Circuit Structure and Switching Function Verification

MIN-WEN DU AND C. DENNIS WEISS

Abstract—A new approach is presented for the design of multiple fault detection tests in which the structure of a combinational circuit is used to reduce the number of input combinations required. The structure is defined by the interconnection of the basic elements, each of arbitrary complexity. The fault model assumes that the functions realized by the basic elements may undergo any deviation whatsoever, but that the circuit structure is fault free. Thus, arbitrary combinations of multiple faults within one or more basic elements are included in the model. Decomposition theory can be used to verify that a set of input combinations is a multiple fault detection test set under this model. A process called expansion will be introduced to simplify this task. A well-defined procedure is given for deriving a suitable test set which for some circuits is minimal or near minimal. It will yield a multiple fault detection test of length less than $2^n$ for any circuit with a nontrivial nondisjoint decomposition, defined by a basic-element partition. Higher order basic-element partitions are introduced as a generalization. An upper bound is given on the length of a multiple fault detection test for any circuit with a given structure, independent of the function realized on the structure. The bound is tighter when function information is also used.

Index Terms—Combinational logic networks, fault detection, functional decomposition, multiple faults.

I. INTRODUCTION

Designing tests for multiple faults in combinational circuits has proven to be extremely difficult, primarily because most approaches require the consideration of each combination of possible signal faults, usually assumed to be of the stuck-at-1 or stuck-at-0 variety. If there are $n$ lines in a circuit, there are $3^n - 1$ possible multiple stuck faults. Although these may be considered in equivalence classes [1], [2], the number of such classes may still be quite large.

The most familiar approaches to the design of fault detection tests involve either simulation [3] or employ the basic path sensitization idea [4]–[6], sometimes with the use of the Boolean difference [7]–[10]. In the present study, a quite different approach is possible, in which knowledge of the structure of a combinational circuit is used to reduce the number of input combinations required to test the circuit. The fault free approach employed in [11] also exploits circuit structure, and the results reported here can be incorporated into the procedures in [11].

A circuit's structure is defined by the interconnection of the basic elements, each of arbitrary complexity. The idea is to exploit the fact that the structure of a circuit implies constraints among certain sets of subfunctions of $f_C$, the circuit output function. These constraints enable one to draw inferences of the following form.

If a circuit $C$ with known structure produces outputs $f_C(S)$...
Fig. 1. (a) Circuit diagram. (b) Structure defined by G and H. (c) Normal circuit output function f for Example 1.

Fig. 2. f(S) for Example 1.

A combinational circuit C will be viewed as an interconnection of basic switching elements E₁, · · · , Eₖ. Each is either an individual gate or itself a combinational circuit with inputs and one output.² C will be assumed to be free of feedback loops and to have only one circuit output fC. Each input of a basic element is driven by either a primary input variable from the set I = {I₁, · · · , Iₙ}, or by the output of another basic element. 2[1] will denote the Cartesian product {0, 1}ⁿ.

Definition 1: The variation of a basic element Eᵢ, var (Eᵢ) is the set of possible faulty functions that Eᵢ may realize. Here we let \( \text{var}(Eᵢ) = \{g \in \{0, 1\}ⁿ : g(Iᵢ) \neq Eᵢ \} \), where \( mᵢ \) is the number of inputs to \( Eᵢ \). Thus, var (Eᵢ) is the set of all \( mᵢ \)-place switching functions. var (C) = var (E₁) × · · · × var (Eₖ) will denote the variation of C, where E₁, · · · , Eₖ are all the basic elements in circuit C.

Rather than selecting functions for var (Eᵢ) that could be obtained by assuming all possible combinations of single or multiple stuck faults on lines within \( Eᵢ \), we have chosen a more encompassing model that includes these as well as more general multiple faults due perhaps to incorrect part selection or wiring within any basic element \( Eᵢ \). This model has been explored by Kautz [12], Thurber [13], and Hornbuckle and Spann [14], ordinarily under the assumption that only a single basic element is faulty at any time. (In [12] and [13] only array circuits are considered.)

Let \( fₕ \) denote the correct output function realized by C, and let \( fᵢ \) denote a circuit output function obtained if, for all \( i \), the \( i \)th basic element in C realizes the \( i \)th component of \( g = f(h) = f(i₂) = 0 \). Finally, we conclude that S is in fact a multiple fault detection test since the correctness of the circuit output on S implies, using only structure information, that the circuit output is correct for all input combinations.

In subsequent sections, we will formalize and develop the ideas in Example 1 and show the following.

1) How to use the circuit structure to choose appropriate sets of subfunctions of the output function among which useful constraints hold.

2) How to find a subset \( S \) of input combinations such that if the circuit response to \( S \) is correct, then it is necessarily correct for all remaining input combinations regardless of what multiple fault may be present, as long as the assumed structural properties are unchanged.

II. FORMULATION OF THE FUNCTION VERIFICATION PROBLEM
$$(g_1, \ldots, g_k), g \in \text{var}(C)$$.

Then the set of functions realizable on $C$ is $F_C = \{f | f = g, g \in \text{var}(C)\}$. Note that $F_C$ does not depend on the particular functions realized by the basic elements in $C$ but only on the topological structure of the circuit, defined by the interconnections among $E_1, \ldots, E_k$.

Let $f_{obs}$ denote the output function realized by a possibly faulty circuit $C$ under observation and test. That is, $f_{obs}$ is an arbitrary function in $F_C$.

**Definition 2:** A set of input combinations $S$ is a function verification test set (FVTS) if

$$f_{obs}(S) = f_C(S) \Rightarrow f_{obs} = f_C.$$ 

That is, suppose the observed response of a possibly faulty circuit $C$ agrees with the normal output $f_C$ on $S$. Then if $S$ is an FVTS, this implies that the circuit realizes $f_C$, i.e., that $f_{obs}$ is identical to $f_C$.

**Definition 3:** A set of input combinations $S$ verifies a set $T$ if

$$f_{obs}(S) = f_C(S) \Rightarrow f_{obs}(T) = f_C(T).$$

Clearly, if $T = \{0, 1\}^n$, then $S$ is an FVTS for the circuit. In any event, $T$ satisfies the property $\forall f \in F_C, f(S) = f_C(S) \Rightarrow f(T) = f_C(T)$.

This paper develops techniques for deriving from $S$ a set $T$ verified by it. Although the present techniques do not always yield a maximal set $T$, they are computationally simpler than general procedures that do [11].

### III. Basic Element Partitions and Sets of Comparable Configurations

One can systematically derive from a circuit $C$ those sets of subfunctions of $f_C$ among which useful constraints exist. They are useful in the sense that they permit selection of a set $S$ that will verify a set $T$ properly containing $S$.

We begin by defining certain partitions of the input variable set $I$, each associated with a basic element of the circuit.

Consider the general partition of a circuit $C$ defined by basic element $E_j$ as in Fig. 3.2 $X$, $Y$, and $Z$ are disjoint subsets of the input variables $I$. More formally, let $\text{in}(E_j) = \{I_j | I_j \subset I \text{ and there exists a path from } I_j \text{ to } E_j\}$, $\text{out}(E_j) = \{I_j | I_j \subset I \text{ and there exists a path from } I \text{ to the circuit output that does not pass through } E_j\}$. In Fig. 3, $\text{in}(E_j) = X \cup Z$; $\text{out}(E_j) = Y \cup Z$.

**Definition 4:** For any basic element $E_j$ in $C$, a basic-element partition $P_j$ is defined by $P_j = [X|Y|Z]$ where $Z = \text{in}(E_j) \cap \text{out}(E_j)$, $X = \text{in}(E_j) - Z$, and $Y = \text{out}(E_j) - Z$. If $Z = \phi$, then write $P_j = [X|Y]$. If $|X| < 2$ or $Y = \phi$, then $P_j$ is called a trivial partition.

The significance of $P_j$ is that it defines a decomposition of the circuit output function $f_C(I)$ as

$$f_C(I) = f_j(h_j(X, Z), Y, Z)$$

where $h_j$ is the output function of basic element $E_j$ expressed in terms of primary input variables, and $f_j$ is the circuit output function expressed in terms of the output of $E_j$ and the primary input variables. Clearly, variables in $X$ control the circuit output only by affecting the value of $h_j$; and variables in $Y$ have no effect on the value of $h_j$. Note that if $Z$ is empty, (1) is called a disjoint decomposition of $f_C$; otherwise, it is a non-disjoint decomposition [15]–[18].

**Definition 5:** Let $P_j = [X|Y|Z]$ be a nontrivial basic-element partition of a circuit realizing $f_C$, and $x, y, z$ denote specific binary assignments to the variables in $X, Y$, and $Z$, respectively. Then a subfunction $f_C(X, y, z)$ will be called a configuration (CFG) of $f_C$. With $Z$ fixed at $z$, the set $\{f_C(X, y, z) | y \in \{1, 0\}\}$ is a complete set of comparable configurations associated with $P_j$.

Note that there are $|2^{[Z]}|$ complete sets of comparable configurations associated with the single partition $P_j = [X|Y|Z]$.

The following theorem is a generalization of the basic idea in Ashenhurst [15]. It establishes the constraints among the sets of subfunctions of $f_C$ identified before.

**Theorem 1:** For every complete set of comparable configurations, there exists a function $\phi(X)$ such that every subfunction in the set is one of the functions $0, 1, \phi(X), \psi(X)$.

**Proof:** If $P_j = [X|Y|Z]$, then with $z$ fixed and $y \in \{1, 0\}$, $f_C(X, y, z) = f_j(h_j(X, z), y, z) \in \{0, 1, \phi(X), \psi(X)\}$ where $\psi(X) = h_j(X, z)$. For a particular $P_j$ and $z$, $\phi(X)$ may be identically 0 or 1.

**Definition 6:** Every configuration $f_C(X, y, z)$ in a complete set will be referred to as a $0$-CFG, $1$-CFG, or $\phi$-CFG accordingly as $f_C(X, y, z) = 0, 1, \phi$, or $\psi(X)$.

$C$ is a tree circuit if every primary input variable and basic element output goes to only one basic element input, and there are no loops. For tree circuits, $Z = \phi$ in any basic-element partition $P_j$. For such a circuit, decomposition theory provides a simple description of the necessary and sufficient conditions on a function in order that it be realizable on a given treelike structure.

**Theorem 2—Ashenhurst [15], Karp [17]:** If $C$ is a treelike structure, then $f \in F_C$ if and only if for every $P_j = [X|Y]$, $\{f(X, y, z) | y \in \{1, 0\}\} \subseteq \{0, 1, \phi, \psi\}$ for some function $\phi(X)$.

**Example 2:** For the circuit in Fig. 4(a),
Fig. 5. $i_a$ and $i_b$ are comparable: $i_a \sim i_b$.

![Diagram](image)

**Fig. 6.** (a) If $i_a, i_b \in T$, the CFG is a $\psi$-CFG verified by $T$. (b) If $i_a, i_b \in T$, then this pair of CFG’s is linked by $T$. (c) If $i_a, i_b, i_c, i_d \in T$, then CFG is a $k$-CFG verified by $T$, where $k \in \{0, 1\}$.

input combinations forming the domain of the subfunction $f_C(x y z)$. Also, given two comparable configurations (from the same complete set) $CFG_1 = f_C(x y z)$, $CFG_2 = f_C(y x z)$, if $i_a = (x y z) \in D(CFG_1)$, $i_b = (x y z) \in D(CFG_2)$, then $i_a$ and $i_b$ are called comparable input combinations because they agree in the bit positions corresponding to the $X$ and $Z$ variables (see Fig. 5). This will be denoted $i_a \sim i_b$.

**Definition 7:** Let $T \subset 2^{[1]}$ be a set of input combinations.

a) A configuration $CFG = f_C(x y z)$ is a $\psi$-CFG verified by $T$ if there exists a pair $i_a, i_b \in D(CFG) \cap T$ such that $f_C(i_b) \neq f_C(i_a)$. See Fig. 6(a).

b) Let $CFG_1$ and $CFG_2$ be comparable $\psi$-CFG’s verified by $T$. They are linked by $T$ if there exists a pair $i_a \in (CFG_1) \cap T, i_b \in (CFG_2) \cap T$ for which $i_a \sim i_b$. See Fig. 6(b).

c) For $k \in \{0, 1\}$, $CFG_1$ is a $k$-CFG verified by $T$ if a comparable $\psi$-configuration $CFG_2$ and four input combinations are found with $i_a, i_b \in D(CFG_1) \cap T, i_c, i_d \in D(CFG_2) \cap T$ such that $i_a \sim i_c, i_b \sim i_d$ and $f_C(i_a) = f_C(i_b) = k, f_C(i_c) \neq f_C(i_d)$. See Fig. 6(c).

**Theorem 3:** Let $S \subset 2^{[1]}$ and $T$ be a set verified by $S$. Let $i \in ST$ denote that input combination $i$ can be added to $T$ (if not already present) without destroying its property of being verified by $S$.

a) $\psi$-implication rule: Let $CFG_1, CFG_2$ be two comparable $\psi$-CFG’s verified and linked by $T$. Then $i \in D(CFG_1) \cap T \Rightarrow i \in ST$ where $i \sim i$ and $i \in D(CFG_2)$.

b) 0-implication rule: Let CFG be a 0-CF verified by $T$. Then $i \in D(CFG) \Rightarrow i \in ST$.

c) 1-implication rule: Let CFG be a 1-CF verified by $T$. Then $i \in D(CFG) \Rightarrow i \in ST$.

The proof follows directly from Theorem 1. Note that in a),

---

**IV. USING COMPARABLE CONFIGURATIONS TO DERIVE INPUTS VERIFIED BY $S$**

Given a set $S \subset 2^{[1]}$, it is desired to derive a larger set $T$ of input combinations verified by $S$ (in the sense of Definition 3). This will be done using the constraints established by Theorem 1. Three implication rules will be applied recursively to derive new input combinations verified by $S$. To specify them, the following terminology is useful.

Let CFG denote some configuration $f_C(x y z)$ with domain $D(CFG) = \{(x y z) : x \in 2^{[1]}\}$. That is, $D(CFG)$ is the set of
$f_c(i) = f_c(i)$ if $\text{CFG}_1$ and $\text{CFG}_2$ are both $\psi$ or both $\neg \psi$; otherwise, $f_c(i) = \neg f_c(i)$. In b) $f_c(i) = 0$; in c) $f_c(i) = 1$.

We can now define a special set $T$ that is verified by a given set of input combinations $S$.

**Definition 8:** Let $S \subseteq 2^{|I|}$. Then $T$ is the basic expansion of $S$ for a circuit $C$ if $T$ is the largest set to which $S$ can be expanded by recursive application of the $\psi$-$\neg \psi$ implication, $0$-implication, and $1$-implication rules, employing all complete sets of comparable configurations defined by the nontrivial basic-element partitions of the circuit $C$.

It is easily seen that the order in which $S$ is expanded does not affect the final set $T$, so that the basic expansion set $T$ is uniquely defined by $S$ and the normal circuit $C$.

**Example 3:** Consider the circuit in Fig. 7(a) with nontrivial basic-element partitions $P_2 = \{i_2, i_3, i_4, i_5\}$, $P_3 = \{i_3, i_4, i_5\}$. Let $S = \{i_0, i_1, i_8, i_9, i_{12}, i_{13}\}$. The basic expansion set $T$ is found as follows.

a) $\text{CFG}_2$ is a $\psi$-CFG verified by $S$ and $\text{CFG}_1$ is a $0$-CFG verified by $S$. Using the $0$-implication rule on $\text{CFG}_{i_2}, \text{CFG}_{i_3}, \text{CFG}_{i_4}, \text{CFG}_{i_5}, \text{CFG}_{i_6}, \text{CFG}_{i_7}, \in T$.

b) $\text{CFG}_3$, $\text{CFG}_4$ are linked $\psi$-CFG's verified by $S$. The $\psi$-$\neg \psi$ implication rule implies $i_{10}, i_{13} \in T$. No further rules are applicable, so that $f_c(T)$ is shown in the final map in Fig. 7(c).

**Theorem 4:** A sufficient condition that $S \subseteq 2^{|I|}$ be an FVTS for a circuit $C$ is that $T = 2^{|I|}$, where $T$ is the basic expansion of $S$.

**Proof:** This is a direct consequence of Theorem 1. It is not, in general, necessary that $T = 2^{|I|}$ if $S$ is an FVTS, as shown by the following example.

**Example 4:** Consider the circuit shown in Fig. 8. There are $2^2 = 2^4$ distinct functions from $2^{|I|}$ to $\{0, 1\}$, but the above structure can realize at most $2^2 \cdot (2^2)^2 = 2^6$ functions, the total number of distinct ways of assigning functions to the basic elements. Hence, there exists $f \notin F_C$. But then there exist $f_i \in F_C, f_2 \notin F_C$, which differ for only one domain element $u$.

To see this, let $g_1 = f_c \in F_C$, and define a sequence of functions $g_1, \cdots, g_r$ where $g_j$ differs from $g_{j+1}$ for only one domain element, $j = 1, \cdots, r - 1$, and $g_r = f \in F_C$. Let $g_1$ be the last function in the sequence that is a function in $F_C$. Then let $f_1 = g_1, f_2 = g_2 \cdots, g_{r-1}$. We have $f_1(\{u\}) = f_2(\{u\})$ and $f_1(u) \neq f_2(u)$. But now $S = 2^{|I|} - \{u\}$ is an FVTS of $f_1$ because the structure can realize $f_1$ since $f_1 \in F_C$, but cannot realize $f_2$ since $f_2 \notin F_C$. On the other hand, $S = T$ for $f_1$ since the above circuit has no nontrivial basic-element partitions. Therefore, the converse of Theorem 4 is not always true.

We conjecture that the converse of Theorem 4 is true for tree circuits. No proof has yet been found, but efforts at constructing counterexamples have failed. In particular, if $T \neq 2^{|I|}$, where $C$ is a tree circuit, then the following construction appears to always yield a function $f \in F_C, f \neq f_c$. Let the basic expansion of $S$ be $T_i \subseteq 2^{|I|}$. Then select some $u_i \notin T_i$ such that the basic expansion $S \cup \{u_i\}$ is $T_i \subset 2^{|I|}$. Continue until the basic expansion of $S \cup \{u_1, \cdots, u_i\}$ is $T_i \subset 2^{|I|}$, but for every $u \notin T_i$, the basic expansion of $S \cup \{u_1, \cdots, u_{i-1}, u\}$ is $2^{|I|}$. Then choose $f$ such that

\[
f(T_i) = f_c(T_i)
\]

In all cases investigated, $f$ does not violate Theorem 2 so that $S$ is not an FVTS of the tree circuit $C$.

**V. A PROCEDURE TO FIND AN FVTS FOR a CIRCUIT**

A simple procedure is presented here for finding an FVTS for a given circuit. The strategy will be to add input combinations to an originally empty set $S$ (and hence to $T$) so that $T$, the basic expansion of $S$, verifies all $\psi$, $0$, and 1-2-3's and all comparable $\psi$-CFG's are linked. $T$ is recomputed using the rules in Theorem 3 after each combination is added to $S$, thereby avoiding the placement of some unnecessary combinations into $S$. Finally, each input combination not in $T$ is added to $S$, one by one, again with recomputation of $T$ at each step, until $T = 2^{|I|}$.

**Procedure 1:** For constructing an FVTS for a circuit with at least one nontrivial basic-element partition.

In the following steps, combinations are added to $S$; and $T$, the basic expansion of $S$, is recomputed (using rules in Theorem 3) after each addition to $S$.

**Step 1:** Order the collection of complete sets of comparable
configurations (by size of configuration in the set, for example).

Step 2: In each complete set of comparable configurations, select one \( \psi^- \)-configuration \( C \) and add to \( S \) (as in Definition 3) a pair such that \( T \) verifies \( C \).

Step 3: For each \( 0 \)-CFG and \( 1 \)-CFG that is contained\(^4\) in a \( \psi^- \)-CFG verified in Step 2, do the following when possible: add combinations to \( S \) (as in Definition 7 c) so that \( T \) verifies these.

Step 4: If \( \psi^- \)-CFG is a \( \psi^- \)-CFG verified by \( T \), then for each \( \psi^- \)-configuration \( C \), comparable to it, add combinations to \( S \) (as in Definition 7 a, b) so that \( T \) verifies \( C \), and binds it to \( T \).

Step 5: Add combinations to \( S \) (as in Definition 7 c) so that as many of the remaining \( 0 \)- and \( 1 \)-CFG's as possible are verified by \( T \).

Step 6: One by one, add to \( S \) combinations not in \( T \) until \( T = 2[I] \).

Step 7 (optional): For each \( i \in S \), if the basic expansion of \( S - \{i\} \) is also \( 2[I] \), delete \( i \) from \( S \).

Example 5: For the circuit in Fig. 9, the nontrivial basic-element partitions are

\[
P_1 = [I_2 I_3 | I_4 I_6 I_6] \]
\[
P_2 = [I_5 I_6 | I_1 I_2 I_4] \]
\[
P_3 = [I_1 I_2 I_3 | I_4 I_6 I_5] \]
\[
P_4 = [I_4 I_5 I_6 | I_1 I_2 I_3] .
\]

Let \( R_j(C) \) denote the \( C \)-configuration (subfunction of \( f_C \) associated with the \( j \)-th row (ith column) of the map in Fig. 9(b)). Also, \( R_j(LEFT), R_j(RIGHT), C(TOP), C(BOTTOM) \) will denote configurations that are half rows and columns.

Step 1: The sets \( G_1, G_2, G_3, \) and \( G_4 \) of comparable configurations associated with \( P_1, P_2, P_3, \) and \( P_4 \) are columns, all half columns (top and bottom), and all half rows (left and right), respectively.

Step 2: In \( G_1 \), add \( i_0 I_0 \) to \( S \) (denoted \( \{i_0, i_0\} \rightarrow S \)) and \( T \) verifies \( \psi^- \)-CFG \( C_1 \). In \( G_2 \), \( \{i_1, i_2\} \rightarrow S \), and \( T \) verifies \( \psi^- \)-CFG \( C_1 \) (BOTTOM). In \( G_3 \), \( \{i_3\} \rightarrow S \), and \( T \) verifies \( \psi^- \)-CFG \( R_1 \). In \( G_4 \), \( \{i_4\} \rightarrow S \), and \( T \) verifies \( \psi^- \)-CFG \( R_1 \) (RIGHT).

Step 3: For the \( 0 \)-CFG \( C_1 \) (TOP) contained in \( C_1 \), let \( \{i_8\} \rightarrow S \) so that \( T \) verifies \( C_1 \) (TOP). [Note that \( C_1 \) (TOP) is comparable to the \( \psi^- \)-CFG \( C_1 \) (BOTTOM).]

For the \( 0 \)-CFG \( C_1 \) (LEFT) contained in \( R_1 \), let \( \{i_8\} \rightarrow S \) so that \( T \) verifies \( R_1 \) (LEFT). [Note that \( R_1 \) (LEFT) is comparable to the \( \psi^- \)-CFG \( R_1 \) (RIGHT).]

At this stage, input combinations in \( S \) are shown as starred locations in Fig. 9(c), and \( f_C(T) \) is the full set of values shown in Fig. 9(c).

Step 4: In order that \( T \) verify and link all \( \psi^- \)-CFG's comparable to \( C_1 \) (in \( G_1 \)), we can identify the \( 1 \)-CFG \( R_6 \) to serve as a \( 2 \)-axis\(^5\) for \( G_1 \). With \( \{i_5\} \rightarrow S \), \( T \) verifies this \( 1 \)-CFG.

\(^4\)CFG is contained in CFG if \( D(CFG) \subseteq D(CFG) \).

\(^5\)A \( 2 \)-axis (0-axis) through a set (not necessarily complete) of comparable \( CFG's \) is a \( 1 \)-CFG \( 0 \)-CFG) whose domain intersects that of each configuration in the set. Having verified such an axis, each \( \psi^- \)-CFG in the set is already linked to the others in the set, and each can be verified by the addition of at most one input combination to \( S \).

basic expansion \( T \) is now as shown in Fig. 9(d). It can be seen that Step 4 is already complete for \( G_1 \) and \( G_2 \). With \( \{i_6\} \rightarrow S \), \( R_8 \) becomes a \( \psi^- \)-CFG verified by \( T \), and with the verified \( 1 \)-axis \( C_6 \) [see Fig. 9(d)], Step 4 is complete for \( G_3 \) and \( G_4 \). Also, all \( 0 \)-CFG's are now verified by \( T \).

Step 5: With \( \{i_8\}, \{i_6\}, \{i_7\} \rightarrow S \), all \( 1 \)-CFG's are verified by \( T \).

Step 6: At this point, \( T = 2[I] \).

Step 7: None of the elements in \( S = \{i_0, i_1, i_4, i_5, i_6, i_7, i_8, i_9, i_{10}, i_{11} \} \) can be deleted, so that a length-12 FVT is obtained.

By applying somewhat more complicated rules, a length-10 FVTs \( S = \{i_0, i_5, i_{10}, i_{14}, i_{15}, i_{16}, i_{17}, i_{18}, i_{19}, i_{20} \} \) may be found.

Example 6: A nontree circuit that has nontrivial basic-element partitions is shown in Fig. 10(a). \( E_1, E_3, E_4, \) and \( E_7 \) each realize the odd parity functions of \( k \) inputs; \( E_2 \) and \( E_5 \) realize \( \text{nand} \); and \( E_4 \) and \( E_8 \) are \( \text{not} \) gates. The nontrivial basic-element partitions are \( P_5 = [DEF] \cap [AB], P_6 = [EFABC]D \).

The complete sets of comparable CFG's defined by \( P_5 \) are \( G_1 = \{R_1, R_2\}, G_2 = \{R_3, R_4\}, G_3 = \{R_5, R_6\}, G_4 = \{R_7, R_8\} \).
The complete sets of comparable CFG’s defined by $P_9$ are
$G_9 = \{ R1 (LETF), \cdots, R8 (LETF) \}$, $G_6 = \{ R1 (RIGHT), \cdots, R8 (RIGHT) \}$. Applying Procedure 1 to find an FVTS, we obtain the following.

In Step 2, $\{i_16, i_17\}, \{i_52, i_3\} \rightarrow S$. $R3, R3 (LEFT), R7$ and $R7 (RIGHT)$ are verified to be $\psi$-CFG’s.

In Step 3, $\{i_59, i_21\}, \{i_48, i_49\} \rightarrow S$. $R3 (RIGHT)$ and $R7 (LEFT)$ are verified to be 0-CFG’s.

In Step 4, $\{i_54, i_56\}, \{i_50, i_52\} \rightarrow S$. $R4$ and $R8$ are verified to be $\psi$-CFG’s. Also, they are linked to $R3$ and $R7$, respectively.

In Step 5, $\{i_0, i_1\}, \{i_6, i_3\}, \{i_52, i_53\}, \{i_40, i_41\}, \{i_4, i_5\}, \{i_1, i_3\}, \{i_46, i_47\}, \{i_44, i_45\} \rightarrow S$. All 0- and 1-CFG’s are verified.

In Step 6, $\{i_{10}\}, \{i_{15}\} \rightarrow S$. $T$ becomes $2^{[4]}$.

In Step 7, none of the elements in $S$ can be deleted. The final combinations in $S$ are starred in the map of Fig. 10(c). This is an FVTS of length 30.

VI. GENERALIZATION AND AN UPPER BOUND ON THE LENGTH OF THE MINIMAL FVTS

Theorem 1 can be generalized so that a wider class of circuits can be treated. An upper bound on the length of the minimal FVTS will then be derived.

In Theorem 1, each basic element is considered separately to find the basic-element partitions. When $k$ basic elements are considered at a time, a $k$th-order basic-element partition can be constructed.

Definition 9: Let $J$ be the index set of $k$ basic elements. The $k$th-order basic-element partition for $\{E_j ; j \in J\}$ is $P^k_J = [X|Y|Z]$, where $Z = \cup_{j \in J} \mathbb{I}n(E_j) \cap \cup_{j \in J} \mathbb{O}ut(E_j)$, $X = \cup_{j \in J} \mathbb{I}n(E_j)$, and $Y = \cup_{j \in J} \mathbb{O}ut(E_j) - Z$.

If $k = 1$, the first-order partitions of Definition 4 are obtained.

Theorem 5—Based on Karp [17]: Let $[X|Y|Z]$ be a $k$th-order basic-element partition. For any $z \in 2^{[Z]}$, the set of subfunctions $\{f_C(x_i, Y, z) : x_i \in 2^{[X]}\}$ contains at most $2^k$ distinct functions.

Proof: The partition defines a multiple nondisjoint decomposition of $f_C$. $f_C(x, Y, z) = g(\beta_1(X, Z), \cdots, \beta_k(X, Z), Y, z)$ where the $\beta_i$ are the outputs of the $k$ basic elements defining the partition. For $z \in 2^{[Z]}$, $\{f_C(x_i, Y, z) : x_i \in 2^{[X]}\} = \{g(\beta_1(x_i, z), \cdots, \beta_k(x_i, z), Y, z) : x_i \in 2^{[X]}\}$, where $(\beta_1(x_i, z), \cdots, \beta_k(x_i, z)) \in \{0, 1\}^k$. Thus, the set contains at most $2^k$ distinct functions; each of the form $g(u, Y, z)$ for $u \in \{0, 1\}^k$. This completes the proof.

The set of subfunctions $\{f_C(x_i, Y, z) : x_i \in 2^{[X]}\}$ are columns in the map of Fig. 11. Theorem 5 asserts that there are at most $2^k$ different columns on the map.

Theorem 5 can be used to modify Procedure 1 to derive function verification test sets using higher order basic-element partitions [19]. Here we will derive an upper bound for the minimal FVTS of a circuit $C$ based on a single $k$th-order basic-element partition.

In a map of $f_C(X, Y, z)$, if there are exactly $2^k$ distinct columns, then after verifying all function values appearing in one copy of each such column, we can verify all function values in any other column with at most $2^k - 1$ combinations. Thus, at most $2^k \cdot 2^{[Y]} + (2^k - 1) \cdot (2^k - 2^k)$ combinations are sufficient to verify $f_C(X, Y, z)$. On the other hand, if there are fewer than $2^k$ distinct columns, consideration of the CFG’s $\{f_C(x_i; x_j) : y_i \in 2^{[Y]}\}$ itself does not permit reduction in the number of combinations required; all $2^{[X \cup Y]}$ will be used.

From any $k$th-order basic-element partition $P^k_J = [X|Y|Z]$, an upper bound on the length of the minimal FVTS can be derived as follows. If there are $m$ maps $f_C(X, Y, z)$, each with $2^k$ distinct columns, then the length of the minimal FVTS is no greater than

$$L_j^k = m [2^k \cdot 2^{[Y]} + (2^k - 1) \cdot (2^k - 2^k)] + (2^{[Z]} - m) \cdot 2^{[X \cup Y]}.$$  

(2)

Thus $\min_{1 \leq k \leq 1} \min_{\text{over all } P^k_J} L_j^k$ is an upper bound on the length of the minimal FVTS of the circuit. In this formula,
is the largest integer solution to \( t + 1 + \log_2 (t + 1) \leq |I| \) because for a map to have more than \( 2^k \) columns and \( 2^k \) distinct columns, \( k \) must satisfy \( 2^k < 2^{|X|} \), \( k \leq 2^{|Y|} \). The case \( k = 2^{|Y|} \) leads to a map (subfunction) in which all combinations must be used since all \( k \) entries in each column must be known to classify it with respect to the \( 2^k \) distinct columns. Thus, if not all entries are to be used, \( k \) should satisfy \( k + 1 \leq 2^{|Y|} \). Therefore, it is necessary that \( k + 1 + \log_2 (k + 1) \leq |X| + |Y| \leq |I| \).

Let \([X|Y|Z]\) be a nontrivial first-order basic-element partition (\( k = 1 \)). Then an upper bound that is independent of the function realized on the structure is

\[
2 \cdot 2^{|Y|} + (2^{|X|} - 2) + (2^{|Z|} - 1) 
\]

\[
\cdot 2^{|X|}
\]

(3)

This follows from (2), for if the function is nondegenerate, \( m = 1 \) is guaranteed because at least one \( f_C(Xyz) \) is a \( \psi \)-CFG.

In the following example, we find an upper bound on an FVTS using a \( k \)th-order basic-element partition with \( k = 2 \).

**Example 7:** Consider the circuit in Fig. 12 with the second-order basic-element partition \([ABCD|EF\overline{GH}]\). Assuming only that there exist four distinct columns in the map shown, an upper bound on the length of an FVTS is \( 2^2 \cdot 2^4 + (2^2 - 1) \cdot 2^2 \cdot 2^2 = 100 \), whereas \( 2^{|I|} = 256 \).

If the circuit had only a single line from \( E_2 \) to \( E_1 \), then the bound in (3) can be applied (assuming \( f \) depends nonvacuously on at least one \( x_i \)). At most, 46 combinations are then required in an FVTS and, hence, in the most general multiple fault detection test. Note that only the structure of the circuit has been used to find this bound.

**VII. Conclusion**

A new approach has been presented for finding multiple fault detection tests. It is also expected to have application to the fault location problem and to the design of circuits that are easily tested. The basic expansion technique in Procedure 1 is computationally faster than finding the total expansion of a set of input combinations as discussed in [11], and can be incorporated into the more general algorithm.

**References**


Min-Wen Du (S'70--M'72), for a photograph and biography see page 240 of the March 1973 issue of this *Transactions*.

C. Dennis Weiss (S'62--M'66), for a photograph and biography see page 240 of the March 1973 issue of this *Transactions*. 