The design of self-checking multi-output combinational circuits*

by D. C. KO

Burroughs Corporation
Mission Viejo, California

and

M. A. BREUER

University of Southern California
Los Angeles, California

ABSTRACT

In this paper we present a technique, called Extended-Parity Checking, for the design of error-detecting circuits for combinational logic networks. Its concept is derived from the conventional parity checking technique, which is applicable only for odd number of errors, yet it can detect errors of any degree. A structural model, called the fanout-graph, is introduced which contains a minimum number of nodes sufficient to determine the fundamental causes of multiple errors in a circuit. Output functions are then expressed in a special form, called the Fanout-Observed Output Function (FOOF), which facilitate the analysis of errors. Based on this information and certain circuit parameters, a set of design methods are presented for producing self-checking circuits. Among them, one deals with the addition of external leads by augmenting some of the fanout nodes in the original circuit, while others involve duplicating or checking independently parts of the logic.

INTRODUCTION

The implementation of a self checking system requires appropriate error detecting circuitry. This circuit should generate an *error signal* whenever an output error occurs. This signal can be used to stop computation, signal manual repair work, or initiate a reconfiguration process.

Shown in Figure 1 is a general model of a self-checking system. It consists of two circuits, namely C and D. C is the operating circuit being checked, having input X and output F, both vector-valued. D is a single-output circuit, called the error detecting circuit (or logic), whose output, denoted by ϵ, is the required error signal (subject to timing control). Y is a set of internal signals of C which, depending on circuit constraints, may or may not be available to D. Under our present investigation both C and D are assumed to be acyclic combinational circuits.

The simplest form for a self-checking system is complete duplication in which D properly contains C. In this case, a redundancy ratio of more than 2:1 is expected. Depending on the particular function which C implements, and its structure, some other techniques exist which may sometimes yield a smaller redundancy ratio.[7]

This paper deals with the design of checking circuits. Our goal is to try to achieve a redundancy ratio of less than or equal to 2:1. The technique we are going to investigate is called Extended-Parity Checking (EPC). Its concept is derived from the conventional parity checking scheme. It is well-known that parity checking will fail in case of an even number of errors occurring on the circuit outputs. The EPC on the other hand, will *not* have this deficiency.

FAULT ANALYSIS AND ERROR DETECTABILITY

Let $f:\{0, 1\}^n \rightarrow \{0, 1\}$ be a single-output Boolean switching function over the set of variables $X=\{x_1, x_2, \ldots, x_n\}$. A multi-output Boolean switching function is denoted by $F:\{0, 1\}^n \rightarrow \{0, 1\}^m$ and consists of m single-output functions, i.e., $F=(f_1, f_2, \ldots, f_m)$ where $f_i=f_i(x_1, \ldots, x_n)$ for $i=1, 2, \ldots, m$. Let C be a combinational circuit which realizes F. The set of input lines $\{x_1, x_2, \ldots, x_n\}$ are called *primary inputs* (PI) and the set of outputs $\{f_1, f_2, \ldots, f_n\}$ are called *primary outputs* (PO). We denote an input vector to C by $X_i=(x_1, x_2, \ldots, x_n)$ and the corresponding output vector by $F_j=(f_1, f_2, \ldots, f_m)$.

Let X_i represent the binary input vector whose decimal value is i, e.g. $X_3=(00\cdots011)$ and set $\chi=\{X_i | i=0, 1, \ldots, 2^n-1\}$. By $F(X_k)$ we mean the value of F for input $X=X_k$.

We assume circuits are made up of single-output gate elements such as AND, NAND, OR, NOR, etc. Below are some basic definitions concerning circuit topology.

* This work was supported in part by the National Science Foundation under Grant GK-23886 and in part by the Office of Naval Research under Grant N00014-67-A-0269-0019 (NR 048-299).

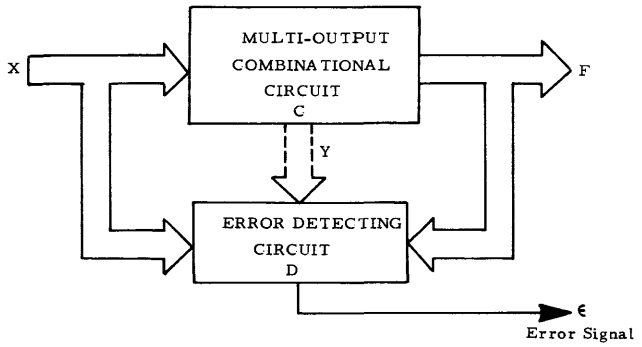


Figure 1—General model of a self-checking system

1. Every PI, PO and gate is a node, called a *signal node* (SN). (We do not differentiate between a gate and its output.)
2. A node is an *internal node* (IN) if it is neither a PI nor a PO nor a node directly connected to a PO.
3. The *fanout value* τ_α of a node α is equal to the number of nodes to which it fans out.
4. A node α is called a *fanout node* (FN), if $\tau_\alpha \geqq 2$.
5. A *signal path* is a sequence of nodes of the form $\alpha_1\alpha_2 \ldots \alpha_n$, where $n \geqq 2$ and α_{i-1} is an input to α_i. A path is *simple* if $\tau_{\alpha_i}=1$ for $i=2, 3, \ldots, n-1$.
6. A *reconvergent node* (RN) is a node having some pair of inputs which are the terminal nodes of paths having a common source node (a FN).
7. A *limited fanout-free* (LFF) circuit is a circuit in which the only FN's are PI's.

In order to simplify the presentation of our results we assume that all circuits being dealt with contain no redundancy. The general case which includes redundancy is dealt with in Reference 8.

In our work we will assume a single permanent stuck-at fault model.

Let C be an irredundant combinational circuit realizing the switching function F, and let $\Delta=\{\delta_1, \delta_2, \ldots, \delta_p\}$ be a set of faults associated with C. Then we denote the circuit C containing fault δ_j by C^j, where $C^0(=C)$ represents the fault-free circuit. C^j realizes the function $F^j(X)=(f_1{}^j(X), f_2{}^j(X), \ldots, f_n{}^j(X))$. If $F^j(X_k)\neq F^0(X_k)$, then fault δ_j is *detectable* by input X_k.

Let $H=\Delta\times\chi$ be the set of all fault-input pairs. Then the *error indicators* $\Delta^j f_i(X_k)$ and $\Delta^j F(X_k)$ are defined as follows. For each $h_{jk}=(\delta_j, X_k)\in H$ we have

$$\Delta^j f_i(X_k)=f_i{}^j(X_k)\oplus f_i{}^0(X_k),$$

for $i=1, 2, \ldots, m$, and

$$\Delta^j F(X_k)=F^j(X_k)\oplus F^0(X_k)$$
$$=(\Delta^j f_1(X_k), \Delta^j f_2(X_k), \ldots, \Delta^j f_m(X_k)).$$

If $\Delta^j F(X_k)=(0, 0, \ldots, 0)=0$ then δ_j is not detected by X_k, otherwise it is. The norm $|\Delta^j F(X_k)|$ is said to be the *Hamming weight* of the vector $\Delta^j F(X_k)$, and equals the number of 1's in the vector.

Suppose that $|\Delta^j F(X_k)|=q$, $q=1, 2, 3$, or $n(4\leqq n\leqq m)$. Then we say there is a single-error, double-error, triple-error, or n-bit error on the circuit output respectively. We call "q" the *degree* of the output error.

Suppose we append to C an associated error detecting circuit D having the following property. If an error in the output of C occurs, the output of D, called the *error signal* and denoted by ϵ, will be set to 1; otherwise its value will be 0. Hence $\epsilon=1$ indicates the detection of an output error in C, and the fault which caused this error is thus detected.

We will be concerned with the design of D.

Let $\chi(\delta_j)$ be the set of inputs which detect δ_j in C, i.e. $\Delta^j F(X_i)\neq 0$ for each $X_i\in\chi(\delta_j)$. Since C is irredundant $\chi(\delta_j)\neq\varphi$.

Let χ' be a subset of $\chi(\delta_j)$ such that for each $X_i\in\chi'$, if δ_j is present then $\epsilon=1$.

(a) If $\chi'=\chi(\delta_j)$ then δ_j is said to be *totally checked*.
(b) If $\chi'=\varphi$, then δ_j is said to be *unchecked*, and
(c) If $\varphi\subset\chi'\subset\chi(\delta_j)$ then δ_j is said to be *conditionally checked*.

If all δ_j are totally checked then C is said to be *totally checked*, and if some faults in C are totally checked while others are conditionally checked then C is said to be *conditionally checked*. If some faults are unchecked, then C is said to be *partially checked*.

The combined circuit (C, D) forms a self-checking system which in turn is subject to faults. Our error detecting criteria is defined as follows. For each $h_{jk}\in H'=\Delta'\times\chi$, where Δ' is the set of faults associated with the new circuit (C, D), we require

$$\epsilon=\begin{cases}1 \text{ if } q\neq 0, \text{ or D has a fault}\\ 0 \text{ otherwise.}\end{cases} \tag{2.1}$$

We assume that the fault ϵ *s-a-0 fault* and any fault equivalent to it in D is not included in Δ'. The problem of detecting output faults in a checker is discussed in Reference 3.

The general form of our forced parity checking system is shown in Figure 2. Here we augment C with the logic $\hat{c}$ having outputs $\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_\lambda$. $\hat{c}$ is designed such that any single fault in C or $\hat{c}$ causes an odd number of outputs from $(C, \hat{c})$ to be in error. $\tilde{P}$ is a circuit which implements the parity function defined by the expression

$$\left(\bigoplus_{i=1}^{n} f_i\right)\oplus\left(\bigoplus_{i=1}^{\lambda}\hat{\alpha}_i\right).$$

The outputs of $\tilde{P}$ and $\tilde{P}'$ are then compared by T' to see if an error has occurred.

ALGEBRAIC STUDY OF CIRCUIT OUTPUT ERRORS

In a multi-output combinational circuit there is the possibility of several output lines being jointly dependent on one signal. If a fault causes an error on that signal then multiple errors may occur on the outputs.

Assume under condition $h_{jk}=(\delta_j, X_k)$ that f_p and f_q are in error, and that δ_j occurs at signal α, i.e. δ_j corresponds to α

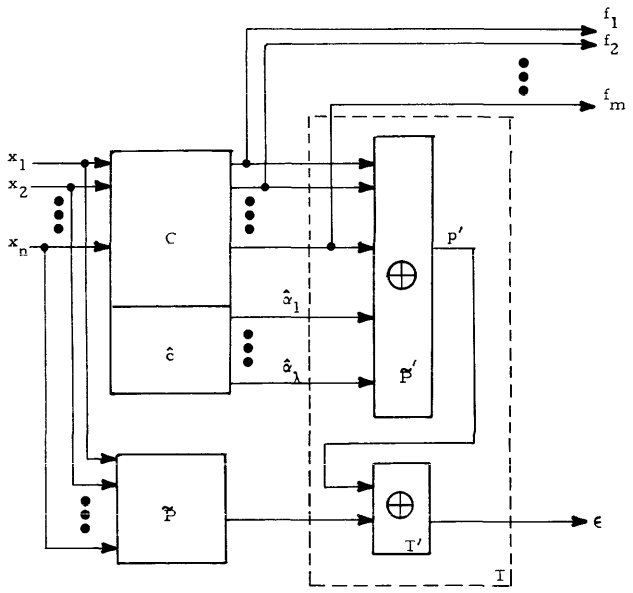


Figure 2—A forced-parity checking system

s-a-1 or α s-a-0. Then (p, q) is said to be the output error *location* of a double-error, and α is the location of the fault. If for condition h_{jk} output lines $f_{l_1}, f_{l_2}, \ldots, f_{l_q}$ are in error, then $(l_1, l_2, \ldots, l_q)$ is said to be the output error location and it consists of the following set of double-error locations: $\{(l_a, l_b) | a<b; l_a, l_b \in \{l_1, l_2, \ldots, l_q\}\}$. The number of output lines in error, their indices, and the output error pattern, are a few parameters one needs to know before an error detecting circuit can be implemented.

In a parity checking system, multiple errors of odd degree (q odd) can always be detected. It is those of even degree which will fail to be detected. We will first consider the case for q=2, since it is the simplest. We will then show how to extend the results to q=4, 6, . . . , etc.

An error is said to be *located* when the output lines in error are identified. One way of locating all possible double-errors in a circuit is to enumerate all pairs of outputs. This can be done for each fault which may exist in the circuit. Depending on the circuit structure, not all output pairs and faults need to be considered. Note, for example, that under the single fault assumption no multiple errors can exist in a fanout-free circuit.

Structural modeling and graph theoretic results

Consider two nodes in C, say α and β. If every path from node β to every PO includes α (α can itself be a PO), then α is said to be *essential* to β, and this is denoted by $\alpha E \beta$. In this case, if a fault in β causes an error in the signal at α when X_i is applied, then one can always assume there is a fault labeled "α s-a-$\bar{\alpha}$" occurring at this time. Thus, it is sufficient to deal with double errors due to faults occurring at α and we can ignore double errors due to faults at β.

A node α is called a *prime node* (PN) in C if there exists no other node β, $\beta \neq \alpha$, such that $\beta E \alpha$ (note that a PI can never be a PN). α is called a *prime fanout node* (PFN) if it is also a fanout node.

In the circuit of Figure 3, nodes α_2, α_3, and α_4 are the only three PFN's. The four primary outputs are PN's only.

Theorem 1: The set of all PFN's form a minimum set of nodes where faults associated with these nodes are sufficient to cause all multiple-output errors in a circuit. □

If all PFN's in a circuit can be identified, then the location of all possible double-errors can be made more easily and efficiently. In addition, the design of error detecting circuits, as we will see in the next section, depends heavily on this information. One method of identifying all the PFN's of a circuit is through a structural modeling process of Ko.[8] In this process, the final circuit model is represented by a directed graph G showing all the PFN's of the circuit. We call this graph G the fanout-graph for the circuit C and it has the following properties:

1. Every node in G is a prime node in C and G contains the complete set of prime nodes of C.
2. There are as many disjoint subgraphs in G as there are disjoint sub-circuits in C (assume PI's can be shared).
3. All nodes in G are singular if C is fanout-free or limited fanout-free.

From these properties and Theorem 1 we immediately conclude that no multiple-output errors can occur in C if C is fanout-free or limited fanout-free. The fanout graph shown in Figure 4 is obtained by applying the structural modeling process to the circuit of Figure 4.

Analyzing circuit output errors using the boolean difference

The *Boolean difference* of a switching function $f=f(x_1, x_2, \ldots, x_n)$ with respect to x_i is defined as

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

and can be written as

$$\frac{df}{dx_i} = h(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n).$$

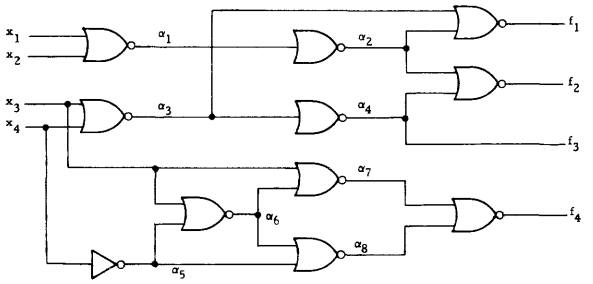


Figure 3—Example Circuit

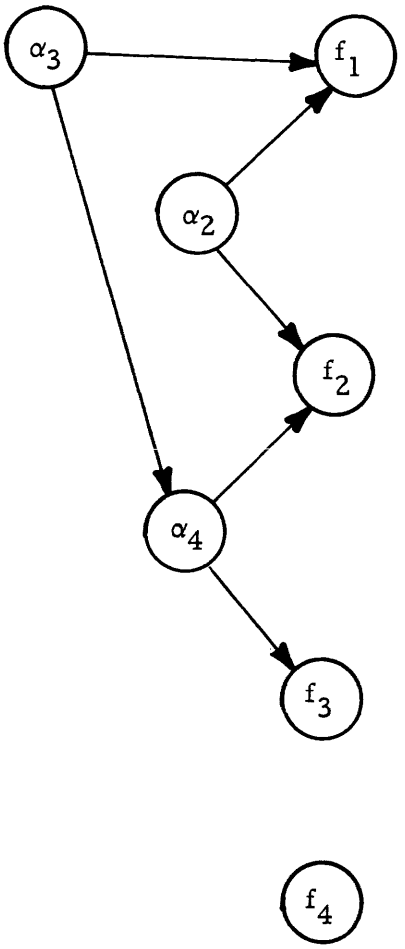


Figure 4—Fanout graph obtained by applying structural modeling process to the circuit of Figure 3

The Boolean difference of f with respect to an internal signal α rather than a primary input can be derived as follows. Write the output function f in the form $f=g(\alpha, X)$ where $\alpha=\alpha(X)$. Thus α becomes an explicit variable of f. Then

$$\frac{df}{d\alpha}=\frac{dg(\alpha, X)}{d\alpha}=g(0, X)\oplus g(1, X).$$

In general, let $f=g(\alpha_1, X)$ and $\alpha_i=\beta_i(\alpha_{i+1}, X)$ for $i=1, 2, \ldots, n-1$. Then

$$\frac{df}{d\alpha_n}=\frac{df}{d\alpha_1}\cdot\frac{d\alpha_1}{d\alpha_2}\cdots\frac{d\alpha_{n-1}}{d\alpha_n}. \tag{3.1}$$

Equation (3.1) is called the Boolean difference *chain* or the *partial* Boolean difference. Two important properties about the Boolean difference are that $df/d\bar{\alpha}=df/d\alpha$ and $d\bar{f}/d\alpha=df/d\alpha$. Note that: (a) if $df/d\alpha\equiv 0$, then an error in α will not cause an error in f; (b) if $df/d\alpha\equiv 1$, then an error in α will always cause an error in f; and (c) if $df/d\alpha=h(X)$ then an error in α will cause an error in f if and only if $h(X)=1$. Thus $df/d\alpha$ actually defines an *error function* whose value will be used in determining whether or not an error can be sensitized to the output. Let $w_i^{\alpha}=df_i/d\alpha$ be the error function of the output f_i with respect to α. We define a *pairwise error function* $w_{ij}^{\alpha}=w_i^{\alpha}\cdot w_j^{\alpha}$ as the logical product of two error functions. Three cases exist.

Case 1. If $w_{ij}^{\alpha}\equiv 0$ then an error in α will cause either a single-error or no error on the output pair f_i and f_j.

Case 2. If $w_{ij}^{\alpha}\equiv 1$ then an error in α will always cause a double-error on the output pair f_i and f_j.

Case 3. If $w_{ij}^{\alpha}=h(X)$ then an error in α will cause a double-error on the output pair f_i and f_j if and only if $h(X)=1$.

Example 1: Consider the circuit of Figure 3. Its fanout graph is shown in Figure 4. From Theorem 1, only the three PFN's namely α_2, α_3, and α_4 need be considered for possible causes of multiple errors. Since f_4 is a singular node, it can be ignored. For the remaining three terminating nodes we express their output functions in terms of the PFN's, i.e.,

$$f_1=\overline{\alpha_2+\alpha_3}=\bar{\alpha}_2\bar{\alpha}_3$$

$$f_2=\overline{\alpha_2+\alpha_4}=\bar{\alpha}_2\bar{\alpha}_4=\bar{\alpha}_2\alpha_3$$

$$f_3=\alpha_4=\bar{\alpha}_3.$$

Since $m=l=3$, there exist a total of $\binom{3}{2}\times 3=9$ pairwise error functions. Among them, 4 are trivial. For instance, f_1 is not a function of α_4, hence $w_{12}^{\alpha_4}$ and $w_{13}^{\alpha_4}$ must be 0. The 5 non-trivial ones are

$$w_{12}^{\alpha_2}=\bar{\alpha}_3\cdot\alpha_3=0 \qquad w_{12}^{\alpha_3}=\bar{\alpha}_2\cdot\bar{\alpha}_2=\bar{\alpha}_2$$

$$w_{13}^{\alpha_3}=\bar{\alpha}_2\cdot 1=\bar{\alpha}_2=w_{23}^{\alpha_3}=w_{23}^{\alpha_4}.$$

Thus when $\bar{\alpha}_2=x_1+x_2=1$, multiple errors can occur whenever there is an error in α_3 or α_4. The error in α_3 or α_4 can either be a fault in the node itself, or it can be caused by some other fault in a preceding node. In this example, since $w_{12}^{\alpha_3}=w_{13}^{\alpha_3}=w_{23}^{\alpha_3}$, an error in α_3 can cause all three output pairs simultaneously to be in error leaving a net result of a triple-error. Note that an error in α_2 can never cause any double-error because $w_{ij}^{\alpha_2}=0$ for all $1\leqq i, j\leqq 4$.

A simple analysis reveals that a double-error $01\leftrightarrow 10$ will occur on the outputs f_1 and f_2 whenever there is an error in α_3 and the input condition is one which causes $\alpha_2=0$. A different type of double-error, namely $00\leftrightarrow 11$ can be found in the circuit for the output pair f_1 and f_3. We call "$01\leftrightarrow 10$" and "$00\leftrightarrow 11$" *error patterns*. Theorem 2 in the next section will be devoted to determining such error patterns.

In general, for $q>2$ a q-bit error can be considered as a group of k double-errors where $k=\binom{q}{2}$. Once all pairwise double-errors are located, multiple errors of higher degree can also be located. In order to do this, we introduce the notation of an *error-graph*. It is a non-directed graph such as the one shown in Figure 5(a). In this graph, every node is a terminating node of a fanout-graph. A *link* is entered into the graph if the pairwise error function of two outputs is not

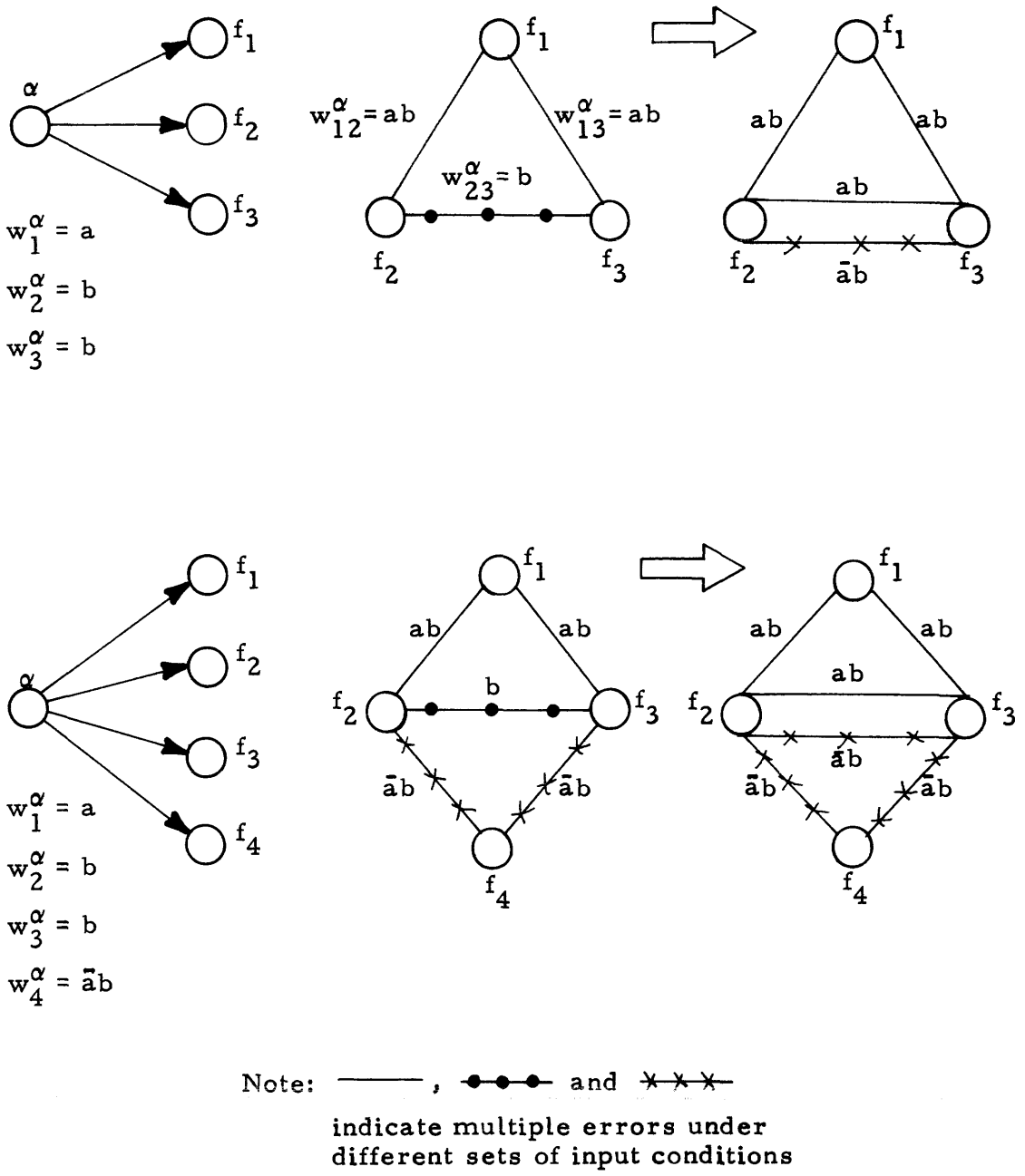


Figure 5—Fanout-graphs and error-graphs of two circuits

zero. Thus a link actually indicates the possibility of a double-error on its two end-nodes. We use different marks on the links to represent double-errors under different sets of input conditions. A closed triangle of identically marked links represents a possible triple-error.

For the graph of Figure 5(a) we have three pairwise double-errors. They are indicated by the three links representing $w_{12}=ab$, $w_{13}=ab$, and $w_{23}=b$. By rewriting w_{23} as the sum of two product terms ab and $\bar{a}b$ and using two distinct links, the resultant graph, shown in Figure 5(b), indicates a triple-error plus one double-error. This obviously will enable us to predict circuit output errors more precisely. Shown in Figure 5(c) is a second graph which shows five pairwise double-errors. By applying the same technique, only two triple-errors can be found in the final graph. With the addition of an extra output f_4 (as compared with the first circuit), this circuit becomes free of any multiple error of even degree. Therefore, this circuit can be parity checked without any further work.

Fanout-observed output functions

Given a circuit C, let α be a prime fanout node in C. If for some output f there exists at least one path between α and

f, then the output function of f derived after making a cut at node α (or treating α as a PI) can be expressed as follows:

$$f(\alpha, X)=A(X)\alpha+B(X)\bar{\alpha}+C(X). \quad (3.2)$$

We call Equation (3.2) the *F*anout-*O*bserved *O*utput *F*unction (FOOF) of f with respect to α. The Boolean switching functions A(X), B(X), and C(X) are called the Boolean *coefficients* of $f(\alpha, X)$. If A(X)=B(X), then f is independent of α. For f to be dependent on α, at least one of the two coefficients A(X) and B(X) must not be zero. A shorthand form of the above equation is

$$f=A\alpha+B\bar{\alpha}+C.$$

By denoting $\alpha^1=\alpha$ and $\alpha^0=\bar{\alpha}$, two special forms of the FOOF can be written as follows:

1. $f=a\alpha^u+b$ (3.3)
2. $f=a\alpha^u\oplus b$ (3.4)

where $u\in\{0, 1\}$; a and b are arbitrary switching functions independent of α.

We call Equation (3.3) the *+-form* and Equation (3.4) the *⊕-form* of the FOOF. In the +-form, f is unate in α. In the ⊕-form, both α and $\bar{\alpha}$ can appear in a minimal normal form expression for f unless the Boolean coefficient b has a constant value.

Lemma 1: If no linear gate or reconvergent fanout exists between α and f, then f can be expressed by a FOOF of the +-form only. □

Consider a FOOF $f(\alpha, X)$ where $\frac{df(\alpha, X)}{d\alpha}\neq 0$. There must exist at least one input $X_k\in\chi$ such that $\left.\frac{df(\alpha, X)}{d\alpha}\right|_{X=X_k}=1$. When this condition is satisfied, α will be sensitized to the output. The value of f under this condition will be $f(\alpha, X_k)$. Let

$$Y_f=\left\{X_k\in\chi \left| \left.\frac{df(\alpha, X)}{d\alpha}\right|_{X=X_k}=1\right.\right\} \quad (3.5)$$

be a non-empty set of all inputs under which α can be sensitized to the output. Set

$$Z_f=\{f(\alpha, X_k)\,|\,X_k\in Y_f\}.$$

Then Z_f is the non-empty set of all possible switching functions realized by f when sensitized by α. Note that Z_f is undefined if $Y_f=\emptyset$, or equivalently $df/d\alpha\equiv 0$.

Lemma 2: $Z_f\subseteq\{\alpha, \bar{\alpha}\}$. □

For the next theorem we need the following definitions. Let f and g be two FOOF's with respect to α, where $\frac{df}{d\alpha}\cdot\frac{dg}{d\alpha}\neq 0$. We define

$$Y_{fg}=\left\{X_k\in\chi \left| \left.\frac{df}{d\alpha}\cdot\frac{dg}{d\alpha}\right|_{X=X_k}=1\right.\right\}=Y_f\cap Y_g$$

and

$$Z_{fg}=\{f(\alpha, X_k)\cdot g(\alpha, X_k)\,|\,X_k\in Y_{fg}\}.$$

Lemma 3: $Z_{fg}\subseteq\{0, \alpha, \bar{\alpha}\}$. □

Theorem 2: Let $f(\alpha, X)$ and $g(\alpha, X)$ be the two FOOF's of a pair of outputs f and g. Then an error in α will cause a $01\leftrightarrow 10$ error pattern if and only if $Z_{fg}=\{0\}$. It will cause a $00\leftrightarrow 11$ error pattern if and only if $Z_{fg}\subseteq\{\alpha, \bar{\alpha}\}$. □

Now we will define the "variance" of a FOOF. The uniqueness of a double-error pattern can be determined by the variance of two FOOF's.

A FOOF $f(\alpha, X)$ is said to be *α-invariant* if $Z_f=\{\alpha\}$ or $\{\bar{\alpha}\}$. A FOOF $f(\alpha, X)$ is said to be *α-variant* if $Z_f=\{\alpha, \bar{\alpha}\}$.

A pair of outputs is said to have a *unique* double-error pattern if all possible double-errors associated with the outputs are of either $01\leftrightarrow 10$ pattern or $00\leftrightarrow 11$ pattern but not both.

Lemma 4: Any FOOF of the +-form is α-invariant. □

Theorem 3: A pair of outputs have a unique double-error pattern if both FOOF's are α-invariant (assume error in α only). □

Corollary 1: In a non-reconvergent fanout circuit, if no linear gates are in the circuit, then all double-errors have a unique error pattern. □

In the remainder of this section we will discuss some equivalent forms of FOOF's. An α-augmented function will then be introduced and an augmented parity function will be investigated. These results will aid us in the design of error detecting circuitry. Their application can be found in the next section.

Lemma 5: If $f=a\alpha^u+b$, then $f=\frac{df}{d\alpha}\alpha^u+b$. □

Lemma 6: For $\#\in\{+, \oplus\}$, if $f=a\alpha^u\#b$ then $f=\frac{df}{d\alpha}\alpha^u\oplus b$. □

Theorem 4: Any FOOF of the form $f=a\alpha^u\#b$ can be expressed in one of the following two forms:

1. $f=\frac{df}{d\alpha}\alpha^u\#b$
2. $f=\frac{df}{d\alpha}\alpha^u\oplus b$. □

We now define an *α-augmented function* as a Boolean switching function of the form $Q(\alpha, X)=w(X)\alpha^u$ where w(X) is an arbitrary switching function (for our application, w(X) is an error function). Note that $Q(\alpha, X)$ is also a FOOF of a special form. This function can be implemented to augment a prime fanout node α of a circuit such that a q-bit output error (q even) can be transformed into a (q+1)-bit error.

Consider the case when there exist two outputs f and g in a circuit C, where $f=a\alpha^u\#b$ and $g=c\alpha^u\#d$ are the two associated FOOF's. The pairwise error function is $\frac{df}{d\alpha}\cdot\frac{dg}{d\alpha}$. Suppose $\frac{df}{d\alpha}\cdot\frac{dg}{d\alpha}\not\equiv 0$, then let $Q=\left(\frac{df}{d\alpha}\cdot\frac{dg}{d\alpha}\right)\alpha^u$ be an α-augmented function. Under any input $X_k\in Y_{fg}$, an error in α will cause a double-error on the two outputs f and g.

Since $\left.\frac{dQ}{d\alpha}\right|_{x=x_k}=1$, the output Q will also be in error. The net result is a triple-error on the outputs f, g, and Q. The parity function for these three outputs is P=f⊕g⊕Q and is called the *augmented parity function*.

Lemma 7: $P=\left(\frac{df}{d\alpha}+\frac{dg}{d\alpha}\right)\alpha^u \oplus b \oplus d$. □

It is seen that both P and Q contain the terms $\frac{df}{d\alpha}$ and $\frac{dg}{d\alpha}$. In the implementation of P and Q, if $\frac{df}{d\alpha}$ and $\frac{dg}{d\alpha}$ can be built once and shared by both P and Q, then a saving in the hardware can be achieved. Provision must be made that a fault in the node $\frac{df}{d\alpha}$ or $\frac{dg}{d\alpha}$ must not cause any double-error on the outputs P and Q. Otherwise, it cannot be detected. Let $\beta=\frac{df}{d\alpha}$ and $\gamma=\frac{dg}{d\alpha}$ be the two nodes of interest. We require $w_{PQ}{}^{\beta}=w_{PQ}{}^{\gamma}\equiv 0$ where w's are the pairwise error functions for P and Q.

Theorem 5: $w_{PQ}{}^{\beta}=w_{PQ}{}^{\gamma}\equiv 0$. □

EXTENDED-PARITY CHECKING

In this section we will show how to apply the preceding theory to the design of checking circuits.

Forced-parity methods

Two methods will now be presented in which additional hardware is introduced. By using these methods one can be assured that the output error of a circuit will always be of odd degree. In this case, a parity checker alone is sufficient to detect all output errors. This approach is invalid if certain PFN's of a circuit are inaccessible. However, it can serve as a design guide in the initial layout of self-checking circuitry.

Fanout degeneration

Given a circuit C, let α be a PFN of C. The fanout value τ_α can be interpreted in two ways. One is the actual number of fanouts of α in C. The other one is the outdegree of α as it appears in the fanout-graph for C. Unless otherwise indicated we will use the latter definition.

Now consider a circuit C whose fanout-graph G is shown in Figure 6(a). The only PFN is α and $\tau_\alpha=2$. It has two associated outputs f_i and f_j. Assume an error in α can cause a double-error on the two outputs. In order to eliminate this double-error, we can remove either one of the two branches αf_i or αf_j from G. The resultant graph G′ with αf_j removed is shown in Figure 6(b). In G′, α is no longer a prime node (since $f_i E \alpha$) and hence can be deleted. The final graph G″ is shown in Figure 6(c). Since G″ is a singular graph, no multiple errors can occur on the outputs.

The removing of the branch αf_j from G corresponds to a

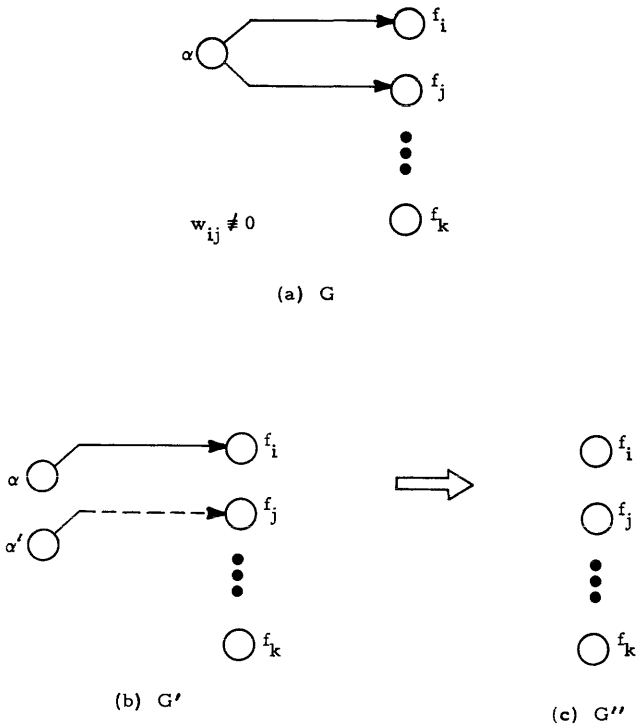


Figure 6—Fanout-graphs for a circuit before and after fanout degeneration

degeneration in the number of fanouts of α in C. This can be accomplished by constructing a new signal α' to replace one or more† of the fanout signals of α. Here α' is logically identical to, yet structurally independent of α. In other words, if $C(\alpha)$ and $C(\alpha')$ are two sub-circuits whose outputs are α and α' respectively, we have $\alpha=\alpha'$ with or without some commonly shared components. A check for new multiple errors must be made, unless $C(\alpha')$ is a duplicate of $C(\alpha)$ and is fed only by PI's. The new circuit, labeled $\tilde{C}$, can be parity checked. Note that for G″ to be singular does not necessarily imply $\tilde{C}$ is fanout-free. This method is essentially a resynthesis procedure since no new output leads are formed.

The method can be greatly enhanced if, instead of completely removing a PFN α from G, α is allowed to stay in G so long as an error in α cannot cause any multiple error of even degree in C. Consider a circuit whose fanout-graph G is shown in Figure 7(a). The error characteristics of this circuit are represented by a Venn diagram shown in Figure 7(b). In this diagram each element Y_i, Y_j or Y_k is a set of input n-tuples defined by Equation (3.5). There are two possible double-errors in the circuit as are indicated by their intersection Y_{ik} and Y_{jk}. By removing the set Y_k from this diagram, all double-errors can be eliminated. The removal of Y_k corresponds to the deletion of a directed branch αf_k from G. So we conclude that only one signal α' needs to be generated. This signal α' will be used to implement f_k. The result is a reduction in τ_a from 3 to 2, and the resulting circuit will be free of any multiple errors.

† f_j can be a reconvergent node.

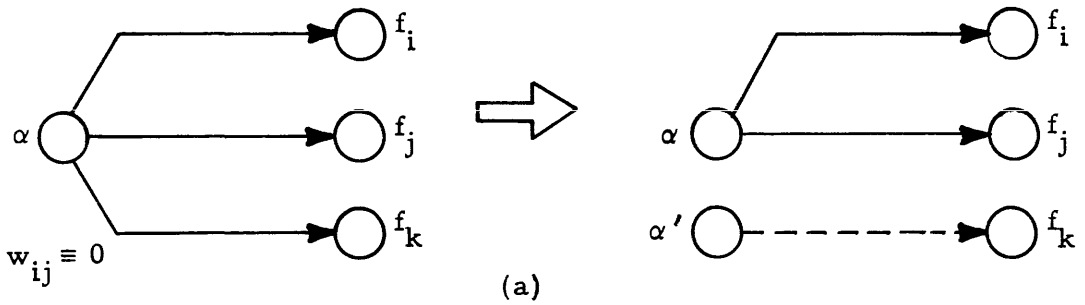
Shown in Figure 7(c) is the change in the error-graph for this circuit. Since there is a close resemblance between the Venn diagram representation and the error-graph, we will use the latter as a working model in our future applications.

For the circuit just presented, the decision on removing αf_k is obvious. For more complicated problems a general procedure is required in order to select pairs (αf_i) to be removed from G. One such heuristic procedure is given in Ko [8], and for brevity, will not be presented here.

Once a node has gone through the degeneration process the fanout-graph is simplified accordingly, and the process is repeated for another PFN. Since each iteration of this process removes a PFN from G, this procedure will terminate when the final graph reaches a singular graph.

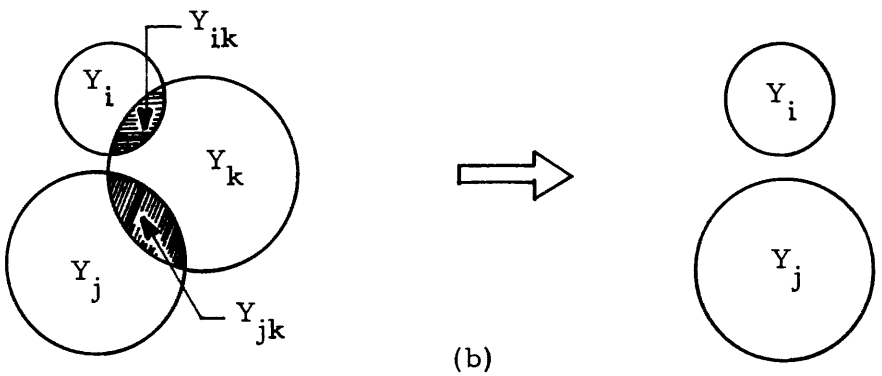
Fanout augmentation

Contrary to the previous method, the Fanout Augmentation method does not require any duplication of the fanout



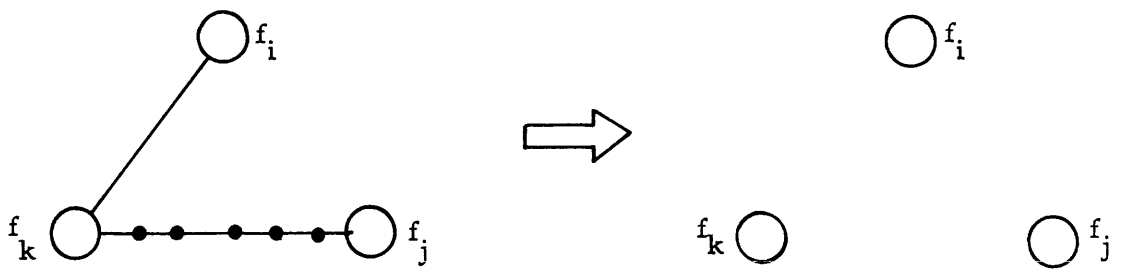
(a)

The Fanout-Graphs



(b)

The Venn Diagrams



(c)

The Error-Graphs

NOTE: ——— and —●—●—● indicate double-errors under different sets of input conditions

Figure 7—Graphic results showing net changes in a fanout degeneration process

signals. Instead, a line is tapped off on a fanout node which after some gating logic is sent to a parity checker. The new output signal, say $\hat{\alpha}$ will perform the function of transforming any multiple error of even degree into odd degree as long as it is caused by an error in α.

Consider the fanout-graph of Figure 6(a). Instead of removing the branch αf_j from G, we want to add a new branch $\alpha\hat{\alpha}$ to G such that the double-error on f_i and f_j can be transformed into a triple-error on f_i, f_j and $\hat{\alpha}$. The transformation can be achieved by implementing an α-augmented function $\hat{\alpha}(\alpha, X)=w_{ij}(X)\cdot\alpha^u$ where w_{ij} (the error function) will be our gating function. Since $d\hat{\alpha}/d\alpha=w_{ij}$, an error in α will also cause $\hat{\alpha}$ to be in error whenever $w_{ij}=1$ under some input in Y_{ij}.

We will now show how this method will affect the fanout-graph and error-graph of a circuit. Consider the circuit whose fanout-graph is shown in Figure 7(a). In this circuit, there exist two possible double-errors on the output pairs (f_i, f_k) and (f_j, f_k). By implementing a new signal $\hat{\alpha}=(w_{ik}+w_{jk})\cdot\alpha^u$, each double-error can be transformed into a triple error. The resultant graphs showing these changes can be found in Figure 8.

Theorem 6: Given a fanout-graph G. If $B_\alpha=\{\alpha f_{l_1}, \alpha f_{l_2}, \ldots, \alpha f_{l_M}\}$ where f_{l_i}, $1\leq i\leq M$, is a terminating node in G, then

$$\hat{\alpha}=\left[\left(\overline{\bigoplus_{i=1}^{M} w_i}\right)\cdot\left(\sum_{i=1}^{M} w_i\right)\right]\cdot\alpha^u \quad \text{where} \quad w_i=\frac{df_{l_i}}{d\alpha}. \quad \square \qquad (4.1)$$

Theorem 7: Let α be a PFN in G and $B_\alpha=\{\alpha\alpha_1, \alpha\alpha_2, \ldots, \alpha\alpha_M\}$ be the set of all M directed branches whose starting-node is α, and end-nodes are α_i, $1\leq i\leq M$. Associated with each node α_i is a set N_i consisting of all terminating nodes having a path from α_i. We will allow the case where $N_i=\{\alpha_i\}$ and call $\alpha_i\alpha_i$ a legitimate path. If $N_i\cap N_j=\emptyset$ for all $i\neq j$, and if every PFN in G is to be processed by the augmentation technique, then

$$\hat{\alpha}=\left[\left(\overline{\bigoplus_{i=1}^{M} W_i}\right)\cdot\left(\sum_{i=1}^{M} W_i\right)\right]\cdot\alpha^u \qquad (4.2)$$

where $W_i=\sum_{f_k\in N_i}\frac{df_k}{d\alpha}$. $\square$

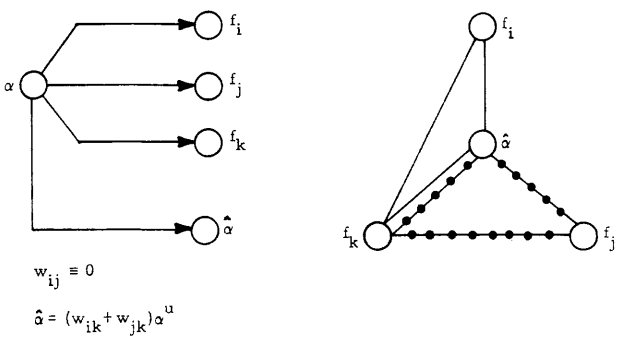


Figure 8—The resultant graphs for Figure 7 after a fanout augmentation process

Note that if every node α_i, $1\leq i\leq M$ is a terminating node, then $W_i=w_i$. In this case, Equation (4.1) and Equation (4.2) are the same.

Line sensing techniques

Under circumstances when circuit constraints or other factors prohibit the use of forced-parity techniques, the *Line Sensing* techniques should be investigated since they may provide a good result. In this section we will discuss two methods which do not require accessing to the PFN's.

Conditional line sensing

In the fanout degeneration method, if a branch αf_i is removed from G, we need to build a new signal α' to replace the line(s) being cut in C. In this method we will build the same α' (or its complement), not for replacement but for comparison. Consider the example of Figure 6 where $B_\alpha=\{\alpha f_i, \alpha f_j\}$. Suppose f_i is α-invariant and we decide to sense f_i. Under input conditions such that $w_i=1$ we will have $Z_i=\{\alpha\}$ or $\{\bar{\alpha}\}$. Since Z_i contains only a single element, we can associate with f_i a switching function α^u and call it the function realized by f_i under $w_i=1$. Let $\alpha'=\alpha^u$ and it can be used to compare with the "line" f_i under the "condition" of $w_i=1$. Shown in Figure 9(a) is such a scheme, and we call it the *Conditional Line Sensing* method. In this method, an Exclusive-OR gate is used to compare f_i with $\alpha^u(X)$. Its output is then gated by the switching function $w_i(X)$. The final output is an error signal ϵ_i which will be set to 1 whenever an error in α causes an error on f_i independent of whether or not f_j is in error. Let ϵ_p be the output of a parity checker checking on all the outputs. Then $\epsilon=\epsilon_p+\epsilon_i$ will be our overall error signal for the circuit. Note that an error of f_i caused by some other source may or may not set $\epsilon_i=1$. That is why f_i should also be included in the parity checking.

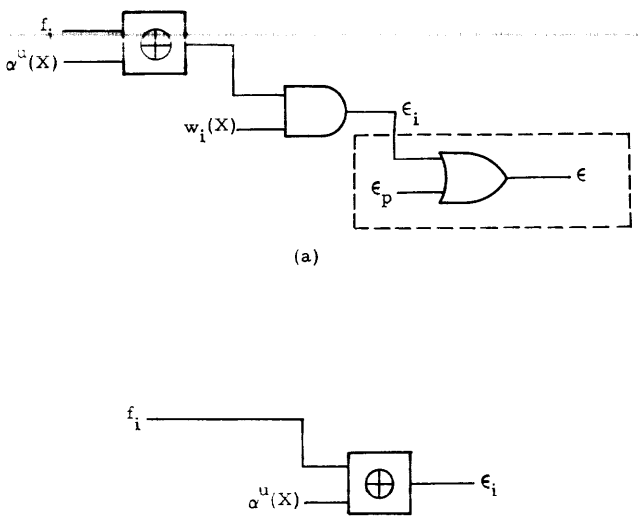


Figure 9—Conditional line sensing

Mathematically, we have

$$\epsilon_i = w_i(f_i \oplus \alpha^u)$$

A special situation is $w_i \equiv 1$ in which case

$$\epsilon_i = f_i \oplus \alpha^u.$$

Its implementation is shown in Figure 9(b).

It should be pointed out that the gating function w_i can actually be replaced by the pairwise error function w_{ij}. When this is done, ϵ_i will be set to 1 whenever a double-error occurs on the outputs f_i and f_j. Thus what is undetected by the parity checker ($\epsilon_p = 0$) will now be detected by the line sensing mechanism ($\epsilon_i = 1$) and the result is $\epsilon = 1$. For the graph of Figure 7 a gating function of $w_{ik} + w_{jk}$ will also be appropriate if f_k is α-invariant. In any event, one should select that implementation which is of least cost.

Now let us consider the case when an output function f_i is α-variant. Since $Z_i \subseteq \{\alpha, \bar{\alpha}\}$, α^u alone will no longer be sufficient to serve as a reference signal. In order to solve this problem, we express f_i in the general form $f_i = A\alpha + B\bar{\alpha} + C$. The error function w_i is readily found to be $(A \oplus B)\bar{C} = A\bar{B}\bar{C} + \bar{A}B\bar{C} = W_1 + W_2$, where $W_1 = A\bar{B}\bar{C}$, and $W_2 = \bar{A}B\bar{C}$. It is seen that when $W_1 = 1$, f_i will be equal to α, and when $W_2 = 1$, f_i will be equal to $\bar{\alpha}$. So clearly we can write the following equation:

$$\epsilon_i = W_1(f_i \oplus \alpha) + W_2(f_i \oplus \bar{\alpha}).$$

Again, all the previous arguments will still hold for each term in ϵ_i.

For any circuit, if more than one ϵ_i is generated, then ϵ should be set as follows:

$$\epsilon = \epsilon_p + \sum_i \epsilon_i. \qquad (4.3)$$

Unconditional line sensing

In this method the input condition plays no important role in the design. First of all, the double-error patterns of a pair of outputs (f_i, f_j) have to be determined. If it has a unique double-error pattern then before duplicating a line f_i, we first perform a functional mapping on f_i and f_j as follows:

1. If (f_i, f_j) has a unique $00 \leftrightarrow 11$ pattern, then let g_i be a function defined by any one of the following expressions:

   (a) f_i (d) $\bar{f}_i$
   (b) $f_i + f_j$ (e) $\bar{f}_i \cdot \bar{f}_j$
   (c) $f_i \cdot f_j$ (f) $\bar{f}_i + \bar{f}_j$

2. If (f_i, f_j) has a unique $01 \leftrightarrow 10$ pattern, then define g_i to be any one of the following:

   (a) f_i (d) $\bar{f}_i$
   (b) $f_i + \bar{f}_j$ (e) $\bar{f}_i \cdot f_j$
   (c) $f_i \cdot \bar{f}_j$ (f) $\bar{f}_i + f_j$

The criterion in selecting one of the expressions in each group as g_i is based on the cost of implementing such a function. Once g_i is selected, we can implement a comparison scheme such as the one shown in Figure 10. Let G_i be a circuit which realizes g_i and is implemented using only the signal PI's as inputs. We construct another circuit G_i' which realizes the same function g_i but its inputs are now taken directly from f_i and f_j. Call its output g_i'. We can perform the following comparison

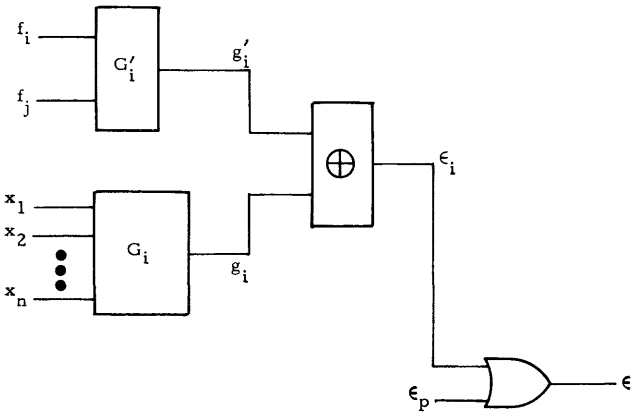


Figure 10—Unconditional line sensing

$$\epsilon_i = g_i(X) \oplus g_i'(f_i, f_j)$$

using only one Exclusive-OR gate. We call this method the *Unconditional Line Sensing* method since the function of w_i is no longer involved.

In this method, unless g_i is chosen to be f_i or $\bar{f}_i$, all the outputs are still required to be sent to a parity checker. On the other hand, if g_i equals f_i or $\bar{f}_i$, then f_i can be excluded from the parity checking. In this case, a partial duplication is implied. For the case when a pair of outputs do not have a unique double-error pattern, we require g_i to be either f_i or $\bar{f}_i$. As was mentioned before, if more than one ϵ_i is generated, then Equation (4.3) will have to be used.

As a final note we would like to point out that this method can be extended to include the case of setting $g_i = \alpha_i$ if a PFN α_i is accessible. We call this method the *Fanout Duplication* method.

In conclusion, these methods have been used to design checking circuits for a number of functional devices as well as random logic. Examples and conclusions dealing with the suitability of specific techniques of different types of logic circuits, such as iterative arrays can be found in Reference 8.

REFERENCES

1. Bouricius, W. G., W. C. Carter, K. A. Duke, J. P. Roth, and P. R. Schneider, "Interactive Design of Self-Testing Circuitry," *Proc. Purdue Centennial Year Symp. on Inform. Processing*, April 1969, pp. 73-80.
2. Carter, W. C. and P. R. Schneider, "Design of Dynamically Checked Computers," *Proc. IFIP*, Vol. 2, August 1968, pp. 878-883.

3. Carter, W. C., D. C. Jessep, and A. B. Wadia, "Error-Free Decoding for Failure-Tolerant Memories," *Proc. IEEE International Computer Group Conference,* June 1970, pp. 229-239.
4. Friedman, A. D. and P. R. Menon, *Fault Detection in Digital Circuits,* Prentice-Hall, Englewood Cliffs, New Jersey, 1971.
5. Gerrand, F. and H. S. Rasmussen, "Self-Correction in Large Scale Digital Computers," *Proc. National Symposium on Reliability and Quality Control,* Philadelphia, Pennsylvania, January 1961, pp. 351-360.
6. Harary, F., R. Z. Norman, and D. Cartwright, *Structural Models—An Introduction to the Theory of Directed Graphs,* J. Wiley, New York, 1965.
7. Kautz, W. H., "Automatic Fault Detection in Combinational Switching Networks," *Proc. 2nd Annual Symposium on Switching Circuit Theory and Logical Design,* 1961, pp. 195-214.
8. Ko, D. C., "Self-Checking of Multi-Output Combinational Circuits Using Forced-Parity Techniques," University of Southern California Electronic Sciences Laboratory Report No. 451, June 1973.
9. Russell, J. D. and C. R. Kime, "Structural Factors in the Fault Diagnosis of Combinational Networks," *IEEE Trans. on Computers,* Vol. C-20, November 1971, pp. 1276-1285.
10. Sellers, F. F., M. Y. Hsiao, L. W. Bearnson, "Analyzing Errors with the Boolean Difference," *IEEE Trans. on Computers,* Vol. C-17, July 1968, pp. 676-683.
11. Sellers, F. F., M. Y. Hsiao, and L. W. Bearnson, *Error Detecting Logic for Digital Computers,* McGraw-Hill, New York, 1968.
12. Wilcox, R. H. and W. C. Mann, eds., *Redundancy Techniques for Computing Systems,* Spartan Books, Washington, D.C., pp. 205-228, 1962.