Several methods on the design of minimum length fault tests for combinational circuits

by

Jonqpyng Yuan

A Research Project Report

submitted to

Department of Computer Science

Oregon State University

In partial fulfillment of

the requirement for the

degree of

Master of Science

December 10, 1976

Approved:

Robert A. Short

Robert A. Short, Chairman

Department of Computer Science

in charge of major

Date presented 10th December, 1976

Typed by Sue-I Tung

Contents

I Introduction

The problems of determining whether or not a digital circuit operates correctly, and of ensuring its correct operation even when some of its parts fail are of both practical concern and theoretical interest. Present-day digital systems may be disabled by almost any internal failure, and the trend toward miniaturization makes the problems of maintenance and fault detection even more urgent. Digital systems may suffer two classes of faults: temporary faults, which occur due to noise and the nonideal transient behavior of switching components, and permanent faults, which result from component failures. We shall primarily be concerned with permanent faults due to component failures. It is assumed that other procedures will be employed to protect the circuit against the effects of transient faults.

One way of determining whether a combinational circuit operates properly is by applying to the circuit all possible input combinations and comparing the resulting outputs with either the corresponding truth table or a faultless version of the same circuit. Any deviation indicates the presence of some fault. Moreover, if a known relationship exists between the various possible faults and the deviations of output patterns, it is possible to diagnose the fault and to classify it at least within a subset of faults whose effects on the circuit outputs are identical. Such exhaustive tests are generally very long, and in many cases impractical.

In order to arrive at fault-detection procedures which will be relatively easy to devise and short enough to be practical, it is necessary to make several simplifying assumptions. In this paper we shall develop procedures for circuits composed of loop-free interconnections of AND, OR,

NOT, NAND, and NOR gates. That is, feedback loops are not allowed in the circuits being tested. We shall be concerned primarily with procedures for the detection of single fault. All the tests are developed for irredundant combinational circuits. If a circuit contains logical redundancies, not all faults are detectable, since a redundant connection may be cut without altering the logical value of the function.

A permanently open diode, or a cut input connection to an AND or NAND gate, is logically equivalent to having the input "stuck-at-one". A permanently open diode, or a cut input connection to an OR or NOR gate, is logically equivalent to having that input "stuck-at-zero". In designing fault-detection tests we shall be concerned only with those faults which cause any wire to be (or appear logically to be) stuck-at-zero or stuck-at-one (abbreviated s-a-0 and s-a-1). Restricting our consideration to just this class of faults is technically justified since most circuit failures fall in this class, and many other failures exhibit symptomatically identical effects.

Now, several methods have been commonly used for the design of minimum length fault tests for combinational circuits. Most of the methods use either of the four techniques: "Map", "Tabular", "Matrix", or "Scoring". There are four typical methods considered in the following sections, and each of these corresponds to one of the techniques named above. Also, these four typical methods are relatively easy to use and result in tests that are short enough to be practical. However, there is no method that can give an absolutely minimum length fault test among all possible designs except those on the simple circuits containing only a few variables. The four methods we shall consider have been gleaned from the original literature and some critical comments on each original

paper will be provided. We discuss two-level circuits in sections II-1, II-2, II-3, and III, for which the "map", "tabular", and "fixed schedule ($F \longrightarrow G$ matrix)" methods are applied. Multilevel circuits are considered in section II-4 and IV, and for those the "tabular" and "scoring" methods are applied.

II Map and Tabular method

II-1 Boolean difference function[1] fault tests

Before developing test generation by this technique, let us first discuss the Boolean difference function itself, which is the theoretical support on which the map and tabular methods are based. The Boolean difference function can be expanded to form two analytical expressions that can be used to calculate the tests for any s-a-0 and s-a-1 fault within combinational circuits. The map and tabular methods of developing a fault test can be developed directly from these analytical expressions. The minimum test for the complete circuit can then be taken directly from the map or table as we shall illustrate subsequently below.

For switching function F, the Boolean difference function with respect to variable x_i is defined as:

$$\frac{dF(x)}{dx_i} = F(x_1, x_2, \ldots, x_i, \ldots, x_n) \oplus F(x_1, x_2, \ldots, \bar{x}_i, \ldots, x_n)$$

which describes the condition which by a change in a variable x_i will cause a change in the output. Of course F is the logical function corresponding to a combinational circuit. Notice, the variable x_i does not appear at the right side of $\frac{dF(x)}{dx_i}$, as follows immediately from an argument based on the usual Shannon expansion about variable x_1:

$$F(x_1) = x_1 f(1,x_2,\ldots,x_n) + x_1' f(0,x_2,\ldots,x_n) = x_1 f_1 + x_1' f_0$$

$$F(x_1') = x_1' f(1,x_2,\ldots,x_n) + x_1 f(0,x_2,\ldots,x_n) = x_1' f_1 + x_1 f_0$$

$$\frac{dF(x_1)}{dx_1} = (x_1 f_1 + x_1' f_0) + (x_1' f_1 + x_1 f_0)$$

$$= (x_1 f_1 + x_1' f_0)(x_1' f_1 + x_1 f_0)' + (x_1 f_1 + x_1' f_0)'(x_1' f_1 + x_1 f_0)$$

$$= x_1 f_1 f_0' + x_1' f_1' f_0 + x_1' f_0' f_1 + x_1 f_1' f_0$$

$$= f_1 f_0' + f_1' f_0$$

$$= g(x_2,\ldots,x_n)$$

For example:

$$F(x,y) = x'y$$

For x:	For y:
$f_1 = 0$	$f_1 = x'$
$f_1' = 1$	$f_1' = x$
$f_0 = y$	$f_0 = 0$
$f_0' = y'$	$f_0' = 1$

$$\frac{dF(x,y)}{dx} = f_1 f_0' + f_1' f_0 = 0 \cdot y' + 1 \cdot y = y$$

$$\frac{dF(x,y)}{dy} = x' \cdot 1 + x \cdot 0 = x'$$

This definition of the Boolean difference function has some disadvantages: the first disadvantage is due to the fact that the Boolean difference of a function with respect to a variable does not distinguish between the effects of that variable changing from a one to a zero and from it changing from a zero to a one. It also doesn't give any information about the effect of the variable perturbation on the output, i.e., whether the output changes from a one to a zero or a zero to a one. In order to overcome the problem of not knowing which test is valid for

a s-a-0 fault condition, the variable can be logically appended to the Boolean difference taken with respect to the variable question, i.e., $x_i(dF(x)/dx_i)$ will yield the equation for which a change in x_i from one to zero will cause a change in the circuit output. Conversely, the equation for a change in x_i from zero to one can be obtained by $\bar{x}_i(dF(x)/dx_i)$. We will use the symbols $\nabla x_i F$ for $x_i(dF(x)/dx_i)$ and $\Delta x_i F$ for $\bar{x}_i(dF(x)/dx_i)$.

The problem of determining what the correct output should be for a given test can be solved by intersecting the tests for a s-a-1 and s-a-0 with both the function and its inverse. We now define two equations that define completely the test for an individual fault:

$$\nabla x_i F = x_i \frac{dF}{dx_i}(F , \bar{F}) \qquad (1)$$

$$\Delta x_i F = \bar{x}_i \frac{dF}{dx_i}(F , \bar{F}) \qquad (2)$$

In these two equations, the tests for the variable x_i s-a-1 and s-a-0 are intersected with the ordered pair $(F , \bar{F})$. Equation (1) gives the test that will detect a s-a-0 fault in variable x_i, and furthermore it partitions these tests into tests corresponding to a one or zero output for the overall circuit output F. Equation (2) yields the tests for variable x_i s-a-1 and partitions them according to their effect on the circuit output. For example,

$$F = A'B + BD + ACD$$

then

$$\frac{dF}{dA} = (A'B + BD + ACD) \oplus (AB + BD + A'CD)$$

$$= (A'B+BD+ACD)'(AB+BD+A'CD)+(A'B+BD+ACD)(AB+BD+A'CD)'$$

$$= A'B'CD + ABD' + A'BD' + AB'CD$$

$$= B'CD + BD'$$

From this we see that

$$\nabla_A F = A \frac{dF}{dA} = AB'CD + ABD'$$

So the test will be AB'CD (ABD' is undetermined on C, so neglected) for s-a-0 on A. Now, what should the output of F be if we apply AB'CD (i.e. 1011) as an input?

Since $F = A'B + BD + ACD$

whence $\bar{F} = A'B' + B'C' + B'D' + AD'$

by conventional algebraic methods.

Now, AB'CD is intersected with F (since ACD is in F), but not with $\bar{F}$, so the output of F will be 1 if we apply AB'CD (1011) as an input.

Also

$$\Delta_A F = \bar{A} \frac{dF}{dA} = A'B'CD + A'BD'$$

A'B'CD is intersected with $\bar{F}$ (since A'B' is in $\bar{F}$), but not $\bar{F}$, so the output of F will be 0 if we apply A'B'CD (0011) as an input to test A s-a-1.

If we complete each derivative with respect to B, C, and D, then we will get a test set:

(ABC'D 1), (AB'CD 1), (A'BCD' 1)

(ABCD' 0), (A'B'CD 0), (AB'C'D 0)

The one or zero in these terms mean fault-free output.

In following sections II-2 and II-3, we will discuss map and tabular methods, and both of these two methods are based on the Boolean difference function.

II-2 Map method

At first minimum, two-level, sum-of-products circuits will be considered. We will discuss multilevel circuits later, since they need "path sensitizing" techniques besides those yielded by map and tabular techniques. Assume, then, that we have a sum-of-product expression consisting of three product terms:

$$F = f(x) = x_1\Pi_1(x) + x_2\Pi_2(x) + \Pi_3(x)$$

where x_1, x_2 are literals that we wish to test, and x is another set of independent literals. Π_1, Π_2, and Π_3 are in product forms. Now, we examine the literal x_1. The Boolean difference with respect to x_1 is:

$$\frac{dF}{dx_1} = \Pi_1(x)\ \overline{(x_2\Pi_2(x) + \Pi_3(x))}$$

and

$$\nabla x_1 F = x_1\Pi_1(x)\ \overline{(x_2\Pi_2(x) + \Pi_3(x))}\ (F,\bar{F})$$

and also

$$\Delta x_1 F = \bar{x}_1\Pi_1(x)\ \overline{(x_2\Pi_2(x) + \Pi_3(x))}\ (F,\bar{F})$$

as illustrated in section II-1

Considering $\nabla x_1 F$ carefully, one can easily find that the test for x_1 s-a-0 must be chosen from the unique minterms of the function which is chosen from $\nabla x_1 F$ (since there is no complement symbol on the above of $x_1\Pi_1(x)$). Unique minterms of the function are the minterms in a given term but not in any other term of the function. Similarly, a test for x_1 s-a-1 must be chosen from $\Delta x_1 F$. Following these observations, the map method for obtaining a complete and minimum test for a sum-of-

products implementation of a logic expression is given by the following steps:

Step 1: Choose one unique minterm from each term in the expression.

Step 2: Choose a set of x_i- adjacencies from $\bar{F}$ that completely test all literals of each term.

Of these two steps, the more difficult is that of choosing the proper set of x_i- adjacencies[1] in step 2. In looking for a minimum set of adjacencies we are in reality interested in looking for the adjacency with the most coverage. ("Coverage" is used here to mean that an adjacency is adjacent to more than one term of the function.) To aid in the selection of adjacencies it is advantageous to label each adjacency with a number that corresponds to the number of terms to which it is adjacent. For example, consider the map in Fig. 2 for the circuit of Fig. 1. In the map of Fig. 2 the minterm A'B'CD is labeled with a 3 since it is adjacent to the terms BD, A'B, and ACD. Blank seqares are not x_i- adjacent to any term of F.

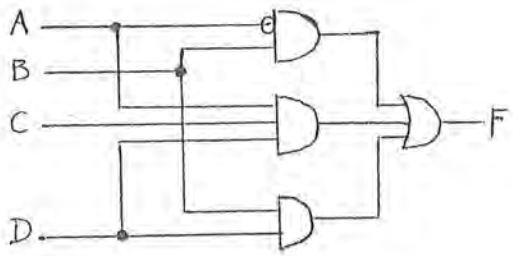


Fig.1 Circuit for F=A'B+BD+ACD

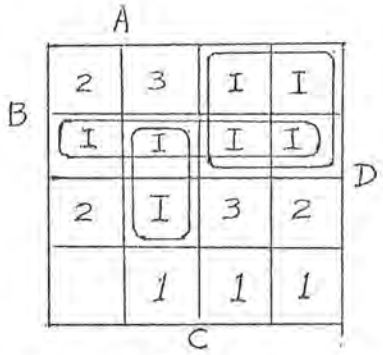


Fig.2 Map for circuit of Fig.1

The minterms of AB'C'D and A'B'CD, labeled 2 and 3, respectively, are essential since AB'C'D is the only term that is C-adjacent to ACD.

---

1 A minterm m_1 is x_i- adjacent to m_2 if $x_i \longrightarrow \bar{x}_i$ implies that $m_1 \longrightarrow m_2$, where $x_i \epsilon m_1$, x_i is a variable in m_1.

As a by-product which are also B-adjacent to BD and A'B. After choosing these two essential tests we must choose the remaining tests so as to complete the entire test schedule. We proceed by selecting the minterm with the highest number of in Fig. 2. In this case the 3 corresponding to the term ABCD' is investigated, it is D-adjacent to ACD, D-adjacent to BD, and A-adjacent to A'B, with the selection of test ABCD' we have a complete test as shown in Fig. 2. A minimum test is then

(ABC'D 1), (AB'CD 1), (A'BCD' 1)

(ABCD' 0), (A'B'CD 0), (AB'C'D 0)

The one or zero in these terms indicates the fault-free output when the inputs are set to the value described by the associated minterm. Now we have determined that the test schedule indicated is (1101, 1011, 0110, 1110, 0011, 1001).

II-3 Tabular method[3]

The procedure in the map method suffers from the limitation inherent in any operation involving a Karnaugh map. When the number of variables increases, the manipulation of the map becomes unwieldy. In this case the tabular method and other methods such as those described in the following sections must be considered.

To apply the tabular method we shall construct a so-called testing table which consists of an array of u columns and v rows. In this context, u designates the total number of inputs to all the AND gates, and v designates the number of possible s-a-0 and s-a-1 tests. Each AND gate input defines a column heading and

every possible test defines a row heading. The s-a-0 and s-a-1 tests are arranged in two disjoint groups and the column headings are grouped by gates, as shown in Fig. 3 for the circuit of our running example:

$$f(A,B,C,D) = A'C' + C'D + ABD + BCD'$$

The table contains ten columns corresponding to the ten AND gate inputs, and 13 rows (It should be 16 rows in the table. However, 1,5,13 are removed since they are contained in two true subcubes) corresponding to the possible s-a-0 and s-a-1 tests. (Recall that every input combination for which f=1 and which is contained in only one true subcube is a valid s-a-0 test, and every input combination for which f=0 and which is contained in a subcube adjacent to a true subcube is a valid s-a-1 test). The entries of the ith column in the table consist of x's placed at the intersection with the rows that correspond to s-a-0 and s-a-1 tests for the ith input. The problem now is to select minimal subsets of s-a-0 and s-a-1 tests, such that each column contains at least one x in an s-a-0 row and one x in an s-a-1 row corresponding to the selected subsets. We can summarize the process by showing how we first determine the essential tests and then proceed to find a cover. (Sometimes, two or more tests are valid for detecting the same fault. In this case, only one of them is used.) In the present example, test 9 and 15 are essential in the s-a-0 group, and tests 7, 11, and 12 are essential in the s-a-1 group. The complete experiment consists of the following eight tests:

$$\{7,\ 9,\ 11,\ 12,\ 15,\ 0 \text{ or } 4,\ 6 \text{ or } 14,\ 2 \text{ or } 10\}$$

If one uses the map method, the same result will obtain, as the interested reader can verify. Again, we can distinguish the faults between s-a-0 and s-a-1 with the above eight inputs. (Usually, it is unnecessary to make this distinction, if we are only concern about the fault detection.)

		Gate 1		Gate 2		Gate 3			Gate 4		
		A′	C′	C′	D	A	B	D	B	C	D′
s-a-0 tests	0	X	X								
	4	X	X								
	6								X	X	X
	9			X	X						
	14								X	X	X
	15					X	X	X			
s-a-1 tests	2		X						X		
	3		X	X							
	7		X	X		X					X
	8	X			X						
	10								X		
	11			X			X				
	12	X			X			X		X	

Fig.3 Testing Table

Although this method was demonstrated for the case of AND-OR logic, its extension to OR-AND logic and NAND-NOR logic is straight-forward and will not be discussed further.

II-4 Multilevel Circuits

II-4-1. Path sensitizing[4]

Before examining the details for multilevel circuits, we shall first discuss the idea of "path sensitizing". The main idea behind the path-sensitizing procedure will be illustrated by deriving a test which detects a s-a-1 fault at input A of Fig.4.

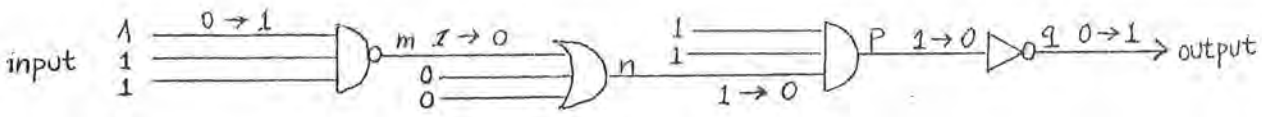


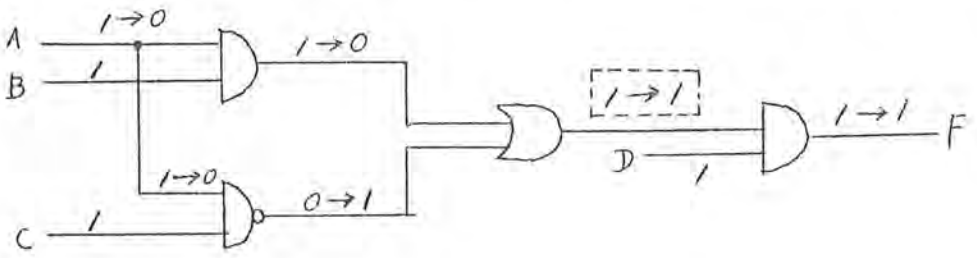
Fig.4 A portion of a circuit describing a sensitized path

Suppose that this path is the only one from A to the circuit output. In order to test a s-a-1 fault at input A, it is necessary to apply a 0 to A and 1's to all remaining inputs to the AND and NAND gates in the path, and 0's to all remaining inputs to the OR and NOR gates in the path. This ensures that all the gates will allow the propagation of the signal from A to the circuit output, and that only this signal will reach the circuit output. This assignment of values is shown in Fig. 4. The path is now said to be sensitized.

If input A is s-a-1, then m, the output of the NAND gate, changes from 1 to 0, and this change propagates through connections n and p and results in point q changing from 0 to 1. Clearly, in addition to detecting a s-a-1 fault at A, this test also detects s-a-0 faults at m, n, and p and a s-a-1 fault at q. A s-a-0 fault at A is detected in a similar manner. A 1 is applied to A, while other gate inputs remain as before. This second test would also detect the complementary set of faults to those detected by the previous test on this path. Thus the two tests together are sufficient to detect all s-a-0 and s-a-1 faults on this path.

Consequently, if we can select a set of tests which sensitize a set of paths containing all connections in the circuit, then it is

sufficient for the detection of just those faults which appear on the circuit inputs. In circuits in which each gate output is connected to just one gate input, there is only one path from each circuit input to the output, and thus the set of paths originating at the inputs will indeed contain all connections. In circuits which do not possess the above property, care must be taken to include all connectinns in the selected set of sensitized paths. As we shall see, such sets of sensitized paths can be found for most circuits, but there is no guarantee that they exist for every circuit. For example:



If often occurs that an assignment of values to the circuit inputs sensitizes several paths simultaneously. Moreover, in general, a path may be sensitized by assigning values only to a subset of the circuit variables. (Since some other variables are not involved in through the sensitized path.) In many cases this makes possible the assignment of the remaining variables in such a way that several paths will be sensitized simultaneously in one test. Unfortunately, no efficient direct methods are available for the selection of these tests for multilevel circuits. In order to utilize these properties and to find a good (i.e., nearly minimal) set of tests, it is necessary to first transform the circuit into its "equivalent normal form" and

then to derive these tests from the transformed circuit.

The equivalent normal form of a switching circuit is obtained by expressing the circuit output as a sum of products such that each path has a corresponding distinct product term. As an example, consider the circuit of Fig. 5, which has four input variables, A, B, C, and D, and one output T. The gates are numbered 1 through 7, and the internal connections are labeled by the letters i, j, k, m, u, v.

The test that can be derived from the equivalent normal from are obtained by tracing all paths from the circuit output to every circuit input, while recording the gates through which the paths pass. Starting with gate 7, we obtain

$$T = (u + v)_7$$

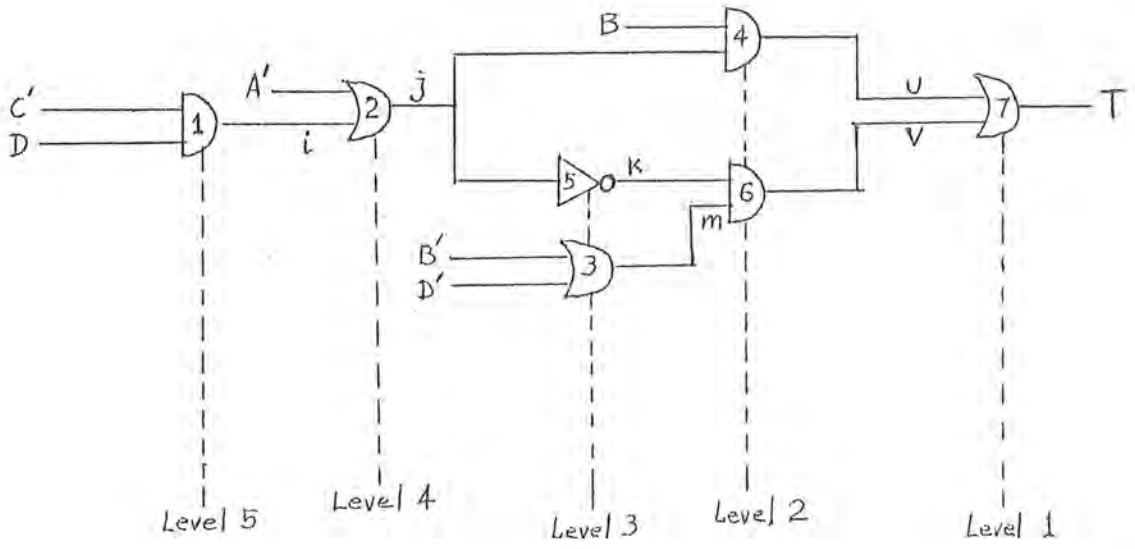


Fig.5 A circuit to be tested

where the subscript 7 identifies the gate through which inputs u and v pass. Proceeding to the second level of gates, we find

$$T = ((Bj)_4 + (km)_6)_7$$

In a similar manner we expand T in terms of inputs to remaining gate levels.

$$\begin{aligned} T &= ((Bj)_4 + ((j')_5(B' + D')_3)_6)_7 \\ &= ((B(A' + i)_2)_4 + (((A' + i)_2')_5(B' + D')_3)_6)_7 \\ &= ((B(A' + (C'D)_1)_2)_4 + (((A(C + D')_1)_2)_5(B' + D')_3)_6)_7 \end{aligned} \tag{3}$$

The output T is now expressed as a function of only the external inputs. The subscript associated with each pair of parentheses is now "distributed" to all the terms within the parentheses, and as a result the expression for the equivalent normal form is obtained as follows:

$$\begin{aligned} T = {} & A'_{247}B_{47} + B_{47}C'_{1247}D_{1247} + A_{2567}C_{12567}B'_{367} \\ & + A_{2567}C_{12567}D'_{367} + A_{2567}D'_{12567}B'_{367} + A_{2567}D'_{12567}D'_{367} \end{aligned} \tag{4}$$

Each variable of the equivalent normal form, together with its subscripts, uniquely specifies a path from the corresponding circuit input to the output. For example, B'_{367} in the third term of T specifies the path from input B' into gate 3, through gates 6 and 7 to the output, while literal A_{2567} identifies the path from input gate 2, through gates 5, 6, and 7 to the output. In general, if the number of inversion elements within a path (i.e., NOT, NAND, and NOR gates) is odd, the corresponding variables will appear in the equivalent normal form in a polarity opposite (say, A $\longrightarrow$ A', or D' $\longrightarrow$ D, etc.) to the one it has in the circuit. If the number of inversion elements

is even, the polarity of the input variable is unaltered. Although the circuit of Fig.5 is in minimal form, its corresponding equivalent normal form contains several logically redundant terms, e.g., the fifth term. Note also that, for the prupose of fault detection, the literals D'_{12567} and D'_{367} are not equivalent, since they identify two different paths.

We will also need the complemented equivalent normal form when we discuss the "Scoring method" in section IV. In that section, we only test the s-a-1 fault in both equivalent normal form and its complemented equivalent normal form, instead of testing the equivalent normal form for both s-a-0 and s-a-1 faults. The complemented equivalent normal form for our running example is found by complementing Eq. 3 and expanding it. This results in

$$\begin{aligned} T' = {} & A'_{2567}B'_{47} + A_{247}A'_{2567}C_{1247} + A_{247}A'_{2567}D'_{1247} \\ & + B'_{47}C'_{12567}D_{12567} + A_{247}C_{1247}C'_{12567}D_{12567} \\ & + A_{247}C'_{12567}D'_{1247}D_{12567} + B'_{47}B_{367}D_{367} \\ & + A_{247}B_{367}C_{1247}D_{367} + A_{247}B_{367}D'_{1247}D_{367} \end{aligned} \tag{5}$$

A s-a-1 test for A'_{2567} in Eq. 5 is a s-a-0 test for A_{2567} in Eq. 4.

In order to simplify the notation, let us replace the subscript sequences in Eqs. 4 and 5 as follows: $247 \longrightarrow \alpha$, $47 \longrightarrow \beta$, $1247 \longrightarrow \gamma$, $2567 \longrightarrow \delta$, $367 \longrightarrow \epsilon$, $12567 \longrightarrow \lambda$. The modified equations are repeated as follows:

$$T = A'_{\alpha}B_{\beta} + B_{\beta}C'_{\gamma}D_{\gamma} + A_{\delta}C_{\lambda}B'_{\epsilon} + A_{\delta}C_{\lambda}D'_{\epsilon} + A_{\delta}D'_{\lambda}B'_{\epsilon} + A_{\delta}D'_{\lambda}D'_{\epsilon} \tag{6}$$

$$T' = A'_\delta B'_\beta + A_\alpha A'_\delta C_\gamma + A_\alpha A'_\delta D'_\nu + B'_\beta C'_\lambda D_\lambda + A_\alpha C_\gamma C'_\lambda D_\lambda$$
$$+ A_\alpha C'_\lambda D'_\nu D_\lambda + B'_\beta B_\epsilon D_\epsilon + A_\alpha B_\epsilon C_\gamma D_\epsilon + A_\alpha B_\epsilon D'_\nu D_\epsilon \qquad (7)$$

II-4-2 Design of the Experiment

We shall now use the ideas of "path sensitization" and "equivalent normal form" for the design of fault-detection experiments for multilevel combinational circuits. (These two ideas will also be used again in section IV for designing tests by using the method called the "Scoring Function".) An experiment will be designed for the equivalent normal form (two-level form) of the mutilevel circuit, and the experiment can be shown to be valid for the original multilevel circuit as well[14]. The technique is rather complicated, and involves separate computations for s-a-0 and s-a-1 faults. It also depends on the order of testing of the literals. This order is determined by assigning a score to each of the inputs of the equivalent normal form, and, as is often the case, different scoring procedures yield different experiments. The testing method presented in this section (no scoring is used) overcomes some of these limitations. However, like the method in reference (5), it does not guarantee minimal experiments, nor is there a guarantee that a set of sensitized paths can be found for every circuit. The equivalent normal form of a switching circuit is obtained by expressing the circuit output as a sum of products such that each path has a corresponding distinct product term. Thus the equivalent normal form for the circuit of Fig. 5 is shown in Fig.5-1.

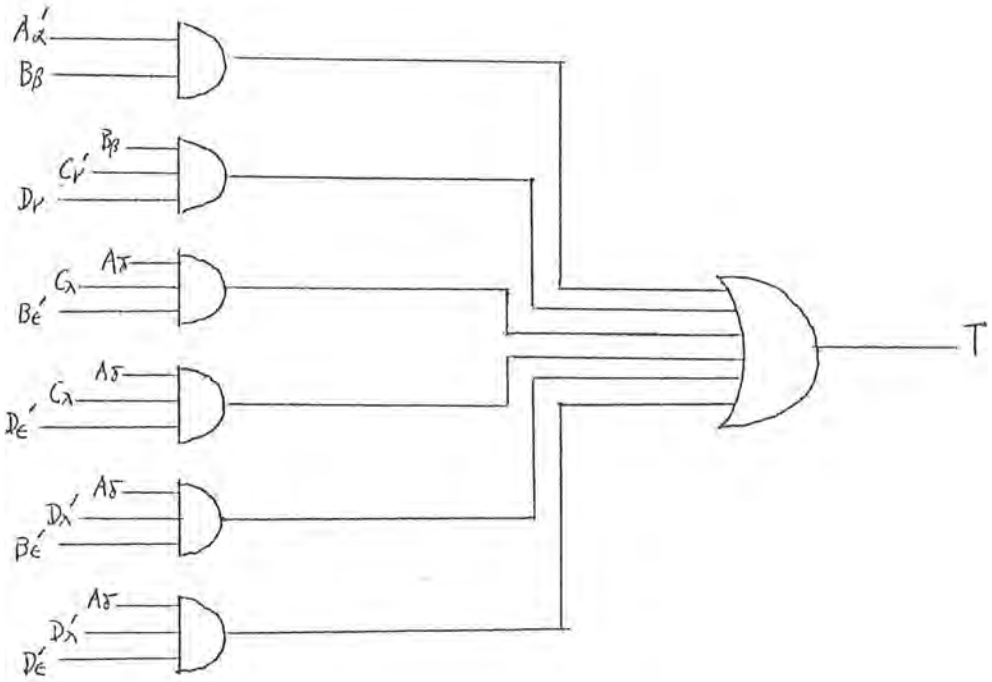


Fig. 5-1 ENF for circuit in Fig. 5.

The techniques developed earlier for two-level circuits can now be applied except for the situation where the signal diverges into two different paths and then reconverges. In our example A'_α corresponds to the upper path and A_δ corresponds to the lower one. Two cases must now be considered:

1) Both paths have either an even or an odd number of inversions in them. (An inversion is obtained by a NOT, a NAND, or a NOR gate.) If these literals appear at different AND gates in the equivalent normal form, they may be tested simultaneously for s-a-1 faults. Simultaneous s-a-0 tests, however, should be avoided.
2) One path has an even number of inversions, while the other has an odd number of inversions. In this case no input combination can

be selected so as to test two complementary literals simultaneously for s-a-0 and s-a-1 faults. An immediate corollary is that if two complementary literals appear as inputs to the same AND gate in the equivalent normal form, no test can be found for any literal at the input of that gate.

The fault-detection experiment for the equivalent normal form in Fig. 5 will now be constructed. As before, we shall use the tabular method. Each distinct literal is assigned at the head of a column, and every s-a-0 and s-a-1 test is assigned as a row heading. Literals associated with the same input but different paths (e.g., A'_α, A_δ) are grouped together. The testing table is shown in Fig. 6. The table entries can be found with the aid of a map, Fig. 7, literal by literal.

		A'_α	A_δ	C_λ	C'_ν	D'_λ	D_ν	B_β	B'_ϵ	D'_ϵ
s-a-1 tests	0							X		
	1							X		
	2		X					X		
	3		X					X		
	9					X		X		
	15	X			X				X	X
s-a-0 tests	4							X		
	5							X		
	6							X		
	7	X						X		
	8		X			X				
	10		X							
	11		X	X					X	
	12									X
	13				X		X	X		
	14									X

Fig.6 Testing table for equivalent normal form

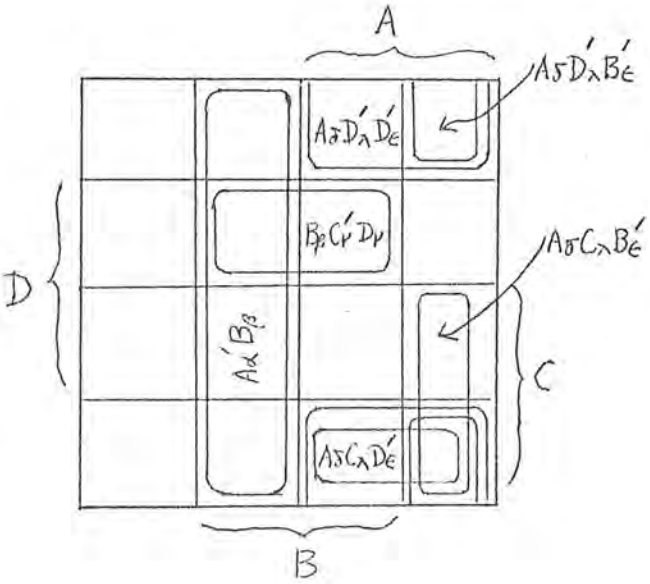


Fig.7 Equivalent normal form map

All the procedures are the same as in the two-level circuit except for that the one special situation listed in page 18 should be included.

As an example, then, we shall consider two of the tests required in the test set on the circuit of Fig. 5. Only two examples will be listed here for clarity:

1) Literal A'_α : Form the map in Fig. 7 we observe that A'_α is contained in only one subcube. Theoretically, the possible s-a-0 tests are 4, 6, and 7. But since 4 and 6 are adjacent to a subcube containing A_δ, they are invalid tests, because each of them tests simultaneously the same original input in two complementary forms for s-a-0 and s-a-1. Thus the only s-a-0 test is 7. The s-a-1 tests are determined by observing that 15 is the only adjacent cell for which T = 0 and in which the value of A is complementary to its value in the subcube being tested. We now enter the x in the testing table in column A'_α rows 7 and 15.

2) Literal D'_λ : This literal is contained in two true subcubes. One of these subcubes is $A_\delta D'_\lambda D'_\epsilon$. But because this subcube contains two entries of the same variable (i.e., D'), which correspond to different inputs in the original circuit, none of the D' inputs in this subcube can be tested for s-a-1 faults. Hence the s-a-1 test is 9. (Test 15 is invalid because it tests only the "untestable" $A_\delta D'_\lambda D'_\epsilon$) the s-a-0 test is 8, because 10 and 14 are contained in other true subcubes while 12 is adjacent to subcube $B_\beta C'_\gamma D_\gamma$ and will simultaneously test D_γ for s-a-1.

If we carry out this procedure we shall find that the complete experiment is $\{7, 8, 9, 11, 13, 15, (0 \text{ or } 2 \text{ or } 3), (12 \text{ or } 14)\}$ in this case.

We will see that this result is the same as in section IV which discusses the "Scoring function" method.

The main reason for not being able to obtain minimal experiments is not due to the testing table, but to the way the equivalent normal form is constructed. A single circuit input may result in a number of different literals in the equivalent normal form. Each of these literals must now be tested for s-a-0 and s-a-1 faults. This may result in redundant tests, which will increase the length of fault detection test. Again, we can distinguish between s-a-0 and s-a-1 faults if the circuit does not operate properly.

III Fixed schedule method: ($F \longrightarrow$ G matrix method) [6]

The fixed-schedule solutions offered here can be carried out satisfactorily by hand for small networks, and on a digital computer for any combinational network having up to eight to ten inputs, several outputs, and no more than about 100 faults. While some networks, much larger than this, can also be handled, at present there are no procedures for generating even reasonable good (almost minimum) test schedules for very large arbitrary networks. This problem and the problem of fault diagnosis in sequential networks are considered to be the most important subjects for further research on this area.

Some of the procedures described below are contained implicitly

in the literature (7), (8).

Given a single-output combinational network, an analysis of it can be carried out in order to determine the effect that each of various hypothetical faults has on its output. The result of such an analysis may be expressed in a multioutput table of combinations like the one below:

x_n ----- x_2 x_1	f_0	f_1 f_2 ------ f_j -----
0 ----- 0 0	0	1 0 ------- 0 -------
0 ----- 0 1	1	1 0 ------- 0 ------
0 ----- 1 0	0	1 0 ------- 1 ------
⋮	⋮	⋮ ⋮ ⋮
1 ----- 1 1	0	0 0 ------- 1 -------

where x_1, x_2, ..., x_n are the input variables to the network; $f_0=f(x_1,x_2,\ldots,x_n)$ is the the fault-free (correct) output, and f_1, f_2,...., f_j,.... are the erroneous outputs, each corresponding to one of the possible faults which the desired test schedule is supposed to check. The left side of the table simply lists all 2^n possible combinations of the input variables. Since we are going to consider irredundant networks, no column f_j is identical to column f, also no two columns f_j and f_k are identical. So all of the columns $f = f_0, f_1, f_2, \ldots, f_m$ (say, for m distinguishable faults) will be different. We may collect these m+1 columns into a 2^n row binary "fault table" or "fault matrix" F:

$$F = \begin{bmatrix} 0 & 1 & 0 & \cdots \\ 1 & 1 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots \end{bmatrix}$$

If a fixed schedule of input tests is to be employed to check a possibly faulty network, we are interested in economizing on the number of different test inputs (i.e., the "length") of such a test. The problem is therefore one of selecting a minimal subset of rows of the matrix F that preserves a certain degree of distinguishability among the columns. More precisely, for the "detection" of any of the m faults, we want to delete from F as many rows as possible so that the first column is different from all other columns.

We now show how the fault-detection problem may be converted to a familiar switching circuit minimization problem. Since we are only concerned about maintaining the distinctness between certain columns of F, we may conveniently express each of the conditions given above in terms of matrix G, each of those columns is the modulo-2 sum of a different "pair" of columns of F that are supposed to remain different. That is, in the same row of both of the matrices F and G, a 1 is entered in that column of matrix G labeled with the pair $(0,j)$ if the digits in the two columns of matrix F labeled f_0 and f_j are different, otherwise, a 0 is entered. When corresponding rows of F and G are deleted during minimization, two columns of F will then remain distinct if and only if the corresponding single column of G does not become a column of all 0's. Thus all the conditions stated before may be expressed as a single condition on the G matrix, namely:

Delete from G as many rows as possible, so that every column is nonzero. (Condition X)

The G matrix has just m columns, one for each column pair (f_0, f_j) $(j = 1, 2, \ldots\ldots, m)$ in F. For the above example, we have

$$G = \begin{array}{c} \begin{matrix} o1 & o2 & & om \end{matrix} \\ \begin{bmatrix} 1 & 0 & \cdots & \\ 0 & 1 & \cdots & \\ 1 & 0 & \cdots & \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \end{bmatrix} \end{array}$$

Condition X expresses precisely the problem of finding a minimal prime-implicant cover of a given switching function from its prime-implicant table. Good solutions to this problem are well known and have been programmed for execution on computers for quite large tables.[9] We shall now describe a version of this procedure which is adequate for solving simple problems by hand, and which also illustrates the two main steps in all of the programmed algorithms, as follows:

1) Simplification of the table to delete certain superfluous rows and columns.
2) Final selection of one or more minimal row subsets from the residual table.

The justification of the following simplifications for step 1) above is fairly obvious.

a) Delete any row whose 1's all fall in the same columns as the 1's in some other row, that is, delete any row which is covered by, or is the same as, some other row.
b) Delete any column which has 1's in all of the rows in which another column has 1's. That is, delete any column which covers, or is the same as, some other column.

These steps may be applied in any order until neither is applicable.

The selection of a minimal row subset S is made by first labeling the rows of the simplified matrix G* with binary variables a,b,c,..... each of which will have the value 1 if its row is to be included in S, otherwise its value will be 0. We now form a Boolean function L(G*) as a product of sums, one sum per column of G*, such that each sum contains just those row variables assigned to rows in which the corresponding column of G* has 1's. The function L(G*) will therefore have the value 1 when and only when a sufficient subset of row variables a,b,c,... has the value 1, namely, when every column G* is represented. Expansion of this product of sums into a sum of products then discloses as individual products all of the alternative row subsets which satisfy the column condition X. This allows the set S to be selected on the basis of any product which has the least number of variables.

As an example of this procedure, consider the following circuit for faults for which n = 4 and we have m = 4 the F and G matrice are thus:
(Note: I use the same example as in section II-2, i.e., F = A'B+BD+ACD)

$$F = \begin{array}{cccccc} & 1 & 2 & 3 & 4 & \\ & A & B & C & D & \\ \left(\right. & 0 & 0 & 1 & 0 & 0 \left.\right) & a & 0000 \\ & 0 & 0 & 1 & 0 & 0 & b & 0001 \\ & 0 & 0 & 1 & 0 & 0 & c & 0010 \\ & 0 & 1 & 1 & 0 & 0 & d & 0011 \\ & 1 & 0 & 0 & 1 & 1 & e & 0100 \\ & 1 & 1 & 0 & 1 & 1 & f & 0101 \\ & 1 & 0 & 0 & 1 & 1 & g & 0110 \\ & 1 & 1 & 0 & 1 & 1 & h & 0111 \\ & 0 & 0 & 0 & 0 & 0 & i & 1000 \\ & 0 & 0 & 1 & 1 & 0 & j & 1001 \\ & 0 & 0 & 0 & 0 & 1 & k & 1010 \\ & 1 & 0 & 1 & 0 & 0 & l & 1011 \\ & 0 & 1 & 0 & 0 & 1 & m & 1100 \\ & 1 & 1 & 0 & 1 & 0 & n & 1101 \\ & 0 & 1 & 0 & 0 & 1 & o & 1110 \\ \left.\right) & 1 & 1 & 1 & 1 & 0 \left.\right) & p & 1111 \end{array}$$

$$G = \begin{array}{cccc} 01 & 02 & 03 & 04 \\ \left(\right. & 0 & 1 & 0 & 0 \left.\right) \\ & 0 & 1 & 0 & 0 \\ & 0 & 1 & 0 & 0 \\ & 1 & 1 & 0 & 0 \\ & 1 & 1 & 0 & 0 \\ & 0 & 1 & 0 & 0 \\ & 1 & 1 & 0 & 0 \\ & 0 & 1 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & 0 & 1 & 1 & 0 \\ & 0 & 0 & 0 & 1 \\ & 1 & 0 & 1 & 1 \\ & 1 & 0 & 0 & 1 \\ & 0 & 1 & 0 & 1 \\ & 1 & 0 & 0 & 1 \\ \left.\right) & 0 & 0 & 0 & 1 \left.\right) \end{array}$$

After simplification by using rules (a) and (b), we will have:

$$G^* = \begin{array}{c} \begin{matrix} 01 & 02 & 03 & 04 \end{matrix} \\ \left[\begin{matrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{matrix}\right] \end{array} \begin{matrix} g \\ j \\ l \\ n \end{matrix}$$

$$L(G^*) = (g \vee l)(g \vee j \vee n)(j \vee l)(l \vee n)$$

$$= lg \vee lj \vee ln \vee \cdots\cdots$$

One minimal set S therefore consists of rows l and g. (Each group separated by symbol V contains at least two items. Reader may verify it.) So the length of fault-detection test in this case is two, which is shorter than six obtained in section II-2.

IV Scoring Function method (14):

IV-1 The Underlying Principles and Scoring Procedure

Our first task is to select, whenever possible, among all sets of paths which cover all network interconnections, one set to which there corresponds a minimal set of fault-detection tests. In order to obtain a minimal set of paths, it is generally necessary to check all of them, a task we try hard to avoid, since it implies, again, a covering problem, which has been shown to be impractical for even medium-size networks. Instead, we aim for less and try to obtain a practically effective experiment. In most cases, however, the experiments derived using the algorithm presented in this section are nearly minimal, and often absolutely minimal.

Consider the circuit in Fig. 8 (same as Fig. 5) and Eqs. 6a and

7a. The first test to be chosen will be one which tests for either s-a-1 or s-a-0 for as many different literals of the equivalent normal form as possible. The tested literals are checked off, and the next test is chosen so as to test as many as possible of the remaining literals, and so on. The smallest number of tests so selected, which test at least one appearance of every literal for both s-a-1 and s-a-0 faults, constitutes the fault-detection experiment.

In order to sensitize as many paths as possible in one test, values are initially assigned to the smallest possible number of input variables necessary for the sensitization of at least one path. Among the yet unassigned input variables, another minimal subset is selected, and values are assigned so as to sensitize one or more additional paths. This process terminates when values have been assigned to all input variables. The assignment of values is based on two criteria. First, a product term in the equivalent normal form (equivalent to a two-level circuit) containing the least number of literals is chosen. This ensures that the assignment of values to these literals will leave as many literals as possible unassigned, thus retaining maximum freedom in assigning values in the remaining unassigned terms. Second, aomong the literals of the term selected for assignment, find the literal whose complement has a maximum number of appearances in other product terms of the equivalent normal form. Assign value 0 (for s-a-1 test) to that literal and value 1 to the remaining literals in the term. As a result, a large number of literals in other terms are assigned a value 1, thus improving the chances of sensitizing as many paths as possible in one test, because the desired condition for s-a-1 tests is to assign just a single 0 in each term and

assign the remaining literals the value 1.

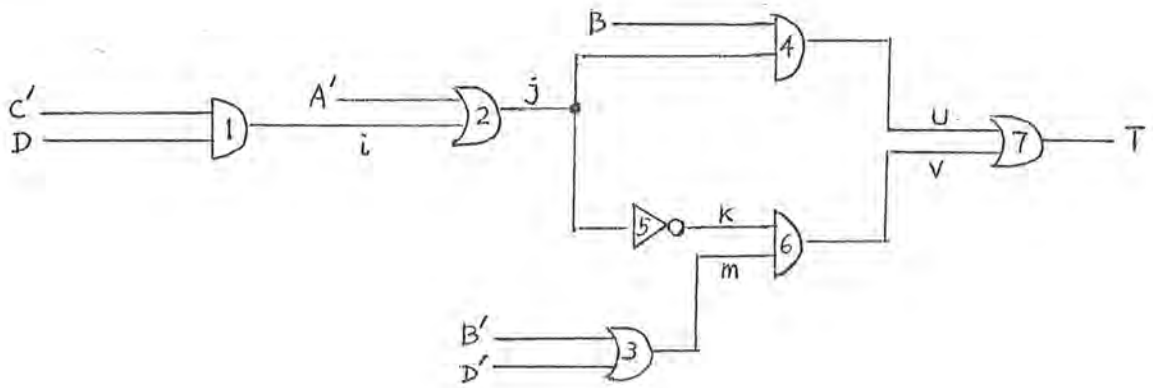


Fig.8 A circuit to be tested

Again:

$$T = A'_{\alpha}B_{\beta} + B_{\beta}C'_{\gamma}D_{\gamma} + A_{\delta}C_{\lambda}B'_{\epsilon} + A_{\delta}C_{\lambda}D'_{\epsilon} + A_{\delta}D'_{\lambda}B'_{\epsilon} + A_{\delta}D'_{\lambda}D'_{\epsilon} \quad (6a)$$

where $\alpha \longrightarrow 247$, $\beta \longrightarrow 47$, $\gamma \longrightarrow 1247$

$\delta \longrightarrow 2567$, $\epsilon \longrightarrow 367$, $\lambda \longrightarrow 12567$

First we note that the term containing the least number of literals is $A'_{247}B_{47}$. Next it is necessary to check whether the complement of A' (i.e., A) or that of B (i.e., B') has the larger number of appearances in the remaining terms. Since A appears in four other terms while B' appears in only two terms, A'_{247} is assigned the value 0 and B_{47} is assigned 1. This assignment, which sensitizes one path in the circuit, specifies the value of variables A and B to 1 in all remaining terms and leaves the values of C and D unspecified. The specification of these latter variables is accomplished in a similar manner so as to achieve a maximum number of sensitized paths in one assignment of values. Before proceeding

with this example, we shall further develop the experiment-construction procedure. The construction of s-a-0 tests requires different criteria from the ones used for s-a-1 tests. But since two sets of criteria may considerably complicate the procedure, it is preferable to find whether s-a-0 faults can be detected by employing just s-a-1 type tests. This can indeed be accomplished by using the complemented equivalent normal form network. In fact, a s-a-0 test for a particular literal in the equivlaent normal form network is a s-a-1 test for the corresponding literal in the complemented normal form network. Hence, instead of testing the equivalent normal form for both s-a-0 and s-a-1 faults, it is sufficient to test both the equivalent normal form and its complement for only s-a-1 faults.

The complemented equivalent normal form of circuit Fig. 8 is

$$T' = A'_\delta B'_\beta + A_\alpha A'_\delta C_\gamma + A_\alpha A'_\delta D'_\gamma + B'_\beta C'_\lambda D_\lambda + A_\alpha C_\gamma C'_\lambda D_\lambda + A_\alpha C'_\lambda D'_\gamma D_\lambda + B'_\beta B_\epsilon D_\epsilon + A_\alpha B_\epsilon C_\gamma D_\epsilon + A_\alpha B_\epsilon D'_\gamma D_\epsilon \qquad 7(a)$$

The determination of the order for the testing of literals cannot be accomplished effectively by mere inspection of the equations of T nnd T'. The problem becomes more involved because a s-a-1 test in T may detect in addition many s-a-0 faults, which are s-a-1 faults in T', as demonstrated by the analysis of the sensitized path in the circuit of Fig.8. Therefore the next test is constructed according to the appearances of the unchecked literals in both the equivalent normal form and its complement. To keep track of the order of testing literals, so that the previously specified criteria will be satisfied, we derive a scoring function which assigns a score to each literal appearance. At any stage of the construction of the tests, the literal to be assigned is

selected among those as yet unchecked on the number of whose literal appearances possesses the highest score.

It must be emphasized from the outset that, like many other scoring functions, the function to be derived in this section is not unique. We have, however, confidence that different scoring functions, based on the above ideas, when appropriately derived, will assign identical, or "almost identical", relative scores.

To illustrate the process, let S_k be the s-a-1 score for the kth literal in the equivalent normal form. Let n be the total number of variables, and w be the total number of literal appearances. Thus, in the equivalent normal form shown in Eq. (6a), $n = 4$ and $w = 17$.

Now define m_j to be the number of distinct variables in the jth term of the equivalent normal form, where this term contains the kth literal. Then the first criterion is covered by the term $(1-m_j/m)$. Also let p be defined in the following manner:

$$p = \begin{cases} \text{Number of unprimed appearances of the ith variable in the equivalent normal form, if the kth literal is primed.} \\ \text{Number of primed appearances of the ith variable in the equivalent normal form, if the kth literal is unprimed.} \end{cases}$$

The relative weights due to the second criterion can thus be represented by the term p/w. The scoring function is therefore given by:

$$S_k = \left(1 - \frac{m_j}{m}\right) + \frac{p}{w}$$

Before assigning the scores of each variable, it is convenient to compute the number of primed and unprimed appearances of each variable

in T, as shown below:

A = 4	C = 2
A'= 1	C'= 1
B = 2	D = 1
B'= 2	D'= 4

To find the score for A_δ in the last term of Eq. (6a), we write

$$S_{A_\delta} = (1 - \frac{2}{4}) + \frac{1}{17} = \frac{152}{272}$$

The common denominator 272 has been selected for the scores of the literals in both T and T'. This denominator will subsequently be omitted. Note that $m_6 = 2$ in that term, although the number of literals in that term is 3. The scores for all literals in T are found in a similar manner, as shown in the second row of table 1.

IV-2 Deriving the tests

Two tables are constructed in this process: Table 1 for the equivalent normal form, and table 2 for its complement. The first row in each table lists the terms whose literals are testable for s-a-1 faults. Thus all six terms of T are listed in Table 1, while only five of the nine terms of T' are listed, because none of the literals in the remaining four terms are testable for s-a-1 faults, as will be explained later. The second row in each table contains the s-a-1 scores corresponding to each literal appearance, as derived earlier. No score is associated with the two appearances of variable D in the last term of Table 1, because a s-a-1 test for either literal requires an assignment of 0 to both, which invalidates the entire

test. The third row of each table gives an ordering of the literal appearances, derived according to their scores, that is, the literal whose score is the highest is labeled 1, the next highest 2, and so on. If two literals have identical scores, the ordering is arbitrary. It will be convenient in the following discussion to refer to each literal by the number assigned to it in this row. The remaining rows contain the binary values which should be assigned to each literal so that a s-a-1 test is achieved. As an example, we shall now derive the first tests generated by the algorithm.

In the first step we select the literal appearance in the two tables whose score is the highest. This is literal 1 in Table 1, whose score is 200. Consequently, the first test to be derived is determined by means of Table 1. This test will detect at least s-a-1 faults in A' whithin the first term of T. According to the s-a-1 testing procedure, literal 1 is assigned the value 0, and the remaining literal in this term (i.e., literal 2) is assigned value 1. This assignment specifies the value of input variables A and B to be 1. Thus all appearances of the variables A and B in table 1 are assigned accordingly. It is now necessary to check whether the values of additional variables are determined by the assignment. In general, a test is unacceptable if a variable assignment is such that it produces one or more terms all of whose literals are assigned 1, because this implies that not all AND gates in the two-level equivalent-normal-form hypothetical circuit are disabled, which is contrary to the s-a-1 testing procedure. The last term in table 1 clearly implies that the value of D must be 1. This assignment in turn implies that variable C must be assigned 1. Therefore all appearances of variables D and C

Table 1 Tests for equivalent normal form

equivalent normal form	1 4 A_α' B_β	∨ 1 X B_β C_γ' D_γ	∨ X 8 A_δ B_ϵ' D_λ'	∨ 1 8 A_δ B_ϵ' C_λ	∨ ∨ 1 A_δ C_λ D_ϵ'	4 X X A_δ D_λ' D_ϵ'
score	200 168	100 100 132	84 100 84	84 100 84	84 84 84	152 — —
ordering of literals	1 2	5 6 4	9 7 10	11 8 12	13 14 15	3 — —
Test 1	0 1	1 0 1	1 0 0	1 0 1	1 1 0	1 0 0
Test 4	1 0	0 0 0	0 1 1	0 1 1	0 1 1	0 1 1
Test 8	0 0	0 1 1	1 1 0	1 1 0	1 0 0	1 0 0

Table 2 Tests for complemented equivalent normal form

complemented equivalent normal form	3 2 A_δ' B_β'	∨ 3 7 B_β' C_λ' D_λ	∨ 3 X B_β' B_ϵ D_ϵ	2 ∨ 5 ∨ A_α B_ϵ C_γ D_ϵ	X X 5 6 A_α B_ϵ D_γ' D_ϵ
score	170 187	119 85 85	187 187 –	17 51 17 17	— — 136 85
ordering of literals	4 1	6 7 8	2 3 –	11 10 12 13	— — 5 9
Test 2	1 0	0 0 1	0 1 1	0 1 1 1	0 1 0 1
Test 3	0 1	1 0 1	1 0 1	1 0 1 1	1 0 0 1
Test 5	0 0	0 1 1	0 1 1	1 1 0 1	1 1 0 1
Test 6	0 0	0 0 0	0 1 0	1 1 1 0	1 1 1 0
Test 7	0 1	1 1 0	1 0 0	1 0 0 0	1 0 1 0

are assigned accordingly. Since all four variables have been assigned values, the construction of the test is complete.

It now remains to determine the faults that are detectable by this test. Any s-a-1 fault in literal 1 is clearly detectable. Consequently, a 1 is placed above that literal appearance in table 1, to signify that it has been checked by test 1. Evidently, this test will also detect a s-a-1 fault at literal 6, because it is assigned value 0, while all remaining literals in the term are assigned 1. In a similar manner, we find that four distinct literals have been tested, which is recorded by placing a 1 above each of them, a check mark is also placed above all other appearances of these literals in the first row of table 1. Again, we can distinguish between s-a-0 and s-a-1 faults.

It is now necessary to explain the reasons for ignoring the four terms of T' while constructing table 2. Consider the ignored term $A_\alpha A'_\beta C_\gamma$, which corresponds to $A_{247}A'_{2567}C_{1247}$. An inspection of Fig.8 reveals that the paths associated with A_{247} and A'_{2567} emanate from gate 2, proceed through the upper and lower paths, and reconverge in gate 7. Since only the lower path contains an inverter, the values of A_{247} and A'_{2567} are complementary, regardless of whether their corresponding input A' to gate 2 is s-a-1 or is intact. Therefore neither A_{247} nor A'_{2567} can be tested in this particular term. Morever, since one of them will have the value 0 at all times, no other literal in this term can be tested. Consequently, the entire term can be disregarded when deriving the tests, an in the third term of Eq. (7a). Similar arguments show that the fifth and sixth terms of Eq. (7a) can also be ignored.

V Conclusions:

Given the four methods described, let us compare the bounds on length of tests for each method. If n is the number of variables of a circuit (it doesn't matter if it is a two-level or a multi-level circuit), then, whatever method is chosen, an upper bound for a test schedule is obviously 2^n. Even if the number of places, m, of possible faults in a circuit might greater than 2^n, the number 2^n is still the maximum number of possible input combinations. Hence the upper bound is $\min(2m,2^n)$, where 2m accounts for the two kinds of faults, s-a-1 and s-a-0. A lower bound for the minimum length of tests schedule required for each method is obviously 1. Those upper and lower bounds are valid for all of the four methods described before if we are only concerned with fault detection. (For fault location, the upper and lower bounds may be different if different methods are applied.[6]) So, in fact, no method is better than the others as measured by these bounds on the length of the test schedules.

However, the fixed test schedule method is really better than the other methods if $n \geq 5$. The most important reason for recommending this method is that it is very easy to program once the F matrix is established. The only thing we have to do is to establish the F matrix very carefully. One may get a different fault-detection test set with the different order of applying reduction rules (a) and (b) listed in page 24. From L(G*), we also know that there are more than one choice of the test set. However, all the test sets obtained from the fixed test schedule method must be a subset of the test set obtained from the map or tabular method. Also, all

the methods except the fixed schedule method can make a conclusion whether the fault is s-a-1 or s-a-0 after the fault is detected. A question is just what advantage or accrues if we can detect the two different faults. The fixed schedule method best for just detecting a fault and not determining whether it is s-a-1 or s-a-0 fault. The best method to use in order to determine the type of fault depends on the number of variables. If $n \leq 4$, the map method is applied (explained in next paragraph). After considering those situations, the fixed schedule method seems to be more practical and yields a shorter test for a two-level circuit, even though it may take some time to establish the F matrix. The fixed schedule method can be carried out satisfactorily by hand for small networks, and on a digital computer for any combinational network having up to eight to ten inputs, several outputs, and no more than about 100 faults.[6] While some networks much larger than this can also be handled, at present there are no procedures for generating even reasonable good (almost minimum) test schedules for very large arbitrary networks. This problem is considered to be a most important open problem on this area.

For small networks, $n \leq 4$, although the fixed schedule solutions can be carried out satisfactorily by hand, we prefer to use map method. The reason is that for $n \leq 4$, the Karnaugh map can be drawn very easily (at least easier than establishing the F matrix), also the test sets can be picked up from the map directly with the two steps listed in page 8.

The reader may wonder why we don't use equations (1) and (2) to derive fault detection tests directly. If a logical expression contains only

a few variables, then it will still be time consuming and impractical to apply equations (1) and (2) on each literal (see example on page 5 and 6). Therefore the map method is more practical.

It seems at present that the easiest way of designing minimal experiments for multi-level circuits is by transforming them into some sort of a two-level equivalent normal form, then using the method discussed in section III; i.e., let computer finish the calculations, or develop a different scoring procedure. Perhaps a different definition of the equivalent normal form, or another approach, may yield a more systematic procedure for obtaining minimal experiments.

For the multiple-output case, the entries in the f_j columns of the fault table F, as well as in the entire F matrix, are now q-digit binary numbers instead of single binary digits. Two entries of the fault table should be considered to be identical when and only when all of the corresponding q digits of the two entries are identical. Thus, two columns of the fault table can be considered to be the same only if all corresponding q-digit entries in these columns are the same. As before, the matrix G will have for its entries only single binary digits, according to the following rule: if two q-digit entries in the same row of F differ in any of their corresponding digits, then the corresponding entry in G for these two columns is a 1; otherwise it is a 0. This rule follows directly from the fact that a fault may be distinguished on any one or more of the q networks outputs. The rest of the minimization procedure on G is carried out just as it was for the single-output case. Note, however, that since the matrix G now usually

has a greater number of 1's in it, we can expect the length N of the test schedule to be smaller, assuming that the other parameters remain the same.

The following list summarizes some of the unsolved problems in this area of research:

1) Development of techniques for selecting an economical small number of test points or additional inputs for a given combinational network, in order to drastically reduce the length of the test schedule required for fault detection.
2) Development of practical procedures for deriving economical test schedules for very large combinational networks.
3) Study of the ways in which the original design of a digital network or subsystem may be modified in order to make it more susceptible to the diagnostic procedures.
4) Development of techniques for the design of special-purpose diagnostic circuitry, including estimates of the economies to be achieved through its use.

The following alternative approaches might lead to further insight:

1) Define another function which is different from the Boolean difference function so that one can detect a fault with minimal length tests.
2) Develop a program (i.e., give an algorithm) that accomplishes systematically the procedures discussed in section III.
3) Develop different definitions of the equivalent normal form, different scoring methods, or other different approach for

fault detection may yield better and more systematic procedures for obtaining minimal experiments.

So far we have been only concerned about the fault-detection of irredundant circuits. We would like to mention here that the fixed test schedule method can be extended and applied to fault-location of any irredundant circuit. Obviously, the establishment of F and G matrices will be more complicated than to just detect a fault.[6] Also, one may possibly define another function different from the Boolean difference function so that it can be used to locate the fault. So, these techniques can be expanded to "locate" a fault in any irredundant circuit as well.

References:

(1) L. W. Bearnson and Chester C. Carroll. "On the design of minimum length fault tests for combinational circuits". IEEE Trans. Electron. Comput., 1971 Jul-Dec.

(2) F. F. Sellers, M. Y. Hsiao, and L. W. Bearnson. "Analyzing errors with the Boolean difference". IEEE Trans. Comput. Vol. c-17, pp 676-683. July 1968.

(3) Zvi Kohavi, Dewayne A. Spires. "Designing Sets of Fault-Detection Tests for Combinational Logical Circuits". IEEE Trans. Comput. Vol. c-20. pp 1463, Dec. 1971.

(4) Z. Kohavi, Switching and Finite Automata Theory. New York: McGraw-Hill, 1970.

(5) D. B. Armstrong, "On finding a nearly minimal set of fault detection tests for combinational logic nets." IEEE Trans. Electron. Comput., Vol. EC-15, Feb. 1966, pp 66-73.

(6) William H. Kautz. "Fault Testing and Diagnosis in Combinational Digital Circuits". IEEE. Trans. Computer., pp 352-366, April 1968.

(7) A. Gill, "Minimum-Scan pattern recognition." IRE Trans. Information Theory, Vol. IT-5, pp 52-58, June 1959.

(8) E. C. Riekeman, A. Glovazky, and E. J. McCluskey, Jr., "Determination of redundancies in a set of patterns", IRE Trans. Information Theory (Correspondence), Vol IT-3, pp 167, June 1957.

(9) I. B. Pyne and E. J. McCluskey, Jr., "The reduction of redundancy in solving prime implicant tables". IRE Trans. Electronic Computers. Vol. EC-11, pp 473-482, August 1962.

(10) R. A. Short, "A theory of relations between sequential and combinational realizations of switching functions." Stanford Electronics Laboratories, Stanford University, Stanford, Calif, Rept. 098-1, 1962.

(11) --------, "The design of complementary output networks." IRE Trans. Electronic Computers, Vol. EC-11, pp 743-752, Dec. 1962.

(12) C. Y. Lee, "Representation of switching circuits by binary decision programs." Bell Sys. Tech. J., Vol. 38, pp 985-999, 1959.

(13) Herberty Y. Chang, "An algorithm for selecting an optimum set of diagnostic tests." IEEE Trans. Electronic Computers. Vol. EC-14, No. 5. pp 706-711. October 1965.

(14) Zvi Kohavi, "Switching and Finite Automata Theory." Section 8-4. "Constructing fault-detection experiments." pp 210-222. McGraw-Hill, Computer Science Series.