ x_{14}$ (b^1, c^1, 14^0, ; 21^1), (8,G1), (2,1)

$T_5(8) = ab\bar{c}\bar{d}e\ \bar{x}_{14}$ (b^0, d^1, ; 21^1), (9,Ø), (2,2)

X $T_5(9) = a\bar{b}cd\bar{e}\ \bar{x}_{14}$ (c^0, d^1, ; 21^1), (10,Ø), (2,2)

$T_5(10) = \bar{a}bcd\bar{e}\ \bar{x}_{14}$ (a^1, b^0, ; 21^1), (11,Ø), (2,2)

X $T_5(11) = \bar{a}bcde\ \bar{x}_{14}$ (a^1, b^0, ; 21^1), (11,Ø), (2,2)

X $T_5(12) = \bar{a}b\bar{c}de\ \bar{x}_{14}$ (a^1, b^0, ; 21^1), (12,Ø), (2,2)

$T_5(13) = abc\bar{d}\bar{e}\ \bar{x}_{14}$ (d^1, ; 21^1), (13,Ø), (2,2)

$T_5(14) = abcd\bar{e}\ \bar{x}_{14}$ (d^1, ; 21^1), (13,Ø), (2,2)

$T_5(15) = \bar{a}\bar{b}cd\bar{e}\ \bar{x}_{14}$ (c^0, d^1, ; 21^1), (14,Ø), (2,2)

X $T_5(16) = \bar{a}b\bar{c}\bar{d}e\ \bar{x}_{14}$ (b^0, ; 21^1), (15,Ø), (2,2)

$T_5(17) = \bar{a}bcd\bar{e}\ \bar{x}_{14}$ (Ø, ; 21^1), (16,Ø), (2,2)

X $T_5(18) = \bar{a}bcd\bar{e}\ \bar{x}_{14}$ (Ø, ; 21^1), (16,Ø), (2,2)

X $T_5(19) = ab\bar{c}\bar{d}e\ x_{14}$ (b^0, d^1, ; 21^1), (17,Ø), (2,3)

$T_5(20) = a\bar{b}cd\bar{e}\ x_{14}$ (c^0, d^1, ; 21^1), (18,Ø), (2,3)

X $T_5(21) = \bar{a}bcd\bar{e}\ x_{14}$ (a^1, b^0, ; 21^1), (19,Ø), (2,3)

$$\chi_{T_5(22)} = \bar{a}bcde\ x_{14}\ (a^1,\ b^0,\ ;\ 21^1),\ (19,\emptyset),\ (2,3)$$

$$\chi_{T_5(23)} = \bar{a}b\bar{c}de\ x_{14}\ (a^1,\ b^0,\ ;\ 21^1),\ (20,\emptyset),\ (2,3)$$

$$\chi_{T_5(24)} = abc\bar{d}\bar{e}\ x_{14}\ (d^1,\ ;\ 21^1),\ (21,\emptyset),\ (2,3)$$

$$\chi_{T_5(25)} = abc\bar{d}e\ x_{14}\ (d^1,\ ;\ 21^1),\ (21,\emptyset),\ (2,3)$$

$$\chi_{T_5(26)} = \bar{a}\bar{b}cde\ x_{14}\ (c^0,\ d^1,\ ;\ 21^1),\ (22,\emptyset),\ (2,3)$$

$$\chi_{T_5(27)} = \bar{a}bcde\ x_{14}\ (b^0,\ ;\ 21^1),\ (23,\emptyset),\ (2,3)$$

$$\chi_{T_5(28)} = \bar{a}bc\bar{d}\bar{e}\ x_{14}\ (\emptyset,\ ;\ 21^1),\ (24,\emptyset),\ (2,3)$$

$$\chi_{T_5(29)} = \bar{a}bc\bar{d}e\ x_{14}\ (\emptyset,\ ;\ 21^1),\ (24,\emptyset),\ (2,3)$$

$$T_5(30) = ab\bar{c}\bar{d}\bar{e}\ \bar{x}_{14}\ (b^0,\ d^1,\ ;\ 21^1),\ (25,\emptyset),\ (2,4)$$

$$\chi_{T_5(31)} = a\bar{b}cd\bar{e}\ \bar{x}_{14}\ (c^0,\ d^1,\ ;\ 21^1),\ (26,\emptyset),\ (2,4)$$

$$T_5(32) = \bar{a}b\bar{c}d\bar{e}\ \bar{x}_{14}\ (a^1,\ b^0,\ ;\ 21^1),\ (28,\emptyset),\ (2,4)$$

$$T_5(33) = \bar{a}\bar{b}cd\bar{e}\ \bar{x}_{14}\ (c^0,\ d^1,\ ;\ 21^1),\ (30,\emptyset),\ (2,4)$$

$$\chi_{T_5(34)} = \bar{a}bc\bar{d}\bar{e}\ \bar{x}_{14}\ (b^0,\ ;\ 21^1),\ (31,\emptyset),\ (2,4)$$

$$T_5(35) = \bar{a}\bar{b}\bar{c}de\ \bar{x}_{14}\ (b^1,\ c^1,\ 14^1,\ ;\ 21^0),\ (33,\emptyset),\ (2,5)$$

$$T_5(36) = \bar{a}\bar{b}cd\bar{e}\ \bar{x}_{14}\ (b^1,\ d^0,\ 14^1,\ ;\ 21^0),\ (34,\emptyset),\ (2,5)$$

$$T_5(37) = ab\bar{c}d\bar{e}\ \bar{x}_{14}\ (a^0,\ d^0,\ 14^1,\ ;\ 21^0),\ (35,\emptyset),\ (2,5)$$

$$T_5(38) = \bar{a}\bar{b}cde\ \bar{x}_{14}\ (b^1,\ 14^1,\ ;\ 21^0),\ (36,\emptyset),\ (2,5)$$

$$\chi_{T_5(39)} = a\bar{b}\bar{c}de\ \bar{x}_{14}\ (b^1,\ c^1,\ 14^1,\ ;\ 21^0),\ (38,\emptyset),\ (2,5)$$

$T_5(40) = a\bar{b}cd\bar{e}\ \bar{x}_{14}\ (d^0,\ 14^1,$; $21^0),\ (39,\emptyset),\ (2,5)$

$T_5(41) = a\bar{b}\bar{c}\bar{d}e\ \bar{x}_{14}\ (14^1,$; $21^0),\ (40,\emptyset),\ (2,5)$

$T_5(42) = \bar{a}\bar{b}cde\ \bar{x}_{14}\ (b^1,\ d^0,$; $21^0),\ (42,\emptyset),\ (2,6)$

$T_5(43) = ab\bar{c}de\ \bar{x}_{14}\ (a^0,\ d^0,$; $21^0),\ (43,\emptyset),\ (2,6)$

$T_5(44) = abcd\bar{e}\ \bar{x}_{14}\ (a^0,\ d^0,$; $21^0),\ (45,\emptyset),\ (2,6)$

$T_5(45) = abcde\ \bar{x}_{14}\ (a^0,\ d^0,$; $21^0),\ (45,\emptyset),\ (2,6)$

$T_5(46) = a\bar{b}cde\ \bar{x}_{14}\ (d^0,$; $21^0),\ (47,\emptyset),\ (2,6)$

$T_5(47) = \bar{a}\bar{b}cde\ x_{14}\ (b^1,\ d^0,\ e^0,\ c^0,$; $21^0),\ (50,E1),\ (2,7)$

$T_5(48) = ab\bar{c}de\ x_{14}\ (a^0,\ d^0,\ b^0,\ e^0,$; $21^0),\ (51,E2),\ (2,7)$

$T_5(49) = abcd\bar{e}\ x_{14}\ (a^0,\ d^0,\ b^0,\ c^0,$; $21^0),\ (53,E3),\ (2,7)$

$T_5(50) = abcde\ x_{14}\ (a^0,\ d^0,$; $21^0),\ (53,E4),\ (2,7)$

$T_5(51) = a\bar{b}cde\ x_{14}\ (d^0,\ e^0,\ c^0,$; $21^0),\ (55,E1),\ (2,7)$

$T_5(52) = ab\bar{c}\bar{d}\bar{e}\ x_{14}\ (\emptyset,$; $21^0),\ (57,\emptyset),\ (2,8)$

$T_5(53) = a\bar{b}cd\bar{e}\ x_{14}\ (\emptyset,$; $21^0),\ (58,\emptyset),\ (2,8)$

$T_5(54) = \bar{a}bc\bar{d}\bar{e}\ x_{14}\ (\emptyset,$; $21^0),\ (60,\emptyset),\ (2,8)$

$T_5(55) = \bar{a}\bar{b}cd\bar{e}\ x_{14}\ (\emptyset,$; $21^0),\ (62,\emptyset),\ (2,8)$

$T_5(56) = \bar{a}bc\bar{d}\bar{e}\ x_{14}\ (\emptyset,$; $21^0),\ (63,\emptyset),\ (2,8)$

From the PTI's of T_5, we see that T_5 is a sufficient test set for B_5. We note that in T_4 the following tests have not been used: 5, 27, 29, 32, 37, 41, 44, 46, 48, 49, 52, 54, 56, 59, 61, 64. We delete these in T_4 and check the PTI's of the remaining elements in T_4. It is easy to see that the PTI's of the remaining elements form a cover of the G/E tables of B_4 and contain all the elements in T_3. Therefore, the remaining set of T_4 still satisfies the sufficiency requirement.

Then we proceed to merge T_5 with the tests for B_6. We let

$$x_8 = a, x_9 = 6, x_{10} = e, x_{11} = f$$

$T_6(1) = \bar{a}\bar{b}cd\bar{e}\bar{f}$ $(b^1_1, e^1_1, b^1_1, e^1_1, f^1_1; 21^1)$, (2,E1), (2,1)

$T_6(2) = a\bar{b}\bar{c}d\bar{e}f$ $(b^1_1, c^1_1, a^0_1, b^1$; $21^1)$, (5,E3), (2,1)

$T_6(3) = a\bar{b}\bar{c}def$ $(b^1_1, c^1_1, a^0_1, b^1$; $21^1)$, (5,E5), (2,1)

$T_6(4) = a\bar{b}cd\bar{e}f$ $(b^1_1, e^1_1, a^0_1, b^1$; $21^1)$, (6,E2), (2,1)

$T_6(5) = a\bar{b}cd\bar{e}\bar{f}$ $(b^1_1, e^1_1, b^1$; $21^1)$, (6,E4), (2,1)

$T_6(6) = a\bar{b}\bar{c}d\bar{e}f$ $(b^1_1, c^1_1, a^0_1, b^1$; $21^1)$, (7,E3), (2,1)

$T_6(7) = a\bar{b}\bar{c}def$ $(b^1_1, c^1_1, a^0_1, b^1$; $21^1)$, (7,E5), (2,1)

$T_6(8) = ab\bar{c}\bar{d}ef$ $(b^0_1, d^1$; $21^1)$, (8,$\emptyset$), (2,2)

$T_6(9) = ab\bar{c}\bar{d}\bar{e}f$ $(b^0_1, d^1$; $21^1)$, (8,$\emptyset$), (2,2)

$T_6(10) = \bar{a}bcd\bar{e}\bar{f}$ $(a^1,b^0$; $21^1)$, (10,Ø), (2,2)

$T_6(11) = abc\bar{d}\bar{e}\bar{f}$ $(d^1$; $21^1)$, (13,Ø), (2,2)

$T_6(12) = abc\bar{d}\bar{e}\bar{f}$ $(d^1$; $21^1)$, (13,Ø), (2,2)

$T_6(13) = abc\bar{d}ef$ $(d^1$; $21^1)$, (14,Ø), (2,2)

$T_6(14) = abc\bar{d}e\bar{f}$ $(d^1$; $21^1)$, (14,Ø), (2,2)

$T_6(15) = \bar{a}\bar{b}cd\bar{e}\bar{f}$ $(c^0,d^1$; $21^1)$, (15,Ø), (2,2)

$T_6(16) = \bar{a}\bar{b}cde\bar{f}$ $(c^0,d^1$; $21^1)$, (15,Ø), (2,2)

$T_6(17) = \bar{a}bcd\bar{e}\bar{f}$ $(Ø$; $21^1)$, (17,Ø), (2,2)

$T_6(18) = a\bar{b}cd\bar{e}\bar{f}$ $(c^0,d^1$; $21^1)$, (20,Ø), (2,3)

$T_6(19) = a\bar{b}cde\bar{f}$ $(c^0,d^1$; $21^1)$, (20,Ø), (2,3)

$T_6(20) = ab\bar{c}d\bar{e}\bar{f}$ $(b^0,d^1$; $21^1)$, (30,Ø), (2,4)

$T_6(21) = ab\bar{c}\bar{d}e\bar{f}$ $(b^0,d^1$; $21^1)$, (30,Ø), (2,4)

$T_6(22) = \bar{a}bc\bar{d}\bar{e}\bar{f}$ $(a^1,b^0$; $21^1)$, (32,Ø), (2,4)

$T_6(23) = \bar{a}\bar{b}cd\bar{e}\bar{f}$ $(c^0,d^1$; $21^1)$, (33,Ø), (2,4)

$T_6(24) = \bar{a}bcd\bar{e}\bar{f}$ $(b^0$; $21^1)$, (33,Ø), (2,4)

$T_6(25) = \bar{a}\bar{b}\bar{c}de\bar{f}$ $(b^1,c^1,a^1,e^0$; $21^1)$, (35,G1),(2,5)

$T_6(26) = \bar{a}\bar{b}\bar{c}def$ $(b^1,c^1,a^1$; $21^1)$, (35,G3),(2,5)

$T_6(27) = \bar{a}\bar{b}cde\bar{f}$ $(b^1,d^0,a^1,f^0$; $21^0)$, (36,G3),(2,5)

$T_6(28) = ab\bar{c}d\bar{e}\bar{f}$ $(a^0,d^0,b^0$; $21^0)$, (37,G5),(2,5)

$T_6(29) = ab\bar{c}de\bar{f}$ $(a^0,d^0,b^0$; $21^0)$, (37,G7),(2,5)

$T_6(30) = \bar{a}\bar{b}cde\bar{f}$ $(b^1,a^1,e^0$; $21^0)$, (38,G1),(2,5)

$T_6(31) = \bar{a}\bar{b}cdef$ $(b^1,a^1$; $21^0)$, (38,G3),(2,5)

$T_6(32) = \bar{a}\bar{b}cdef\bar{}$ ($b^1_,d^o$; 21^o), (42,ø), (2,6)

$T_6(33) = \bar{a}\bar{b}cdef$ ($b^1_,d^o$; 21^o), (42,ø), (2,6)

$T_6(34) = ab\bar{c}def$ ($a^o_,d^o$; 21^o), (43,ø), (2,6)

$T_6(35) = ab\bar{c}def\bar{}$ ($a^o_,d^o$; 21^o), (43,ø), (2,6)

$T_6(36) = abcd\bar{e}f\bar{}$ ($a^o_,d^o$; 21^o), (44,ø), (2,6)

$T_6(37) = abcd\bar{e}f$ ($a^o_,d^o$; 21^o), (44,ø), (2,6)

$T_6(38) = abcdef$ ($a^o_,d^o$; 21^o), (45,ø), (2,6)

$T_6(39) = abcdef\bar{}$ ($a^o_,d^o$; 21^o), (45,ø), (2,6)

$T_6(40) = a\bar{b}cdef\bar{}$ ($d^o_,e^o_,c^o$; 21^o), (51,ø), (2,7)

$T_6(41) = a\bar{b}cdef$ ($d^o_,e^o_,c^o$; 21^o), (51,ø), (2,7)

$T_6(42) = a\bar{b}cd\bar{e}f\bar{}$ (ø ; 21^o), (53,ø), (2,8)

$T_6(43) = a\bar{b}cd\bar{e}f$ (ø ; 21^o), (53,ø), (2,8)

$T_6(44) = \bar{a}\bar{b}cd\bar{e}f\bar{}$ (ø ; 21^o), (55,ø), (2,8)

It is easy to check that G4 or G6 of B_6 are needed, but in T_6 neither is included in the PTI. The following elements of T_5 have not been merged: 1, 3, 4, 9, 11, 12, 16, 18, 19, 21, 22, 23, 24, 25, 27, 28, 29, 31, 34, 39, 40, 41, 46, 47, 48, 49, 50, 52, 54, 56 and must be eliminated from T_5. Then we check the PTI's of the surviving elements in T_5 and discover that G2, E2, E3 of B_5 must be put into the missing list.

Then we go on checking the sufficiency of T_4. First

notice that in T_4 the following elements should be removed: 1, 3, 4, 10, 12, 15, 17, 19, 20, 21, 22, 23, 24, 26, 31, 38, 39, 40, 47, 50, 51, 53, 57, 60, 63. It is also easy to check that the surviving elements of T_4 are still a sufficient test set for B_4, and contain every element of T_3.

Thus the complete missing list in $\{$(5,G2), (5,E2), (5,E3), (6,G4), (6,G6)$\}$.

G Catch-up Process

It is apparent that the above described merging process has certain similarities to the D-algorithm (Refs. 4, 9). Our growth and existence tables are like a "D-table", i.e., we have a D-cover for our blocks, which are "super-gates". (Note the similarity between the CL and the D symbol.) Unlike finding D-covers, our table entries are computed without tracing explicitly for sensitive paths. In the merging process, however, because we choose not to merge with all possible inputs to a block, but only with the test-set for that block, we may fail to find essential tests. It is the purpose of the catch-up process to compute those necessary tests that were missed.

First, we consider the question of when to perform the catch-up process. Several possibilities exist. Each one works particularly well for certain networks, and the issue is related to local minimization versus global minimization.

Approach 1 would catch up immediately after each

merging process when a new block has been processed. Approach 2 catches up only after all the blocks of the current level have been processed. Approach 3 catches up at the very end of the merging process. Approach 2 is a compromise and was selected here. However, any of the three approaches can be fitted into our overall algorithm. A switch could be built into the program and used to select the appropriate one for a given network, after enough experience to make a good decision has been obtained.

In catching up, we can try to establish a single sensitized path from the block with the missing test to the output. If we are not successful, we can then try to find a double sensitized path, and so on. On the other hand, we can use the D-algorithm to drive the test to the output.

The first approach (i.e., trying a single path first) will work much faster for a network with a small number of fanouts. For a network with a lot of reconvergent fanout, it may happen that certain tests can only be found by sensitizing a great many paths simultaneously. If we use the first approach, the cost of generating such tests would be particularly high, and the D-algorithm (i.e., the second approach) would work better for this particular case. However, it is true that such situations are rare, and so we expect to have a higher average performance by using the first approach. Note that in our running example

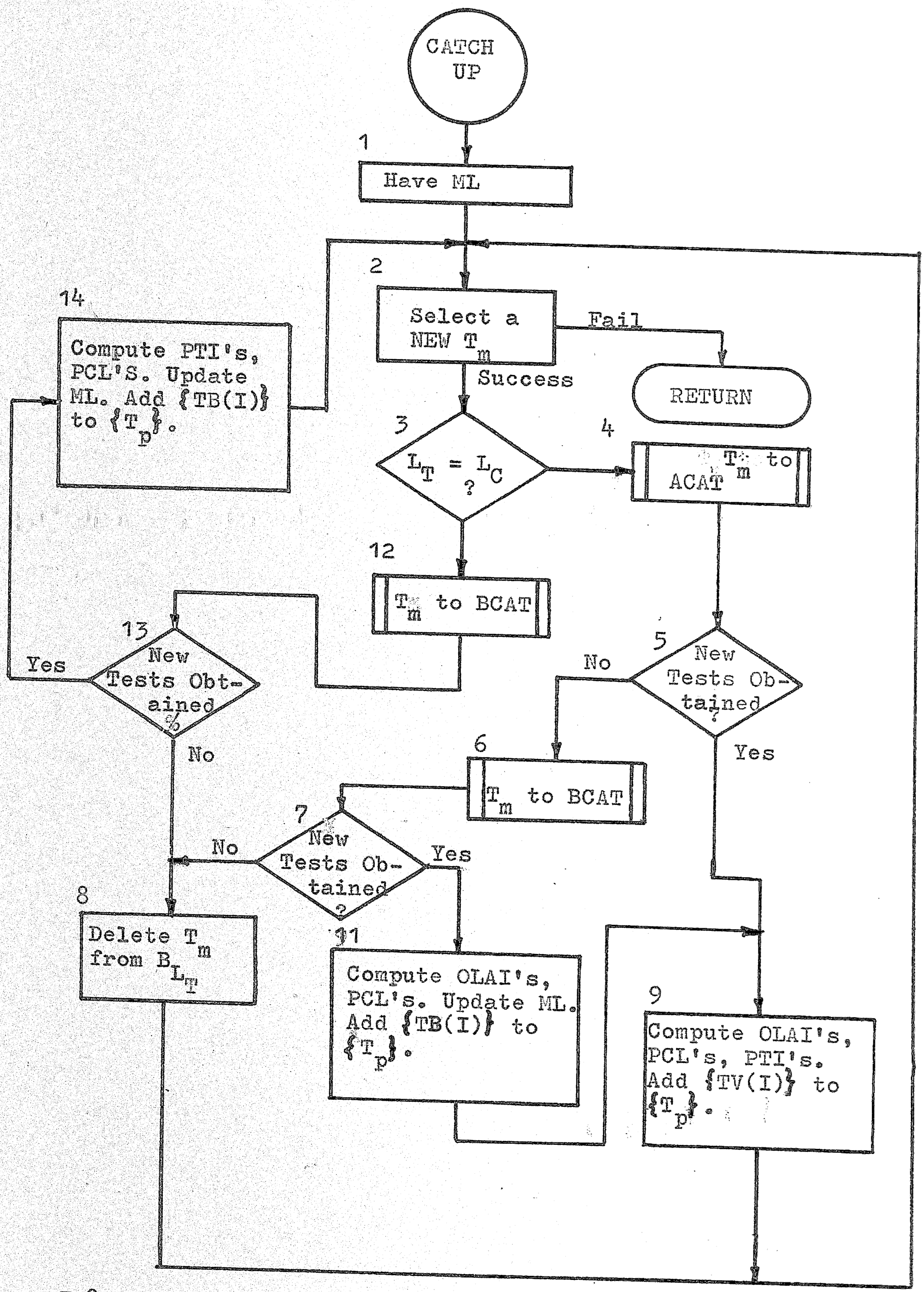

CATCH UP
1
Have ML
2
Select a NEW T_m
Fail
RETURN
Success
3
$L_T = L_C$?
4
T_m to ACAT
12
T_m to BCAT
5
New Tests Obtained?
No
Yes
6
T_m to BCAT
7
New Tests Obtained?
No
Yes
8
Delete T_m from B_{L_T}
11
Compute OLAI's, PCL's. Update ML. Add {TB(I)} to $\{T_p\}$.
9
Compute OLAI's, PCL's, PTI's. Add {TV(I)} to $\{T_p\}$.
13
New Tests Obtained?
Yes
No
14
Compute PTI's, PCL'S. Update ML. Add {TB(I)} to $\{T_p\}$.

Figure 3.8

it happens to be the case that all the tests that are on the missing list in fact correspond to undetectable faults in the partitioned network. So it happens that our merging process is pretty good in this particular example.

Our overall catch-up process consists of two main parts. One of them finds the inputs that should be applied at level i in order to have some specified signal pattern on leads at the input to blocks at level j, where level j is down-stream from level i. The other part finds the inputs at level i so that there is a sensitized path from the desired output leads of the blocks at level i to some observable output lead. We name the first process ACAT, and the second BCAT.

Algorithm CATCHUP:

1) Store all the tests still needed to be caught up in ML (missing list) and arrange them in ascending order according to the level number of the block they test, i.e., tests for the lowest level number first.

2) Select a new element from the top of ML (i.e., an element that has not been selected before) and name it T_m. If all the elements in ML have been selected before, then exit from the CATCHUP process. Otherwise go to the next step.

3) If T_m corresponds to a block at the current level in the merging process, then go to step 12. We name the block associated with T_m the "trouble-block". We use L_T

to denote the level number of the trouble-block and L_C to denote the current level number in the merging process. So this step is equivalent to asking the question "Is L_T equal to L_C?"

4) Find the set of all tests in the test set found at the end of the level L_T that includes the test T_m. Pass these elements to the process ACAT and execute ACAT.

5) If ACAT in step 4 finds no test, then go to step 6. Otherwise go to step 9.

6) Pass T_m to the process BCAT and execute BCAT.

7) If BCAT in step 6 cannot obtain any test, then go to step 8. Otherwise go to step 11.

8) Delete T_m from the test-set of the trouble-block and output the information that T_m is related to undetectable faults according to the way we partitioned the network. Then go to step 2.

9) Add the TV(I)'s* computed by ACAT to each of the corresponding test sets T_p and WTB(L_c)* to the current test set which has been found in the merging process. Update the associated OLAI's. Their PTI's are left empty. The I-part of each PCL is also left empty, but the 0 part of each PCL is the same as the 0 part of

*Refer to page 76.

$TV(L_T)$ successfully used in ACAT. (If $TV(L_T)$ is one of the tests that appeared in the working test set before, its PTI should be restored.)

10) Go to step 2.

11) Add TB(I)** found in process BCAT to the corresponding test set found in the merging process. (I should be equal to or greater than the level number of the trouble-block.) Set the I part of the associated PCL of each element in TB(I) equal to TCL(I) and the O part determined by TB(I) for I equals the output level number because it is also a test for the block at the output level. Update the associated OLAI's. Eliminate the elements in the missing list that are contained in the TB(I)'s and append to each TB(I) an ordered pair indicating the block and test number that it serves. Go to step 9.

12) Pass T_m to the process BCAT and execute BCAT.

13) If we cannot obtain any test from step 12, go to step 8.

14) Add $WBT(L_C)$ found in process BCAT to the current test set and the TB(I)'s to their corresponding test sets. Set the I part of the associated PCL of each element in TB(I) equal to TCL(I), and the O part to the TB(I) of the output level. Any element still in ML and contained in $\{TB(I)\}$ should be deleted from ML. Append

**Refer to page 78.

to each TB(I) an ordered pair indicating the block and test number that it serves. Then go to step 2.

In order to improve the speed of the process ACAT, we will assume that not only do we have the F-expression for each block, but also the $\bar{F}$-expression for each block, where $\bar{F}$ is the complement of the block response. This assumption is also made to facilitate the explanation of the algorithm. Process ACAT;

1) Set index I equal to the level number of the trouble-block, i.e., L_T.

2) Denote the set of elements passed on from the main program by WTB(I). [The elements in WTB(I) are the patterns of signal values at the input to level L_T that would result in the missing test.]

3) Select a new element from WTB(I) i.e., select an element of WTB(I) that has not been selected before, and name it TV(I). If all the elements in WTB(I) have been selected before, go to step 8.

4) In TV(I) replace each 1(0) by the corresponding F($\bar{F}$) expression for the logic in level (I-1), expressed in terms of the lead variables at the output of the blocks at level (I-2). Check the expressions for consistency, i.e., determine that all of the specified logic values can be applied to the input of level I with a single value for each output from level (I-2).

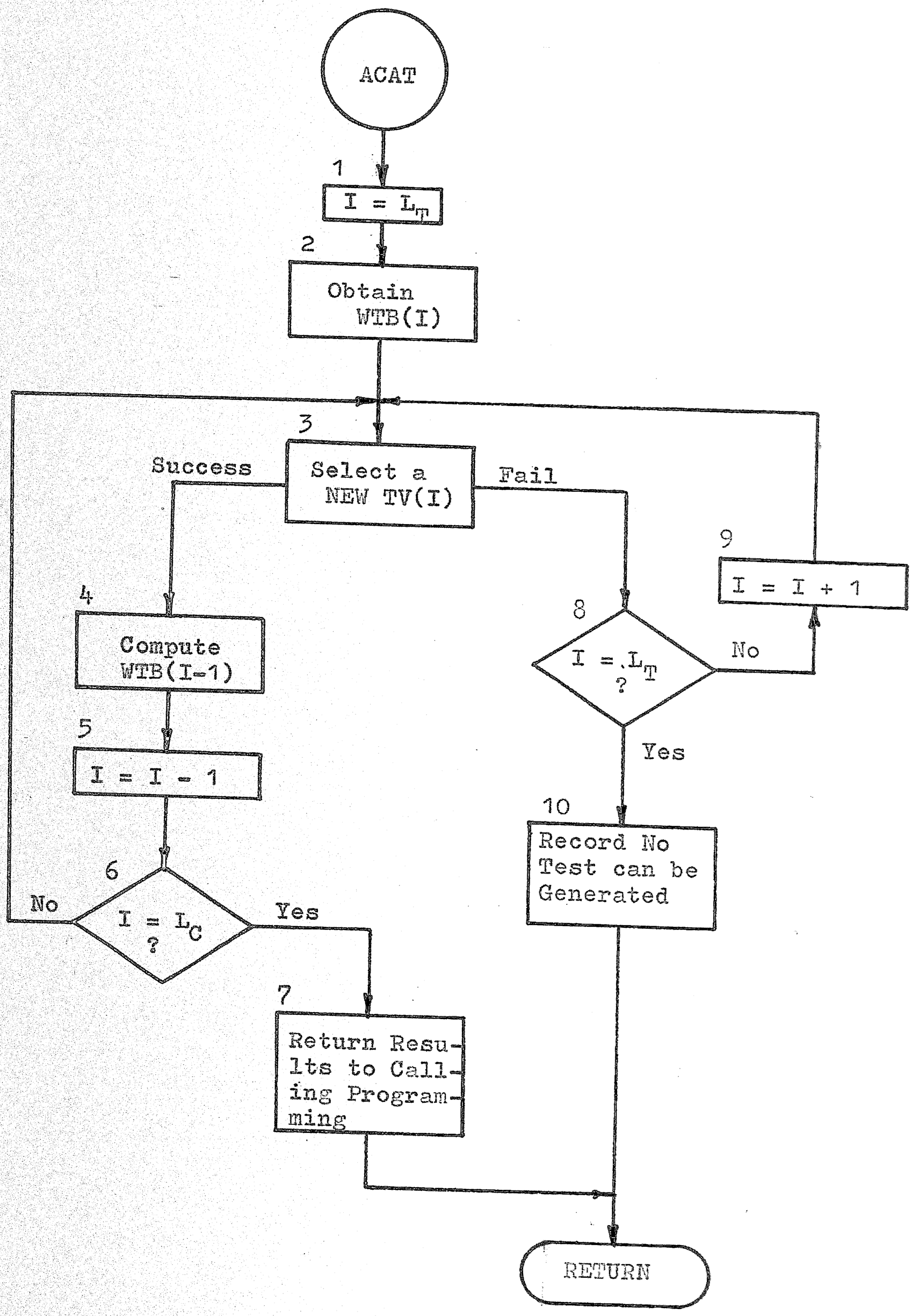

ACAT
1
I = L_T
2
Obtain WTB(I)
3
Select a NEW TV(I)
Success
Fail
4
Compute WTB(I-1)
5
I = I - 1
6
I = L_C ?
No
Yes
7
Return Results to Calling Programming
8
I = L_T ?
No
Yes
9
I = I + 1
10
Record No Test can be Generated
RETURN

Figure 3.9

[Example: Let p=q=1 be required, and p=ab and q=$\bar{a}$ Then there is an inconsistency, because no one value for a can make p=q=1.] Record consistent values for signals at level (I-2) in WTB I-1.

5) Decrease I by one.

6) If I equals the current level number (i.e., I = L_C), go to step 7. Otherwise go to step 3.

7) Return WTB(L_C) and the corresponding TV(I) for I = Lc+1 to L_T to the calling program, and exit from the process.

8) If I equals L_T, then go to step 10.

9) Increase the value of I by one. Then go to step 3.

10) Inform the main program that no test can be generated, and exit from the process.

The process BCAT is more complicated. The idea used is somewhat like the D-algorithm, i.e., we look forward to seek a sensitized path first, then backward to find consistent input values that provide the selected sensitized path. Process BCAT:

1) Let index I equal the level number of the trouble-block, i.e., L_T.

2) Store in TA(I) the missing input pattern at L_T which is passed on by the main program.

3) If I equals the output level number, go to step 28.

4) Find the set of leads at level (I + 1) that are connected to the output of the trouble-block. Denote the resulting

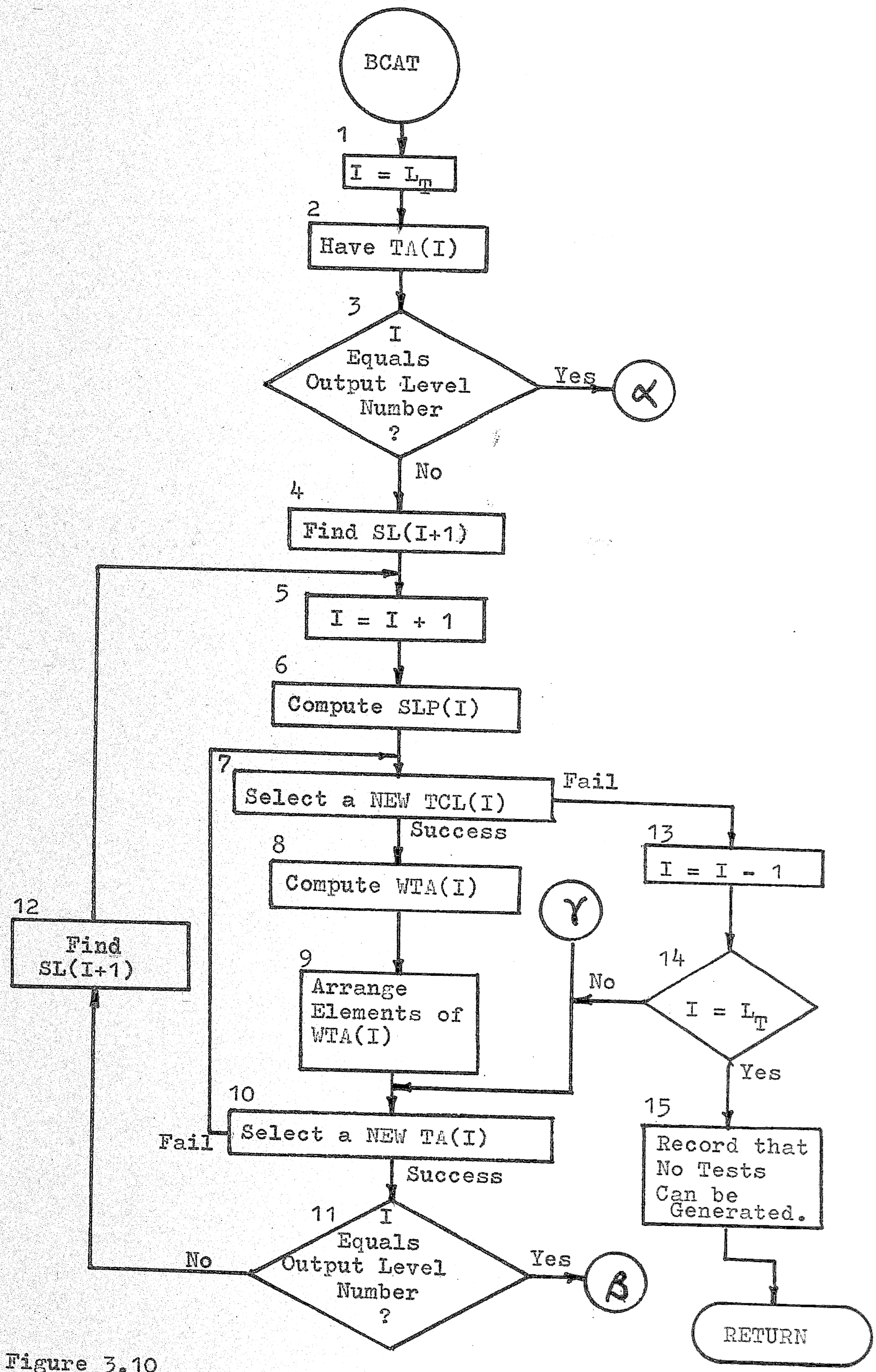

BCAT
1
I = L_T
2
Have TA(I)
3
I Equals Output Level Number ?
Yes
α
No
4
Find SL(I+1)
5
I = I + 1
6
Compute SLP(I)
7
Select a NEW TCL(I)
Fail
Success
8
Compute WTA(I)
γ
9
Arrange Elements of WTA(I)
10
Select a NEW TA(I)
Fail
Success
11
I Equals Output Level Number ?
No
Yes
β
12
Find SL(I+1)
13
I = I - 1
14
I = L_T
No
Yes
15
Record that No Tests Can be Generated.
RETURN

Figure 3.10

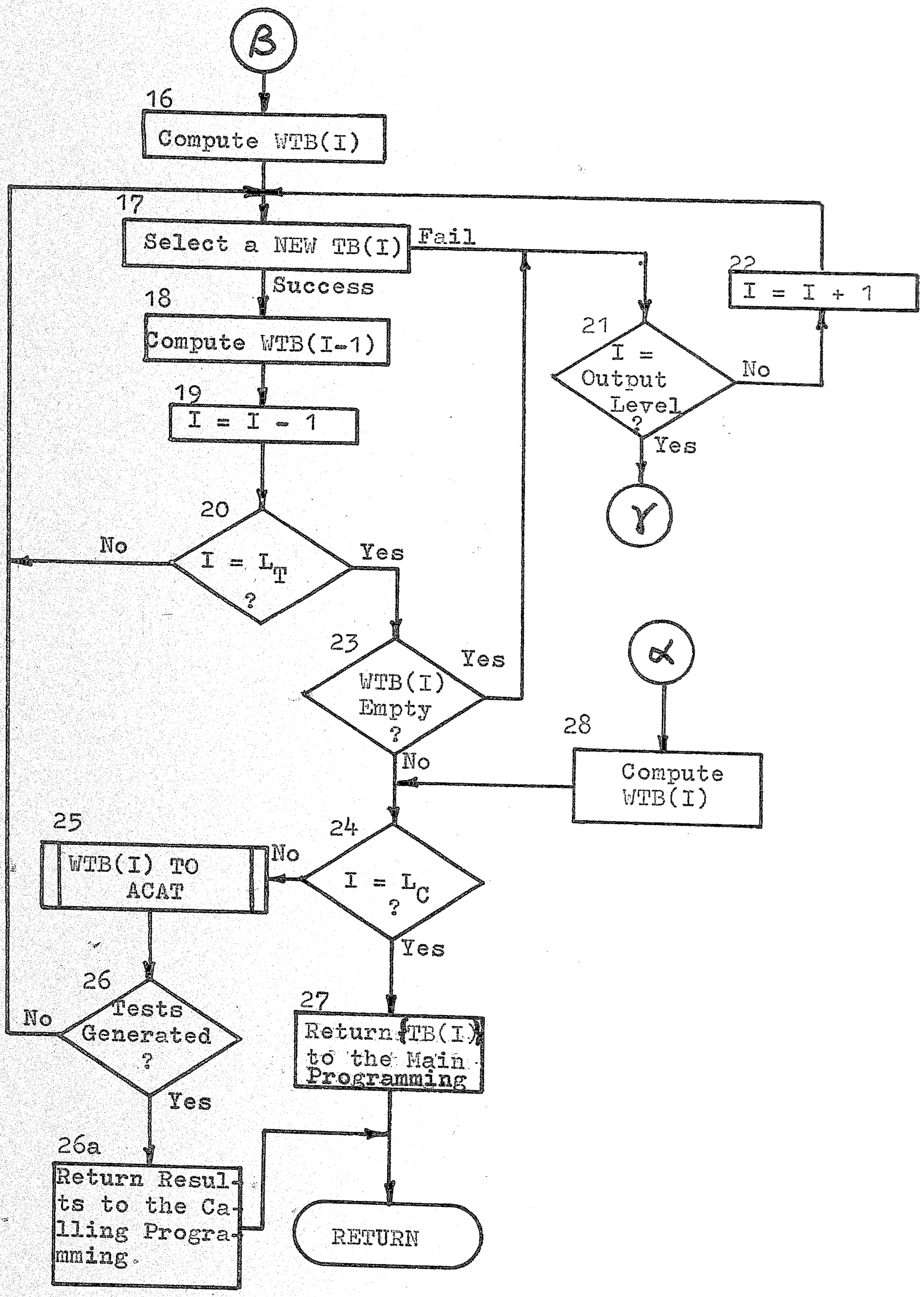


Figure 3.10 cont.

set by SL(I + 1). (We use SL as mnemonics for sensitive leads. The elements of SL(I + 1) are superscripted with 1(0) if that lead is sensitized to stick at 1(0).

5) Increase the value of I by one.

6) Form all the subsets of SL(I), putting them into SLP(I). Arrange the elements of SLP(I) in ascending order according to their size.

7) Select a new element from SLP(I) and name it TCL(I). If all the elements of SLP(I) have been previously selected, then go to step 13.

8) For the current TCL(I), determine the set WTA(I) of inputs to the blocks at level I that make changes in the signal values of the leads in TCL(I) which result in changes in the outputs of level (I+1). [If only one path is sensitized (i.e., TCL(I) is a one-literal element), then select those inputs to the sensitive block which has the element of TCL(I) contained in the I-part of its associated PCL. If more than one block has been sensitized, (i.e., TCL(I) has more than one literal, and each literal corresponds to a different block), one selects inputs such that they satisfy each sensitive block and are mutually consistent. If two (or more) sensitized paths are passing through the same block, then the desired inputs of the sensitized block cannot be found by just looking at the PCL's of the tests of the sensitized blocks. Instead, the desired inputs

are those which simultaneously satisfy both $\bar{F}$ and the expression for F <u>with the sensitized leads complemented</u>. This condition is the "multiple Boolean difference". It assures that when the sensitized leads are stuck, the output changes value from the normal situation.]

9) Arrange the elements of WTA(I) into different groups such that each group sensitizes identical lead patterns at the output of level (I + 1).

10) Select a <u>new</u> group from WTA(I), and name it TA(I). If no group is available, go to step 7.

11) If I equals the output level number, then go to step 16.

12) Find the set of leads at level (I + 1) that are connected to the sensitive output leads of blocks at level I (i.e., are connected to the leads that are sensitized by TA(I)). Denote the resulting set by SL(I+1). Then go to step 5.

13) Decrease the value of I by one.

14) If I equals the level number of the trouble block, then go to step 15. Otherwise, go to step 10.

15) Inform the main program that TA(I) (i.e., T_m in the main program) corresponds to some undetectable faults in our partitioning scheme. Then exit from the process.

16) Find the set of inputs to the blocks at the output level that has the sensitized pattern as TCL(I) subject to the constraints of TA(I). Name the resulting set WTB(I).

17) Select a new element from WTB(I) and name it TB(I). If

none is available, go to step 21. Otherwise, go to the next step.

18) Solve for the lead values on level (I-1) that satisfy both TB(I) and TA(I-1) as computed before. The resulting solutions are placed in WTB(I-1).

19) Decrease the value of I by one.

20) If the value of I equals L_T, (i.e., the level number of the trouble-block) then go to step 23. Otherwise go to step 17.

21) If the value of I equals the level number of the output blocks, then go to step 10.

22) Increase the value of I by one. Then go to step 17.

23) If WTB(I) is empty, then go to step 21.

24) If the value of I equals L_C (i.e., the current level number), then go to step 27.

25) Pass the set WTB(I) to ACAT.

26) If a test was generated, then return the sets $\{TV(I), I = L_C + 1 \text{ to } L_T\}$, $WTB(L_C)$, $\{TB(I); I = L_T \text{ to output level}\}$, to the main program, then exit from the process. If no test was found by the ACAT process, then go to step 17.

27) Pass the set $\{TB(I); I = L_C \text{ to output level number}\}$ to the main program, then exit from the process.

28) Find the set of inputs to the blocks at the output level, subject to the constraints of TA(I). Denote the resulting set by WTB(I). Then go to step 24.

Now we may continue the example previously started. For later convenience we list all the F and $\bar{F}$ expressions of each block here.

For block 1,

$$F = x_{21} = x_{19}\bar{x}_{20} + \bar{x}_{19}x_{20}$$

$$\bar{F} = \bar{x}_{21} = \bar{x}_{19}\bar{x}_{20} + x_{19}x_{20}$$

For block 2,

$$F = x_{19} = x_{15}\bar{x}_{16} + \bar{x}_{15}x_{16}$$

$$\bar{F} = \bar{x}_{19} = \bar{x}_{15}\bar{x}_{16} + x_{15}x_{16}$$

For block 3,

$$F = x_{20} = x_{17} + x_{14}x_{18}$$

$$\bar{F} = \bar{x}_{20} = \bar{x}_{14}\bar{x}_{17} + \bar{x}_{17}\bar{x}_{18}$$

For block 4,

$$F = x_{12} = x_1x_2x_4 + \bar{x}_2\bar{x}_3 + \bar{x}_2x_4$$

$$\bar{F} = \bar{x}_{12} = \bar{x}_1x_2 + x_2\bar{x}_4 + x_3\bar{x}_4$$

For block 5,

$$F = x_{13} = x_5x_6 + x_6x_7 + x_7x_5$$

$$\bar{F} = \bar{x}_{13} = \bar{x}_5\bar{x}_6 + \bar{x}_6\bar{x}_7 + \bar{x}_7\bar{x}_5$$

For block 6,

$$F = x_{14} = x_8\bar{x}_9x_{10} + x_8\bar{x}_9x_{11} + \bar{x}_9\bar{x}_{10}\bar{x}_{11}$$

$$\bar{F} = \bar{x}_{14} = x_9 + \bar{x}_8x_{10} + \bar{x}_8x_{11}$$

Note that the F-expressions listed here are not exactly the same as those listed before, but are "reduced" E-expressions because we may now perform any kind of simplification.

This is so because we use F and $\bar{F}$ expressions only to represent the set of true (or false) vertices of a given block, and not to find tests.

From the previous work, we have the missing list $\{(5,G2), (5,E2), (5,E3), (6,G4), (6G6)\}$. Because all the missing tests are in the same level, we arbitrarily select the first element in ML, which makes $T_m = (5,G2)$. Continuation of the working example,

Because block 5 is at the current level in the merging process, we pass T_m to the process BCAT.

Because the level number of block 5 is 1, we get I = 1.

$$TA(1) = (5,G2) = x_5\bar{x}_6\bar{x}_7\ (6^1_{,}7^1_{;}13^1) = b\ \bar{c}\ \bar{e}$$

$$SL(2) = L^1_{13} = \{L^1_{16}, L^1_{17}\}$$

$$I = 2$$

$$SLP(2) = \{(L^1_{16}),\ (L^1_{17}),\ (L^1_{16}, L^1_{17})\}$$

First select (L^1_{16}), thus $TCL(2) = (L^1_{16})$

$$WTA(2) = \{x_{15}\bar{x}_{16}\ (15^0_{,}\ 16^1;\ 19^0);\ \bar{x}_{15}\bar{x}_{16}(15^1,\ 16^1;\ 19^1)\}$$

$$TA(2) = \{x_{15}\bar{x}_{16}\}$$

$$SL(3) = \{L_{19}{}^0\}$$

$$I = 3$$

$$SLP(3) = \{(L_{19}{}^0)\}$$

$TCL(3) = (L_{19}^{0})$

$WTA(3) = \{x_{19}\, x_{20}\, (19^{0}, 20^{0}; 21^{1}),\ x_{19}\bar{x}_{20}\, (19^{0}, 20^{1}; 21^{0})\}$

$TA(3) = \{x_{19}x_{20}\}$

Now I equals the output level number.

$WTB(3) = \{x_{19}x_{20}\}$

$TB(3) = \{x_{19}x_{20}\}$

At this time, $TCL(3) = (L_{19}^{0})$, so lead L_{19} is on the sensitized-path, but not lead L_{20}.

$WTB(2) = TA(2) \cap F_3$

$$= \{x_{15}\bar{x}_{16}\} \cap \{x_{17} + x_{14}x_{18}\}$$

$$= \{x_{12}\bar{x}_{13}\} \cap \{x_{13} + x_{14}x_{12}\} = \{x_{12}\bar{x}_{13}x_{14}\}$$

Checking block 3, it is easy to find out that lead L_{17} is also sensitized by the element in WTB(2). But our TCL(2) required that only L_{16} be sensitized. Thus the element of WTB(2) should be deleted, i.e., WTB(2) is empty.

Since WTB(2) is empty, we cannot go upstream further.

I is not equal to the output level number (i.e. 3,) so we increase the value of I by one.

I = 3

The one element of WTB(3) has already been selected, so we fail to have a <u>new</u> element from WTB(3).

At this time, I is equal to the value of the output level number (i.e. 3).

Then we try to select a new element from WTA(3). We succeed

$$TA(3) = \{x_{19}\ \bar{x}_{20}\}$$

$$WTB(3) = \{x_{19}\bar{x}_{20}\}$$

$$TB(3) = \{x_{19}\bar{x}_{20}\}$$

$$WTB(2) = \{x_{15}\bar{x}_{16}\} \cap \bar{F}_3$$

$$= \{x_{15}\bar{x}_{16}\} \cap \{\bar{x}_{14}\bar{x}_{17} + \bar{x}_{17}\bar{x}_{18}\}$$

$$= \{x_{12}\bar{x}_{13}\} \cap \{\bar{x}_{14}\bar{x}_{13} + \bar{x}_{13}\bar{x}_{12}\}$$

$$= \{x_{12}\bar{x}_{13}\bar{x}_{14}\}$$

Checking block 3, it is easy to find out that lead L_{17} is also sensitized by both elements in WTB(2). But our TCL(2) required that only L_{16} be sensitized. Thus both of the elements of WTB(2) should be deleted, i.e., WTB(2) is empty.

So we back-track again several steps.

Because all the elements of WTA(3) have already been selected before, we try to select a new element from SLP(3), but fail again.

Then the value of I is decreased by one. I = 2.

I is not equal to the level number of the trouble block, and WTA(2) is not yet empty. We have $TA(2) = \{\bar{x}_{15}\bar{x}_{16}\}$

$SL(3) = \{L_{19}^{1}\}$

$I = 3$

$SLP(3) = \{(L_{19}^{1})\}$

$TCL(3) = (L_{19}^{1})$

$WTA(3) = \{\bar{x}_{19}\bar{x}_{20}\ (19^1, 20^1; 21^1), \bar{x}_{19}x_{20}(19^1, 20^0; 21^0)\}$

$TA(3) = \{\bar{x}_{19}\bar{x}_{20}\}$

Now I equals the output level number.

$WTB(3) = \{\bar{x}_{19}\bar{x}_{20}\}$

$TB(3) = \{\bar{x}_{19}\bar{x}_{20}\}$

At this time, $TCL(3) = (L_{19}^{1})$, so lead L_{19} is on the sensitized-path, but not lead L_{20}.

$WTB(2) = TA(2) \cap \{\bar{F}_3\}$

$= \{\bar{x}_{15}\bar{x}_{16}\} \cap \{\bar{x}_{16}\bar{x}_{17} + \bar{x}_{17}\bar{x}_{18}\}$

$= \{\bar{x}_{12}\bar{x}_{13}\} \cap \{\bar{x}_{16}\bar{x}_{13} + \bar{x}_{13}\bar{x}_{12}\}$

$= \{\bar{x}_{12}\bar{x}_{13}\bar{x}_{14}, \bar{x}_{12}\bar{x}_{13}x_{14}\}$

Checking block 3, it is easy to find out again that lead L_{17} is also sensitized by both elements in WTB(2). But our TCL(2) required that only L_{16} be sensitized. Thus both of the elements of WTB(2) should be deleted, i.e. WTB(2) is empty.

This time, we have to back track up to the point of selecting a new element from SLP(2) (i.e. step 7 in BCAT).

$TCL(2) = (L_{17}^{1})$

$WTA(2) = \{x_{16}\bar{x}_{17}\bar{x}_{18}\ (17^{1}, 18^{1}; 20^{1}), \bar{x}_{14}\bar{x}_{17}x_{18}(14^{1}, 17^{1}; 20^{1});$

$\bar{x}_{14}\bar{x}_{17}\bar{x}_{18}(17^{1}; 20^{1})\}$

$TA(2) = \{x_{14}\bar{x}_{17}\bar{x}_{18}, \bar{x}_{14}\bar{x}_{17}x_{18}, \bar{x}_{14}\bar{x}_{17}\bar{x}_{18}\}$

because they have the same output, i.e., signal on lead 20 is 0.

$SL(3) = \{L_{20}^{1}\}$

$I = 3$

$SLP(3) = \{(L_{20}^{1})\}$

$TCL(3) = (L_{20}^{1})$

$WTA(3) = \{\bar{x}_{19}\bar{x}_{20}(19^{1}, 20^{1}; 21^{1}); x_{19}\bar{x}_{20}(19^{0}, 20^{1}; 21^{0})\}$

$TA(3) = \{\bar{x}_{19}\bar{x}_{20}\}$

At this time, I equals the output level number.

$WTB(3) = \{\bar{x}_{19}\bar{x}_{20}\}$

$TB(3) = \{\bar{x}_{19}\bar{x}_{20}\}$

Because TCL(3) is (L_{20}^{1}), lead L_{20} is on the sensitized path, but not lead L_{19}.

$$WTB(2) = TA(2) \cap \bar{F}_2$$

$$= \{\bar{x}_{14}\bar{x}_{17} + \bar{x}_{17}\bar{x}_{18}\} \cap \{\bar{x}_{15}\bar{x}_{16} + x_{15}x_{16}\}$$

$$= \{\bar{x}_{14}\bar{x}_{13} + \bar{x}_{13}\bar{x}_{12}\} \cap \{\bar{x}_{12}\bar{x}_{13} + x_{12}x_{13}\}$$

$$= \{\bar{x}_{12}\bar{x}_{13}\}$$

$$= \{\bar{x}_{12}\bar{x}_{13}\bar{x}_{14},\ \bar{x}_{12}\bar{x}_{13}x_{14}\}$$

Because lead L_{16} also happens to be sensitized by $\bar{x}_{12}\bar{x}_{13}$, WTB(2) is empty.

Again we back track and set

$$TA(3) = \{x_{19}\bar{x}_{20}\}$$

$$WTB(3) = \{x_{19}\bar{x}_{20}\}$$

$$TB(3) = \{x_{19}\bar{x}_{20}\}$$

$$WTB(2) = TA(2) \cap F_2$$

$$= \{\bar{x}_{14}\bar{x}_{17} + \bar{x}_{17}\bar{x}_{18}\} \cap \{x_{15}\bar{x}_{16} + \bar{x}_{15}x_{16}\}$$

$$= \{\bar{x}_{14}\bar{x}_{13} + \bar{x}_{13}\bar{x}_{12}\} \cap \{x_{12}\bar{x}_{13} + \bar{x}_{12}x_{13}\}$$

$$= \{x_{12}\bar{x}_{13}\bar{x}_{14}\}$$

But lead L_{16} of block 3 is also sensitized by $x_{12}\bar{x}_{13}\bar{x}_{14}$, so WTB(2) is empty.

We have to back-track again, and have $TCL(2) = (L_{16}^{1}, L_{17}^{1})$ i.e. both paths should be sensitized simultaneously.

$$WTA(2) = \{\bar{x}_{15}\bar{x}_{16} + x_{15}\bar{x}_{16}\} \cap \{x_{14}\bar{x}_{17}\bar{x}_{18} + \bar{x}_{14}\bar{x}_{17}\bar{x}_{18}\}$$

$$= \{\bar{x}_{12}\bar{x}_{13}x_{14},\ \bar{x}_{12}\bar{x}_{13}\bar{x}_{14}\}$$

$$TA(2) = \{\bar{x}_{12}\bar{x}_{13}x_{14},\ \bar{x}_{12}\bar{x}_{13}\bar{x}_{14}\}$$

$$SL(3) = \{L^1_{19},\ L^1_{20}\}$$

It is easy to check that when both input leads of block 1 change, its output response does not change (i.e., the intersection obtained in step 8 of BCAT is empty).

Now we have to back-track all the way to the starting point and find that test (5,G2) cannot be executed.

For brevity, we do not describe catch-up process for the other elements in the missing list. It happens that in this example <u>all</u> the elements in the missing list cannot be applied.

Now we can select a complete single-fault detection test-set from the set T_6 found in the merging process.

Now we select a complete single-fault detection test-set from the set T_6 found in the merging process.

H. The method presented below will give us a nearly minimum test set.

Selection process:

1) Let I equal the highest block number.

2) Build a table with (I + k) columns. Each of the first I-columns corresponds to each block, where the largest

block-number corresponds to the first column. The remaining columns correspond to the k observable output terminals. (In our example, k equals one). The row labels correspond to the tests that will be selected. [These k columns tell what the normal outputs will be].

3) If the elements in column B_I are still not a sufficient test for block I, starting from the beginning of T_I, select a minimum number of tests from T_I such that a sufficient test set for block-I results (make use of the PTI's). Then add as row labels those tests not already listed.

   [Using the PTI's we can trace easily to the output (and/or input) and determine which blocks have been tested for what.]

4) If I equal to zero (i.e. if all blocks have been checked), then exit from the process.

5) Decrease the value of I by one. Then go to step 3. Now we may complete our working example by finding a minimum test set from T_6.

The table we build is in Fig. 3.11. The 11 input vertices listed form a single fault test set for our partitioned network.

INPUTS	B6		B5		B4		B3		B2		B1	OUTPUT
$\bar{a}\bar{b}cd\bar{e}\bar{f}$	E_1	(2)	G_3	(2)	ø	(1)	E_1	(1)	E_1	(1)	G_1	0
$a\bar{b}\bar{c}\bar{d}e\bar{f}$	E_3	(5)	G_1	(6)	ø	(1)	E_1	(1)	E_1	(1)	G_1	0
$a\bar{b}cd\bar{e}f$	E_2	(6)	G_3	(7)	ø	(1)	E_1	(1)	E_1	(1)	G_1	0
$\bar{a}\bar{b}\bar{c}\bar{d}e\bar{f}$	G_1	(35)	ø	(33)	E_1	(5)	B_2	(4)	E_1	(3)	E_1	1
$\bar{a}\bar{b}cd\bar{e}f$	G_2	(36)	ø	(34)	E_2	(5)	G_2	(4)	E_1	(3)	E_1	1
$ab\bar{c}d\bar{e}\bar{f}$	G_5	(37)	ø	(35)	E_3	(5)	G_2	(4)	E_1	(3)	E_1	1
$a\bar{b}cde\bar{f}$	ø	(51)	E_1	(55)	E_7	(7)	E_2	(5)	G_1	(4)	E_2	1
$ab\bar{c}\bar{d}ef$	ø	(8)	ø	(9)	G_1	(2)	E_3	(1)	E_1	(1)	G_1	0
$\bar{a}bcd\bar{e}\bar{f}$	ø	(10)	ø	(11)	G_3	(2)	E_3	(1)	E_1	(1)	G_1	0
$a\bar{b}cd\bar{e}\bar{f}$	ø	(20)	ø	(18)	G_2	(3)	E_4	(2)	E_2	(1)	G_1	0
$a\bar{b}cd\bar{e}\bar{f}$	ø	(53)	ø	(58)	ø	(8)	G_1	(3)	G_2	(2)	G_2	0

Figure 3.11

I. Discussion

While it is expected that most faults in a network can be detected by sensitizing only a single path (Ref. 10, section 3), it is true that reconvergent fanout is troublesome in the well-known D-algorithm. Here we have an approach where reconvergent fanout is less troublesome, because whenever it is contained entirely within a block, it is easily and automatically taken care of by the E-algorithm. Also, since we enlarge the basic unit from a single gate to a block of many gates, we significantly reduce the number of possible test combinations that have to be considered during the "consistency operation" of combining tests from different blocks. For these two reasons it is believed that the approach discussed in this thesis is an improvement over others.

By considering the working example in this chapter, one can get some feeling as to the power of the merging process, even though it is coincidental that 100% of the required tests are obtained in this case in the merging process and no additional tests are found in the catchup process.

The catch-up process is essentially equal to the D-algorithm. The notation we employ here will cause some inefficiency in computation by a computer, compared to the original D-algorithm. On the other hand, the basic unit here is a supergate (a subnetwork) instead of a basic (primitive) gate, and this will hopefully balance out the inefficiency of the catch-up process.

After more experience is obtained, variations in the merging process can be made. For example, consider what happens when we merge $T_{p-1}(i)$ with the test-set of block B_p. Recall that if the PCL of $T_{p-1}(i)$ does not contain the output lead number of block B_p, the result so obtained will not be a test for block B_p. Except when there are some other output lead numbers of other blocks at the same level and these appear in the

PCL of $T_{p-1}(i)$, it will in general not help us a lot in the overall process if all the tests of block B_p are used to merge with $T_{p-1}(i)$. Anyhow, modifications can be made after more experience has been gained with the algorithm.

Chapter IV Conclusion.

In this thesis, two different ways are presented of modifying the E-algorithm for generating error detection test sets of a combinational network.

In chapter 2, we showed how the topological structure of a combinational network can be embodied within its E-expression. Under the single fault assumption, a sufficient test set can be obtained by discarding the so called T-set, i.e., in only one pass in the E-algorithm. Under the single fault assumption, the generation of test sets using the proposed modifications to the original E-algorithm is likely to require somewhat less effort. A quantitative measure of the difference for the general case remains to be found. It is expected that the principal application of this approach, i.e., subscripting the E-expression, will be in fault location problems, even though this approach is guaranteed to be as good as existing fault detection methods. It is a quite straight forward process to extend our approach to handle the fault location problem. The key issue involved is the way one manipulates the fault table.

In chapter 3, we showed how a "D-cover" of a sub-network can be built efficiently, and how tests of each sub-network may be combined together in forming good tests for the whole network. The Armstrong and Roth path sensitizing techniques are modified and imbedded within our merging and catch-up processes. With the help of the set of PTI's, it is quite easy to find a set of reasonably good tests

from the results of the merging-catch-up processes.

Since the partition method is designed to handle large scale combinational networks, it is quite reasonable to expect that extension to cover the sequential case may be possible. Of course the first question that has to be answered is whether the E-algorithm is applicable to the sequential case. This appears to be a promising area for future investigation.

We have attempted to present our methods in such a way that modifications are not difficult to make, since we are aware that it is often reasonable to do so after some practical results are obtained.

REFERENCES

1. Chang, H. Y., Manning E. G., & Metz, G.: Fault Diagnosis of Digital Systems, John Wiley.
2. Klayton, A. R.: Fault Analysis for Computer Memory Systems and Combinational Logic Network, Ph. D. Dissertation, Department of Electical Engineering, Lehigh University, 1971.
3. Susskind, A. K. and Klayton, A. R.: Multiple-Fault-Detection Tests for Loop Free Logic Networks, Conference Digest, 1971 IEEE International Computer Science Conference.
4. Roth, J. P.: Diagosis of Automatic Failures, a Calculus and a Method, IBM Journal, July 1966.
5. Armstrong, D. B.: On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets, IEEE Transactions on Electronic Computers, Volume EC-15, Number 1, Feb. 1966.
6. Howarter, D.R.: The Selection of Failure Location Tests by Path Sensitizing Techniques, Report R-316, Aug. 1966. Coordinated Science Laboratory, University of Illinois.
7. Clegg, F. W.: Algebraic Properties of Faults in Logic Networks, Technical Report No. 4, March 1970, Stanford Electronics Laboratories.
8. Bossen, D. C. & Hong, S. J.: Cause-Effect Analysis for Multiple Fault Detection in Combinational Networks. IEEE Transactions on Computers, C-20, Nov. 1971.
9. Diephuis, R. J.: Faults Analysis for Combinational Logic Networks. Department of Electrical Engineering, MIT, Sept. 1969.
10. Susskind, A. K. et al: Threshold Elements and the Design of Sequential Switching Networks. MIT Electronic Systems Lab. Technical Report No. RADC-TR-68-313, Oct. 1968.
11. Kautz, W. H.: Fault Testing and Diagnosis in Combinational Digital Circuits, IEEE Transactions on Computers, April 1968.
12. Isenhart, R. D.: Optimal Test Sets for the Diagnosis Multiple Output Combinational Nets, Report R-381, May 1968. Coordinated Science Laboratory, University of Illinois.

Vita

Siu-chong Si

Born: February 21, 1947 in Swatow, Canton Province, China.

Parents names: Jackson and Lantan Hoi Sze.

Educations: Lingnam Middle school (1965)
National Taiwan University (1969) B.Sc.

Experience: Electromagnetic Waves Research Center, Taiwan, China. (1968 summer)
Teaching assistant of the Department of Electrical Engineering and Computer Center of National Taiwan University. (1969-1970 academic year)
Join the research project sponsored by Bell Telephone Laboratories at Lehigh University during the summer of 1971.

Membership: IEEE Computer and Information Group.
The Electrical Engineering Association of China.