4-1978

On Combinational Networks with Restricted Fan-Out

K. L. Kodandapani
University of Regina

Sharad C. Seth
University of Nebraska-Lincoln, seth@cse.unl.edu

On Combinational Networks with Restricted Fan-Out

K. L. KODANDAPANI AND SHARAD C. SETH, MEMBER, IEEE

Abstract—**Fan-out-free networks of AND, OR, NOT, EXOR, and MAJORITY gates are considered. Boolean functions for which such networks exist are defined to be fan-out free. The paper solves the following problems regarding the fan-out-free networks and functions.**

1) Characterization of the class of fan-out-free functions: The characterization given is constructive in the sense that if a given function is fan-out free one obtains a fan-out-free network to realize it.

2) Counting the class of fan-out-free functions: After establishing a correspondence between a fan-out-free function and a normalized network realizing it, a series of formulas are developed to count distinct normal networks for any subset of the five gates mentioned above.

3) Fault Diagnosis: Methods are developed to detect multiple faults and to locate single faults in arbitrary fan-out-free networks.

Index Terms—**Characterization of fan-out-free networks, combinational networks, counting of fan-out-free, fan-out-free networks, functional decomposition, localized fan outs, multiple fault detection, single fault location.**

I. Introduction

LOGICAL networks with limited or no fan-out simplify testing and fault diagnosis. In this paper we consider limiting the fan out by localizing it to the modules of a network. In the most general case the modules may realize an arbitrary $(n-1)$ variable function in an n input network. Functions realizable by such networks can be easily shown to correspond to those which have simple disjunctive decompositions. In practice the class of modules available is small and usually fixed, therefore, we restrict discussion to fixed sets of modules, and in particular, to modules which are commonly available. In the context of a given module set we will call a function *fan-out free* if it can be realized by a fan-out-free interconnection of modules. We answer the following questions about fan-out-free functions and networks:

1) How do we characterize and synthesize fan-out-free functions of AND, OR, NOT, EXOR (EXCLUSIVE-OR function), and MAJ (3 input majority function) modules? (See Section II.)

2) How many fan-out-free functions of n variables are there for $n \geq 1$? (See Section III.)

3) How do we detect and locate faults in this generalized class of fan-out-free networks? (See Section IV.)

Even though the results pertain to the fan-out-free functions and networks of this specific set of modules, the methodology is often applicable to any fixed set of modules. The AND, OR, and EXOR modules we consider may be extended gates, that is, they may contain more than two inputs. Also, we note, that since the NAND's and NOR's can be simulated by fan-out-free networks of the modules considered, we do not lose anything by excluding these from the module set as long as the aim is not to synthesize a minimum network in some well-defined sense.

The above questions have been answered for more restricted sets of modules in the literature. Hayes [8], [9] answers questions 1) and 2) for AND, OR, and NOT, while Chakrabarti and Kolp [6] answer the same questions for arbitrary two input modules which is equivalent to considering AND, OR, NOT, and EXOR. Butler [5] gives counting formulas for functions of arbitrary two input modules. The problem of fault diagnosis of EXOR networks is considered by Seth and Kodandapani [11].

II. Characterization and Synthesis Algorithm

A. Background and Notation

Let f be a two-valued Boolean function of n variables $x_1, x_2, \cdots, x_n$. The Boolean difference[1] of f with respect to a variable x_i is denoted by df/dx_i and is given by

$$\frac{df}{dx_i} = f(x_i = 0) \oplus f(x_i = 1),$$

where $\oplus$ is the EXCLUSIVE-OR operation. In general,

$$\frac{d}{dx_1}\left(\frac{d}{dx_2}\cdots\left(\frac{df}{dx_j}\right)\cdots\right) \text{ is denoted by } \frac{df}{dx_1 x_2 \cdots x_j}. \quad (1)$$

It can be verified that

$$\frac{df}{dx_1 x_2 \cdots x_j} = \frac{df}{dx_{i_1} x_{i_2} \cdots x_{i_j}}$$

where $(i_1, i_2, \cdots, i_j)$ is an arbitrary permutation of $(1, 2, \cdots, j)$. Thus, given a set $X = \{x_1, x_2, \cdots, x_j\}$, we can unambiguously represent the Boolean difference (1) by df/dX.

We list below some properties of Boolean difference which will be used in the characterization of fan-out-free networks:

Property 1: If $df/dx_i = df/dx_j$, then $df/dx_i x_j = 0$.

Property 2: Let f be decomposable as

$$f(X) = g(h(Y), X - Y) \quad (2)$$

[1] Readers unfamiliar with Boolean difference are referred to Akers [1] and Sellers *et al.* [10] for further details.

Manuscript received May 28, 1976; revised January 11, 1977.

K. L. Kodandapani was with the Department of Computer Science, University of Regina, Regina, Sask., Canada. He is now with the Department of Computer Science, Wichita State University, Wichita, KS 67208.

S. C. Seth is with the Department of Computer Science, University of Nebraska, Lincoln, NE 68588.

0018-9340/78/0400-0309$00.75

doi: 10.1109/TC.1978.1675103

where Y is a subset of X. Furthermore, let y be a member of Y. Then

$$\frac{df}{dy} = \frac{dg}{dh} \cdot \frac{dh}{dy},$$

where the "$\cdot$" represents the AND operation.

Property 3: Let f be expressed in the Reed–Muller canonical (RMC) form involving only uncomplemented variables as follows:

$$f = \sum_{i=0}^{2^n-1} a_i x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}, \tag{3}$$

where the summation operation is EXCLUSIVE-OR, i_j's are either 0 or 1, such that i is the decimal equivalent of the binary number $i_n \cdots i_2 i_1$, and $x_j^{i_j} = 1$ if $i_j = 0$ and $x_j^{i_j} = x_j$ if $i_j = 1$. Then

$$a_i = \frac{df}{dx_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}} \qquad (x_1 x_2 \cdots x_n = 00 \cdots 0).$$

Property 4: Let f be the linear function

$$f = a_0 \oplus x_1 \oplus x_2 \oplus \cdots \oplus x_n, \qquad \text{where } a_0 \in \{0, 1\}.$$

Then

$$\frac{df}{dx_i} = \frac{df}{dx_j}, \qquad \text{for } i \text{ and } j \in \{1, 2, \cdots, n\}.$$

B. Characterization

We will first investigate the conditions under which f has a simple decomposition of form (2) with the further stipulation that h is a nondegenerate function of Y. If $X = \{x_1, x_2, \cdots, x_n\}$, we will assume, without any loss of generality, that $Y = \{x_1, x_2, \cdots, x_m\}$ where $m \leq n$.

Theorem 1: A function $f(x_1, x_2, \cdots, x_n)$ can be expressed as $g(h(x_1, x_2, \cdots, x_m), x_{m+1}, \cdots, x_n)$, where $h = x_1 \oplus x_2 \oplus \cdots \oplus x_m$, iff

$$\frac{df}{dx_i} = \frac{df}{dx_j}, \qquad \text{for } i \text{ and } j \in \{1, 2, \cdots, m\}.$$

Proof—(Only If Part): Assume f has the indicated decomposition. By Property 2:

$$\frac{df}{dx_i} = \frac{df}{dh} \cdot \frac{dh}{dx_i}$$

and

$$\frac{df}{dx_j} = \frac{df}{dh} \cdot \frac{dh}{dx_j}.$$

But by Property 4 $dh/dx_i = dh/dx_j$, hence $df/dx_i = df/dx_j$.

If Part: Assume $df/dx_i = df/dx_j$ for i and $j \in \{1, 2, \cdots, m\}$. Then from Property 1 it follows that $df/dZ = 0$ for any subset Z of $\{x_1, x_2, \cdots, x_m\}$ containing two or more variables. Furthermore, if W is any subset of $\{x_{m+1}, \cdots, x_n\}$ we must have $df/dx_i W = df/dx_j W$. Thus, it follows from Property 3 that the coefficients of all the terms in the RMC form (3) of f involving two or more variables from $\{x_1, \cdots, x_m\}$ are zero and the coefficients of two terms each involving a single variable from this set are identical. For example, the coefficients of the terms $x_1 x_2$, $x_1 x_3$, $\cdots$, $x_1 x_2 \cdots x_m$ are zero while the coefficients of $x_1 x_{m+1}$, $x_2 x_{m+1}$, $\cdots$, and $x_m x_{m+1}$ will be identical. Hence, the function $h = x_1 \oplus x_2 \oplus \cdots \oplus x_m$ can be factored out of various terms in (3) with the consequent decomposition

$$f = g(h(x_1, \cdots, x_m), x_{m+1}, \cdots, x_n). \qquad \blacksquare$$

Hayes [8] has shown that a necessary and sufficient condition for f to have a decomposition of the form (2) with

$$h(x_1, x_2, \cdots, x_m) = \begin{cases} x_1^* x_2^* \cdots x_m^* & \text{or} \\ x_1^* + x_2^* + \cdots + x_m^*, \end{cases} \qquad x_i^* \in \{x_i, \bar{x}_i\}$$

is that $f(x_i = a_i) = f(x_j = a_j)$ for all i and $j \in \{1, 2, \cdots, m\}$, where $a_i, a_j \in \{0, 1\}$. The following lemma shows that this condition is totally disjoint from that stated in Theorem 1.

Lemma 1: Let f be nonvacuous in variables x_i and x_j and let $f(x_i = a_i) = f(x_j = a_j)$. Then $df/dx_i \neq df/dx_j$.

Proof: Without loss of generality assume $a_i = a_j = 0$. Then according to Hayes [8] f has a simple decomposition:

$$f(X) = g(h(x_i, x_j), X - \{x_i, x_j\}),$$

where

$$h(x_i, x_j) = x_i \cdot x_j.$$

Now by Property 2

$$\frac{df}{dx_i} = \frac{dg}{dh} \cdot \frac{dh}{dx_i} = \frac{dg}{dh} \cdot x_j$$

and

$$\frac{df}{dx_j} = \frac{dg}{dh} \cdot \frac{dh}{dx_i} = \frac{dg}{dh} \cdot x_i.$$

Since f is nonvacuous in x_i and x_j it follows that df/dx_i and df/dx_j cannot be identically zero and therefore dg/dh is not identically equal to 0. Thus

$$\frac{dg}{dh} \cdot x_j \neq \frac{dg}{dh} \cdot x_i. \qquad \blacksquare$$

It is now possible to extend Hayes' concept of adjacency to provide an algorithmic procedure for determining fan-out-free functions of AND, OR, NOT, and EXCLUSIVE-OR gates.

Definition 1: Two variables x_i and x_j of a function $f(x_1, \cdots, x_n)$ are *adjacent* if either of the following conditions are satisfied.

Condition 1: $f(x_i = a_i) = f(x_j = a_j)$ for a_i and $a_j \in \{0, 1\}$.

Condition 2: $df/dx_i = df/dx_j$.

It is easy to verify that adjacency is an equivalence relation and hence, partitions the set of variables into equivalence classes.

Theorem 2: Let Y be an equivalent class under the adjacency relation containing two variables of f. Then

$$f(X) = g(h(Y), X - Y),$$

where h is either an AND, OR, or EXCLUSIVE-OR function of the variables in Y. Furthermore, h is an AND or an OR function if

the variables in Y are adjacent to each other because of Condition 1; otherwise h is an EXCLUSIVE-OR function.

Proof: By Lemma 1 the variables in Y could be adjacent to each other by satisfying either Conditions 1 or 2 but not both. If Condition 1 is satisfied, then, as shown in [8], f has the desired decomposition with h as either an AND or an OR function (possibly of some complemented variables). If Condition 2 is satisfied then h is an EXCLUSIVE-OR function from Theorem 1. ∎

Corollary 1: Let $Y_1, \cdots, Y_k$ be the nonsingleton equivalence classes of the variables of f under the adjacency relation. Then

$$f(X) = g(h_1(Y_1), \cdots, h_k(Y_k), X - (Y_1 \cup \cdots \cup Y_k)),$$

where h_i is an AND or an OR function if variables in Y_i satisfy Condition 1 of Definition 1; h_i is EXCLUSIVE-OR otherwise.

Corollary 2: $f(x_1, \cdots, x_n)$ for $n > 1$ is *not* a fan-out-free function of AND, OR, NOT, and EXCLUSIVE-OR if each equivalence class under the adjacency relation contains a single variable.

Theorem 2 and its corollaries can be used to develop an iterative algorithm to synthesize a fan-out-free network for a given function whenever it exists. The following example illustrates how this can be done.

Example 1: Let

$$f(x_1, x_2, x_3, x_4, x_5) = (\bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 x_3 + x_4)\bar{x}_5 + (\bar{x}_3 \bar{x}_4 + x_1 x_2 \bar{x}_4 + \bar{x}_1 \bar{x}_2 \bar{x}_4)x_5.$$

The equivalence classes are $\{x_1, x_2\}$, $\{x_3\}$, $\{x_4\}$, and $\{x_5\}$ where $df/dx_1 = df/dx_2 = x_3 \bar{x}_4$. Thus, f can be expressed as

$$f = g(x_1 \oplus x_2, x_3, x_4, x_5) \equiv g(h, x_3, x_4, x_5).$$

Now, g can be determined by the use of the decomposition chart, see Curtis [7], as $hx_3\bar{x}_5 + x_4\bar{x}_5 + \bar{h}\bar{x}_4x_5 + \bar{x}_3\bar{x}_4x_5$. The equivalence classes of g are $\{h, x_3\}$, $\{x_4\}$, and $\{x_5\}$ where $g(h = 0) = g(x_3 = 0)$. Thus,

$$g = k(hx_3, x_4, x_5) \equiv k(l, x_4, x_5).$$

Furthermore,

$$k(l, x_4, x_5) = (l + x_4)\bar{x}_5 + \bar{l}\bar{x}_4x_5.$$

The equivalence classes of k are $\{l, x_4\}$ and $\{x_5\}$ where $k(l = 1) = k(x_4 = 1)$. Thus,

$$k = u(l + x_4, x_5) \equiv u(v, x_5) = v\bar{x}_5 + \bar{v}x_5.$$

The equivalence class of u are $\{v, x_5\}$ where $du/dv = du/dx_5$.

Thus, $u = a_0 \oplus v \oplus x_5$ where a_0 is determined to be 0 since $u(v = 0,\ x_5 = 0) = 0$. The resulting fan-out-free network for f is shown in Fig. 1.

Example 2: Consider the majority function of three variables

$$f(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3.$$

It can be easily verified that the equivalence classes of f are $\{x_1\}$, $\{x_2\}$, and $\{x_3\}$. Thus, by Corollary 2 of Theorem 2, f is not a fan-out-free function of AND, OR, NOT, and EXOR.

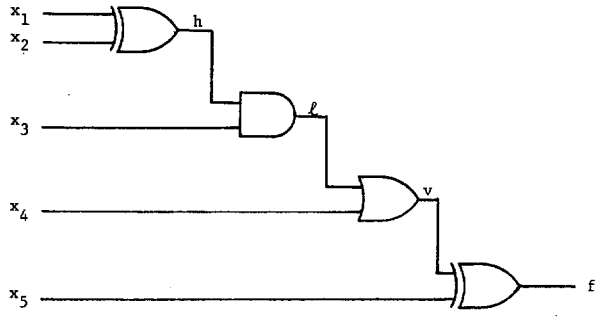

Fig. 1. Decomposed network for the function f in Example 1.

Table I shows which genera of three variable functions are fan-out-free in AND, OR, NOT, and EXOR. Note that the genus numbers marked with a "*" are not fan-out-free without the EXOR function.

It may be observed that all two-variable functions are fan-out free in AND, OR, NOT, and EXOR. Furthermore, a multiinput AND, OR, or EXOR module obviously has a fan-out-free realization in terms of the two-input gates of the same kind. Thus, the class of functions being considered here coincides with the fan-out-free functions of two-input flexible cells considered by Chakrabarti and Kolp [6], Butler [5], and others. However, we believe that our characterization of these functions in terms of the adjacency relation leads to a simpler checking algorithm than available heretofore.

The genera 5, 7, 8, 12, and 13 in Table I are not fan-out free in AND, OR, NOT, and EXOR. We could add one or more of these to the module set and try to answer the three questions in the Introduction for the new module set. From a practical standpoint, however, the majority gate function corresponding to genus number 12 is the only other module commonly available by itself or as part of a full adder. Thus, in the following, we confine ourselves to determining the precise conditions under which the h function in the decomposition of a function f specified by (2) is the majority function.

Theorem 3: Let $Y = (x_i^*, x_j^*, x_k^*)$ where $x_\sigma^* = x_\sigma$ or $\bar{x}_\sigma$ for $\sigma \in \{i, j, k\}$. Then the following conditions are necessary and sufficient for $f(X)$ to be decomposable as $g(h(Y), X - Y)$ where $h(Y) = \text{MAJ}(Y)$.

Condition i):

$$\frac{df}{dx_ix_j} = \frac{df}{dx_jx_k} = \frac{df}{dx_ix_k}.$$

Condition ii):

$$\frac{df}{dx_i}(x_j = x_k = 0) = a_i$$

$$\frac{df}{dx_j}(x_i = x_k = 0) = a_j$$

$$\frac{df}{dx_k}(x_i = x_j = 0) = a_k,$$

such that the following condition is true.

TABLE I

Genus Number	Representative Function	?
1	0	Yes
2	$x_1x_2x_3$	Yes
3	x_1x_2	Yes
4*	$x_1\bar{x}_2\bar{x}_3 + x_1x_2x_3$	Yes
5	$x_1x_2x_3 + \bar{x}_1\bar{x}_2\bar{x}_3$	No
6	$x_1x_2 + x_1x_3$	Yes
7	$x_1x_2 + \bar{x}_1\bar{x}_2x_3$	No
8	$\bar{x}_1x_2x_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3$	No
9	x_1	Yes
10*	$x_1 \oplus x_2$	Yes
11*	$x_1 \oplus x_2 \oplus x_3$	Yes
12	$x_1x_2 + x_1x_3 + x_2x_3$	No
13	$x_1\bar{x}_3 + x_2x_3$	No
14	$\bar{x}_1\bar{x}_2\bar{x}_3 + x_1x_2 + x_1x_3$	Yes

Condition C): Either all three of a_i, a_j, and a_k are 0, or two of a_i, a_j, and a_k are equal and the third is 0.

Proof of Sufficiency: Assume $f(x_1, \cdots, x_n) = g(h(Y), X - Y)$.

$$\frac{df}{dx_i} = \frac{dg}{dh}\cdot\frac{dh}{dx_i}$$

$$\frac{df}{dx_ix_j} = \frac{dg}{dh}\cdot\frac{dh}{dx_ix_j}.$$

Similarly,

$$\frac{df}{dx_ix_k} = \frac{dg}{dh}\cdot\frac{dh}{dx_ix_k};$$

$$\frac{df}{dx_jx_k} = \frac{dg}{dh}\cdot\frac{dh}{dx_jx_k}.$$

Now $h(Y) = \text{MAJ}(Y)$.

Case 1: $Y = (x_i, x_j, x_k)$ or $(\bar{x}_i, \bar{x}_j, \bar{x}_k)$.

Clearly

$$\frac{df}{dx_ix_j} = \frac{df}{dx_ix_k} = \frac{df}{dx_jx_k} = \frac{dg}{dh}.$$

Now

$$\frac{df}{dx_i} = \frac{dg}{dh}(x_j \oplus x_k) = \frac{dg}{dh}(\bar{x}_j \oplus \bar{x}_k)$$

$$\frac{df}{dx_j} = \frac{dg}{dh}(x_i \oplus x_k) = \frac{dg}{dh}(\bar{x}_i \oplus \bar{x}_k)$$

$$\frac{df}{dx_k} = \frac{dg}{dh}(x_i \oplus x_j) = \frac{dg}{dh}(\bar{x}_i \oplus \bar{x}_j)$$

$$\therefore \quad a_i = a_j = a_k = 0.$$

Case 2: $Y = (x_i^*, x_j^*, x_k^*)$ where any one or two of x_i, x_j, x_k are complemented. Without loss of generality, assume

$$Y = (\bar{x}_i, x_j, x_k) \quad \text{or} \quad (x_i, \bar{x}_j, \bar{x}_k).$$

$$\frac{df}{dx_i} = \frac{dg}{dh}(x_j \oplus x_k) = \frac{dg}{dh}(\bar{x}_j \oplus \bar{x}_k)$$

$$\frac{df}{dx_j} = \frac{dg}{dh}(x_i \oplus x_k) = \frac{dg}{dh}(x_i \oplus \bar{x}_k)$$

$$\frac{df}{dx_k} = \frac{dg}{dh}(\bar{x}_i \oplus x_j) = \frac{dg}{dh}(x_i \oplus \bar{x}_j).$$

Then $a_i = 0$, $a_j = a_k = df/dx_ix_j = dg/dh$.

Proof of Necessity: Assume Conditions i) and ii) hold.

Case 1:

$$a_i = a_j = a_k = 0. \tag{4}$$

Consider the RMC expansion of $f(x)$ about the variables x_i, x_j, and x_k.

$$f(x) = b_0 \oplus b_1x_i \oplus b_2x_j \oplus b_3x_ix_j \oplus b_4x_k \oplus b_5x_ix_k \oplus b_6x_jx_k \oplus b_7x_ix_jx_k, \tag{5}$$

where

$$b_r = \left.\frac{df}{dx_i^{r_1}x_j^{r_2}x_k^{r_3}}\right|_{x_ix_jx_k=000}, \qquad r_3r_2r_1$$

is the binary expansion of r. In view of Condition i), $b_3 = b_5 = b_6$ and $b_7 = 0$. In view of (4), $b_1 = b_2 = b_4 = 0$. Hence, (5) becomes

$$f(x) = b_0 \oplus b_3(x_ix_j \oplus x_ix_k \oplus x_jx_k) = (b_0 \oplus b_3) \oplus b_3(\bar{x}_i\bar{x}_j \oplus \bar{x}_i\bar{x}_k \oplus \bar{x}_j\bar{x}_k).$$

Since b_0 and b_3 are independent of x_i, x_j, x_k and

$$x_ix_j \oplus x_ix_k \oplus x_jx_k = \text{MAJ}(x_i, x_j, x_k)$$

$$\bar{x}_i\bar{x}_j \oplus \bar{x}_i\bar{x}_k \oplus \bar{x}_j\bar{x}_k = \text{MAJ}(\bar{x}_i, \bar{x}_j, \bar{x}_k),$$

we have $f(x) = g(h(Y), X - Y)$.

Case 2: We consider only the case $a_i = 0$, $a_j = a_k = df/dx_ix_j$. In this case, in (5), $b_1 = 0, b_3 = b_5 = b_6 = b_2 = b_4$, $b_7 = 0$. Hence, (5) can be written as

$$f(x) = b_0 \oplus b_3(\bar{x}_ix_j \oplus \bar{x}_ix_k \oplus x_jx_k) = (b_0 \oplus b_3) \oplus b_3(x_i\bar{x}_j \oplus x_i\bar{x}_k \oplus \bar{x}_j\bar{x}_k).$$

Hence, $f(x) = g(\text{MAJ}(Y), X - Y)$ where

$$Y = (\bar{x}_i, x_j, x_k) \quad \text{or} \quad (x_i, \bar{x}_j, \bar{x}_k). \qquad \blacksquare$$

Now we give an algorithm to check whether a function has a decomposition of type given by (2), when $h(Y)$ is a majority function of three variables.

Algorithm

Step 1: Compute

$$\frac{df}{dx_1x_2}, \frac{df}{dx_2x_3}, \cdots, \frac{df}{dx_{n-1}x_n}.$$

If for some i, j, k,

$$\frac{df}{dx_ix_j} = \frac{df}{dx_jx_k},$$

then compute df/dx_ix_k and check if

$$\frac{df}{dx_ix_k} = \frac{df}{dx_ix_j}.$$

If so, go to Step 2. Otherwise, f is not decomposable.

Step 2: Compute

$$\frac{df}{dx_i}(x_j = x_k = 0), \qquad \frac{df}{dx_j}(x_i = x_k = 0),$$

and

$$\frac{df}{dx_k}(x_i = x_j = 0),$$

and let them be represented by a_i, a_j, and a_k, respectively. Then, if Condition C) of Theorem 3 is satisfied f is decomposable, otherwise not.

Example 3: Consider the representative function of genus number 5 in Table I

$$f(x_1, x_2, x_3) = x_1x_2x_3 + \bar{x}_1\bar{x}_2\bar{x}_3$$
$$= 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3$$
$$\frac{df}{dx_1} = 1 \oplus x_2 \oplus x_3$$
$$\frac{df}{dx_2} = 1 \oplus x_1 \oplus x_3$$
$$\frac{df}{dx_3} = 1 \oplus x_1 \oplus x_2$$
$$\frac{df}{dx_1x_2} = \frac{df}{dx_1x_3} = \frac{df}{dx_2x_3} = 1$$

which satisfies Condition i) of Theorem 3.

However, $a_1 = a_2 = a_3 = 1$, which violates Condition ii). Hence, the function is not decomposable in the form of a majority function.

Example 4: Let $f = \bar{x}_1x_2 + x_2x_3 + \bar{x}_1x_3 + x_4$. Then

$$\frac{df}{dx_1} = (x_2 \oplus x_3)\bar{x}_4$$
$$\therefore \quad \frac{df}{dx_1x_2} = \frac{df}{dx_1x_3} = \bar{x}_4.$$

Hence, we check for df/dx_2x_3. Now

$$\frac{df}{dx_2} = (x_1 \oplus x_3 \oplus 1)\bar{x}_4$$
$$\therefore \quad \frac{df}{dx_2x_3} = \bar{x}_4.$$

Thus Condition i) of Theorem 3 is satisfied. To check Condition ii) we compute

$$\frac{df}{dx_1}(x_2 = x_3 = 0) = 0$$
$$\frac{df}{dx_2}(x_1 = x_3 = 0) = \bar{x}_4$$
$$\frac{df}{dx_3}(x_1 = x_2 = 0) = (x_1 \oplus x_2 \oplus 1)\bar{x}_4|_{x_1=x_2=00} = \bar{x}_4.$$

Therefore Condition ii) is also satisfied and we must have

$$f = g(M(x_1^*, x_2^*, x_3^*), x_4).$$

The polarities of x_1, x_2, x_3 are determined by looking at which case in the proof of Theorem 3 applies; in this case either $\bar{x}_i$, x_2, x_3 or x_1, $\bar{x}_2$, $\bar{x}_3$ are acceptable polarities. The first choice leads to the decomposition

$$f = M(\bar{x}_1, x_2, x_3) + x_4$$

as determined by the procedure given in Hayes [8].

III. Counting Fan-Out-Free Functions

The fan-out-free networks of AND, OR, NOT, EXOR, and MAJ gates obtained by the method described in the last section can all be "normalized" so that NOT gates, if any, occur only at the input. This is accomplished by successively pushing the NOT gates at the output of other gates to their inputs using the following:

1) DeMorgan's rules for AND or OR gates;

2) the rule $\bar{l}(x_1, x_2, \cdots, x_n) = l(\bar{x}_1, x_2, \cdots, x_n)$ for a linear function l, and

3) the rule $\bar{M}(x_1, x_2, x_3) = M(\bar{x}_1, \bar{x}_2, \bar{x}_3)$ for the majority function M.

One way of counting fan-out-free functions is to count the number of normalized networks corresponding to distinct functions. This will be the approach followed here. First, we consider how the problem can be broken down into simpler counting problems in successive steps.

First, the class of n variable fan-out-free functions, $F(n)$, may be broken down into degenerate (those depending on fewer than n variables) and nondegenerate functions so that

$$F(n) = F_D(n) + F_{ND}(n), \tag{6}$$

where the degenerate functions may be counted by the relation:

$$F_D(n) = \sum_{0 \le m \le n-1} \binom{n}{m} F_{ND}(m). \tag{7}$$

It suffices therefore to count only nondegenerate functions.

Second, the class of nondegenerate functions may itself be broken down into mutually disjoint subclasses, identified according to the module type used at the output in the normal realization. We will use the function symbols A, O, E, M, and N to denote the number of functions with respectively AND, OR, EXOR, MAJ, and NOT gates as the output gates in the normal realizations. Furthermore, let $G = \{A, O, E, M, N\}$, then

$$F_{ND}(n) = \sum_{x \in G} X(n). \tag{8}$$

Third, for an output gate X with p inputs we may consider a normal realization to have the form of Fig. 2 where B_i's represent normal fan-out-free networks of n_i variables. The partition of the input variables in the figure is of the type $T = \{n_1, \cdots, n_p\}$. Each distinct partition type gives distinct classes of normal realizations. Let $X(n, T)$ be the number of n variable functions with output gate X and partition type T. Then

$$X(n) = \sum_T X(n, T), \qquad \text{for } X \in G. \tag{9}$$

Finally, we develop recursive relations to compute $X(n, T)$. We will find an alternate way of representing partition

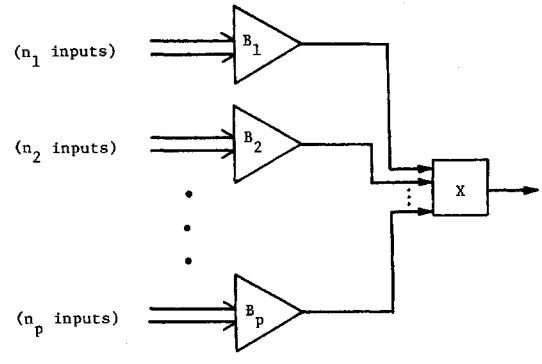

Fig. 2. The form of a fan-out-free network considered in deriving counting formulas. X could be any one of the five gates considered.

types more convenient for our purposes. In this, the repetition frequencies of each integer occurring in the partition type are used as superscripts of that integer. For example, the partition type $\{1, 1, 1, 2, 4\}$ of 9 will be represented as $1^3 2^1 4^1$. The number of partitions of n inputs (or any objects) of the type $T = m_1^{l_1} \cdots m_k^{l_k}$ is given by (see Berge [2]):

$$r(T) = \frac{n!}{(m_1!)^{l_1} \cdots (m_k!)^{l_k} l_1! \cdots l_k!}. \tag{10}$$

The output gate of B_i's in Fig. 2 may be restricted to be other than X in a standard realization if X is an AND, OR, or an EXOR gate. This is because otherwise such an output gate will be subsumed in an extended X. Furthermore, if X is an EXOR gate, only 2 out of 2^p distinct assignments of functions $(f_1^*, f_2^*, \cdots, f_p^*)$, where each f_i^* is either f_i or $\bar{f}_i$, are distinct. Thus we get the following recursive equations:

$$X(n, T) = r(T) \sum_{Y^i \in G - \{X\}} Y^1(n_1) \cdots Y^p(n_p), \quad \text{for } X \in \{A, O\} \tag{11}$$

$$E(n, T) = \frac{r(T)}{2^{p-1}} \sum_{Y^i \in G - \{E\}} Y^1(n_1) \cdots Y^p(n_p) \tag{12}$$

$$M(n, T) = r(T) \sum_{Y^i \in G} Y^1(n_1) \cdots Y^p(n_p). \tag{13}$$

In special cases $X(n, T)$ may be calculated directly. We leave it to the reader to verify that

$$x(1, 1^1) = X(1) = \begin{cases} 2, & \text{if } X = N \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

$$X(n, 1^n) = \begin{cases} 2^n, & \text{if } x \in \{A, O, M\} \\ 2, & \text{if } X = E \end{cases}, \quad \text{for } n > 1 \tag{15}$$

$$A(n) = O(n) \quad \text{(from duality)} \tag{16}$$

$$N(n, T) = 0, \quad \text{for } n > 1 \quad \text{(NOT is never an output gate in normal realizations for } n > 1) \tag{17}$$

and from (4) and (11)

$$N(n) = 0, \quad \text{for } n > 1 \tag{18}$$

$$M(n, T) = 0, \quad \text{if } p \neq 3. \tag{19}$$

Equations (6)–(19) provide an algorithmic way to count fan-out-free functions. Some sample calculations are shown in the following example.

Example 5: Consider fan-out-free functions of three variables. There are two partition types of 3: $T_1 = 1^3$ and $T_2 = 1^1 2^1$. From (15)

$$A(3, T_1) = O(3, T_1) = M(3, T_1) = 8$$

and $E(3, T_1) = 2$.

From (5)

$$r(T_2) = \frac{3!}{1!\,2!} = 3.$$

Therefore using (11) and (12)

$$\begin{aligned} A(3, T_2) &= O(3, T_2) \\ &= 3(N(1)(N(2) + O(2) + E(2) + M(2)) \\ &\quad + O(1)(N(2) + O(2) + E(2) + M(2)) \\ &\quad + E(1)(N(2) + O(2) + E(2) + M(2)) \\ &\quad + M(1)(N(2) + O(2) + E(2) + M(2))). \end{aligned}$$

But from (14) all but the first term within the outermost brackets are 0. Also $M(2) = 0$ from (19) and (9) and $N(2) = 0$ from (18). Thus,

$$A(3, T_2) = 3N(1)(O(2) + E(2)).$$

But

$$O(2) = O(2, 1^2) = 4 \quad \text{and} \quad E(2) = E(2, 1^2) = 2$$

$$\therefore \quad A(3, T_2) = 3 \cdot 2(4 + 2) = 36.$$

Similarly,

$$\begin{aligned} E(3, T_2) &= \tfrac{3}{2}N(1)(O(2) + A(2)) \\ &= 3(4 + 4) = 24. \end{aligned}$$

Now using (9)

$$X(3) = X(3, T_1) + X(3, T_2), \qquad \text{for } X \in G.$$

Therefore

$$\begin{aligned} O(3) &= A(3) = 8 + 36 = 44, \\ M(3) &= 8 + 0 = 8, \\ E(3) &= 2 + 24 = 26, \quad \text{and} \\ N(3) &= 0. \end{aligned}$$

Therefore, from (8)

$$\begin{aligned} F_{ND}(3) &= A(3) + O(3) + M(3) + E(3) + N(3) \\ &= 44 + 44 + 8 + 26 + 0 = 122 \end{aligned}$$

and from (7)

$$\begin{aligned} F_D(3) &= \binom{3}{0} F_{ND}(0) + \binom{3}{1} F_{ND}(1) + \binom{3}{2} F_{ND}(2) \\ &= 2 + 3 \cdot N(1) + 3(A(2) + O(2) + E(2)) \\ &= 2 + 3 \cdot 2 + 3 \cdot 10 = 38. \end{aligned}$$

TABLE II
FAN-OUT-FREE FUNCTIONS OF DIFFERENT MODULE SETS

n	AON			AOEN			AOEMN			AOMN		
	FD(n)	FND(n)	F(n)	FD(n)	FND(n)	F(n)	FD(n)	FND(n)	F(n)	FD(n)	FND(n)	F(n)
1	2	2	4	2	2	4	2	2	4	2	2	44
2	6	8	14	6	10	16	6	10	16	6	8	14
3	32	64	96	38	114	152	38	122	160	32	72	104
4	314	832	1146	526	2154	2680	558	2554	3112	346	1152	1498
5	4892	15104	19996	12022	56946	68968	14102	75386	89488	6572	26304	32876
6	104518	352256	456774	376430	1935210	2311640	493230	2865370	3358600	176678	773376	950054
7	2814520	10037248	12851768	14821942	80371122	95193064	21734582	133191386	154925968	5511738	27792384	33304122

Therefore from (6)

$$F(3) = F_D(3) + F_{ND}(3)$$
$$= 38 + 122 = 160.$$

Table II shows the results of a computer program which implemented the method in this section to count fan-out-free functions of up to 7 variables for 4 different module sets. The module sets {AND, OR, NOT} and {AND, OR, EXOR, NOT} are the same as considered in [8] and [5], respectively. These are included here for reference only. The third module set consists of {AND, OR, EXOR, MAJ, NOT} for which the number of fan-out-free functions $F(n)$ grows faster than the first two module set; for $n = 7$ this number is more than ten times larger than for the first module set and almost twice the size for the second module set. The fourth module set consisting of {AND, OR, MAJ, NOT} is of interest because the fan-out-free functions of this set are a larger subset of unate functions than the fan-out-free functions of {AND, OR, NOT}. Again, the function grows much more rapidly than for the smaller module set and is about $2\frac{1}{2}$ times greater for $n = 7$.

IV. Diagnosis of Faults in Fan-Out-Free Combinational Networks

Fault detection in fan-out-free combinational networks has been considered by a number of authors. In the literature, the gates in the network are usually restricted to AND, OR, NOT, NAND, and NOR types. In [11] multiple fault detection in linear tree networks consisting of two-input EXOR modules has been considered. The fault model in [11] assumes that a fault in an EXOR gate can change the EXOR function to any other function of its two inputs other than the equivalence function. In this section, we consider multiple fault detection and single fault location in fan-out-free networks consisting of the NOT gate, the two-input AND, OR, and EXOR gates, and the three-input MAJ gate. The results of this section can be readily generalized to networks in which the AND, OR, and EXOR gates are extended to include more than two inputs. In our fault model we assume stuck-type faults for the basic AND, OR, and NOT gates, but for the EXOR and the MAJ gate which are usually realized by a network of basic gates, we will allow arbitrary faults with the following exceptions: the EXOR gate cannot change to an EQUIVALENCE (complement of EXOR) gate, and the MAJ gate cannot fail to another majority function with one or more inputs complemented.[2] By a single fault we mean the presence of a fault of the above mentioned type and by a multiple fault we mean the simultaneous presence of a number of single faults.

The multiple fault detection test set is derived in an iterative manner. We will view the fan-out-free network N as a tree whose root node is the output gate. We will denote the root node by R, its left and right subtrees by N_L and N_R, respectively, and its middle subtree, if any, by N_M. We have to consider three cases depending on whether R is an AND or an OR gate, an EXOR gate, or a MAJ gate. Before proving the theorems that specify the multiple fault detection test set for N in each of the above three cases, we state the following results, the proofs of which are similar to those for linear tree networks given in [11].

Lemma 2: Every multiple fault in a fan-out-free network is detectable.

Lemma 3: Assume the correct response of a tree network to a multiple fault detection test set is a binary vector C. Then no multiple fault can change the response to $\bar{C}$.

Corollary 3: No multiple fault can complement the output function of a tree network.

Theorem 4: Let R be an AND(OR) gate. Let $T(N_L)$ and $T(N_R)$ be the multiple fault detection test sets for N_L and N_R, respectively. Partition the two test sets into a set of false tests, i.e., those normally producing a 0; and a set of true tests, i.e., those normally producing a 1.

[2] These restrictions in the fault model appear to be arbitrary, however, they seem to hold for stuck-type faults in nonredundant realizations. The authors have verified this for commonly known implementations of the EXOR gate and for two-level realizations of the MAJ gate.

$$T(N_L) = \{a_1^0, a_2^0, \cdots, a_j^0\} \cup \{a_1^1, a_2^1, \cdots, a_{n_l-j}^1\}$$
$$T(N_R) = \{b_1^0, b_2^0, \cdots, b_k^0\} \cup \{b_1^1, b_2^1, \cdots, b_{n_r-k}^1\},$$

where, the elements of the sets represent vectors and the superscripts denote the normal response.

Define a new test set $T(N) = T_{N_1} \cup T_{N_2}$ where

$$T_{N_1} = \{(x, b_1^1(b_1^0)), \forall x \in T(N_L)\}$$
$$T_{N_2} = \{(a_1^1(a_1^0), x), \forall x \in T(N_R)\}.$$

$T(N)$ detects all multiple faults in N.

Proof: We will prove the result only for the case when R is an AND gate. The case when R is an OR gate follows by duality.

Case 1: The multiple fault includes a fault in R. In view of the fault model, this fault can be ϕ-a-0 or ϕ-a-1 faults at the input or output leads of R. In such a case, it is easy to see that $T(N)$ detects this fault.

Case 2: R is fault-free. The fault is in N_L, N_R, or both. Without loss of generality, assume N_L is faulty. (N_R may or may not be faulty.) In this case T_{N_1} detects the fault because if the output of N_R remains 1 for the input b_1^1 to N_R, then the output of N_L is sensitized to the output of R; if a fault in N_R causes the output of N_R to be 0 when b_1^1 is applied to N_R, then a true test in $T(N_L)$ detects the fault. ∎

Theorem 5: Let R be an EXOR gate and let $T(N_L)$ and $T(N_R)$ be as defined in Theorem 4. Define a new test set $T(N)$ as

$$T(N) = \{(x, b_1^0) \mid x \in T_{N_L}\} \cup \{(a_1^0, x) \mid x \in T_{N_R}\} \cup \{(a_1^1, b_1^1)\}$$

$T(N)$ detects all multiple faults in N.

Theorem 6: All conditions remaining the same as in Theorem 5, assume that at least one of N_L or N_R is nondegenerate, then the test set $T(N)$ given by

$$T(N) = \{(a_1^0, x) \mid x \in T_{N_L}\} \cup \{(x, b_1^0) \mid x \in T_{N_R}, x \neq a_1^1\} \cup \{(a_1^1, b_1^1)\}$$

detects all multiple faults in N.

The proofs of Theorems 5 and 6 are exactly similar to the corresponding theorems for linear tree networks given in [11].

Theorem 7: Let R be a MAJ gate. Let N_L, N_M, and N_R be the three subtrees corresponding to the three inputs of R, and $T(N_L)$, $T(N_M)$, and $T(N_R)$ the corresponding multiple fault detection test sets for N_L, N_M, and N_R, respectively. Then the test set

$$T_N = T_{N_1} \cup T_{N_2} \cup T_{N_3} \cup T_{N_4}$$
$$= \{(x, c_1^0, b_1^1) \mid x \in T_{N_L}\} \cup \{(a_1^1, x, b_1^0) \mid x \in T(N_M)\} \cup \{(a_1^0, c_1^1, x) \mid x \in T_{N_R}\} \cup \{(a_1^0, c_1^0, b_1^0), (a_1^1, c_1^1, b_1^1)\}$$

detects all multiple faults in N.

Proof:

Case 1: The multiple fault includes a fault in R. We will show by contradiction that at least one of the 8 tests $\{(a_1^p, c_1^q, b_1^r) \mid p, q, r \in \{0, 1\}\}$, included in T_N, detects the multiple fault. Assume such is not the case and that for $k \in \{0, 1\}$, α^k, β^k, and γ^k represent the responses of the (possibly faulty) subnetworks N_L, N_R, and N_M when input vectors a_1^k, b_1^k, and c_1^k are applied to them, respectively. The response of the faulty root node for the 8 tests can then be summarized in the form of a table:

Input	Response
$\alpha^0\ \gamma^0\ \beta^0$	0
$\alpha^0\ \gamma^0\ \beta^1$	0
$\alpha^0\ \gamma^1\ \beta^0$	0
$\alpha^0\ \gamma^1\ \beta^1$	1
$\alpha^1\ \gamma^0\ \beta^0$	0
$\alpha^1\ \gamma^0\ \beta^1$	1
$\alpha^1\ \gamma^1\ \beta^0$	1
$\alpha^1\ \gamma^1\ \beta^1$	1

Furthermore, it can be shown by arguments similar to those used in [11] that $\alpha^0 = \overline{\alpha^1}$, $\beta^0 = \overline{\beta^1}$, and $\gamma^0 = \overline{\gamma^1}$. Thus, the above table, indeed, represents a truth table for R when specific binary values are assigned to α^0, β^0, and γ^0. Of the eight possible tables, one corresponds to the fault-free MAJ gate and hence can be ruled out. The other seven correspond to failure modes for the MAJ gate excluded by our fault model. Thus, by contradiction, the assumption that none of the 8 tests detect the fault must be false.

Case 2: The multiple fault does not include a fault in R. Assume N_L is faulty. (N_R or N_M may or may not be faulty.) Consider the application of the tests in T_{N_1} to N. If the outputs of N_M and N_R remain 01 or change to 10, then the output of N_L is sensitized to the output of R and the fault is detected. If the outputs of N_M and N_R change to 00 or 11, then the output of N remains 0 or 1 for all the tests in T_{N_1} and hence the fault is detected. ∎

When one or more of the subtrees in a tree network consists of only AND or OR gates, then the functions produced at the outputs of such trees are unate functions. Minimal multiple fault detection test sets can be obtained easily by the method proposed by Berger and Kohavi [3].

Now we will illustrate the application of the above theorems for the derivation of a multiple fault detection test set for the tree network shown in Fig. 3. The dotted boxes show the successive stages in the derivation of the test set.

Single Fault Location: As in [11] the basic principle used in the location of single faults is that of binary search based on isolation of the fault to either the left subtree N_L, right subtree N_R, or the root R. Thus the fault location procedure is adaptive—the outcome of tests up to a certain time determine, in general, what tests should be applied next. Moreover, the tests applied also depend on the function realized by the current root node.

Assume the root node is an AND gate and suppose the test set T_{N_1} specified in Theorem 4 is applied to the network. Let the fault-free output be V_{C_L} which will also be the output of the left subtree in this case because the output of the right subtree merely acts as a sensitizing input to the root node. Assuming a fault had already been indicated, it is not too difficult to analyze various outputs and partially isolate the fault as shown in Table III. The vectors **0** and **1** in Table III represent all zeros and all ones, respectively. The ambiguity in the first column can be removed by applying T_{N_2} to the

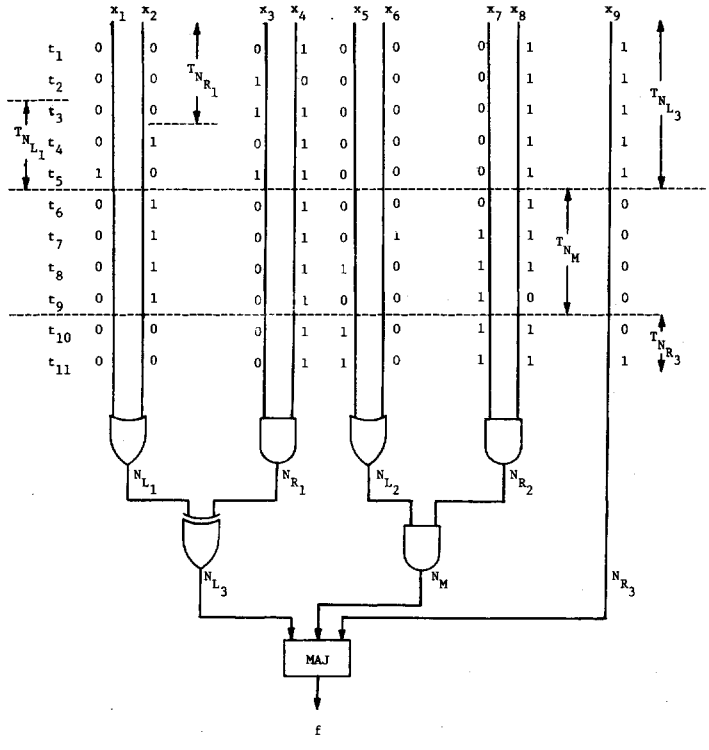

Fig. 3. Derivation of tests for an example network using Theorems 4–7.

TABLE III

Response implies fault in	**0** or V_{C_L} R or N_R	**1** R	other N_L

TABLE IV

		Response to T_{N_2} = **0** or V_{C_R}	**1**	other
Response to T_{N_1} =	**0** or V_{C_L}	R	R	N_R
	1	R	×	×
	other	N_L	×	×

network and assuming V_{C_R} is the correct response. This is illustrated in Table IV. The "×" entries in Table IV represent logically impossible situations.

Similar analysis applies to the situation where the root node is an OR gate, an EXOR gate, or a MAJ gate. The tests to be applied in these cases follow from Theorems 4 (dual case), 5, and 7.

The single fault location procedure recursively applies the appropriate test set to subtree networks (while holding nonsubtree inputs at constant sensitizing values) until a root node is determined to be faulty. Clearly, in the worst case the recursive calls need not exceed the number of levels in the tree.

V. Conclusion

The class of functions considered in this paper is a generalization of the strictly fan-out-free functions of Hayes [8] which are realizable by fan-out-free networks of AND, OR, and NOT gates; the functions we consider have fan-out-free realizations in terms of modules which may themselves be realized by fan out. In other words, we consider networks of AND, OR, and NOT gates in which fan out, if present, is restricted to be local. It can be shown that for any given set of modules the class of fan-out-free functions is still vanishingly small, as compared to the class of Boolean functions. However, a number of practically useful functions (e.g., linear functions) which are not strictly fan-out free may be included in the more general class considered in this paper. Furthermore, we conjecture that the number of strictly fan-out-free functions of n variables, as a fraction of the fan-out-free functions of AND, OR, NOT, and EXOR gates, asymptotically becomes zero for a large value of n.

For further research we suggest the following.

1) Extend the characterization for the majority gate (Theorem 3) to include an arbitrary voting function of n variables.

2) Assume a function cannot be realized by a fan-out-free network of modules. What is the minimum fanout, cf., Hayes [8], to realize this function?

Acknowledgment

The authors wish to thank Dr. J. Butler of Northwestern University for pointing out some computational errors in Table II in an earlier version of the paper.

References

[1] S. B. Akers, Jr., "On the theory of boolean functions," *SIAM J. Appl. Math.*, vol. 7, pp. 487–498, 1959.
[2] C. Berge, *Principles of Combinatorics*. New York: Academic Press, 1971.
[3] I. Berger and Z. Kohavi, "Fault detection in fanout-free combinational networks," *IEEE Trans. Comput.*, vol. C-22, pp. 908–914, Oct. 1973.
[4] R. Betancourt, "Derivation of minimum test sets for unate logical circuits," *IEEE Trans. Comput.*, vol. C-20, pp. 1264–1269, Nov. 1971.
[5] J. T. Butler, "On the number of functions realized by cascades and disjunctive networks," *IEEE Trans. Comput.*, vol. C-24, pp. 681–690, July 1975.
[6] K. Chakrabarti and O. Kolp, "Fan-in constrained tree networks of flexible cell," *IEEE Trans. Comput.*, vol. C-23, pp. 1238–1249, Dec. 1974.
[7] H. A. Curtis, *A New Approach to the Design of Switching Circuits*. Princeton, NJ: Van Nostrand, 1962.
[8] J. P. Hayes, "The fanout structure of switching functions," *J. Assoc. Comput. Mach.*, Oct. 1975.
[9] ——, "Enumeration of fanout free Boolean functions," to be published.
[10] 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.
[11] S. C. Seth and K. L. Kodandapani, "Diagnosis of faults in linear tree networks," *IEEE Trans. Comput.*, vol. C-26, pp. 29–33, Jan. 1977.

K. L. Kodandapani received the B.E. degree in electrical engineering from Mysore University, Mysore, India, the M.E. degree in applied electronics and servomechanisms, and the Ph.D. degree in electrical engineering from the Indian Institute of Science, Bangalore, India, in 1971 and 1974, respectively.

From September 1974 to November 1975, he was a lecturer in the Computer Science Program, Indian Institute of Technology, Kanpur, India. From December 1975 to May 1977, he was a Post-Doctoral Fellow at the University of Regina, Regina, Sask., Canada. Since August 1977, he has been an Assistant Professor of Computer Science, Wichita State University, Wichita, KS. His current research interests include fault-tolerant computing and parallel processing.

Sharad C. Seth (S'66–M'70) was born in Sagar, India, on November 1, 1942. He received the B.E. degree in electronics and telecommunications from Jabalpur, India, the M.S. degree from the Indian Institute of Technology, Kanpur, India, in 1966, and the Ph.D. degree from University of Illinois, Urbana, in 1970.

Since then he has been on the faculty of the Department of Computer Science, University of Nebraska at Lincoln, currently holding an Associate Professor's position. His research interests are in the areas of fault tolerant computing, switching theory, and software verification.

Dr. Seth is a member of the Association for Computing Machinery.