DETECTING BRIDGING AND STUCK-AT FAULTS AT INPUT AND OUTPUT PINS OF STANDARD DIGITAL COMPONENTS

Mark Karpowsky and Stephen Y. H. Su
Research Group on Design Automation
and Fault-tolerant Computing
School of Advanced Technology
State University of New York, Binghamton, NY 13901

ABSTRACT

Due to the advances in the integrated circuit technology, there is an increasing importance in testing bridging (short circuit) failures in digital networks. Unfortunately, very little work has been done in this area. This paper presents the schemes for the detection of feedback bridging between the inputs and outputs through the observation of oscillation and asynchronous behavior of sequential networks created by bridging faults. The lower and upper bounds on the number of tests for detecting all feedback bridging faults are given. Conditions for the undetectability of input bridging are given and a method for testing input bridging is presented. The results are generalised to detect bridging and stuck-at faults in the input and output lines of a multiple-output network. Finally, the complete test sets are given for detecting input, output and feedback bridging as well as stuck-at faults at the input and output pins of the standard integrated circuit chips including shift registers, counters, decoders, multiplexers, adders/subtractors, multipliers, dividers and RAMs. Future unsolved problems in this area are also given.

1. INTRODUCTION

The testing of digital systems has become increasingly important in recent years. Unfortunately, almost all published research papers deal only with the stuck-at faults. A bridging (short circuit) fault is a fault in which two or more leads in the circuit are shorted (wired) together. Technology has moved into VLSI (very large scale integration) where hundreds of thousands of digital components are being fabricated into a tiny IC (integrated circuit) chip. For example, the whole CPU (central processing unit) of a computer was fabricated into a single chip called a microprocessor. Now microprocessor chips of greater complexity are being built. The trend for fabricating more and more components into an IC chip is continuing and this increases the chance of short circuit failures between components and interconnecting wires. Bridging faults may arise in various levels.

This work is supported in part by the Division Mathematical and Computer Science of National Science Foundation under grant No. NSC78-24325.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

1. Inside the chip, they may arise due to either the failure of insulation between adjacent layers of metallization or the bridging of two conductors in the same layer as a result of improper masking or etching.

2. Integrated circuit packaging provides more bridging faults due to the manual labor involved in bonding between the tiny solder pads on the chip and the pins of package. Two neighboring wires may come into contact when one shakes loose between pads.

3. At the circuit board level, shorts may be caused by human error, defective printed circuit traces and feedthroughs, loose, or excess bare wires or solder bridges.

4. Interboard communication is normally done by cabling, backplane wire-wrapping, or printed circuit motherboards. Solder bridges, excess bare wires, defective traces, etc., remain to be the major causes of bridging faults and so do the easily bent wire-wrap pins. Also, the insulation of cable wires may crack in severe conditions and result in bridging faults.

Compared to the stuck-at faults, bridging faults pose more difficulties. These are given below.

1. Bridging faults may change a non-redundant network to a redundant one, which may in turn invalidate a stuck-at fault detection test set. Thus, therefore, poses a practical problem.

2. Bridging faults may introduce a feedback loop in a combinational logic circuit, causing the circuit to oscillate or to behave as an asynchronous sequential circuit. This violates the usual assumption that a fault does not change a combinational circuit into a sequential circuit and complicates the analysis of the fault behavior.

3. In the case of stuck-at faults, if there are n lines in the circuit, then there are 2n possible single stuck-at faults, and 3n-1 possible multiple stuck-at faults. Whereas in the case of bridging faults, if we have to consider 5 digit faults between any 4 lines in the circuit, then the number of single bridging faults alone will be 5 and the number of multiple bridging faults will be very much larger [5].

In the very recent paper by these authors [19], conditions for feedback bridging (short circuit) faults to generate oscillation and asynchronous behavior are given for short circuits among input lines and the primary output. The lower and upper bounds on the number of tests for detecting all feedback bridging faults are given. Necessary and sufficient conditions for the undetectability of input bridging are presented. It is found that any test can detect a single bridging fault if it will also detect all multiple bridging faults containing it. Complete test sets for locating either all input or all feedback bridging of any multiplicities for networks implementing several classes of functions are given.

In this paper, we shall concentrate on the testing of bridging and stuck-at faults among the input and
bridging faults, we believe that algorithms which use smaller memory space and shorter computing time can be developed and should be investigated.

Cheng [3] proposed a fault model for single stuck-
output or stuck-at faults. A two-pass technique may be used for the detection of such faults. The first pass detects bridging faults and the second pass detects stuck-at faults. The algorithm presented by Cheng is based on the observation that a bridging fault can be detected by applying a test pattern to the input and output of the stuck-at fault. The test pattern is a sequence of high and low levels that are applied to the input. The output of the stuck-at fault is then compared to the expected output. If the output is different, then the stuck-at fault is detected. If the output is the same, then the bridging fault is detected.

Friedman [4] has shown that in a fanout-free network, a single stuck-at fault test set will also detect all bridging faults. Furthermore, it has been established that there exists a test for some stuck-at faults which can also detect an AND (OR) type bridging fault at the two input leads of an OR (AND) gate. This test was provided that at least one of the inputs does not have any fanout. However, it is still not known whether a bridging fault between two input leads with fanouts is detectable by a single fault detection test set or even detectable at all. Also, the possibility exists for such an undetectable fault to invalidate a single fault detection test set. Friedman [4] has modified single fault detection test sets for detecting all detectable bridging faults at the inputs of a gate.

Mei [5] considered feedback bridging faults in fanout-free combinational networks using NAND gates. He showed that a single stuck-at fault detection test set is also capable of detecting a feedback bridging fault if there is an odd number of inverters in the path between the two points that are bridged. If there is an even number of inverters between the two bridging points, then the detection is relatively more difficult. These faults are referred to as non-inverting feedback bridging faults (NFBF). He showed that the detection of these faults would require a certain amount of ordering of the test vectors.

Pradhan and Goudsmit [6] gave an algebraic equation to test whether a non-feedback bridging fault of two lines was detectable. They generalized their results to the case of bridging of an arbitrary number of lines (Theorem 4). They also proved that in a nonredundant two-level AND-OR (NAND-NOR) network, all intragate bridging faults are detectable.

Iosupovic [7] proved that for two-level AND-OR tree, a minimum fault detection set for a single stuck-at fault could be found which would also detect all single bridging faults. He also gives a procedure, called maximum dissimilarity procedure, for finding that minimal fault test set.

Danilov, Evarovskv and Moskaliev [8] considered the problem of detection and location of short circuits and open circuits of an arbitrary multiplicity in non-oriented graphs and contact networks. The exact bounds for the minimum number of tests for both detection and location were given. In [9] they applied the method developed for the contact networks to the detection of stuck-at errors in an arbitrary combinational network with many outputs.

In the paper by Liu and Su [10], single feedback bridging faults present between two lines in general combinational networks were examined. Sufficient conditions independent of circuit implementations were derived to determine the set of feedback faults which caused a logic network to oscillate or convert it to a sequential network. Test patterns for detecting these faults were generated and dominance relations between feedback bridging faults were studied.
III. DETECTION OF FEEDBACK BRIDGING FAULTS

In this section, first we shall present our results on the detection of feedback bridging (short circuit) faults between the primary output and the primary inputs in a single-output logic network. The results on the detection of bridging faults among the primary inputs shall be given in the next section.

Instead of considering the bridging between two lines, we shall consider the general case of the bridging among the primary output and a primary input line, called feedback bridging of multiplicity $s$. Similarly, the input bridging among $s$ input lines is called input bridging of the multiplicity $s$. Without loss of generality, we assume that for a network implementing function $F(x_1, x_2, \ldots, x_n)$, if the $s$ input lines which are bridged together (either with the primary output or among themselves only) are known, then these lines are $x_{1'}, x_{2'}, \ldots, x_{s'}$. $(X', x_{2'}, \ldots, x_n)$ and $(x_1', x_{2'}, \ldots, x_n)$ denote feedback and input bridging of multiplicity $s$, respectively.

Let us consider a combinational network implementing $F(x_1, x_2, \ldots, x_n)$. If the AND-type bridging fault exists between the primary output and a input line $x_1', x_2', \ldots, x_{s'}$, then the primary output $Y$ is equal to the AND function of the original output of the network and $x_1', x_2', \ldots, x_{s'}$. Each one of the first $s$ primary inputs becomes $x_1', x_2', \ldots, x_{s'}$. This can be represented by the model shown in Fig. 1. Such a model will be used throughout the paper for feedback bridging faults.

The following definitions can be found in [8].

Definition 1. A circuit oscillates under certain input combination (pattern), if the output of the circuit at the next instant is the complement of current output, i.e., $Y = Y' \uparrow$ where $\uparrow$ is the output at time $t'$.

Definition 2. A circuit has asynchronous behavior under certain input combination if the circuit is stable and the present output is a function of its previous inputs and $Y = Y' \uparrow$.

The proofs and the examples for the following theorems can be found in the very recent paper by these authors [1].

Theorem 1. Under feedback bridging $(X', x_2', \ldots, x_n)$ any network $N$ implementing $F(x_1', x_2', \ldots, x_n)$ oscillates if the binary input $n$-tuple $(x_1', x_2', \ldots, x_n)$ satisfies the following conditions:

$$x_{1'}, x_2', \ldots, x_n \in F(0, 0, \ldots, 0, x_1', x_2', \ldots, x_n)$$

$$F(1, 1, \ldots, 1, x_1', x_2', \ldots, x_n) = 1$$

(1)

We will have the asynchronous behavior if

$$x_1', x_2', \ldots, x_n \in F(0, 0, \ldots, 0, x_1', x_2', \ldots, x_n)$$

$$F(0, 1, \ldots, 1, x_1', x_2', \ldots, x_n) = 1$$

(2)

Corollary 1. Under feedback bridging $(Y, x_1)$, any network $N$ implementing $F(x_1, x_2', \ldots, x_n)$ oscillates if the binary input $n$-tuple $(x_1, x_2', \ldots, x_n)$ satisfies

$$x_1 \in F(1, x_2', \ldots, x_n) \cup F(0, x_2', \ldots, x_n) \cup 1$$

(3)

$N$ has asynchronous behavior if

$$F(x_1, x_2, \ldots, x_n) = 1$$

(4)

Note that for the network to oscillate, the total number of inversions in the feedback loop must be odd. If the number of inversions in the feedback loop is even, the bridging faults in the network can be detected by utilizing the asynchronous behavior property.

If Eq. (2) is satisfied, then

$$F(0, 0, \ldots, 0, 0, x_1', x_2', \ldots, x_n) = 0$$

which means that the circuit output can be reset to 0 as long as the first $s$ input variables are 0's. After resetting the output to 0, if we apply an input pattern $Y$ such that $F(Y) = 1$, then from the model shown in Fig. 1, the output response is 1 for the fault-free network and 0 for the network with feedback bridging of a multiplicity $s$.

From Theorem 1, if there exists $x_{1'}, x_2', \ldots, x_n \in \{0, 1\}$ such that $F(0, 0, \ldots, 0, x_1', x_2', \ldots, x_n) \neq 0$, then either Eq. (1) or Eq. (2) is satisfied for an input pattern with the first $s$ variables equal to 0's. Therefore, the bridging $(Y, x_1', x_2', \ldots, x_n)$ can be detected by observing the oscillation or asynchronous behavior of the faulty network.

It is interesting to note that sometimes the same test pattern can detect oscillation for the bridging of the given multiplicity $s$ but cannot detect bridging faults of multiplicity less than $s$. Formally speaking, if test pattern $\tau$ satisfies Eq. (1) or Eq. (2) for the bridging $(Y, x_1', x_2', \ldots, x_n)$, then $\tau$ not necessarily satisfies Eq. (1) or Eq. (2) for a bridging $(Y, x_1', x_2', \ldots, x_n)$ ($s = 1$) [1].

Theorem 1 is devoted to the conditions of the oscillation or of the asynchronous behavior and to the detection of the given feedback bridging. The following theorem will be devoted to the case when we don't know which input lines are bridged and only the multiplicity $s$ of a fault is given. Let $[x]$ denote the number of 1's in the binary $n$-tuple $x_1', x_2', \ldots, x_n$.

$$[x] = 2^k$$

and $F([x], [x] - 1) = 1$.

(5)

We note that, as it follows from the model of a faulty network (see Fig. 1), if $\tau$ is a single test pattern generating the oscillation for all possible bridging of the given multiplicity $s$ ($1 \leq s \leq n$), then $\tau = \tau^* (1, 1, \ldots, 1)$.

Theorem 2 (4) The single test vector $\tau^* (1, 1, \ldots, 1)$ detects all possible feedback bridging of the given multiplicity $s$ by oscillation in a network realizing $F(x_1', x_2', \ldots, x_n)$ if

$$F([x] = 0),$$

$$F([x] = 0).$$

(6)

$\tau^* (1, 1, \ldots, 1)$. The test sequence $(\tau, \tau^*)$ detects all possible feedback bridging of the given multiplicity by asynchronous behavior in a network realizing $F(x_1', x_2', \ldots, x_n)$ if

$$F([x] = 0),$$

$$F([x] = 0).$$

(7)

where $K$ is the input pattern which resets the output to 0.
Example 1. (1) For the parity checker $F = x_1$ and weight, $J = (1, 1, 1, \ldots)$ generates the oscillation for any feedback bridging of odd multiplicity. 

(11) For a K-bit parallel adder $F(x_1, y_1)$ where $K = 1$:

$$x = \sum_{i=0}^{K-1} x_i 2^i, \quad y = \sum_{i=0}^{K-1} y_i 2^i, \quad F = \sum_{i=0}^{K-1} f_i 2^i$$

where $x_0, x_1, \ldots, x_K$ and $y_0, y_1, \ldots, y_K$ are the inputs and outputs, respectively.

Theorem 2 deals with the case when the multiplicity of bridges is known. The following result is for the general case where the multiplicity of the bridges is unknown.

Theorem 3. Let $H_{\text{det}}(0)$ be the minimum number of tests for detecting feedbackbridges of any multiplicity in a network implementing $F$, where $F$ represents the logical function. Then:

$$H_{\text{det}}(0) < n$$

Since $F$ is the function representing the network and $H_{\text{det}}(0)$ is the number of tests required to detect feedback bridges, we can determine the minimum number of tests needed.

Example 2. For a K-bit comparator $F(x_i, y_i)$, $x_i = y_i$ if $x_i = y_i$, $F = \overline{x_i}$ because $x_i$ and $y_i$ are the inputs. Then:

$$x = \sum_{i=0}^{K-1} x_i 2^i, \quad y = \sum_{i=0}^{K-1} y_i 2^i$$

and since $x = y$ if $i = 0$.

Theorem 4 states that for any feedback bridge in a network implementing $F(x_1, y_1)$, the output test patterns are detected.

Example 3. For a K-bit adder (see Example 1(11)) where $F(x_1, y_1)$, $F = \sum_{i=0}^{K-1} x_i y_i 2^i$ and two input patterns $x_0, x_1$ and $y_0, y_1$, we can determine any feedback bridge between inputs $x_0, x_1$ and $y_0, y_1$.

Example 4. For a K-bit multiplexer with $K$ data inputs ($x_1, x_2, \ldots, x_K$) and one output $F$, we have $F(x_1, x_2, \ldots, x_K) = x_j$ where $j = 0, 1, \ldots, 2^k - 1$ and $x_j = x_j$. Choose $F = \sum_{i=0}^{K-1} x_i 2^i$ where $x_0, x_1, \ldots, x_K$ are the inputs.

Then $F(x_1, x_2, \ldots, x_K)$.
Note that the generalizations of the results for locating feedback and input bridging is also given in [19].

5. Fault Detection in Multiple-Output Networks

Our previous results were mostly devoted to the case of networks with single output. In this section, we consider the problem of test generation for multiple-output networks. The problem of the detection of input, output and feedback bridging and stuck-at faults in these networks is very important for the maintenance testing of remote networks e.g blocks of an on-board computer.

Let us consider a combinational network with $n$ inputs and $m$ outputs implementing the system of $m$ functions

$$T_k = f_k(x_1, \ldots, x_n) \quad k = 1, \ldots, m$$

We shall consider the following five classes of faults: (1) Feedback bridging; (2) Input bridging; (3) Output bridging; (4) Input stuck-at faults; (5) Output stuck-at faults.

5.1 Feedback Bridgins

First, let us consider feedback bridging. If $F_k(0^m)=0$ (where $1 \leq k \leq m$ and $0^m = (0, \ldots, 0)$), then input vector $0^m$ detects all feedback bridgings (since output vector for a faulty network contains zero). Similarly, if there exists $t=(t_0, \ldots, t_n)$ such that $F_k(t)=F_k(0^m)$ and $t_0 \neq 0$, then two input vectors $t, t_0$ detect all feedback bridgings. In general, any sequence $(t_1, \ldots, t_n)$ of input patterns such that $F_k(t_i)=F_k(0^m)$ and $j=1, \ldots, n$ and

$$t_{i-1} \neq 0 \quad \text{is the test sequence detecting all feedback bridgings.}$$

Another approach for generating tests for feedback bridging is based on the observation of the oscillation of a synchronous behavior. In this case, we may use the corresponding results from Section III.

Let $T_k$ be a minimal test for a part of the network implementing $F_k(1, \ldots, 1)$. Then we always, can use the set $T = \bigcup_{k=1}^m T_k$ as a test set for the detection of all bridging faults for the network with a single input and $m$ outputs.

$$1 \leq s_k(0^m) \leq \left(\bigcup_{k=1}^m T_k \right) \leq n = m$$

For any IC chip with 10-input, 10-output variables, in the worst case, it takes only 100 test patterns to detect the feedback bridging. We suspect that the upper bound smaller than 100 can be obtained and we plan to look into this.

5.2 Input Bridgings

Let us consider now input bridging. First we generalize the conditions of undetectability of input bridging to the case of multiple-output. The bridging $(x_1, \ldots, x_n)$ is undetectable if

$$F_k(0_1, \ldots, 0_n, x_1, \ldots, x_n) = F_k(0_1, \ldots, 0_n, x_1, \ldots, x_n)$$

for all $\alpha = (\alpha_1, \ldots, \alpha_n)$, $\alpha \neq 0$,

$$x_1, \ldots, x_n \in \{0,1\}, \quad i=1, \ldots, m.$$  

We note that there exist system $F_k = \{F_k(x_i), i=1, \ldots, n\}$ such that each one of the functions of the system cannot detect all input bridgings of the given multiplicity, but the whole system detects them. In other words, there exists a multiple-output network such that not all input bridgings can be detected by observing any one output but all input bridgings can be detected by observing all outputs.

Example: Let us consider single input bridging in a network computing $F_1 \cdot F_2$ (see Table 1, here $m=3$, $n=2$).

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_1 \cdot F_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1

It follows from (11) that the function $F_1$ cannot detect bridging $(x_1, x_2)$ and the function $F_2$ cannot detect $(x_2, x_1)$, but the whole system detects all single input bridging, e.g., by the test set $T = \{(0,1), (0,0)\}$ since $T \cup \{(1, 1)\}$ detects the bridging between $x_1$ and $x_2$ or $x_2$ and $x_1$ and $(0,0)$ test bridging between $x_1$ and $x_2$.

If $T = \{(0,0), \ldots, (1,1)\}$ is a test sequence for detecting all input bridgings then in the K x 3 matrix $T$ with rows $t_1, \ldots, t_3$, all columns will be different.

We note also that, if $T_k$ is a minimal test set for detecting input bridging by observing output $F_k$ only, then we always can use $T_k$ as a test sequence for a whole network. Thus, if $N_{s_k}(m, n)$ is the minimal number of tests, for input bridging, then we have $N_{s_k}(m, n) \geq N_{s_k}(1, m)$.

Since for a 4-output network, there are 10 instead of one observation points, the actual number of tests will be smaller than the smallest $[T_k]$.

5.3 Output Bridgings

For output bridging, we note first that all these bridgings are detectable if our network is irreduntant (nonredundant). Indeed, if $t$ detects output bridging $(F_k(t_1), \ldots, F_k(t_n))$, then $t$ detects output bridging $(F_k(t_1) \oplus F_k(t_k))$ (since for a network with an A-M bridging, $F_k(t) = F_k(t) = 0$).

We note that $T = \{(0,1), \ldots, (1,1)\}$ is a test sequence for all output bridging, if and only if in the $(m \times n)$ binary output matrix.
For the corresponding output matrix $F(T)$ we have

$$
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
\end{array}
$$

$$
\text{If } F(t) = \{\text{Sign}Z, \text{sign}Z, \ldots, \text{Sign}Z\}, \quad |F(t)| = 1 \\
\sum_{i=1}^{N} \text{sign}Z_i \\
\text{The general case of K-bit algebraic multiplier is considered in Table 2.}
$$

Note also that the approach to the test generation described above for combinational networks may also be applied for networks with memory if storage elements are directly readable and writable from primary inputs and outputs, which is the standard requirement in the design for testability. Several examples of test sequences for the devices with memory (registers, counters, RAM) are considered in Table 2.

### VI. APPLICATIONS: TESTING STANDARD COMPONENTS

In this section we shall apply our previous results to the simultaneous detection of bridging and stuck-at faults at primary inputs and outputs. The testing of faults in input/output pins is an important part of field testing. Only AND-type bridges will be considered but the results may easily be modified to the case of OR-type bridges. The following five types of faults will be detected: (1) stuck-at faults at input lines; (2) stuck-at faults at output lines; (3) bridging among input lines; (4) bridging among output lines; (5) feedback bridging. Detection of feedback bridging will be based on observation of the asynchronous behaviour (model in Fig. 1) and Theorems 4.

Table 2 contains tests for detecting all single stuck-at and bridging faults for shift registers (Shifters) and counters with parallel load, decoders, algebraic adders, multipliers, dividers and RAM. These tests detect all single faults and about 95% of multiple faults.

Table 2 also contains $N$, the number of test patterns as a function of $i$, the number of input pins for each component. (Input pins for power supply, clock and enable are not included). Numbers in the above table are expressed in the notation of the previous section. For devices with memory (registers, counters, RAM) we assume that all flip-flops are directly testable and readable from the corresponding input and output pins.

Table 2 also contains examples of test sequences for K-bit arithmetical and logical devices with a small $K$. The following notations are used in Table 2.
I is a number of input pins.
N is a number of faults for detecting any single fault in the input and output pins of a component.
For a test sequence $T = (t_1, t_2, \ldots, t_m)$ where
\[ t_i = (t_{i1}, t_{i2}, \ldots, t_{in}) \]
$T$ is in the (nxk) binary input matrix with the row $t_i = (t_{i1}, t_{i2}, \ldots, t_{in})$.
For a device with $V$ input lines and $R$ output lines implementing the changes of Boolean functions, $N = 2^V$ ($x_1, x_2, \ldots, x_V$) where $1, 2, \ldots, R$ we denote $F(C)$ the (nxk) binary output matrix with the element in the ith row and the jth column denotes the jth output response to the ith test.
$N = 2^R$, $x_1, x_2, \ldots, x_V$-matrix containing all possible binary vectors with a components per column: $N_{ij}$ is the transposed matrix for $M_{ij}$, $M_{ij}$ being the matrix for the whole all (one-all) vector column: $N_{ij}$ is the matrix obtained by the negation of all entries in $M_{ij}$: it is the identity matrix.
Let A and B be two (mxn)-binary matrices with rows $A_1, A_2, \ldots, A_m$ and $B_1, B_2, \ldots, B_n$. $A \oplus B$ (the (mxn)-matrix with rows $A_1 \oplus B_1, A_2 \oplus B_2, \ldots, A_m \oplus B_m$. We now outline the approach we took for constructing Table 2 and explain why the tests (shown in the Table) detect the aforementioned five types of faults.

First, let us consider the counter (device No. 2). The first rows in matrices $T$ and $F(C)$ in Table 2 show that when counter control 0-1, the counter counts up to 13, which changes all 0's to 1's and vice versa. $M_{ij}^{(0)}$ is the content of the counter remains unchanged. Matrix $T$ is selected in such a way that all columns of $T$ are distinct and not equal 0 or 1. If one column contains all 0's (1's) then $x_1, x_2, \ldots, x_V$ cannot be detected. Therefore, any binary $y$ that are any two inputs will change at least one column and hence change the value of output response for at least one test pattern. For instance, if there is an AND-biasing fault between $x_1$ and $x_2$, then the last column of $F(C)$ will change to all 0's. The first test pattern will detect any feedback bridging of any multiplicity between any output and any input (including input signal C) because any feedback bridging $x_1, x_2, \ldots, x_V$ will change $y_1$ from 0 to 1 (see the model in Fig. 1) and hence change the outputs. For instance, if there is a bridging of $x_1$ and $x_2$, then the first row of $F(C)$ will become $(1,1,1,1,1,1)$ and thereby changes the first row of $F(C)$ to $(1,1,1,1,1,1)$. For stuck-at faults, the first test will detect any s-a-1 faults at inputs or s-a-1 at outputs. The second test will detect $x_1, x_2, s-a-1$ or $x_4, x_5$ and $x_6, s-a-0$. The last two tests detect the remaining stuck-at faults.

Based on the two matrices in this example, we develop a general form of $T$ for a counter with any number of bits. Note that in this example, the columns of the submatrix $T$ in which the first two to 4 contain all possible combinations of two variables except (1,1,1,1). We denote it by $N_{ij}$ in the general form for $T$ in Table 2. Similarly, $M_{ij}$ in $F(C)$ denotes a matrix containing all possible combinations of its columns except $M_{ij}^{(0)}$ and $M_{ij}^{(1)}$. Finally, the number of tests, $N$, for a counter with $T$ is $2^R-1$ inputs (including control signal C) is $\log_2(2^R-1) + 1$.

Let us now look at the adder/subtractor in Table 2. Each row of matrix $T$ contains the sign $S$ and magnitude (y_1, y_2, y_3) of the first number $X$ and the sign $S$ and the magnitude $(y_1, y_2, y_3)$ of the second number Y. The sum $X + Y$ is given in the corresponding row in $F(C)$, the matrix for the sum. Again we choose $T$ such that all columns are distinct and not equal 0 or 1. This yields the $F(C)$ with distinct columns not equal 0 or 1. Hence all input and output bridgings are detected. The feedback bridging is detected by using the asynchronous behavior property of sequential circuits (Fig. 1 and Theorem 1). The output response to the first test is all 0's. Now if there is a feedback bridging between $x_1$ and $x_2$ for any 1 and 0, then the bridging will change the second test from 1 (0,1,0,1,0) to a pattern with all 2's except a 0 at the 5th position. This in turn will change the sum shown in the second row of $F(C)$ and hence the fault is detected. Finally, the first test in $T$ detects any stuck-at-1 faults at any input or output. The second test detects all stuck-at-0 faults on the inputs and all stuck-at-0 faults except one (i.e., $x_2$) at the outputs. $x_1, x_2, x_3$ is detected by the third test. The test patterns for an adder with any number of bits are given and the total number of tests is $\log_2(2^R-1)$ where $\log_2$ is the number of rows in $M_{ij}$, as shown in Table 2. With a simple modification, Table 2 can be used for generating tests 1's with any number of inputs $T$.

By the same approach, the complete test set and the total number of tests for other standard digital components can be found.

The number of tests for detecting any single fault in different K-bit devices and RAM is given in Table 3.

<table>
<thead>
<tr>
<th>Device</th>
<th>No. of bits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
</tr>
<tr>
<td>Shift Register</td>
<td>6</td>
</tr>
<tr>
<td>Counter</td>
<td>5</td>
</tr>
<tr>
<td>Up &amp; Down Counter</td>
<td>6</td>
</tr>
<tr>
<td>Algebraic Adder</td>
<td>6</td>
</tr>
<tr>
<td>Multiplier</td>
<td>8</td>
</tr>
<tr>
<td>Divider</td>
<td>8</td>
</tr>
<tr>
<td>RAM</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 3. The number of tests for K-bits devices and RAM for detecting input, output and feedback bridge and stuck-at faults.

As we can see from Table 3, for the testing of arithmetical devices and RAM of practical size without respect to input, output and feedback bridgings and stuck-at faults, we need very few tests.

From Table 2 we can see that the complete detection set can be stored in a small PROM and these test patterns can rapidly be applied to the network-under-test to determine whether there is a fault.
CONCLUSION AND FUTURE WORK

With the advances in LSI and VLSI, the problem of testing bridging faults will become worse unless efficient testing strategies are developed very soon. This paper and the very recent paper by these authors [18] presents an initial step toward solving this important problem. Some theories and methods for testing the bridging faults at the input and output pins of integrated circuit chips for some standard digital components have been presented. The complete test set for detecting bridging and stuck-at failures in these components have been generated. Research funds are being sought to continue performing research on the following topics in this important area:

1) Generalization of our results for testing basic standard digital components to the test generation for detecting input, output and feedback bridging and stuck-at faults for ALU, PLA, CPU and some commercially available microprocessors (maintenance testing).

2) Algorithms for generating efficient tests for detecting all internal bridges (manufacturing testing).

3) Lower and upper bounds for the minimal number of tests for detecting all internal bridges (testing time for internal bridges), and the mixture of bridging and stuck-at faults.

4) Techniques and algorithms for the construction of efficient test sets for detecting all bridging and stuck-at faults in multiple-output digital networks.

5) Computer implementation of algorithms developed in items 1, 2, and 4, i.e., program the algorithms for generating tests for detecting internal bridging and the mixture of stuck-at and bridging faults.

REFERENCES


Fig. 1. Logical model of feedback bridging \( \{x_1, \ldots, x_n\} \)

Fig. 2. Logical model of input bridging \( \{x_1, \ldots, x_n\} \)
<table>
<thead>
<tr>
<th>DEVICE</th>
<th>DEFINITION</th>
</tr>
</thead>
</table>
| NO. 1  | F(\text{Shl}, \text{Shr}, x_1, x_2, \ldots, x_k) =  \\
| K-bit register with parallel load or shifter |
|        | (x_1, x_2, \ldots, x_k) if Shl=Shr=0; |
|        | (x_1, x_2, \ldots, x_k, 0) if Shl=1, Shr=0; |
|        | (0, x_1, x_2, \ldots, x_k, 2^r x_{k-1}) if Shl=0, Shr=1; |
|        | \( K = 2^k \leq 2^r \). |

| NO. 2  | \(|Z|=F(C, X) =  \\
|        | C=1+(\text{mod } 2^k) |
|        | C=0,1 |
|        | i=1 |
|        | X = \( x_1 x_2 \ldots x_k \cdot 2^r 1\). |
|        | Z = \( x_2 \ldots x_k \). |

| NO. 3  | \(|Z|=F(C, X, \text{down}) =  \\
|        | C=1+(\text{mod } 2^k) |
|        | C=0,1 |
|        | i=1 |
|        | X = \( x_1 x_2 \ldots x_k \cdot 2^r 1\). |
|        | Z = \( x_2 \ldots x_k \). |

| NO. 4  | \( F(\text{Enable}, x_1, \ldots, x_k) =  \\
|        | (P_1, P_2, \ldots, P_k) |
|        | \text{Decoder-} |
|        | \text{Demultiplexer} |
|        | \text{Program Enable-f} |
|        | \text{Enable} |
|        | K |

---

**Table 2 Complete Detection Sets for Standard Digital Components**

<table>
<thead>
<tr>
<th>K=3 T=4 N=9</th>
<th>F(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>00001</td>
</tr>
<tr>
<td>01000000</td>
<td>010100</td>
</tr>
<tr>
<td>00010000</td>
<td>000101</td>
</tr>
<tr>
<td>00001000</td>
<td>000110</td>
</tr>
<tr>
<td>00000100</td>
<td>000111</td>
</tr>
<tr>
<td>00000010</td>
<td>001000</td>
</tr>
<tr>
<td>00000001</td>
<td>001001</td>
</tr>
<tr>
<td>00000000</td>
<td>001010</td>
</tr>
<tr>
<td>00000000</td>
<td>001011</td>
</tr>
<tr>
<td>00000000</td>
<td>010000</td>
</tr>
<tr>
<td>00000000</td>
<td>010001</td>
</tr>
<tr>
<td>00000000</td>
<td>010010</td>
</tr>
<tr>
<td>00000000</td>
<td>010011</td>
</tr>
<tr>
<td>00000000</td>
<td>010100</td>
</tr>
<tr>
<td>00000000</td>
<td>010101</td>
</tr>
<tr>
<td>00000000</td>
<td>010110</td>
</tr>
<tr>
<td>00000000</td>
<td>010111</td>
</tr>
<tr>
<td>00000000</td>
<td>011000</td>
</tr>
<tr>
<td>00000000</td>
<td>011001</td>
</tr>
<tr>
<td>00000000</td>
<td>011010</td>
</tr>
<tr>
<td>00000000</td>
<td>011011</td>
</tr>
<tr>
<td>00000000</td>
<td>011100</td>
</tr>
<tr>
<td>00000000</td>
<td>011101</td>
</tr>
<tr>
<td>00000000</td>
<td>011110</td>
</tr>
<tr>
<td>00000000</td>
<td>011111</td>
</tr>
<tr>
<td>00000000</td>
<td>100000</td>
</tr>
<tr>
<td>00000000</td>
<td>100001</td>
</tr>
<tr>
<td>00000000</td>
<td>100010</td>
</tr>
<tr>
<td>00000000</td>
<td>100011</td>
</tr>
<tr>
<td>00000000</td>
<td>100100</td>
</tr>
<tr>
<td>00000000</td>
<td>100101</td>
</tr>
<tr>
<td>00000000</td>
<td>100110</td>
</tr>
<tr>
<td>00000000</td>
<td>100111</td>
</tr>
<tr>
<td>00000000</td>
<td>101000</td>
</tr>
<tr>
<td>00000000</td>
<td>101001</td>
</tr>
<tr>
<td>00000000</td>
<td>101010</td>
</tr>
<tr>
<td>00000000</td>
<td>101011</td>
</tr>
<tr>
<td>00000000</td>
<td>101100</td>
</tr>
<tr>
<td>00000000</td>
<td>101101</td>
</tr>
<tr>
<td>00000000</td>
<td>101110</td>
</tr>
<tr>
<td>00000000</td>
<td>101111</td>
</tr>
<tr>
<td>00000000</td>
<td>110000</td>
</tr>
<tr>
<td>00000000</td>
<td>110001</td>
</tr>
<tr>
<td>00000000</td>
<td>110010</td>
</tr>
<tr>
<td>00000000</td>
<td>110011</td>
</tr>
<tr>
<td>00000000</td>
<td>110100</td>
</tr>
<tr>
<td>00000000</td>
<td>110101</td>
</tr>
<tr>
<td>00000000</td>
<td>110110</td>
</tr>
<tr>
<td>00000000</td>
<td>110111</td>
</tr>
<tr>
<td>00000000</td>
<td>111000</td>
</tr>
<tr>
<td>00000000</td>
<td>111001</td>
</tr>
<tr>
<td>00000000</td>
<td>111010</td>
</tr>
<tr>
<td>00000000</td>
<td>111011</td>
</tr>
<tr>
<td>00000000</td>
<td>111100</td>
</tr>
<tr>
<td>00000000</td>
<td>111101</td>
</tr>
<tr>
<td>00000000</td>
<td>111110</td>
</tr>
<tr>
<td>00000000</td>
<td>111111</td>
</tr>
</tbody>
</table>
### Table 2 (Cont'd) Complete Detection Set for Standard Digital Components

<table>
<thead>
<tr>
<th>A_1</th>
<th>A_2</th>
<th>\ldots</th>
<th>A_n</th>
<th>Z</th>
<th>X_1</th>
<th>X_2</th>
<th>\ldots</th>
<th>X_K</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>\ldots</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>\ldots</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>\ldots</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>\ldots</td>
<td>1</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>\ldots</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>\ldots</td>
<td>1</td>
</tr>
</tbody>
</table>

**Notes:**
- **C** = 1 if Read mode
- **C** = 0 if Write mode
- **n** = number of address bits
- **K** = number of data bits
- **Z** = \(Z = \sum_{i=0}^{K} X_i \cdot 2^i\)
- **X_i** = \(X_i = (x_{i1}, x_{i2}, \ldots, x_{iK})\)
- **F** is the parity function for detecting all single-bit faults.