A PRACTICAL APPROACH TO FAULT DETECTION IN COMBINATIONAL NETWORKS

Adi Tzidon*
Israel Berger**
Michael Yoeli***

Technical Report No. 73
April 1976

* Israel Aircraft Industry, Lod, Israel. This paper is part of the first author's D.Sc. dissertation, Dept. of Computer Science, Technion.

** IBM - Israel Scientific Center, Haifa, Israel.

*** Computer Science Dept. Technion - Israel Institute of Technology, Haifa, Israel.
ABSTRACT

In this paper the advantages of exhaustive testing of combinational networks are investigated. The method consists of applying all possible input combinations and checking only some attributes of the output vector. It is shown that by abandoning the requirement of minimal testing time (practically insignificant for medium-sized networks) a substantial reduction of testing data to be stored is obtained, and the generation process is simplified. It is shown that

a) In tree networks the detection of any multiple stuck-at fault is accomplished by checking the parity of the number of ones in the truth vector (odd - for any faultless tree network; even - whenever at least one stuck-at fault occurred.) Generation stage is superfluous.

b) Any single stuck-at fault in any network realizing an unate function affects the number of ones in the truth vector. Thus the generation stage consists of computing the said number and the detection stage of checking this number by counting.

c) For general functions realized by networks without internal fanout a further function-dependent classification is made and methods of selecting suitable attributes for fault detection are described. At most \(2n\) bits of testing information are necessary to perform single stuck-at faults detection for functions of \(n\) variables.

Basic structure of testing device and efficient algorithm to compute the number of ones for a given expression are also presented.
TABLE OF CONTENTS

1. Introduction................................................................. 1

2. Notation, Definitions and Basic Relations................................. 2

3. Tree Networks................................................................. 6

4. NILE Networks................................................................. 8

5. OIL Networks................................................................. 10

6. OIL Networks - A Simplified Approach..................................... 19

7. Generalization................................................................. 23

8. An Efficient Algorithm to Compute \( w(f) \)................................. 26

References.............................................................................. 28
1. INTRODUCTION

The goal of most existing methods of fault detection is to find a minimal experiment sufficient to cover all faults in a given 'fault list' (see e.g. [1-5]).

If one reexamines the possible advantages of the minimal experiment approach, the following can be argued in its favor:

a) reduction of testing data to be stored
b) reduction of human effort required in manual testing
c) saving of testing time.

However, for medium-sized networks, say up to 25 inputs, argument c) is of no significance, since the testing time is negligible in comparison with the set-up time required. For example, the testing of a 20-input network at the relatively slow rate of 1MHz would require about 1 second.

In this paper, we present a new approach to the testing of combinational networks. We abandon the requirement of minimal testing time and show that in exchange considerable improvements are obtained regarding features a) and b) above, and furthermore text generation is significantly simplified. We adopt the idea of exhaustive testing while storing a very limited amount of information. The amount of information required by our methods is substantially smaller than in the minimal-experiment techniques (e.g. any single or multiple stuck-at fault in any tree network is detected by checking the parity of number of "ones" in the truth vector, as shown in Theorem 3.1).

This new approach is very suitable from a practical viewpoint for medium sized networks but is not feasible for large networks.
2. NOTATION, DEFINITIONS AND BASIC RELATIONS

The following terms are assumed to be known: combinational network, tree network, unate function, reconvergent paths, redundancy, stuck-at errors, experiment (for definitions see [1,2]).

Definition 2.1

Let \( f(x_1, x_2, \ldots, x_n) \) be a Boolean function. If
\[
    f = \sum_{i=0}^{2^n-1} a_i \cdot m_i(x_1, x_2, \ldots, x_n)
\]
where
\( m_i(x_1, x_2, \ldots, x_n) \) is the \( i \)-th minterm
and
\( a_i \in \{0,1\} \)
then
\[
    (a_0, a_1, a_2, \ldots, a_{2^n-1})
\]
is the \textbf{characteristic sequence (CHS)} of \( f \).

Example 2.1.1

The CHS of
\[
    f(x_1, x_2, x_3) = x_1x_2 + x_3
\]
is \((0,1,0,1,0,1,1,1)\).

Definition 2.2

An \textbf{attribute} \( A(f) \) of a Boolean function \( f(x_1, \ldots, x_n) \) is any property
which partitions the set of all Boolean functions of \( n \) variables into at least two non-empty subsets.

Let \( N \) be a network realizing \( f \). We shall denote by \( A(N) \) the value of the attribute \( A \) of \( f \); i.e. \( A(N) = A(f) \).

Some examples of attributes:

**Example 2.2.1**

Let

\[
g(x_1, x_2, x_3) = x_1 x_2 + x_3 \quad (2.2.1)
\]

and

\[
h(x_1, x_2, x_3, x_4) = x_1 x_2 + x_3 \quad (2.2.2)
\]

(a) \( w(f) \) is the number of 'ones' in the CHS of \( f \).

\[\text{e.g. } w(g) = 5\]

\[\text{whereas } w(h) = 10\]

(b) \( p(f) \triangleq w(f) \mod 2 \)

\[\text{e.g. } p(g) = 1\]

\[p(h) = 0\]

(c) \( a_i(f) \) is the value of the \( i \)-th component in the CHS of \( f \).

\[\text{e.g. } a_4(g) = 0\]

\[a_4(h) = 0\]
Let $N$ be a faultless realization of a function $f$. Let $N_j$ be a network resulting from the occurrence of fault $\phi_j$ in $N$.

**Definition 2.3**

An attribute $A$ is said to cover fault $\phi_j$ iff

$$A(N) \neq A(N_j).$$

**Example 2.3.1**

The attribute $p$ as defined in Ex. 2.2.1 (b) covers any single or multiple stuck-at error in a tree realization of $g = x_1x_2 + x_3$.

**Definition 2.4**

Let $N$ be a combinational network, $F$ an arbitrary set of faults in $N$, and $S$ an ordered set of attributes. $S$ is called a signature of $(N,F)$ iff any fault $\phi \in F$ is covered by at least one attribute in $S$.

**Definition 2.5**

The weight $w(f)$ of a Boolean function $f$ is the number of 'ones' in the CHS of $f$, i.e. the cardinality of $f^{-1}(1)$.

We shall show in the sequel that the weight is a one-element signature for a wide range of networks and faults.
Consider two Boolean functions \( f \) and \( g \) of disjoint sets of variables:

\[
\begin{align*}
    f &= f(x_1, x_2, \ldots, x_n) \\
    g &= g(z_1, z_2, \ldots, z_k)
\end{align*}
\]

One easily verifies:

\[
\begin{align*}
    w(f \cdot g) &= w(f) \cdot w(g) \quad (2.5.1) \\
    w(f + g) &= 2^k \cdot w(f) + 2^h \cdot w(g) - w(f) \cdot w(g). \quad (2.5.2)
\end{align*}
\]

In the following sections we present simple signatures for some basic types of networks. The fault detection will be actually performed by evaluating the signature of the UUT and comparing the result with the known signature of the corresponding faultless network.
3. TREE NETWORKS

Let \( T \) be an arbitrary fanout-free network (tree network) composed of NAND, NOR, AND, OR and NOT gates only.

Let \( F_M \) be the complete set of multiple stuck-at errors of \( T \).

Theorem 3.1

(a) \( p(T) = 1 \) \hspace{1cm} (3.1.1)

(b) \((p)\) is a signature for \((T, F_M)\).

Proof:

(a) We shall show by induction on the number of variables that any tree-realizable function \( f \) has odd weight \( w(f) \). This is true for all non-trivial functions of one variable.

We can represent the fanout-free realizable function \( f(x_1, \ldots, x_n) \), \( n > 1 \), as a sum or product of two variable-disjoint fanout-free realizable functions \( g \) of \( n_g \) variables, and \( h \) of \( n_h \) variables, where \( 1 \leq n_g < n \) and \( 1 \leq n_h < n \).

Assume now that any fanout-free realizable function of less than \( n \) variables has odd weight.

Thus \( w(g) \) and \( w(h) \) are both odd. If \( f = g \circ h \), then

\[
w(f) = w(g)2^{n_h} + w(h)2^{n_g} - w(g)w(h) \hspace{1cm} (3.1.2)
\]

and \( w(f) \) is odd, consisting of one odd and two even terms.
The same conclusion holds for

\[ f = g \cdot h, \text{ since} \]

\[ w(f) = w(g \cdot h) = w(g) \cdot w(h), \]

which is odd.

Thus, (a) is proven.

(b) We show that any single or multiple stuck-at error in a tree network results in an even weight of the faulty network.

Assume the given tree network \( T \) realizes \( f(x_1, \ldots, x_n) \) and the faulty network realizes the function \( g(x_1, \ldots, x_n) \). Any stuck-at error (single or mult.) causes \( g \) to be vacuous in some non-empty subset of input variables. Hence \( w(g) \) is even.

Q.E.D.

This Theorem provides a very practical tool for fault-detection of fanout-free networks, since no test generation is required, and the testing device consists of one \( n \)-bit counter, one flip-flop and a clock as shown in Fig. 3.1.

![Diagram of testing device](Fig. 3.1)
4. NILE NETWORKS

A NILE Network is a network, in which the number of inverters along any loop is even. This definition is equivalent to the commonly used term 'equal inversions parity along all two reconvergent paths'. Clearly, only unate functions can be realized by NILE networks.

Let N be a given NILE network in an irredundant realization using the previously specified types of gates, and FS the set of all its single stuck-at errors.

Theorem 4.1

(w) is a signature for (N,FS).

Proof: Any stuck-at error will cause a change in the number of ones at the output, since the realization is irredundant and all the paths from the faulty line to the output have equal parity of number on inversions (PNI).

Q.E.D.

Example 4.1 The network Na shown in Fig. 4.1 is a NILE network, thus (w) is a signature for (Na,FS).

![Diagram of Na network]
Example 4.2 The network $N_b$ of Fig. 4.2 realizes the non-unate function $f = x_1 \oplus x_2$. $N$ is not a NILE and $(w)$ is not a signature, since $w(f)$ does not vary if one of the input lines is stuck-at.

![Fig. 4.2](image)

Example 4.3 Fig. 4.3 shows a network $N_c$ realizing $f = x_1 x_2 + x_1 x_3 + x_2 x_3$. The network is a NILE and $(w)$ is a signature for $(N_c, F_c)$.

![Fig. 4.3](image)
5. OIL NETWORKS

Let us consider multilevel irredundant networks, where Only Input Lines may fanout with unequal PNs. Networks of this type will be called OIL networks (e.g. network \( N_b \) of Fig. 4.2).

To clarify our concept of a single stuck-at fault of a line with fanout, we indicate in Fig. 5.1 (a) the possible locations of such faults by \( x \).

![Diagram](attachment:diagram.png)

Fig. 5.1

It follows that even if the actual circuit looks as in Fig. 5.1(b), only faults at locations indicated by \( x \) are taken into account. In view of printed circuit technologies, this assumption is justified from a practical point of view.

Actually, the need for this limitation appears only at junctions wherefrom
paths with non-equal PNI's emerge. Since such junctions do not exist in any of the previously discussed types of networks there was no need to introduce this constraint earlier.

Obtaining a convenient signature for OIL networks is of practical importance since any Boolean function can be easily, and usually is, implemented by means of an OIL network.

Let $N$ be an OIL realization of $f$. Whenever a faulty network will be considered we shall denote it by $N'$.

**Definition 5.1**

Let $f(x_1, \ldots, x_n)$ be a Boolean function. We define

$$w_i(f) = w(f \cdot x_i) \quad \text{for } i = 1, \ldots, n.$$  

For a given Boolean function $f(x_1, \ldots, x_n)$ we set:

$$Q(f) \triangleq \{i \in \{1, \ldots, n\} | w_i(f) = \frac{1}{2} w(f)\}$$

The set $Q$ defines a partitioning of the set of all lines in $N$;

- $L_q$ - the set of input lines $x_i$, $i \in Q$
- $\overline{L}_q$ - the set of all other (including internal) lines.

We denote by $F(L)$ and $F(\overline{L})$ the complete sets of single stuck-at faults on lines belonging to $L_q$ and $\overline{L}_q$ respectively.
Theorem 5.1

\( (w) \) is a signature for \((N,F(L))\).

Proof:

From Theorem 4.1 it follows that all stuck-at faults on internal lines are covered by \((w)\). If a primary line \(x_i\) is stuck, the corresponding value \(w(N')\) becomes either \(2w_i(f)\) or \(2(w(f)-w_i(f))\). Neither value equals to \(w(N)\) iff \(i \notin Q\). Thus, \((w)\) covers all stuck-at faults, excluding those in \(F(L)\).

Q.E.D.

Corollary 5.1

If \(Q(f) = \emptyset\) then \((w)\) is a signature for \((N,F_S)\).

Theorem 5.2

If \(w(N)\) is odd then \((w)\) is a signature for \((N,F_S)\).

Proof:

If \(w(N)\) is odd then for every \(i\) \(w_i(N) \neq \frac{1}{2}w(N)\). Thus \(Q = \emptyset\) and \(F_S = F(L)\).

Q.E.D.

If \(w\) is odd the signature \((w)\) also covers all multiple stuck-at faults on primary input lines since any combination of those causes an even \(w(N')\).
Definition 5.2

For $i = 1, \ldots, n$ we define $e_i(f)$ as follows:

$$e_i(f) = \begin{cases} 
1 & \text{if } f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \neq f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) \\
0 & \text{for some combination } (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \\
0 & \text{otherwise}
\end{cases}$$

Theorem 5.3

If $w_i(N)$ is odd then $(w, p(w_i), e_i)$ is a signature for $(N, F)$. 

Proof:

The faults in $F(L)$ are covered by $(w)$. Now

$$w_i(f) = w(x_i \cdot x_j \cdot f) + w(x_i \cdot \bar{x}_j \cdot f).$$

Since $w_i(f)$ is odd, we have $w(x_i \cdot x_j \cdot f) \neq w(x_i \cdot \bar{x}_j \cdot f)$, But any stuck-at fault on any input line $j, j \neq i$ will cause $w_i(N')$ to become even, thus $p(w_i(f))$ detects faults on all primary lines, $x_i$ not necessarily included. Stuck-at faults on line $x_i$ are covered by $e_i$.

Q.E.D.

Theorem 5.4

If $w_i(N)$ is odd and $i \notin Q$ then $(w, p(w_i))$ is a signature for $(N, F)$. 

Proof: Since $i \notin Q$, $w$ covers also the faults on line $x_i$, thus making $e_i$ of Theorem 5.3 superfluous.

Q.E.D.
NOTES. 1) If \(|Q| = 1\), say \(Q = \{j\}\), the ordered set \((w, e_j)\) is a signature, the value of \(p(w_j)\) being immaterial. For instance any 0IL network realizing

\[ f = x_2x_3x_4 + x_1x_3x_4 + x_1x_2x_4 + x_1x_2x_3 + x_1x_3x_4 \]

is covered by \((w, e_2)\) since \(Q(f) = \{2\}\).

2) Another signature for \((N, F_S)\) is \((w, p(w_i), p(w_j))\) where \(i, j \in Q\), \(i \neq j\), and \(w_i\) is odd. Sometimes the task of testing these attributes may be more convenient since the logic for evaluating \(e_i\) is not required. However, it should be added that the evaluation of \(e_i\) poses no difficulty for the case \(i = n\), since only adjacent components of the CHS have to be compared.

For the case \(i \neq n\) the facility to interchange \(x_i\) and \(x_n\) should be provided by the testing device.

3) Under the conditions of Theorem 5.4, both \(p(w_i)\) and \(e_i\) equal to 1 for faultless networks, thus the information needed to perform the testing consists of values of \(w\) and \(i\) only.

4) Clearly, when \(w = 2 \mod 4\), then \(i \in Q\) implies that \(w_i(N)\) is odd. Thus \((w, p(w_i), e_i)\) is a signature for \((N, F_S)\).

Observe that any function not covered by signatures described so far must
meet the following conditions:

\[ a) \quad w(f) = 0 \mod 4 \]  
\[ b) \quad \forall i: w_i(f) = 0 \mod 2 \]  
\[ c) \quad |Q(f)| > 1 \]  

We shall present now a general signature to cover faults in \( F_S \) for any 0IL network realization of any Boolean function with no constraints on \( w \) and \( w_i \).

Theorem 5.5

Let \( Q = \{i_1, i_2, \ldots, i_k\} \).

\((w, e_{i_1}, e_{i_2}, \ldots, e_{i_k})\) is a signature for \((N, F_S)\).

Proof: The only faults not covered by \( (w) \) are stuck-at faults in \( F(L) \).
But these are covered by the ordered set \((e_{i_1}, \ldots, e_{i_k})\).

Q.E.D.

Notes:
1) Corollary 5.1 presents a special case of Theorem 5.5.
2) A testing device based on \((w, e_1, \ldots, e_n)\) as signature is easily hardware-implemented, provided the \( e_i \)'s are checked serially one at a time. The testing time of such a device will be proportional to \( n \cdot 2^n \). Thus 0IL networks with up to 20 inputs may be tested by this method for all single stuck-at faults in less time than networks with 25 inputs tested within a single "run" (application of all possible input combinations).
Theorem 5.6

\((w, w_1, w_2, \ldots, w_s)\) is a signature for \((N, F_s)\) if for every \(j, \quad j \in Q,\) there exists an \(i, i \in \{i_1, i_2, \ldots, i_s\},\) such that \(w_i(x_j \cdot f) \neq w_i(\bar{x}_j \cdot f).\)

Proof: \(w\) covers all faults in \(F(L).\) Stuck-at faults on primary line \(x_j\) are detected by a change of value of the corresponding \(w_i(f)\).

Q.E.D.

The above theorem implies that the set of functions not covered by "short" signatures does not necessarily require the signature \((w, e_1, e_2, \ldots, e_k),\)

\(k\) being the cardinality of \(Q.\

Allowing some extra computation for decomposing each \(w_i\) into \(w_i(x_j \cdot f)\) and \(w_i(\bar{x}_j \cdot f)\) and combining the results of Theorems 5.5 and 5.6 we can obtain a shorter (less than \(k+1\) attributes) signature and smaller number of serial "runs", thus reducing the duration of the testing phase.

Let \(N_M\) be a multiple-output OIL network, realizing the Boolean functions \(f_1, f_2, \ldots, f_k.\) Denote the corresponding weights by \(w(f_1), w(f_2), \ldots, w(f_k).\)

Let \(\{F^i(L), F^i(\bar{L})\}\) be the partition of all single stuck-at faults on all lines feeding the output \(f_i.\)

Theorem 5.7

The ordered set \((w(f_1), w(f_2), \ldots, w(f_k))\) is a signature for \((N_M, \cup_{i=1}^{k} F^i(L)).\)

Proof: The result follows directly from Theorem 5.1 Q.E.D.
Theorem 5.7 provides a practical and easy tool to detect most single stuck-at faults in multiple-output OIL networks. The undetected faults may occur only on a part of the input lines.

In order to design fully diagnosable (for all single stuck-at errors in $F_s$) multiple-output networks realizing arbitrary Boolean functions, the following design model is proposed.

Let $N$ be an OIL network realizing $f_1, f_2, \ldots, f_k$ of the variables $x_1, \ldots, x_n$. For every set of lines $L_i$ feeding $f_i$, we define $L_q^i, L_q^i$ as above.

Let $L_Q = \bigcap_i L_q^i$.

![Diagram](https://via.placeholder.com/150)

Fig. 5.2
Now consider the extended network $N^*$ as shown in Fig. 5.2, where $T_o$ is an arbitrary tree network having as inputs the lines in $L_Q$, taken after the corresponding 'divergence' points.

Let $F^*_S$ be the set of all single stuck-at errors in $N^*$.

We denote by $f_{k+1}(x_1,\ldots,x_n)$ the output of $T_o$.

**Theorem 5.8**

The ordered set $(w(f_1),w(f_2),\ldots,w(f_k),w(f_{k+1}))$ is a signature for $(N^*,F^*_S)$.

**Proof:** Since in $N^*$ $n\bigcap L_i = \emptyset$ it follows that $\bigcup_{i=1}^{k+1} F_i(L) = F^*_S$, hence our theorem results from Theorem 5.7.

Q.E.D.

Consequently, any $k$-output OIL network of $n$ input variables can be made fully diagnosable for single stuck-at errors by addition of a tree network (essentially a single gate) of less than $n$ variables. A signature for such network will consist of $k+1$ attributes.

Actually the attribute $w(f_{k+1})$ in Theorem 5.8 can be replaced by the evenness of

$$w(f_{k+1}) \mod 2^{n-|L_Q|}.$$ 

Note that by Theorem 3.1 any single or multiple fault of the additional tree network is covered by the above signature.
6. OIL NETWORKS - A SIMPLIFIED APPROACH

A possible objection can be made as to the practical value of the above results requiring the afore determination of the set \( Q \) for a given function \( f \).

Although the amount of calculation needed to evaluate all \( w_i \)'s is not significant provided a computer is used (see algorithm in Section 8), we propose in this section an alternative method of testing based on "weakening" all previous theorems, and thus omitting the need to know \( Q \).

Instead of partitioning the lines of the network into \( L_q \) and \( \bar{L} \) we now suggest the following partition:

\[ L_o : \text{the set of input lines fanning out into at least two reconverging paths with unequal PNI's} \]
\[ L_e : \text{the set of all remaining lines (obviously, all internal lines are in } L_e \). \]

To those two sets correspond two disjoint sets of stuck-at errors \( F(L_o) \) and \( F(L_e) \), respectively:

\[ F(L_o) : \text{the complete set of single stuck-at faults on lines in } L_o \]
\[ L(L_e) : F_S - F(L_o). \]

We now define \( Q' \triangleq \{ i \in \{ 1, \ldots, n \} \mid x_i \in L_o \} \)
Theorem 6.1

For any OIL network

\[ Q \subseteq Q' \]

and

\[ F(L) \subseteq F(L_O). \]

**Proof:** If a stuck-at fault on the input line \( x_i \) does not affect the value of \( w(f) \), clearly \( f \) is not unate in respect to the variable \( x_i \), thus \( x_i \) must fanout into at least two reconverging paths with different PNI's.

Thus, \( i \in Q \Rightarrow (\text{line } x_i) \in L_O \) and the theorem follows.

Q.E.D.

One easily modifies Theorems 5.1, 5.3, 5.4, 5.6, 5.7, and 5.8 by substituting \( Q' \) for \( Q \), \( F(L_e) \) for \( F(L) \), and \( FL_O \) for \( F(L) \). We obtain "weaker" theorems but we now need less, if any, computation during the generation phase.

Thus we replace a tree \( T_Q \) of Fig. 5.2 by a tree with some superfluous but easily selectable inputs.
The following table summarizes different signatures for some types of
networks of \( n \) inputs for the set of single stuck-at faults:

<table>
<thead>
<tr>
<th>Type of network</th>
<th>Signature</th>
<th># of information bits in a signature</th>
<th># of &quot;runs&quot;</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Tree</td>
<td>((p(w)))</td>
<td>1</td>
<td>1</td>
<td>- fixed value signature - faults in ( F_M ) also covered</td>
</tr>
<tr>
<td>2 NIL</td>
<td>((w))</td>
<td>(n)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3 OIL w odd</td>
<td>((w))</td>
<td>(n)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4 OIL w even but some ( w_1 ) odd</td>
<td>((w,e_1,p(w_1)))</td>
<td>(n + \log_2 n)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5 OIL general case</td>
<td>((w,e_1,e_2...e_k))</td>
<td>(2n)</td>
<td>(k)</td>
<td>- One may use ( Q' ) instead computing ( Q ). - Testing time proportional to (</td>
</tr>
<tr>
<td>6 OIL general case with tree added</td>
<td>((w,w_T))</td>
<td>(2n)</td>
<td>1</td>
<td>(w_T \triangleq ) the weight of the additional tree.</td>
</tr>
</tbody>
</table>
By comparing the entries (5) and (6) one can see that the testing of a general OIL network may be accomplished in one of the two basic ways, offering to the user a choice of trade-offs:

(5) - keeps testing simple but prolongs its duration
(6) - adds circuitry to the UUT, the tester is very simple and testing time not affected.

The basic structure of a tester based on our approach is presented in Fig. 6.1.

![BLOCK DIAGRAM OF A TYPICAL TESTER](Fig. 6.1)
7. GENERALIZATION

The use of the attributes \( w, w_i, e_i \) has proven successfully in providing single signatures for a wide range of Boolean functions in OIL realizations.

However, one should bear in mind that \( w_i \) is only one of many ways to define attributes of interest. A more general set of attributes is the following:

\[
g_i = w(f \cdot G_i(x_1, x_2, \ldots, x_n))
\]

where \( G_i \) is an arbitrary Boolean function of \((x_1, x_2, \ldots, x_n)\).

Example 7.1 \((w, g_1)\) is a signature for \((N, F_S)\), where \( N \) is any realization of

\[
f = x_1 x_2 x_3 x_4 + x_1 x_2 x_3 x_4 + x_1 x_2 x_3 x_4 + x_1 x_2 x_3 x_4
\]

of

\[
G_1 = x_1 \oplus x_2 \oplus x_3 \oplus x_4.
\]

Clearly, \( w(N) \triangleq 4 \) and \( g_1(N) = 0 \) while an occurrence of any single stuck-at fault in \( F_S \) alters at least one of these values.

Observe that within the constraint of using \( w \) and \( w_i \) the signature for \((N, F_S)\) would consist of \((w, w_1, w_2, w_3, w_4)\), which is more complicated than \((w, g_1)\).

We now consider some properties of such generalized signatures for an OIL network \( N \), realizing the Boolean function \( f(x_1, x_2, \ldots, x_n) \).
Definition 7.1

Let \( G(x_1, x_2, \ldots, x_n) \) be a Boolean function vacuous in variables whose indices belong to the set \( \gamma \), i.e.:

\[
\gamma = \{ m \mid \text{G is vacuous in } x_m \} \quad \text{where } m \in \{1, 2, \ldots, n\}
\]

With \( \gamma \) we can associate the set of lines \( L_\gamma \) in \( N \) and the corresponding set of faults \( F_\gamma \), where \( F_\gamma \) is the complete set of single stuck-at faults on input lines in \( L_\gamma \).

Let \( g(f) \triangleq w(G \cdot f) \).

Theorem 7.1

If \( g(f) \) is odd then \((p(g))\) is a signature for \((N, F_\gamma)\).

Proof:

\[
g(f) = w(x_j \cdot f \cdot G) + w(x_j \cdot f \cdot \neg G)
\]

\[
\triangleq w' + w''.
\]

Clearly \( w' \neq w'' \) since their sum is odd.

But for a faulty network \( N' \) we have:

\[
g(N') = 2w'
\]

or

\[
g(N') = 2w''
\]

i.e. the sum \( g(f) \) is even in both cases.

Q.E.D.
Let $G_1, G_2, \ldots, G_k$ ($k \leq n$) be Boolean functions and denote by $g_i$ and $\gamma_i$ the corresponding concepts as defined in Def. 7.1.

**Theorem 7.2**

If $g_i(f)$ is odd for $i = 1, \ldots, k$ and $\bigvee_{i=1}^k \gamma_i \supseteq \emptyset$ then

$$(w, p(g_1), p(g_2), \ldots, p(g_k))$$

is a signature for $(N, F_S)$.

**Proof:** All faults in $\bar{L}_q$ are covered by $w$, and each $p(g_i)$ covers all faults in $\gamma_i$, by Theorem 7.1. 

Q.E.D.

The merits of certain function $G_i$ depend on the simplicity of the corresponding circuitry in the testing device and in obtaining short signature for $(N, F_S)$ or $(N, F_M)$.

Although we do not recommend extension of the universal testing device toward generalized attributes, their use can simplify testing whenever special purpose built-in-testing facilities are incorporated in the unit under test.
8. AN EFFICIENT ALGORITHM TO COMPUTE $w(f)$

Most of the previous testing procedures make use of $w(f)$ as one of the attributes. In this section we describe an algorithm by which $w(f)$ is easily computed.

Let $f(x_1, \ldots, x_n)$ be given as sum of products, $f = p_1 + \ldots + p_m$.

**Algorithm**

1) Set $x_1 = p_1$

2) Let $s_i = \sum_j q_j - \sum_k r_k$ for $i = 1, \ldots, m-1$

where the $q_j$ and $r_k$ are products of literals;

Set $s_{i+1} = \sum_j q_j + p_{i+1} + \sum_k p_{i+1} \cdot r_k - (\sum_k r_k + \sum_j p_{i+1} \cdot q_j)$. (8.1.1)

Simplify the expression for $s_{i+1}$ by using the rule $x_i \cdot \bar{x}_i = 0$, but do not apply the absorption law.

3) In $s_m$ substitute each product of $t$ literals by $2^{n-t}$, keeping all signs unchanged. The value of the resulting expression yields $w(f)$.

Note that the above algorithm is based on the following laws of Boolean algebra:

Let $x - y \equiv x \cdot \bar{y}$ \hspace{1cm} (8.1.2)

Then $(x-y) \cdot z = x \cdot z - y \cdot z$ \hspace{1cm} (8.1.3)

and $x \geq z \Rightarrow x - (y-z) = x - y + z$. \hspace{1cm} (8.1.4)
Example 8.1

\[ f(x_1, x_2, x_3, x_4) = x_1x_2 + x_1\overline{x_3}x_4 + x_2x_3x_4 + \overline{x_1}\overline{x_2}x_3x_4 \]  

(8.1.5)

\[ S_1 = x_1x_2 \]

\[ S_2 = x_1x_2 + x_1\overline{x_3}x_4 - x_1\overline{x_2}x_3x_4 \]

\[ S_3 = x_1x_2 + x_1\overline{x_3}x_4 + x_2x_3x_4 - (x_1x_2x_3x_4 + x_1x_2x_3\overline{x_4}) \]

\[ S_4 = x_1x_2 + x_1\overline{x_3}x_4 + x_2x_3x_4 + \overline{x_1}\overline{x_2}x_3\overline{x_4} - (x_1x_2x_3x_4 + x_1x_2x_3\overline{x_4}) \]

\[ w(f) = 2^2 + 2^1 + 2^1 + 2^0 - 2^0 - 2^0 = 7 \]  

(8.1.6)

CONCLUSIONS

We have presented very simple and practical methods for detecting stuck-at faults in a large class of combinational networks of up to about 25 inputs. This class is sufficient to realize any Boolean function and practically most of the existing implementations are included. Especially all two-level realizations such as PLA-designs belong to this class. The only networks excluded are those having internally emerging reconverging paths with unequal parity of number of inversions.

All our testing methods are easily implemented by means of simple hardware devices, consisting of a few counters and some additional logic.
REFERENCES


ISRAEL SCIENTIFIC CENTER TECHNICAL REPORTS

001 Fault Detection in Fanout-Free Combinational Networks
   Israel Berger, Zvi Kohavi
   January 1973

002 Sequential Diagnostic Tree Generation
   Michael Sperling, Bezalel Gavish
   November 1973

003 Simulation of a Cardiovascular System on a Digital Computer
   I. Berger, A. Hauber, Y. Kishon
   December 1973

004 Queue Regulation Policies Using Full Information
   Bezalel Gavish, Paul Schweitzer
   December 1973

005 When You Talk to Your Computer in FORTRAN Does It Answer You in Hex?
   or A System/360 FORTRAN Debugging Package
   Harry J. Saal
   December 1973

006 Inclusion Relations between Power Methods and Matrix Methods of Limitation
   Abraham Ziv
   December 1973

007 The Effect of Arterial Pressure and Ventricular Volume on the Response of the Cardiovascular System
   I. Berger, M. Itzkovitz, Y. Kishon
   December 1973

008 On the Optimality of the Exponential Function for Some Minimax Problems
   Shmuel Gal, Dan Chazan
   January 1974

009 On the Determination of Addresses in Variable Length Addressing
   Gideon Frieder, Harry J. Saal
   January 1974

010 Optimal Test Design for Reliability Demonstration
   Shmuel Gal
   January 1974

011 Estimation of Cardiac Output from Respiratory Data
   Joseph Etsyon, Dan Chazan, Meira Itzkovitz, Simon Bursztein
   January 1974

012 Algorithms for Binary Coded Balanced and Ordinary Ternary Operations
   Gideon Frieder, Clement Luk
   February 1974

013 A Generalization of the Non-Equivalence Theorem of Zeller
   Abraham Ziv
   February 1974

014 Computer Assisted Diagnosis with an Application to Endocrinology
   W. Schid, B. Gavish, B. Lunenfeld
   April 1974

015 On the Numerical Treatment of Stiff, Highly-Oscillatory Systems
   Vardy Amdursky, Abraham Ziv
   April 1974

016 A Method for Obtaining All the Solutions of a Perfect Matching Problem
   Shmuel Gal, Yuri Breitbart
   May 1974

017 Fault Diagnosis in Combinational Tree Networks
   Israel Berger, Zvi Kohavi
   June 1974

018 Inclusion Relations between Power Methods of Limitation
   Abraham Ziv
   August 1974

019 A Model for Selecting an Optimal Set of Medical Facts for a Questionnaire
   M. Ben-Bassat, B. Gavish
   August 1974
020 A Method for Standardizing School Grades
Shmuel Gal, Alex Hauber
December 1974

021 Four Programming Aids for APLSV
Gideon Frieder, Harry J. Saal, Willy Schild, Zvi Weiss
December 1974

022 A Markovian Model for a Central Blood Bank
Dan Chazan, Shmuel Gal
January 1975

023 Coding for Data Compaction
Jacques H. Mommens, Josef Raviv
February 1975

024 Some Properties of APL Programs
Harry J. Saal, Zvi Weiss
March 1975

025 Memoryless Execution: A Programmer's Viewpoint
Israel Gat, Harry J. Saal
March 1975

026 New Bounds for the Probability of Error in Classification Problems
Moshe Ben-Bassat, Josef Raviv
April 1975

027 The Zero Pivots Phenomena in Transportation and Assignment Problems and Its Computational Implications
Bezalel Gavish, Paul Schweitzer, Eliezer Shlifer
June 1975

028 Analysis of Algorithms for the Evaluation of Monotonic Boolean Functions
Yuri Breitbart, Shmuel Gal
July 1975

029 The M/M/1 Queue with Bounded Virtual Waiting Time
Paul Schweitzer, Bezalel Gavish
August 1975

030 Sequential and Parallel Search for Finding a Root
Shmuel Gal, Willard Miranker
August 1975

031 A New Information Measure for Feature Selection
Moshe Ben-Bassat
September 1975

032 On the Numerical Solution of Stiff Linear Systems of the Oscillatory Type
Abraham Ziv, Vardy Amdursky
September 1975

033 Myopic Policies in Sequential Classification
Moshe Ben-Bassat
October 1975

034 Properties and Convergence of A Posteriori Probabilities in Classification Problems
Moshe Ben-Bassat, Shmuel Gal
November 1975

035 A Study in Computer-Assisted Interpretation of Laboratory Tests
D. Motzkin, Z. Kraiem, B. Gavish
January 1976

036 A Relaxation Algorithm for Building Undominated Portfolios
Bezalel Gavish
January 1976

037 Computer Communication Via Satellites - A Queueing Model
B. Gavish, A. Konheim
January 1976

038 A System for Computer Aided Diagnosis
W. Schild, B. Lunenfeld, B. Gavish
February 1976

039 The Numerical Treatment of Linear Highly-Oscillatory O.D.E. Systems by Reduction to Non-Oscillatory Type
Vardy Amdursky, Abraham Ziv
Appendix by Boris Bachells
March 1976