PARTITIONING LOGIC CIRCUITS TO MAXIMIZE FAULT RESOLUTION*

by

Ayee Goundan and John P. Hayes

Department of Electrical Engineering
University of Southern California
Los Angeles, California 90007

Summary

Two techniques are discussed for partitioning logic circuits to maximize the resolution of stuck-line faults. One technique exploits the inherent fault resolution of the circuit by attempting to force equivalent faults into the same module. The other involves inserting control points to separate members of equivalent fault classes, a technique called fault class splitting. Some new methods for identifying equivalent faults are also presented.

I. Introduction

Logic network partitioning may be defined as the process of assigning logic elements to replaceable physical modules [1]. Partitioning is carried out on several different levels in the digital system design process as indicated in Fig. 1. The partitioning problem is to assign elements to modules so that some objective function is optimized, while some set of constraints is satisfied. A partition that satisfies all the constraints but is not necessarily optimal is said to be a feasible partition. The usual objective is to minimize the total number of modules used, while the constraints place upper bounds on the number of elements per module (space constraints) and the number of external connections per module (pin constraints). Other objective functions that have been considered include minimizing the total number of intermodular connections [2], and minimizing the maximum delay through the network [3].

elements	replaceable modules
gates	integrated circuits (IC's)
IC's	circuit boards
circuit boards	cabinets

Fig. 1. Possible levels of partitioning

In this paper we consider the problem of assigning logic gates to IC modules so that each fault of interest can be diagnosed (located) to as few modules as possible, preferably to a single module. Thus the objective is to maximize the fault resolution obtainable. This problem is important in view of the fact that in most large digital systems, the on-line diagnostic programs used cannot locate every fault to a replaceable module. This results either in the replacement of both fault-free and faulty modules, or else in the use of costly off-line testing procedures.

Two approaches to the design of diagnosable circuits are examined.

(1) Partitioning the given circuit N to maximize its inherent fault resolution. This involves finding a feasible partition in which the maximum number of modules m to which any fault is resolvable is minimized. This means that all members of any set of equivalent (or indistinguishable) faults are confined to m or fewer modules. Although m=1 is desirable, it may not be achievable; it all depends on the given circuit. We therefore also consider a second approach.

(2) Modifying a circuit (including circuits that have already been partitioned) to decrease m. Two techniques are possible: the insertion of extra output lines (test points), or the insertion of extra input lines (control points) [4]. The use of test points to improve fault resolution has been considered by Gaddess [5]. Our approach is to insert control points to "split" equivalent fault classes with members in 2 or more modules. Control inputs have the advantage that they can be readily combined to reduce the number of extra pins that must be added to the network.

2. Identifying Equivalent Faults

Without loss of generality, we will confine our attention to NAND networks. All faults will be assumed to be of the standard stuck-at-1 (s-a-1) and stuck at 0 (s-a-0) type. It is also assumed that all faults are detectable or, equivalently, that the networks are irredundant.

In order to be able to measure the degree of fault resolution possible in a given logic network, it is necessary to be able to identify sets of equivalent faults or fault classes. Two faults are equivalent if the functions realized by the faulty circuits (the fault functions) are identical. The fault classes can, in principle, be determined exhaustively by computing all fault functions, and then comparing them pairwise. For large networks, such exhaustive procedures are not practical. For example, in a network of 1000 lines, there are 2000 fault functions and $\binom{2000}{2} = 1999000$ pairs of fault

*This research was supported by the National Science Foundation under Grant ENG74-18647 and by the Joint Services Electronics Program through the Air Force Office of Scientific Research/AFSC under Contract F44620-71-C-0067.

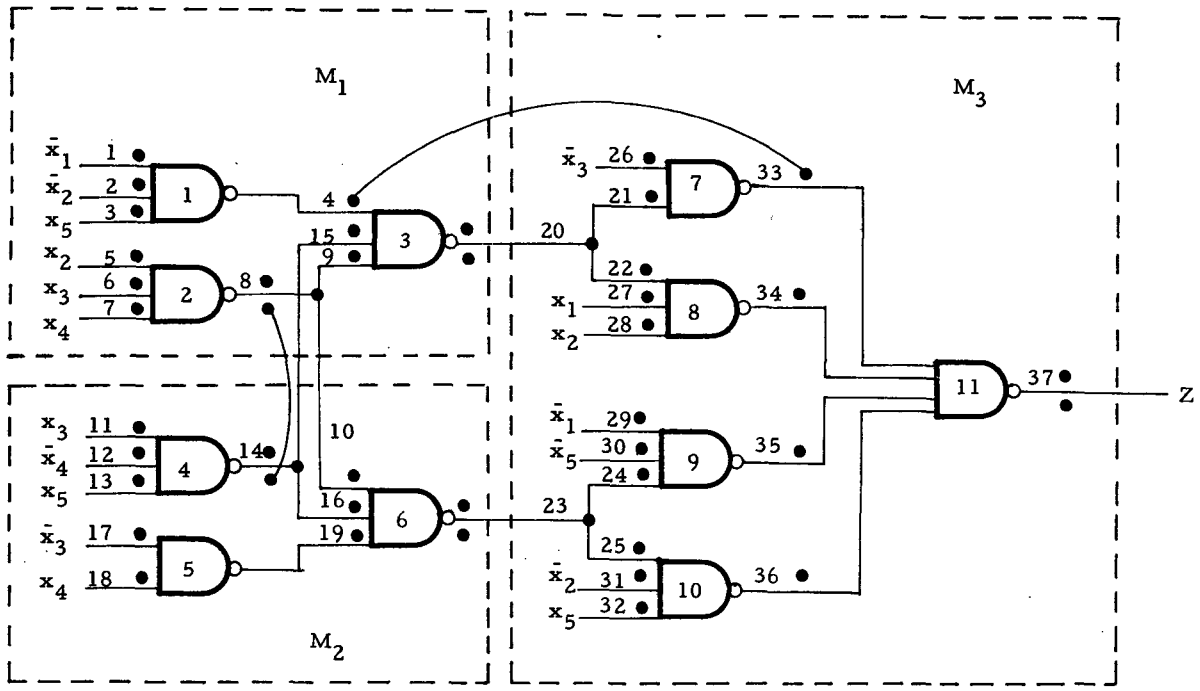


Fig. 2. A NAND network N_G and its representative faults

functions to be compared. Thus, before partitioning can be carried out, it is essential to have a computationally efficient way of identifying fault classes.

A number of attempts have been made to find ways of determining fault equivalence directly from network structure, i.e., without computing fault functions [6, 7, 8]. A general approach is to identify a set of representative faults, and then eliminate faults that are equivalent to the representative faults. Fault collapsing [8] recognizes that the s-a-0 faults on the input lines of a NAND gate are equivalent to the s-a-1 fault on the gate output line. It is convenient to retain the latter s-a-1 fault as a representative, and eliminate the others [6]. Fault collapsing eliminates approximately half the faults in a network, since only the s-a-0 faults on primary output lines and lines that fan out are retained. Faults that are equivalent, but which cannot be identified as equivalent by fault collapsing, are termed nonelementary equivalent faults.

It appears that the number of nonelementary equivalent faults in most circuits is quite small. Consider for example the network N_G in Fig. 2 which is due to Gaddess [5]. There are 37 lines implying 74 distinct s-a-0/1 faults. Fault collapsing reduces this number to 42. Following the notation introduced in [8], these representative faults are indicated by dots near the corresponding lines. A dot above (below) a line indicates the s-a-1 (s-a-0) fault; dots representing equivalent faults are joined.

Let the output line L emanating directly from a gate G be called its primary output. Additional lines that fan out from L are called secondary outputs of G. Thus in Fig. 2, L_{20} is the primary output of G_3 and L_{21} and L_{22} are its secondary outputs. The following theorem is a generalization of the fault collapsing concept.

Theorem 1: In a combinational (NAND) network the s-a-0 faults on the primary output lines of two gates G_1 and G_2 are equivalent if either of the following conditions holds:

(1) The outputs of G_1 and G_2 are connected to the same set of gates.

(2) All gates $\{G_i\}$ connected to the inputs of G_1 are connected to the outputs of G_2, and there is no fan out from the outputs of the intermediate gates $\{G_i\}$.

Gates G_2 and G_4 of Fig. 2 satisfy condition (1) of Theorem 1, hence the s-a-0 faults on lines 8 and 14 are equivalent. Another structural property of networks that facilitates the determination of equivalence classes is inversion parity. We define the inversion parity of line L_i with respect to an output z_k, denoted $p(L_i, z_k)$, as even (odd) if the number of inversions (NAND's in a NAND network) along all possible paths from L_i to z_k is even (odd). $p(L_i, z_k)$ is undefined if there are paths from L_i to z_k with both even and odd numbers of inversions. Let L^d denote the fault: Line L s-a-d, where d = 0 or 1. If $L_i^{d_i}$ and $L_j^{d_j}$ are equivalent, we write $L_i^{d_i} = L_j^{d_j}$.

Theorem 2: Let lines L_i and L_j feed exactly the same set of primary outputs $Z = \{z_1, z_2, \ldots, z_m\}$ of network N. Let the inversion parity of L_i and L_j be defined for all z_k and be p_i and p_j, respectively.

(1) If $p_i = p_j$, then $L_i^0 \neq L_j^1$.

(2) If $p_i \neq p_j$, then $L_i^0 \neq L_j^0$ and $L_i^1 \neq L_j^1$.

Proof: Let y_i denote the function appearing on L_i. We can write $z_k = y_i^{p_i} A + B$ where $y_i^{p_i} = y_i (\overline{y_i})$ if p_i is even (odd), and A and B are independent of y_i. Let $z_k | L_i^d$ denote the fault function to which z_k is changed by the fault L_i^d, and let $<$ denote proper logical implication. When p_i is even

$$z_k | L_i^0 = B < z_k$$

$$z_k | L_i^1 = A + B > z_k .$$

When p_i is odd

$$z_k | L_i^0 = A + B > z_k$$

$$z_k | L_i^1 = B < z_k .$$

A similar set of relations holds for L_j. Since N is irredundant, $z_k | L^d \neq z_k$. If $z' < z_k$ and $z'' > z_k$ then $z' \neq z''$. Hence if $p_i = p_j$, $z_k | L_i^0 \neq z_k | L_j^1$, therefore $L_i^0 \neq L_j^1$. Part (2) of the theorem follows similarly. ■

Inversion parity relationships and Theorem 2 are very useful in reducing the amount of computation required to determine fault classes. For example, the 74 single faults in N_G reduce to 41 after applying conventional fault collapsing and Theorem 1. If equivalence is now determined by pairwise comparison of fault functions, $\binom{41}{2} = 920$ comparisons are needed. By computing inversion parities, we can divide the set of 41 faults into five groups:

(1) E^0: s-a-0 faults associated with even parity lines

(2) E^1: s-a-1 faults associated with even parity lines

(3) O^0: s-a-0 faults associated with odd parity lines

(4) O^1: s-a-1 faults associated with odd parity lines

(5) U : faults associated with lines of undefined parity.

For the circuit under consideration, the 41 remaining faults yield the following:

$$|E^0| = 3,\ |E^1| = 25,\ |O^0| = 1,\ |O^1| = 12,\ |U| = 0 .$$

If pairs of faults that are not equivalent by Theorem 2 are excluded, then the number of remaining pairs is

$$\binom{3}{2} + \binom{25}{2} + \binom{12}{2} + 3(12) + 25(1) = 430 .$$

An additional reduction in computation is obtained if the network is partitioned, in which case pairs of faults associated the same module need not be compared. A fault L_i^d is said to be in module M_j if L_i is either an input or primary output line of a gate in M_j. Fig. 2 shows a 3-module partition where $M_1 = \{G_1, G_2, G_3\}$, $M_2 = \{G_4, G_5, G_6\}$ and $M_3 = \{G_7, G_8, G_9, G_{10}, G_{11}\}$. If pairs of faults in the same module are ignored, the number of pairs derived previously, 430, can be reduced to 284. The fault ambiguity of a modular network N is the smallest number m(N) such that every fault in N can be resolved to m(N) or fewer modules. In the network of Fig. 2 only two fault classes $\{L_4^1, L_{33}^1\}$ and $\{L_8^0, L_{14}^0\}$ span two modules, hence its fault ambiguity is two.

3. Partitioning to Maximize Inherent Resolution

The fault ambiguity of a network can often be significantly decreased by altering module boundaries based on the fault classes present in the network. Clearly m = 1 represents the optimum case, where the network is so partitioned that if two faults are equivalent then they are in the same module. We define a gate cluster of N as the smallest set of gates S such that if $G \in S$, every gate containing a fault equivalent to one in G is also in S. The gate clusters form a partition of the network. It is obviously desirable to place all members of a gate cluster in the same module. We now present an algorithm (Procedure 1) to determine the fault classes and gate clusters in a given unpartitioned network. We then incorporate this algorithm into a general heuristic partitioning method (Procedure 2).

The following procedure is intended to be used in conjunction with a conventional fault simulation and test generation program. A fault partition F/T of a set of faults in N with respect to a test set T is a set of fault classes $\{F_1, F_2, \ldots, F_p\}$ such that two faults L_i^d and L_j^d are in $F_i \in F/T$ if and only if T cannot distinguish them.

Procedure 1: To determine the fault classes and gate clusters of N.

(1) Using the techniques discussed in section 2, generate a set of representative faults F for N. Let $F/T_0 = \{F\}$ and let i = 1.

(2) Generate a new test vector t_i for N.

(3) Compute F/t_i. Intersect F/t_i with the previous fault partition F/T_{i-1} to obtain a new fault partition F/T_i.

(4) If every block in F/T_i contains one member, or if $i = 2^n$, go to step (5). Otherwise set $i = i+1$ and go to step (2).

(5) The blocks of F/T_i constitute the fault classes of N. The gate clusters can be determined directly from these fault classes by associating each $f \in F$ with all faults, not just those in F, with which it is equivalent.

It is important to note that Procedure 1 avoids the explicit generation of fault functions. A very simple test generation technique, e.g. pseudo-random test generation, can be used. Procedure 1 is an algorithm since in the worst case it will compute F/T where T is the set of all 2^n possible input vectors.

We now show how Procedure 1 can be incorpo-

rated into any conventional partitioning technique. It is assumed that a set of conventional packaging constraints limiting the number of gates per module and the number of external connections per module must be satisfied, and that these constraints take precedence over fault resolution.

Procedure 2: To find a partition of N with minimum or near-minimum fault ambiguity.

Let q be the maximum number of modules allowed and, following [1], let $\bar{S}$ denote the maximum space (number of gates) and $\bar{E}$ the maximum number of external pins per module. Let $S(P_i)$ represent the number of gates in the cluster P_i and let $E(P_i)$ represent the number of external pins needed to assign all the gates of P_i to one module.

(1) Using Procedure 1 generate the gate clusters $\{P_i\}$ of N. Order these clusters such that $S(P_1) \geq S(P_2) \geq \ldots \geq S(P_p)$. Let $M_k = \phi$ for $k = 1, 2, \ldots, q$, and let $i = j = 1$.

(2) If $S(P_i) > \bar{S}$ or $E(P_i) > \bar{E}$, go to step (3); otherwise assign P_i to module M_j, set $i = i+1$, and set $j = j+1$. If $i > p$ go to step (5). If $j > q$ go to step (4), otherwise go to step (2).

(3) Assign gates of P_i sequentially to module M_j until the pin or space constraints are violated. Set $P_i = P_i - \{$the gates assigned to $M_j\}$ and set $j = j+1$. If $j > q$ go to step (4), otherwise go to step (2).

(4) Formulate a linear assignment problem to assign the remaining gates to unused space in the modules, with the objective of minimizing fault ambiguity. The cost $c(G_k)$ of assigning gate G_k to a module M_ℓ is infinite if either $S(M_\ell \cup \{G_k\}) > \bar{S}$ or $E(M_\ell \cup \{G_k\}) > \bar{E}$; otherwise it is $-1(0)$ if M_ℓ contains (does not contain) a fault equivalent to a fault in G_k. Solve this assignment problem using any conventional technique and go to step (5).

(5) Stop. If each gate of the original network has been assigned to a module then a partition with minimum or near-minimum fault ambiguity has been obtained.

Note that in step (3) of the above procedure, the larger clusters are assigned to modules first. This is because the smaller clusters can more easily fill the leftover space which is assigned in step (4). Consider again the network N_G of Fig. 2. The gate clusters, which can be determined by Procedure 1 (or directly from the fault classes shown in the figure) are: $P_1 = \{G_1, G_3, G_7, G_8, G_9, G_{10}, G_{11}\}$, $P_2 = \{G_2, G_4\}$ and $P_3 = \{G_5, G_6\}$; clearly these do not match the modules shown in Fig. 2. Suppose a network partition is to be found with space and pin constraints $\bar{S} = 8$ and $\bar{E} = 14$, respectively. On applying Procedure 2 we obtain the partition N' comprising 2 modules $M_1' = P_1 = \{G_1, G_3, G_7, G_8, G_9, G_{10}, G_{11}\}$ and $M_2' = P_2 \cup P_3 = \{G_2, G_4, G_5, G_6\}$. N' is feasible and $m(N) = 1$.

4. Fault Class Splitting Using Control Logic

Partitioning alone may not suffice to increase fault resolution (or, equivalently, reduce fault ambiguity) to an acceptable level. It may be necessary to modify the original network. This can be done by adding extra inputs or outputs to the network [4]. We consider the use of control inputs to split fault classes that span two or more modules. While this can also be done by adding outputs, control inputs have the advantage that they can be readily combined, resulting in the addition of fewer extra pins to the circuit.

Consider a network N_1 to which fault collapsing has been applied yielding a set of representative faults F_1. Let G be any (NAND) gate in N_1. We now present a procedure to isolate the s-a-1 faults of G from the remaining faults in F_1. We assume that the output of G is not a primary output of N_1.

Procedure 3: To split the s-a-1 faults in G from the remaining faults in F_1.

(1) Introduce a new (NAND) gate G' with the same number of inputs, q, as G.

(2) Connect each input L_i of G to a distinct input L_i' of G'. Add control lines c_1 and c_2 to G and G' respectively.

(3) Connect the output of G' to all gates to which the output of G is connected.

The control inputs c_1 and c_2 are used during testing only. During normal operation of the modified circuit at least one of the control lines must be held at logic value 1. Fig. 3 shows the result of applying Procedure 3 to the gate G_1 of Fig. 2; a new gate G_1' is added to the circuit. G_1 and G_1' contain equivalent faults and so must be assigned to the same module. However, the faults L_4^1 and L_{33}^1 are no longer equivalent, therefore all faults in M_3 and the modified module M_1 are distinguishable. Note that the faults on lines L_1, L_2, L_3 of the modified circuit do not introduce any new fault equivalences.

Lemma 1: In an irredundant NAND network no two s-a-1 faults on the input lines of the same gate G are equivalent. No s-a-1 input fault of G is equivalent to a fault on the primary output line of G.

Theorem 3: Let G be a gate in N_1 whose output does not fan out. The circuit N_2 obtained by applying Procedure 3 to G is such that all faults in G and G' are distinguishable from the remaining faults in N_2.

Proof: Fig. 4 shows the relevant parts of N_1 and N_2.

Let F_1 and F_2 be the representative faults in N_1 and N_2 respectively, after applying fault collapsing and Theorem 1. If

$$F_{12} = \{L_{e_1}^1, \ldots, L_{e_q}^1, L_{g_1}^1, \ldots, L_{g_q}^1, L_{k_1}^1, \ldots, L_{k_q}^1, L_{e_0}^1, L_{g_0}^1, L_{h_0}^1, L_{k_1}^0\},$$

then $F_2 = F_1 \cup F_{12}$.

Let F_1/T_1 be the fault partition induced on N_1 by T_1, where T_1 is the set of all 2^n possible input vectors of N_1. By Lemma 1, the faults $L_{g_1}^1$, $L_{g_2}^1$,

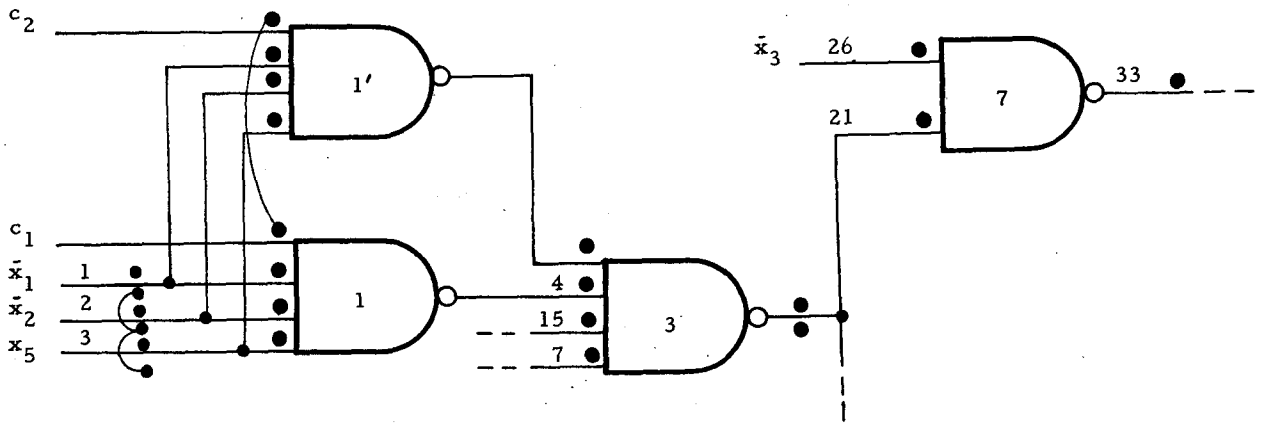
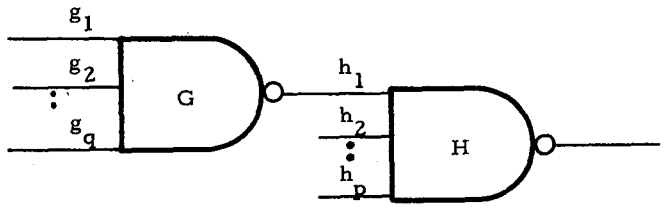
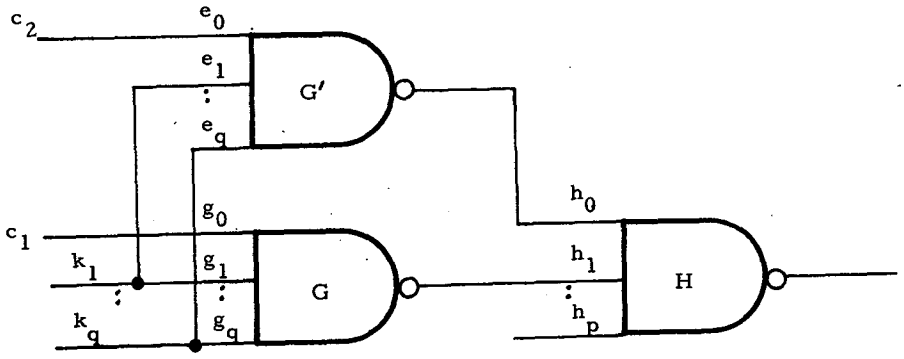


Figure 3. Application of Procedure 3 to G_1 in N_G



(a)



(b)

Fig. 4. Part of a network (a) before and (b) after applying Procedure 3

$\ldots, L^1_{g_q}, L^1_{h_1}$ all lie in different fault classes, hence $F_1/T_1 = \left\{ \{L^1_{g_1}, F_{g_1}\}, \ldots, \{L^1_{g_q}, F_{g_q}\}, \{L^1_{h_1}, F_{h_1}\}, F_r \right\}$ where F_{g_1} is the set of faults equivalent to $L^1_{g_1}$, and F_r is the set of all the remaining fault class in F_1/T_1.

Let T_2 denote all 2^{n+2} input vectors of N_2, and let T_{ij} denote the 2^n members of T_2 with $c_1 = i$ and $c_2 = j$.

Suppose T_{10} is applied to N_2. All vectors in T_{10} apply 0 to c_2 hence the output L_{h_0} of G' is always 1. This implies that

$$F_2/T_{10} = \left\{ \{L^1_{g_1}, L^1_{k_1}, F_{g_1}\}, \ldots, \{L^1_{g_q}, L^1_{k_q}, F_{g_q}\}, \right.$$

$$\{L^1_{h_1}, L^0_{k_1}, F_{h_1}\}, \{L^1_{e_0}, L^1_{e_1}, L^1_{e_2}, \ldots,$$

$$L^1_{e_q}, L^1_{g_0}, L^1_{h_0}\} \} \quad .$$

Since (G, c_1) and (G', c_2) are symmetrical we also have

$$F_2/T_{01} = \{ \{L^1_{e_1}, L^1_{k_1}, F_{g_1}\}, \ldots, \{L^1_{e_q}, L^1_{k_q}, F_{g_q}\}, \{L^1_{h_0}, L^0_{k_1}, F_{h_1}\}, \{L^1_{g_0}, L^1_{g_1}, L^1_{g_2}, \ldots, L^1_{g_q}, L^1_{e_0}, L^1_{h_1}\} \} \quad .$$

T_{00} detects the faults $L^1_{g_0}, L^1_{e_0}$ which are indistinguishable and are not detected by T_{01} or T_{10}. Now

$$F_2/T_2 = F_2/T_{00} \cap F_2/T_{01} \cap F_2/T_{10} \cap F_2/T_{11} = \{ \{L^1_{g_0}, L^1_{e_0}\}, \{L^1_{e_1}\}, \{L^1_{e_2}\}, \ldots, \{L^1_{e_q}\}, \{L^1_{g_1}\}, \{L^1_{g_2}\}, \ldots, \{L^1_{g_q}\}, \{L^1_{h_0}\}, \{L^1_{h_1}\}, \ldots \} \quad .$$

Hence all the faults on the inputs and outputs of G and G' are isolated from the remaining faults in N_2. ■

It is important to note that the s-a-1 faults on the control lines c_1 and c_2 are equivalent. This requires G and G' to be in the same module. However, the fact that both are connected to H by lines without fan out already implies that G, G' and H contain equivalent faults, and therefore should be assigned to the same module.

Corollary 1: Theorem 3 is also valid if the output of G fans out.

The proof of this corollary is similar to that of Theorem 3. Note that if G fans out, and two s-a-1 faults on its secondary output lines are equivalent, these faults are not separated by Procedure 3. They can be separated by applying the procedure to one of the gates to whose inputs the secondary outputs of G are connected.

We call the process of isolating the faults of G from the remaining faults, fault class splitting. Procedure 3 can be applied to many different gates in N_1. If the fault classes being split are disjoint, i.e. they have no common members, then the same control lines may be used to split many fault classes. Suppose F_1 and F_2 are disjoint fault classes and c_1 and c_2 are used to split F_1, then $\bar{c}_1$ and $\bar{c}_2$ may be used to split F_2, which means that only 2 external control inputs need be added to the modified circuit. If more than 3 fault classes must be split, then the following result applies.

Theorem 4: Let $F_1, F_2, \ldots, F_n$ be $n \geq 3$ disjoint fault classes in a circuit. Then each of these fault classes can be split using k external control inputs where k is the smallest number such that $n \leq \binom{k}{2}$.

5. Conclusions

Two approaches to the problem of improving fault resolution in modular networks have been examined. The first attempts to exploit the inherent fault resolution of a given circuit by choosing module boundaries that enclose all members of each fault class in the module. The second method involves modifying the original circuit by the addition of control inputs and extra gates. Previous applications of this approach have been interpreted in terms of observing and controlling internal states of the network [4]. We have interpreted our approach in terms of separating the members of equivalent fault classes. Although the repeated application of Procedure 3 can add significantly to network cost, it has two major advantages.

(1) Control inputs can be shared, so that many fault classes can be split using few additional input pins. No corresponding technique for combining extra outputs (test points) is known.

(2) Since the inserted gates are added in parallel with the original gates, the overall increase in propagation delay (which is due to the extra loading only) is relatively small.

It would be very useful to have guide-lines for designing logic networks with good inherent fault resolution, e.g., few fault classes involving gates that are not directly connected to one another. Theorem 1 suggests one possible design rule: no two gates should have their outputs connected to the same set of gates. Another tentative rule is: lines crossing module boundaries should fan out. In Fig. 2, for example, no fault in $G_6 \in M_2$ is equivalent to a fault in $G_9 \in M_3$. If G_6 were connected to G_9 only (no fanout), then this would no longer be the case. When Procedure 3 is applied to a gate G, fanout is introduced in all the original input lines of G with the effect of separating faults in G from faults in the gates feeding G.

References

[1] U.R. Kodres, "Partitioning and card selection," in Design Automation of Digital Systems (M.A. Breuer, ed.), Prentice-Hall, 1972, pp. 173-212.

[2] F. Luccio and M. Sami, "On the decomposition of networks in minimally interconnected subnetworks," IEEE Trans. Circuit Theory, Vol. CT-16, pp. 184-188, May 1969.

[3] E.L. Lawler, et al, "Module clustering to minimize delay in digital networks," IEEE Trans. Computers, Vol. C-18, pp. 47-57, January 1969.

[4] J.P. Hayes, "On modifying logic networks to improve their diagnosability," IEEE Trans. Computers, Vol. C-23, pp. 56-63, January 1974.

[5] T.G. Gaddess, "Improving the diagnosability of modular combinational logic by test point insertion," Coordinated Science Lab., University of Illinois, Urbana, Report R-409, March 1969.

[6] J.P. Hayes, "A NAND model for fault diagnosis in combinational logic networks," IEEE Trans. Computers, Vol. C-20, pp. 1496-1506, December 1971.

[7] E.J. McCluskey and F.W. Clegg, "Fault equivalence in combinational logic networks," IEEE Trans. Computers, Vol. C-20, pp. 1286-1293, November 1971.

[8] D.R. Schertz and G. Metze, "A new representation of faults in combinational digital circuits," IEEE Trans. Computers, Vol. C-21, pp. 858-866, August 1972.