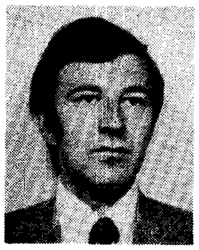
[6] T. L. Booth, *Sequential Machines and Automata Theory*. New York: Wiley, 1967.
[7] P. J. Denning and S. C. Schwartz, "Properties of the working set model," in *Ass. Comput. Mach. 3rd Symp. Operating Syst. Principles*, Oct. 1971, pp. 130-140.
[8] A. Paz, *Introduction to Probabilistic Automata*. New York: Academic, 1971.



Erol Gelenbe (S'67-M'70) was born in Istanbul, Turkey, on August 22, 1945. He received the B.S. degree in electrical engineering from the Middle East Technical University, Ankara, Turkey, in 1966 and the Ph.D. degree in computer science from the Polytechnic Institute of Brooklyn, N.Y., in 1969.

After obtaining his degree he was appointed to an Assistant Professorship at the Polytechnic Institute of Brooklyn and taught courses in programming and in switching and automata theory. During the summers of 1970 and 1972 he was a Visiting Scientist and Lecturer at the Philips Research Laboratories, Eindhoven, The Netherlands, where he did research in the modeling of computer operating systems. Since the fall of 1970 he has been on the faculty of the Department of Electrical and Computer Engineering and of the Graduate Program in Computer, Information and Control Engineering, University of Michigan, Ann Arbor, where he introduced new courses in compiler writing, programming languages, and operating systems theory. On leave of absence from the University of Michigan since January 1972, he has been at the research laboratory (LABORIA) of the Institut de Recherche d'Informatique et d'Automatique (IRIA), Rocquencourt, France, first participating in the design of an operating system, and then initiating and heading a research project in models of computer systems. His primary research interest at the present time is the quantitative evaluation and modeling of operating systems and of other information processing systems. He has published several papers in journals and at symposia on that subject, as well as on automata and formal language theory. He is a referee for several journals in computer science, and is a European Lecturer of the Association for Computing Machinery. He is active in ACM activities in Europe, is a member of the program committee of the 1973 ACM International Computing Symposium, and was co-chairman of the Symposium on Computers and Automata held in New York City in 1971.

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.

Manuscript received October 13, 1971; revised November 27, 1972. A preliminary version of this paper was presented at the 9th Allerton Conference on Circuit and System Theory, October 1971.

M.-W. Du was with the Department of Computer Science, The Johns Hopkins University, Baltimore, Md. 21218. He is now with the College of Engineering, National Chiao Tung University, Hsinchu, Taiwan, Republic of China.
C. D. Weiss was with the Departments of Computer Science and Electrical Engineering, The Johns Hopkins University, Baltimore, Md. He is now with Bell Laboratories, Holmdel, N.J. 07733.

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 $f_C(S)$

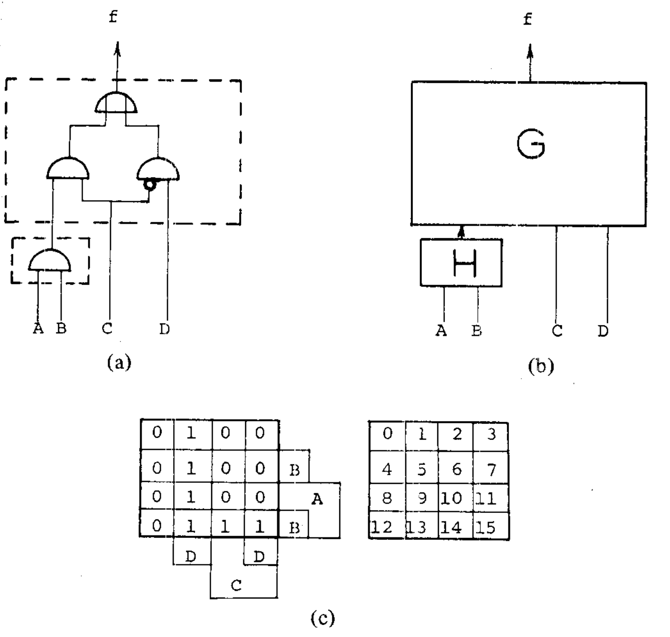


Fig. 1. (a) Circuit diagram. (b) Structure defined by G and H. (c) Normal circuit output function f for Example 1.

to input combinations in a set S, then that circuit must also produce outputs $f_C(T)$ for some set T which, in general, properly includes S. This will be true, furthermore, irrespective of what multiple fault C may have experienced, as long as its structure can be assumed to be correct. This form of inference will be illustrated with a simple example.

Example 1: Fig. 1(a) is a circuit with input variables A, B, C, D. It can be viewed as an interconnection of two basic elements G and H, with the structure shown in Fig. 1(b). The normal circuit output function f is given in Fig. 1(c).

Using Fig. 1(b), it is clear that f can be decomposed in the form $f = G(H(A, B), C, D)$, where G and H are the output functions of the basic elements in that figure.

Consider the four subfunctions

$$f(A, B, 0, 0) = G(H, 0, 0)$$
$$f(A, B, 0, 1) = G(H, 0, 1)$$
$$f(A, B, 1, 0) = G(H, 1, 0)$$
$$f(A, B, 1, 1) = G(H, 1, 1).$$

Each subfunction must be either identically ZERO, identically ONE, H, or $\overline{H}$, where H is a function of A and/or B. Furthermore, this will be true in the presence of any multiple fault involving G and H as long as the structure in Fig. 1(b) is not violated.

Now, assume that the circuit output is observed to be correct for inputs in the set $S = \{i_3, i_7, i_8, i_9, i_{10}, i_{11}, i_{12}, i_{13}, i_{14}, i_{15}\}$, where i_j denotes the binary input combination whose decimal equivalent is j. These outputs are shown in Fig. 2 on maps of the four subfunctions. Since $f(A, B, 1, 1)$ is not identically ZERO or ONE, let $H = f(A, B, 1, 1)$. $f(i_8) = f(i_{12}) = 0$ implies that $f(A, B, 0, 0)$ is neither identically ONE, H, or $\overline{H}$. It must be identically ZERO. Therefore, we conclude that $f(i_0) = f(i_4) = 0$. Similarly, we conclude that $f(A, B, 0, 1)$ must be identically ONE and $f(A, B, 1, 0)$ must be H, so that $f(i_1) =$

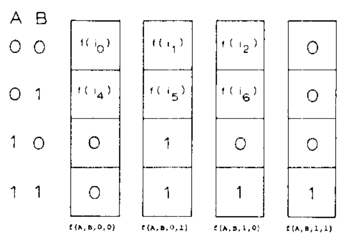


Fig. 2. $f(S)$ for Example 1.

$f(i_5) = 1$ and $f(i_2) = f(i_6) = 0$. 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

A combinational circuit C will be viewed as an interconnection of *basic switching elements* $E_1, \cdots, E_k$. Each is either an individual gate or itself a combinational circuit with inputs and one output.[1] C will be assumed to be free of feedback loops and to have only one circuit output f_C. Each input of a basic element is driven by either a primary input variable from the set $I = \{I_1, \cdots, I_n\}$, or by the output of another basic element. $2^{[I]}$ will denote the Cartesian product $\{0, 1\}^n$.

Definition 1: The *variation* of a basic element E_i, var (E_i) is the set of possible faulty functions that E_i may realize. Here we let var $(E_i) = \{g | g: \{0, 1\}^{m_i} \rightarrow \{0, 1\}\}$, where m_i is the number of inputs to E_i. Thus, var (E_i) is the set of all m_i-place switching functions. var (C) = var $(E_1) \times \cdots \times$ var (E_k) will denote the *variation of* C, where $E_1, \cdots, E_k$ are all the basic elements in circuit C.

Rather than selecting functions for var (E_i) that could be obtained by assuming all possible combinations of single or multiple stuck faults on lines within E_i, 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 E_i. 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 f_C denote the correct output function realized by C, and let f_g denote a circuit output function obtained if, for all i, the ith basic element in C realizes the ith component of g =

[1] A k-output element can be replaced by k parallel single-output basic elements, each with the original inputs.

$(g_1, \cdots, g_k), g \in \text{var}(C)$. Then the set of functions realizable on C is $F_C = \{f | f = f_g, g \in \text{var}(C)\}$. 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 $E_1, \cdots, E_k$.

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

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

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 $\forall f_g \in F_C, f_g(S) = f_C(S) \Rightarrow f_g(T) = f_C(T)$.

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 physical 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 $\text{IN}(E_j) = \{I_r: I_r \in I$ and there exists a path from I_r to $E_j\}$, $\text{OUT}(E_j) = \{I_r: I_r \in I$ and there exists a path from I_r 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.[3]

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

$$f_C(I) = f_j(h_j(X, Z), Y, Z) \tag{1}$$

[2]In general, there may be lines going from subcircuit 1 to subcircuit 2, but where the functions on these lines depend only on Z. The definitions for the sets $\text{IN}(E_j)$ and $\text{OUT}(E_j)$ hold for this case as well, without modification.

[3]The rules in Theorem 3 are not useful for configurations defined by trivial basic-element partitions.

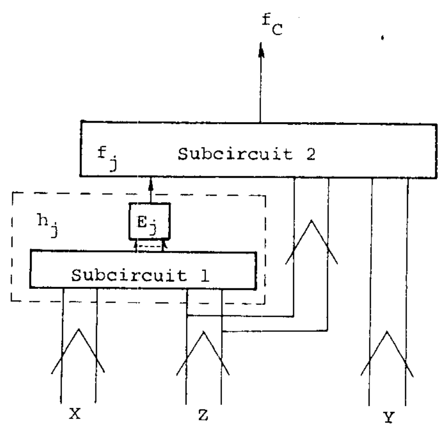


Fig. 3. Element E_j defines the physical partition shown,[2] and a basic-element partition on the inputs: $P_j = [X|Y||Z]$.

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 nondisjoint 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 $\{f_C(X, y_i, z): y_i \in 2^{[Y]}\}$ 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 $\psi(X)$ such that every subfunction in the set is one of the functions $0, 1, \psi(X), \overline{\psi}(X)$.

Proof: If $P_j = [X|Y||Z]$, then with z fixed and $y_i \in 2^{[Y]}$, $f_C(X, y_i, z) = f_j(h_j(X, z), y_i, z) \in \{0, 1, \psi(X), \overline{\psi}(X)\}$ where $\psi(X) = h_j(X, z)$. For a particular P_j and z, $\psi(X)$ 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 *ψ*-CFG* accordingly as $f_C(X, y, z) \equiv 0, \equiv 1$, or $\in \{\psi(X), \overline{\psi}(X)\}$.

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 $P_j = [X|Y]$, $\{f(X, y_i): y_i \in 2^{[Y]}\} \subseteq \{0, 1, \psi, \overline{\psi}\}$ for some function $\psi(X)$.

Example 2: For the circuit in Fig. 4(a),

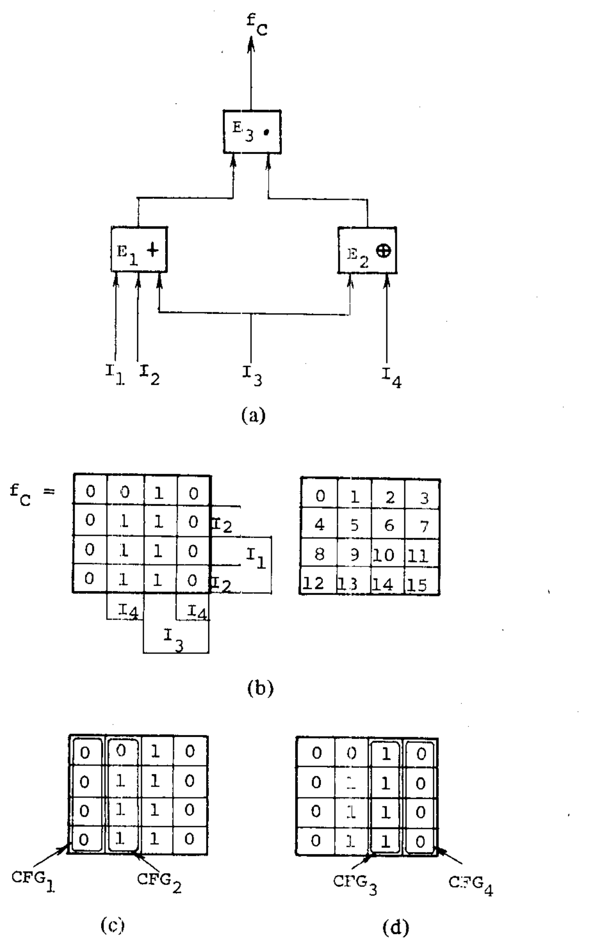


Fig. 4. (a) Circuit for Example 2. (b) Output function f_C. (c) Complete set of two comparable configurations defined by P_1. (d) Another complete set of two comparable configurations defined by P_1.

$$\text{IN}(E_1) = \{I_1, I_2, I_3\} \qquad \text{OUT}(E_1) = \{I_3, I_4\}$$
$$\text{IN}(E_2) = \{I_3, I_4\} \qquad \text{OUT}(E_2) = \{I_1, I_2, I_3\}$$
$$\text{IN}(E_3) = \{I_1, I_2, I_3, I_4\} \qquad \text{OUT}(E_3) = \text{empty}.$$

Thus,

$$P_1 = [I_1 I_2 | I_4 || I_3], \qquad P_2 = [I_4 | I_1 I_2 || I_3],$$
$$P_3 = [I_1 I_2 I_3 I_4 | \text{empty} || \text{empty}].$$

Only P_1 is a nontrivial basic-element partition. Corresponding to P_1, we have the two complete sets of comparable CFG's shown in Fig. 4(c), (d). Each CFG is encircled. Of these four, only CFG$_2$ is a ψ*-CFG. The others are either 1-CFG's or 0-CFG's.

IV. Using Comparable Configurations to Derive Inputs Verified by S

Given a set $S \subset 2^{[I]}$, 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 $D(\text{CFG}) = \{(x\,y\,z) : x \in 2^{[X]}\}$. That is, D(CFG) is the set of

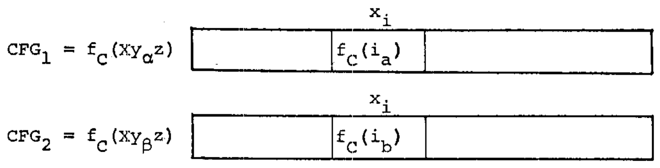


Fig. 5. i_a and i_b are *comparable*: $i_a \sim i_b$.

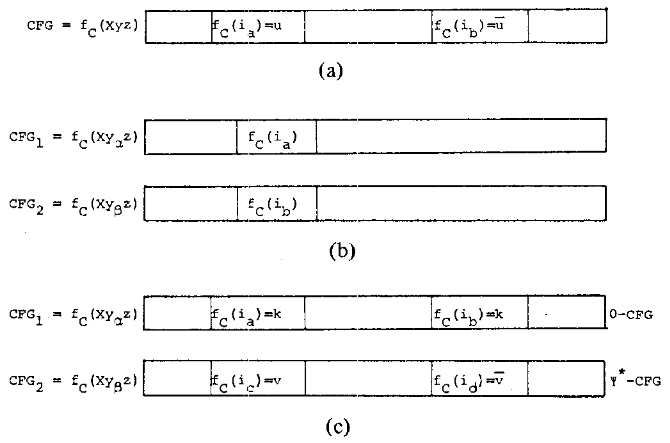


Fig. 6. (a) If $i_a, i_b \in T$, the CFG is a *ψ*-CFG verified by T*. (b) If i_a, $i_b \in T$, then this pair of CFG's is *linked* by T. (c) If $i_a, i_b, i_c, i_d \in T$, then CFG is a *k-CFG verified by T*, where $k \in \{0, 1\}$.

input combinations forming the domain of the subfunction $f_C(X\,y\,z)$. Also, given two comparable configurations (from the same complete set) CFG$_1 = f_C(X\,y_\alpha\,z)$, CFG$_2 = f_C(X\,y_\beta\,z)$, if $i_a = (x\,y_\alpha\,z) \in D(\text{CFG}_1)$, $i_b = (x\,y_\beta\,z) \in D(\text{CFG}_2)$, 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 $T \subset 2^{[I]}$ be a set of input combinations.

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

b) Let CFG$_1$ and CFG$_2$ be comparable ψ*-CFG's verified by T. They are *linked by T* if there exists a pair $i_a \in$ (CFG$_1$) $\cap$ T, $i_b \in D(\text{CFG}_2) \cap T$ 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 ψ*-configuration CFG$_2$ and four input combinations are found with $i_a, i_b \in D(\text{CFG}_1) \cap T$, $i_c, i_d \in D(\text{CFG}_2) \cap T$ 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 $S \subset 2^{[I]}$ and T be a set verified by S. Let $i \in \$T$ denote that input combination i can be added to T (if not already present) without destroying its property of being verified by S.

a) ψ-ψ implication rule: Let CFG$_1$, CFG$_2$ be two comparable ψ*-CFG's verified and linked by T. Then $i \in D(\text{CFG}_1) \cap T \Rightarrow \hat{i} \in \$T$ where $\hat{i} \sim i$ and $\hat{i} \in D(\text{CFG}_2)$.

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

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

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

$f_C(\hat{i}) = f_C(i)$ if CFG_1 and CFG_2 are both ψ or both $\bar{\psi}$; otherwise, $f_C(\hat{i}) = \bar{f}_C(i)$. 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 $S \subset 2^{[I]}$. 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 ψ-ψ implication, 0-implication, and 1-implication rules, employing all complete sets of comparable configurations defined by the nontrivial[3] 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 $P_2 = \{I_2I_3I_4|I_1\}$, $P_3 = [I_3I_4|I_1I_2]$. Let $S = \{i_0, i_1, i_8, i_9, i_{12}, i_{14}\}$. The basic expansion set T is found as follows.

a) CFG_2 is a ψ^*-CFG verified by S and CFG_1 is a 0-CFG verified by S. Using the 0-implication rule on CFG_1, $i_2, i_3, i_4, i_5, i_6, i_7, \in T$.

b) CFG_3, CFG_4 are linked ψ^*-CFG's verified by S. The ψ-ψ 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.

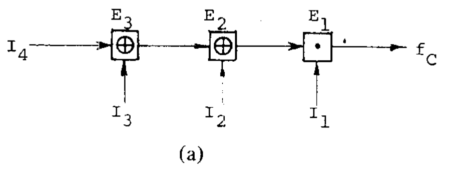
Proof: This is a direct consequence of Theorem 1. It is not, in general, necessary that $T = 2^{[I]}$ if S is an FVTS, as shown by the following example.

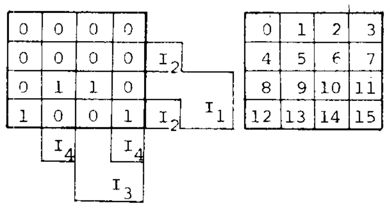
Example 4: Consider the circuit shown in Fig. 8. There are $2^{2^6} = 2^{64}$ distinct functions from $2^{[I]}$ to $\{0, 1\}$, but the above structure can realize at most $2^{2^5} \cdot (2^{2^2})^5 = 2^{52}$ functions, the total number of distinct ways of assigning functions to the basic elements. Hence, there exists $f \notin F_C$. But then there exist $f_1 \in F_C$, $f_2 \notin F_C$, 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_i be the last function in the sequence that is a function in F_C. Then let $f_1 = g_i$, $f_2 = g_{i+1}$. We have $f_1(2^{[I]} - \{u\}) = f_2(2^{[I]} - \{u\})$ 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 $T_1 \subset 2^{[I]}$. Then select some $u_1 \notin T_1$ such that the basic expansion $S \cup \{u_1\}$ is $T_2 \subset 2^{[I]}$. Continue until the basic expansion of $S \cup \{u_1, \cdots, u_{i-1}\}$ 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

$$f(T_i) = f_C(T_i)$$





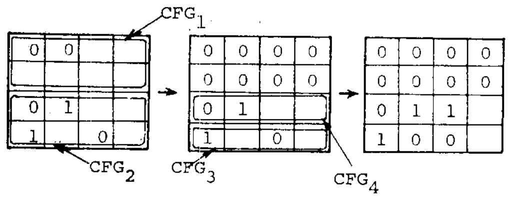


Fig. 7. (a) Circuit for Example 3. (b) f_C and the reference map. (c) Derivation of the basic expansion set T from S.

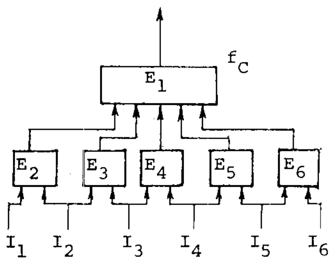


Fig. 8. Circuit for Example 4.

$$f(2^{[I]} - T_i) = \bar{f}_C(2^{[I]} - T_i).$$

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 ψ^*-, 0-, and 1-CFG's and all comparable ψ^*-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 onfigurations (by size of configuration in the set, for example).

Step 2: In each complete set of comparable configurations, elect one ψ^*-configuration CFG and add to S [as in Definition 7 a)] a pair such that T verifies CFG.

Step 3: For each 0-CFG and 1-CFG that is contained[4] in a ψ^*-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 CFG is a ψ^*-CFG verified by T, then for each ψ^*-configuration CFG$_i$ comparable to it, add combinations to S [as in Definition 7 a), b)] so that T verifies CFG$_i$ and links it to CFG.

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

$$P_1 = [I_2I_3|I_1I_4I_5I_6]$$
$$P_2 = [I_5I_6|I_1I_2I_3I_4]$$
$$P_3 = [I_1I_2I_3|I_4I_5I_6]$$
$$P_4 = [I_4I_5I_6|I_1I_2I_3].$$

Let $Rj(Ci)$ denote the CFG (subfunction of f_C) associated with the jth row (ith column) of the map in Fig. 9(b). Also, Rj(LEFT), Rj(RIGHT), Ci(TOP), Ci(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_3, P_1, P_4, and P_2 are all columns, all half columns (top and bottom), all rows, and all half rows (left and right), respectively.

Step 2: In G_1, add i_0, i_{40} to S (denoted $\{i_0, i_{40}\} \to S$), and T verifies ψ^*-CFG $C1$. In G_2, $\{i_{32}\} \to S$, and T verifies ψ^*-CFG $C1$ (BOTTOM). In G_3, $\{i_5\} \to S$, and T verifies ψ^*-CFG $R1$. In G_4, $\{i_4\} \to S$, and T verifies ψ^*-CFG $R1$ (RIGHT).

Step 3: For the 0-CFG $C1$ (TOP) contained in $C1$, let $\{i_8\} \to S$ so that T verifies $C1$ (TOP). [Note that $C1$ (TOP) is comparable to the ψ^*-CFG $C1$ (BOTTOM).]

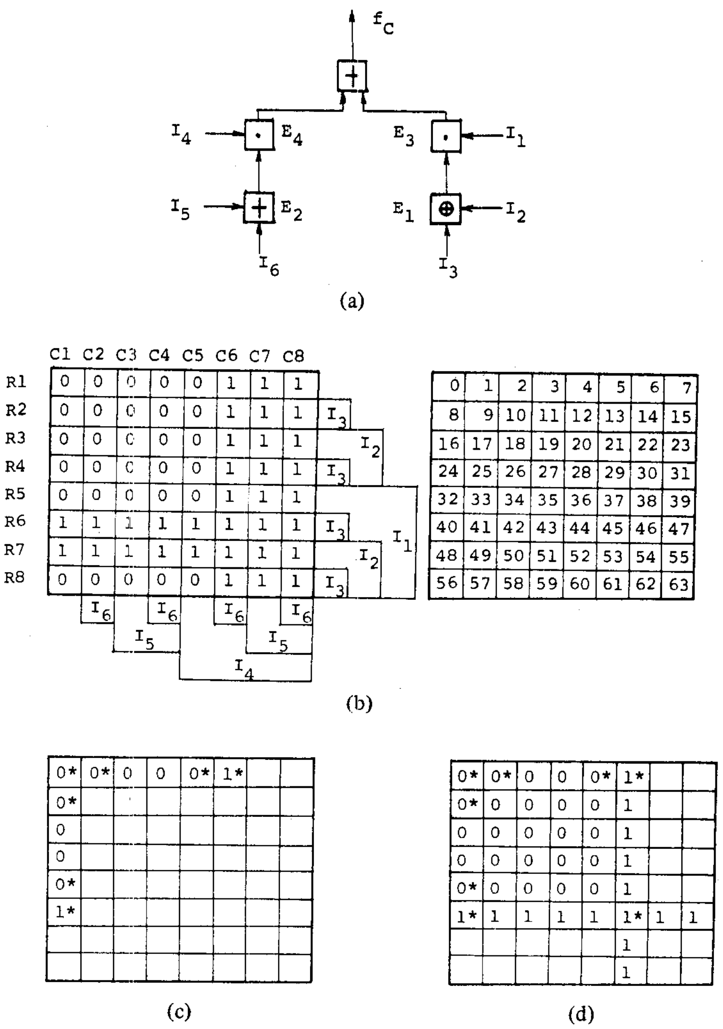
For the 0-CFG $R1$ (LEFT) contained in $R1$, let $\{i_1\} \to S$ so that T verifies $R1$ (LEFT). [Note that $R1$ (LEFT) is comparable to the ψ^*-CFG $R1$ (RIGHT).]

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 ψ^*-CFG's comparable to $C1$ (in G_1), we can identify the 1-CFG $R6$ to serve as a *1-axis*[5] for G_1. With $\{i_{45}\} \to S$, T verifies this 1-CFG. The

[4]CFG$_i$ is contained in CFG$_j$ if $D(\text{CFG}_i) \subset D(\text{CFG}_j)$.

[5]A *1-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 ψ^*-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.



	C1	C2	C3	C4	C5	C6	C7	C8
R1	0	0	0	0	0	1	1	1
R2	0	0	0	0	0	1	1	1
R3	0	0	0	0	0	1	1	1
R4	0	0	0	0	0	1	1	1
R5	0	0	0	0	0	1	1	1
R6	1	1	1	1	1	1	1	1
R7	1	1	1	1	1	1	1	1
R8	0	0	0	0	0	1	1	1

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

0*	0*	0	0	0*	1*		
0*							
0							
0							
0*							
1*							

0*	0*	0	0	0*	1*		
0*	0	0	0	0	1		
0	0	0	0	0	1		
0	0	0	0	0	1		
0*	0	0	0	0	1		
1*	1	1	1	1	1*	1	1
					1		
					1		

Fig. 9. (a) Circuit for Example 5. (b) Output function f_C. (c) S (shown starred) after Step 3. (d) T (all entries) during Step 4.

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 $\{i_{56}\} \to S$, $R8$ becomes a ψ^*-CFG verified by T, and with the verified 1-axis $C6$ [see Fig. 9(d)], Step 4 is complete for G_3 and G_4. Also, all 0-CFG's are now already verified by T.

Step 5: With $\{i_{48}\}$, $\{i_6\}$, $\{i_7\} \to S$, all 1-CFG's are verified by T.

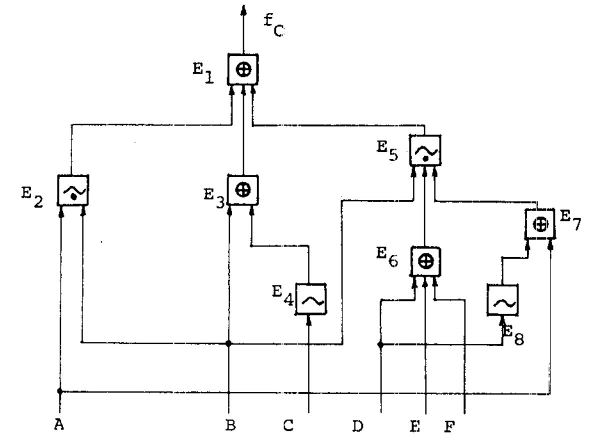
Step 6: At this point, $T = 2^{[I]}$.

Step 7: None of the elements in $S = \{i_0, i_1, i_4, i_5, i_6, i_7, i_8, i_{32}, i_{40}, i_{45}, i_{48}, i_{56}\}$ can be deleted, so that a length-12 FVTS is obtained.

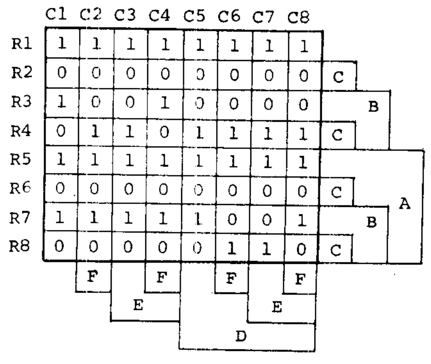
By applying somewhat more complicated rules, a length-10 FVTS $S = \{i_0, i_5, i_{40}, i_{45}, i_9, i_{36}, i_{60}, i_{38}, i_{39}, i_{52}\}$ may be found.

Example 6: A nontree circuit that has nontrivial basic-element partitions is shown in Fig. 10(a). E_1, E_3, E_6, and E_7 each realize the odd parity functions of k inputs; E_2 and E_5 realize NAND; and E_4 and E_8 are NOT gates. The nontrivial basic-element partitions are $P_5 = [DEF|C||AB]$, $P_6 = [EF|ABC||D]$.

The complete sets of comparable CFG's defined by P_5 are $G_1 = \{R1, R2\}$, $G_2 = \{R3, R4\}$, $G_3 = \{R5, R6\}$, $G_4 = \{R7, R8\}$.



(a)



	C1	C2	C3	C4	C5	C6	C7	C8
R1	1	1	1	1	1	1	1	1
R2	0	0	0	0	0	0	0	0
R3	1	0	0	1	0	0	0	0
R4	0	1	1	0	1	1	1	1
R5	1	1	1	1	1	1	1	1
R6	0	0	0	0	0	0	0	0
R7	1	1	1	1	1	0	0	1
R8	0	0	0	0	0	1	1	0

(b)

0*	1*	2	3	4*	5*	6	7
8*	9*	10	11	12*	13*	14	15
16*	17*	18	19*	20*	21*	22	23
24*	25	26*	27	28	29	30	31
32*	33*	34	35	36*	37*	38	39
40*	41*	42	43	44*	45*	46	47
48*	49*	50	51	52*	53*	54	55*
56	57	58	59	60*	61	62*	63

(c)

Fig. 10. (a) Circuit for Example 6. (b) Output function f_C. (c) FVTS found.

The complete sets of comparable CFG's defined by P_6 are $G_5 = \{R1$ (LEFT), $\cdots$, $R8$ (LEFT)$\}$, $G_6 = \{R1$ (RIGHT), $\cdots$, $R8$ (RIGHT)$\}$. Applying Procedure 1 to find an FVTS, we obtain the following.

In Step 2, $\{i_{16}, i_{17}\}$, $\{i_{52}, i_{53}\} \rightarrow S$. $R3$, $R3$ (LEFT), $R7$ and $R7$ (RIGHT) are verified to be ψ*-CFG's.

In Step 3, $\{i_{20}, i_{21}\}$, $\{i_{48}, i_{49}\} \rightarrow S$. $R3$ (RIGHT) and $R7$ (LEFT) are verified to be 0-CFG's.

In Step 4, $\{i_{24}, i_{26}\}$, $\{i_{60}, i_{62}\} \rightarrow S$. $R4$ and $R8$ are verified to be ψ*-CFG's. Also, they are linked to $R3$ and $R7$, respectively.

In Step 5, $\{i_0, i_1\}$, $\{i_8, i_9\}$, $\{i_{32}, i_{33}\}$, $\{i_{40}, i_{41}\}$, $\{i_4, i_5\}$, $\{i_{12}, i_{13}\}$, $\{i_{36}, i_{37}\}$, $\{i_{44}, i_{45}\} \rightarrow S$. All 0- and 1-CFG's are verified.

In Step 6, $\{i_{19}\}$, $\{i_{55}\} \rightarrow S$. T becomes $2^{[I]}$.

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 kth-order basic-element partition can be constructed.

Definition 9: Let J be the index set of k basic elements. The kth-order basic-element partition for $\{E_j: j \in J\}$ is $P_J^k = [X|Y\|Z]$, where $Z = \cup_{j\in J}$ IN$(E_j) \cap \cup_{j\in J}$ OUT(E_j), $X = \cup_{j\in J} \cdot$ IN$(E_j) - Z$, and $Y = \cup_{j \in J}$ OUT$(E_j) - Z$.

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

Theorem 5–Based on Karp [17]: Let $[X|Y\|Z]$ be a kth-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 β_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 kth-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 $2^k \cdot 2^{|Y|} + (2^k - 1) \cdot (2^{|X|} - 2^k)$ combinations are sufficient to verify $f_C(XYz)$. On the other hand, if there are fewer than 2^k distinct columns, consideration of the CFG's $\{f_C(Xy_iz)$: $y_i \in 2^{[Y]}\}$ in itself does not permit reduction in the number of combinations required; all $2^{|X \cup Y|}$ will be used.

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

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

Thus $\min_{1 \leqslant k \leqslant t} \min_{\text{overall } P_J^k} (L_J^k)$ *is an upper bound on the length of the minimal FVTS of the circuit.* In this formula, t

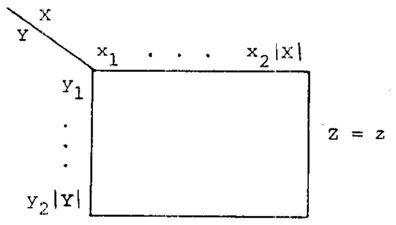


Fig. 11. Map of $f_C(X, Y, z)$.

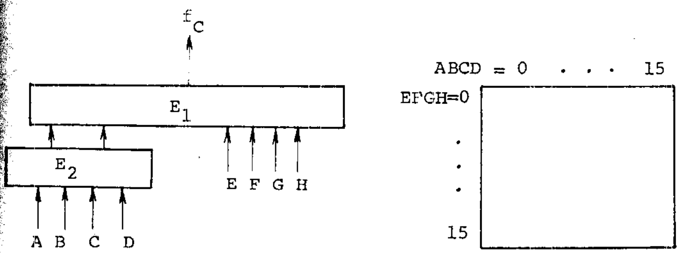


Fig. 12. Circuit for Example 7.

is the largest integer solution to $t + 1 + \log_2(t+1) \leqslant |I|$ because for a map to have more than 2^k columns and 2^k distinct columns, k must satisfy $2^k < 2^{|X|}$, $k \leqslant 2^{|Y|}$. 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 $k + 1 \leqslant 2^{|Y|}$. Therefore, it is necessary that $k + 1 + \log_2(k+1) \leqslant |X| + |Y| \leqslant |I|$.

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

$$2 \cdot 2^{|Y|} + (2^{|X|} - 2) + (2^{|Z|} - 1) \cdot 2^{|X \cup Y|}. \qquad (3)$$

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

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

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

[1] D. R. Schertz and G. A. Metze, "On the indistinguishability of faults in digital systems," in *Proc. 6th Annu. Allerton Conf. Circuit and Systems Theory*, Oct. 1968, pp. 752–760.
[2] E. J. McCluskey and F. W. Clegg, "Fault equivalence in combinational logical networks," *IEEE Trans. Comput.*, vol. C-20, pp. 1286–1293, Nov. 1971.
[3] H. Y. Chang, E. Manning, and G. Metze, *Fault Diagnosis of Digital Systems.* New York: Wiley, 1970.
[4] D. B. Armstrong, "On finding a nearly minimal set of fault detection tests for combinational logic nets," *IEEE Trans. Electron. Comput.*, vol. EC-15, pp. 66–73, Feb. 1966.
[5] J. P. Roth, "Diagnosis of automata failures: A calculus and a method," *IBM J. Res. Develop.*, pp. 278–291. July 1966.
[6] J. P. Roth, W. G. Bouricius, and P. R. Schneider, "Programmed algorithms to compute tests to detect and distinguish between failures in logic circuits," *IEEE Trans. Comput.*, vol. C-16, pp. 567–580, Oct. 1967.
[7] V. Amar and N. Condulmari, "Diagnosis of large combinational networks," *IEEE Trans. Electron. Comput.*, vol. EC-16, pp. 675–680, Oct. 1967.
[8] F. F. Sellers, Jr., M. Y. Hsiao, and L. W. Bearnson, "Analyzing errors with the Boolean difference," *IEEE Trans. Comput.*, vol. C-17, pp. 676–683, July 1968.
[9] P. N. Marinos, "Derivation of minimal complete sets of test-input sequences using Boolean differences," *IEEE Trans. Comput.*, vol. C-20, pp. 25–32, Jan. 1971.
[10] S. S. Yau and Y.-S. Tang, "An efficient algorithm for generating complete test sets for combinational logic circuits," *IEEE Trans. Comput.*, vol. C-20, pp. 1245–1251, Nov. 1971.
[11] M.-W. Du and C. D. Weiss, "Multiple fault detection in combinational circuits: Algorithms and computational results," *IEEE Trans. Comput.*, vol. C-22, pp. 235–240, Mar. 1973.
[12] W. H. Kautz, "Testing for faults in combinational cellular logic arrays," in *Conf. Rec. 8th Annu. Symp. Switching and Automata Theory*, Oct. 1967, pp. 161–174.
[13] K. J. Thurber, "Fault location in cellular arrays," in *1969 Fall Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 35. Montvale, N.J.: AFIPS Press, 1969, pp. 81–88.
[14] G. D. Hornbuckle and R. N. Spann, "Diagnosis of single-gate failures in combinational circuits," *IEEE Trans. Comput.*, vol. C-18, pp. 216–220, Mar. 1969.
[15] R. Ashenhurst, "The decomposition of switching functions," *Ann. Comput. Lab., Harvard Univ.*, vol. 29, 30, pp. 74–116, 1959.
[16] H. A. Curtis, *A New Approach to the Design of Switching Circuits.* Princeton, N.J.: Van Nostrand, 1962.
[17] R. M. Karp, "Functional decomposition and switching circuit design," *SIAM J. Appl. Math.*, vol. 11, pp. 291–335, June 1963.
[18] J. P. Roth and R. M. Karp, "Minimization over Boolean graphs," *IBM J. Res. Develop.*, vol. 4, pp. 227–238, Apr. 1962.
[19] M.-W. Du, "Multiple fault detection in combinational circuits," Ph.D. dissertation, The Johns Hopkins Univ., Baltimore, Md., 1972.

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.