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 E, 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. 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 \( f_i=f_i(x_1, x_2, \ldots, x_n) \) 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 \( \{f_1, f_2, \ldots, f_m\} \) 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 \( F_i=(f_1, f_2, \ldots, f_m) \).

Let \( X_i \) represent the binary input vector whose decimal value is i, e.g. \( X_3=(00\ldots011) \) and set \( \chi=\{X_i|i=0, 1, \ldots, 2^n-1\} \). By \( F(X_\chi) \) we mean the value of F for input \( X=X_\chi \).

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).
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 $\Delta^C = \{\delta_1, \delta_2, \ldots, \delta_n\}$ be a set of faults associated with $C$. Then we denote the circuit $C$ containing fault $\delta_j$ by $C_j$, where $C_j(=C)$ represents the fault-free circuit. $C_j$ realizes the function $F_j(X) = f_{j1}(X), f_{j2}(X), \ldots, f_{jn}(X)$. If $F(X) \neq F_j(X)$, then fault $\delta_j$ is detectable by input $X_k$.

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

$$\Delta f_j(X_k) = f_j(X_k) \oplus f_{j0}(X_k),$$

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

$$\Delta F(X_k) = F(X_k) \oplus F_j(X_k) = (\Delta f_1(X_k), \Delta f_2(X_k), \ldots, \Delta f_m(X_k)).$$

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

Suppose that $|\Delta F(X_k)| = q$, $q = 1, 2, 3$, or $n(4 \leq n \leq m)$.

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 $\epsilon$, 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 $\delta_j$ in $C$, i.e. $\Delta F(X_k) \neq 0$ for each $X_k \in \chi(\delta_j)$. Since $C$ is irredundant $\chi(\delta_j) \neq \emptyset$.

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

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

If all $\delta_j$ 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 uncheckable, 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 $h_{kj} \in H = \Delta^C \times X$, where $\Delta^C$ is the set of faults associated with the new circuit $(C, D)$, we require

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

We assume that the fault $\epsilon$ is a-s-a fault and any fault equivalent to it in $D$ is not included in $\Delta^C$. 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 1. Here we augment $C$ with the logic $\hat{c}$ having outputs $\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_\ell$. $\hat{c}$ 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. $\hat{P}$ is a circuit which implements the parity function defined by the expression

$$\left( \bigoplus_{i=1}^{n} f_i \right) \oplus \left( \bigoplus_{i=1}^{\ell} a_i \right).$$

The outputs of $\hat{P}$ and $\hat{P}'$ 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 $h_{kj} = (\delta_j, X_k)$ that $f_p$ and $f_q$ are in error, and that $\delta_j$ occurs at signal $a_{i}$, i.e. $\delta_i$ corresponds to $a_{i}$.
Design of Self-Checking Multi-Output Combinational Circuits

Design of Self-Checking Multi-Output Combinational Circuits

The design of self-checking multi-output combinational circuits is a critical aspect of digital system design. In such circuits, each output is connected to multiple inputs, which can lead to complex error propagation and detection. The key to self-checking is the ability to detect errors quickly and accurately, without interrupting the circuit's normal operation.

### Structural Modeling

When designing a self-checking circuit, a common approach is to use structural modeling. This involves representing the circuit as a graph, where nodes represent circuit elements and edges represent connections. The goal is to identify essential nodes—those that are crucial for the circuit's operation—and prime fanout nodes (PFNs)—nodes that have a single output that is connected to multiple inputs.

### Analyzing Circuits

Analyzing circuit output errors is a fundamental aspect of self-checking. The boolean difference of a switching function is a key concept in this area. The boolean difference of a function with respect to a variable is used to identify how changes in that variable affect the function's output. This is particularly useful in detecting and locating errors in a circuit.

### Self-Checking Circuits

Self-checking circuits are designed to detect and correct errors as they occur. This is achieved through the use of parity checking systems. These systems are designed to detect multiple errors of odd degree, which are always detectable. However, multiple errors of even degree can fail to be detected, and these are the focus of more advanced self-checking techniques.

### Conclusion

In conclusion, the design of self-checking multi-output combinational circuits is a complex but vital field in digital system design. Techniques such as structural modeling, self-checking circuits, and parity checking systems are essential tools in ensuring the reliability and correctness of digital systems.
The Boolean difference of $f$ with respect to an internal signal $a$ rather than a primary input can be derived as follows. Write the output function $f$ in the form $f=g(a, X)$ where $a=a(X)$. Thus $a$ becomes an explicit variable of $f$.

Then

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

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

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

Equation (3.1) is called the Boolean difference chain or the partial Boolean difference. Two important properties about the Boolean difference are that $df/d\alpha = df/d\alpha$ and $df/d\alpha = df/d\alpha$. Note that: (a) if $df/d\alpha = 0$, then an error in $\alpha$ will not cause an error in $f$; (b) if $df/d\alpha = 1$, then an error in $\alpha$ will always cause an error in $f$; and (c) if $df/d\alpha = h(X)$ then an error in $\alpha$ will cause an error in $f$ if and only if $h(X)=1$.

Thus $df/\alpha_1 = df/\alpha_n$, which is called an error function. In particular, $df/\alpha_i$ is an error function whose value will be used in determining whether or not an error can be sensitized to the output. Let $w_i^n= df/d\alpha$ be the error function of the output $f_i$ with respect to $\alpha$. We define a pairwise error function $w_{ij}^n = w_i^n \cdot w_j^n$ as the logical product of two error functions. Three cases exist:

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

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

Case 3. If $w_{ij}^n=h(X)$ then an error in $\alpha$ 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 $\alpha_2, \alpha_3$, and $\alpha_4$ need be considered for possible causes of multiple errors. Since $f_2$ 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.,

$$f_1 = \alpha_2 + \alpha_3 = \alpha_2 \alpha_3$$

$$f_2 = \alpha_2 + \alpha_4 = \alpha_2 \alpha_4$$

$$f_3 = \alpha_4 = \alpha_3.$$ 

Since $m=3$, there exist a total of $\binom{3}{2} = 9$ pairwise error functions. Among them, 4 are trivial. For instance, $f_1$ is not a function of $\alpha_4$, hence $w_{14}^n$ and $w_{13}^n$ must be 0. The 5 non-trivial ones are

$$w_{12}^n = \alpha_2 \alpha_3 = 0 \quad w_{12}^n = \alpha_2 \alpha_3 = 0$$

$$w_{13}^n = \alpha_2 \alpha_4 = 0 \quad w_{13}^n = \alpha_2 \alpha_4 = 0$$

$$w_{23}^n = \alpha_2 \alpha_3 = 0 \quad w_{23}^n = \alpha_2 \alpha_3 = 0.$$ 

Thus when $\alpha_2 = x_1 + x_2 = 1$, multiple errors can occur whenever there is an error in $\alpha_2$ or $\alpha_3$. The error in $\alpha_2$ or $\alpha_3$ 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 $w_{12}^n = w_{13}^n = w_{24}^n$, an error in $\alpha_2$ can cause all three output pairs simultaneously to be in error leaving a net result of a triple-error. Note that an error in $\alpha_2$ can never cause any double-error because $w_{ij}^n = 0$ for all $1 \leq i, j \leq 4$.

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 $\alpha_2$ 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 $f_2$. We call "01 ↔ 10" and "00 ↔ 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 $k = \left(\begin{array}{c} q-1 \\ 2 \end{array} \right)$. Once all pairwise double-errors are located, multiple errors of higher degree can also be located. In order to do this, we introduce the notion 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
Design of Self-Checking Multi-Output Combinational Circuits

Note: , - - - and * * * indicate multiple errors under different sets of input conditions.

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 \( ab \) 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 \( \alpha \) be a prime fanout node in \( C \). If for some output \( f \) there exists at least one path between \( \alpha \) and
Lemma 1: If no linear gate or reconvergent fanout exists between \(\alpha\) 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 \(X_k \in X\) such that \(\frac{df(\alpha, X)}{d\alpha} |_{X=X_k} = 1\).

When this condition is satisfied, \(\alpha\) will be sensitized to the output. The value of \(f\) under this condition will be \(f(\alpha, X_k)\).

Let

\[
Y_f = \left\{ X_k \in X \mid \frac{df(\alpha, X)}{d\alpha} |_{X=X_k} = 1 \right\}
\]

be a non-empty set of all inputs under which \(\alpha\) can be sensitized to the output. Set

\[
Z_f = \{ f(\alpha, X_k) \mid X_k \in Y_f \}.
\]

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

Lemma 2: \(Z_f \subseteq \{\alpha, \tilde{\alpha}\}\). □

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

\[
Y_g = \left\{ X_k \in X \mid \frac{df}{d\alpha} \frac{dg}{d\alpha} |_{X=X_k} = 1 \right\} = Y_f \cap Y_g
\]

and

\[
Z_{fg} = \{ f(\alpha, X_k) \mid g(\alpha, X_k) \mid X_k \in Y_{fg} \}.
\]

Lemma 3: \(Z_{fg} \subseteq \{0, a, \tilde{a}\}\). □

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 \(\alpha\) will cause a \(01\rightarrow10\) error pattern if and only if \(Z_{fg} = \{0\}\). It will cause a \(00\rightarrow11\) error pattern if and only if \(Z_{fg} \subseteq \{\alpha, \tilde{\alpha}\}\). □

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 \(\alpha\)-invariant if \(Z_f = \{\alpha\}\).

A FOOF \(f(\alpha, X)\) is said to be \(\alpha\)-variant if \(Z_f = \{\alpha, \tilde{\alpha}\}\).

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 \(01\rightarrow10\) pattern or \(00\rightarrow11\) pattern but not both.

Lemma 4: Any FOOF of the \(+\)-form is \(\alpha\)-invariant. □

Theorem 3: A pair of outputs have a unique double-error pattern if both FOOF’s are \(\alpha\)-invariant (assume error in \(\alpha\) 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 \(\alpha\)-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 \(f=aa^*+b\), then \(f=\frac{df}{d\alpha} \alpha^*+b\). □

Lemma 6: For \(*\in\{+, \oplus\}\), if \(f=aa^*\#b\) then \(f=\frac{df}{d\alpha} \alpha^*\#b\). □

Theorem 4: Any FOOF of the form \(f=aa^*\#b\) can be expressed in one of the following two forms:

1. \(f=\frac{df}{d\alpha} \alpha^*\#b\)

2. \(f=\frac{df}{d\alpha} \alpha^*\#b\). □

We now define an \(\alpha\)-augmented function as a Boolean switching function of the form \(Q(\alpha, X)=w(X)\alpha^*\alpha\) 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 \(\alpha\) 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 \(f=aa^*\#b\) and \(g=aa^*\#d\) are the two associated FOOF’s. The pairwise error function is \(\frac{df}{d\alpha} \frac{dg}{d\alpha}\).

Suppose \(\frac{df}{d\alpha} \neq 0\), then let \(Q=\frac{df}{d\alpha} \frac{dg}{d\alpha} \alpha^*\alpha\) be an \(\alpha\)-augmented function. Under any input \(X_k \in Y_{fg}\), an error in \(\alpha\) will cause a double-error on the two outputs \(f\) and \(g\).
Since \( \frac{dQ}{d\alpha} |_{x=x_0} = 1 \), 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 \( P = f \oplus g \oplus Q \) and is called the augmented parity function.

**Lemma 7:**
\[
P = \left( \frac{df}{d\alpha} + \frac{dg}{d\alpha} \right) a^* b \oplus d.
\]

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 \( w_{PQ} f' = w_{PQ} g' = 0 \) where \( w's \) are the pairwise error functions for \( P \) and \( Q \).

**Theorem 5:**
\[
w_{PQ} f' = w_{PQ} g' = 0.
\]

**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 \( \alpha \) be a PFN of \( C \). The fanout value \( \tau_\alpha \) can be interpreted in two ways. One is the actual number of fanouts of \( \alpha \) in \( C \). The other one is the outdegree of \( \alpha \) 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 \( \alpha \) and \( \tau_\alpha = 2 \). It has two associated outputs \( f_1 \) and \( f_2 \). Assume an error in \( \alpha \) 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 \( \alpha f \) or \( \alpha f \) from \( G \). The resultant graph \( G' \) with \( \alpha f \) removed is shown in Figure 6(b). In \( G' \), \( \alpha \) is no longer a prime node (since \( f_1 = 0 \)) 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 \( \alpha f \) from \( G \) corresponds to a degeneration in the number of fanouts of \( \alpha \) in \( C \). This can be accomplished by constructing a new signal \( \alpha' \) to replace one or more of the fanout signals of \( \alpha \). Here \( \alpha' \) is logically identical to, yet structurally independent of \( \alpha \). In other words, if \( C(\alpha) \) and \( C(\alpha') \) are two sub-circuits whose outputs are \( \alpha \) and \( \alpha' \) 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 \( \alpha \) from \( G \), \( \alpha \) is allowed to stay in \( G \) so long as an error in \( \alpha \) 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 \( Y_1, Y_2 \) 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 \( Y_{jk} \) and \( Y_{jk} \). By removing the set \( \bar{Y}_k \) from this diagram, all double-errors can be eliminated. The removal of \( \bar{Y}_k \) corresponds to the deletion of a directed branch \( \alpha f \) from \( G \). So we conclude that only one signal \( \alpha \) needs to be generated. This signal \( \alpha' \) will be used to implement \( f_0 \). The result is a reduction in \( \tau_\alpha \) from 3 to 2, and the resulting circuit will be free of any multiple errors.

\* f 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 \( a_f \) is obvious. For more complicated problems a general procedure is required in order to select pairs \( (a_f) \) 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

---

**The Fanout-Graphs**

![Fanout-Graphs](image)

---

**The Venn Diagrams**

![Venn Diagrams](image)

---

**The Error-Graphs**

![Error-Graphs](image)

---

**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 \( \hat{a} \), 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 \( a \).

Consider the fanout-graph of Figure 6(a). Instead of removing the branch \( a_j \) from \( G \), we want to add a new branch \( a_j \) 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 \( \hat{a} \). The transformation can be achieved by implementing an \( a \)-augmented function \( \hat{a}(a, X) = w_{ij}(X) \cdot a^u \) where \( w_{ij} \) (the error function) will be our gating function. Since \( \frac{d\hat{a}}{da} = w_{ij} \), an error in \( a \) will also cause \( \hat{a} \) to be in error whenever \( w_{ij} = 1 \) under some input in \( Y_i \).

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 \( \hat{a} = (w_{jk} + w_{kj}) \cdot a^u \), 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 \( B_a = \{a_{f_1}, a_{f_2}, \ldots, a_{f_M}\} \) where \( f_i, 1 \leq i \leq M \), is a terminating node in \( G \), then

\[
\hat{a} = \left( \sum_{i=1}^{M} w_i \right) \cdot \left( \sum_{j=1}^{M} w_j \right) \cdot a^u \quad \text{where} \quad w_i = \frac{df_i}{da}.
\]

**Theorem 7:** Let \( \alpha \) be a PFN in \( G \) and \( B_a = \{\alpha_{f_1}, \alpha_{f_2}, \ldots, \alpha_{f_M}\} \) be the set of all \( M \) directed branches whose starting-node is \( \alpha \), and end-nodes are \( \alpha_i, 1 \leq i \leq M \). Associated with each node \( \alpha_i \) is a set \( N_i \) consisting of all terminating nodes having a path from \( \alpha_i \). We will allow the case where \( N_i = \{\alpha_i\} \) and call \( \alpha_{f_i} \) 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

\[
\hat{\alpha} = \left( \sum_{i=1}^{M} w_i \right) \cdot \left( \sum_{j=1}^{M} w_j \right) \cdot \alpha^u
\]

where

\[
w_i = \sum_{I \in E X} \frac{df_i}{d\alpha}.
\]

**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 \( a_{f_i} \) is removed from \( G \), we need to build a new signal \( a' \) to replace the line(s) being cut in \( C \). In this method we will build the same \( a' \) (or its complement), not for replacement but for comparison. Consider the example of Figure 6 where \( B_a = \{a_{f_1}, a_{f_2}\} \). Suppose \( f_j \) is \( \alpha \)-invariant and we decide to sense \( f_j \). Under input conditions such that \( w_j = 1 \) we will have \( Z_j = \{a\} \) or \( \{a\} \). Since \( Z_j \) contains only a single element, we can associate with \( f_j \) a switching function \( a^u \) and call it the function realized by \( f_j \) under \( w_j = 1 \). Let \( a' = a^u + a \) and it can be used to compare with the “line” \( f_j \) under the “condition” of \( w_j = 1 \). 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 \( f_j \) with \( a^u(X) \). Its output is then gated by the switching function \( w_j(X) \). The final output is an error signal \( \epsilon_i \), which will be set to 1 whenever an error in \( \epsilon_j \) causes an error on \( f_j \) independent of whether or not \( f_j \) is in error. Let \( \epsilon_p \) 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 \( \epsilon_i \) caused by some other source may or may not set \( \epsilon_i = 1 \). That is why \( \epsilon_i \) should also be included in the parity checking.
Mathematically, we have

\[ e_i = w_i(f_i \oplus \alpha^m) \]

A special situation is when \( w_i = 1 \) in which case

\[ e_i = f_i \oplus \alpha^m. \]

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 \( \epsilon_i \). When this is done, \( e_i \) 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 \( (\epsilon_i = 0) \) will now be detected by the line sensing mechanism \( (\epsilon_i = 1) \) and the result is \( e = 1 \). For the graph of Figure 7 a gating function of \( w_{ik} + w_{jk} \) will also be appropriate if \( f_i \) is \( \alpha \)-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 \( \alpha \)-variant. Since \( Z_{\alpha}(\alpha, \beta) \), \( \alpha^m \) 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

\[ f_i = A\alpha + B\beta + C. \]

The error function \( w_i \) is readily found to be

\[ (A \oplus B)C = ABC + A\beta C = W_1 + W_2, \]

where \( W_1 = ABC \), and \( W_2 = A\beta C \). It is seen that when \( W_1 = 1 \), \( f_i \) will be equal to \( \alpha \), and when \( W_1 = 1 \), \( f_i \) will be equal to \( \beta \). So clearly we can write the following equation:

\[ e_i = W_1(f_i \oplus \alpha) + W_2(f_i \oplus \beta). \]

Again, all the previous arguments will still hold for each term in \( e_i \).

For any circuit, if more than one \( e_i \) is generated, then \( e \) should be set as follows:

\[ e = e_o + \sum_i e_i. \quad (4.3) \]

**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→11 pattern, then let \( g_i \) be a function defined by any one of the following expressions:
   - \( f_i \)
   - \( f_i + f_j \)
   - \( f_i f_j \)
   - \( f_i f_j \)
   - \( f_i + f_j \)

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

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

\[ e_i = g_i(X) \oplus g_i'(f_i, f_j) \]

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 \( f_j \), all the outputs are still required to be sent to a parity checker. On the other hand, if \( g_i \) equals \( f_i \) or \( f_j \), 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 \( f_j \). As was mentioned before, if more than one \( e_i \) 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 \( \alpha_i \) 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**


