Resistive Bridge Fault Modeling, Simulation and Test Generation

Vijay R. Sar-Dessai
Intel Corporation, FM5-64
1900 Prairie City Road
Folsom CA 95630
Tel: (916) 356-1759
Fax: (916) 377-1300
Email: vijay.sar-dessai@intel.com

D. M. H. Walker
Dept. of Computer Science
Texas A&M University
College Station TX 77843-3112
Tel: (409) 862-4387
Fax: (409) 847-8578
Email: walker@cs.tamu.edu

Abstract

In this work[1] we develop models of resistive bridging faults and study the fault coverage on ISCAS85 circuits of different test sets using resistive and zero-ohm bridges at different supply voltages. These results explain several previously observed anomalous behaviors. In order to serve as a reference, we have developed the first resistive bridging fault ATPG, which attempts to detect the maximum possible bridging resistance at each fault site. We compare the results of the ATPG to the coverage obtained from other test sets, and coverage obtained by using the ATPG in a clean-up mode. Results on ISCAS85 circuits show that stuck-at test sets do quite well, but that the ATPG can still improve the coverage. We have also found that the loss of fault coverage is predominantly due to undetected faults, rather than faults in which only a small resistance is detected. This suggests that lower-cost fault models can be used to obtain high resistive bridge fault coverage.

I. Introduction

There has been much recent discussion about the best way to achieve high coverage of realistic faults. One approach is to use *defect-based* testing [1][2] in which a physically realistic fault model is used for fault simulation and ATPG. The argument is that the stuck-at model is a poor model of realistic faults [3][4] and in particular the correlation between predicted and actual fault coverage falls off at high coverage levels [5]. The second approach is to use the traditional stuck-at fault model, but to modify the ATPG so that the generated vectors achieve high realistic fault coverage [5][6]. As was pointed out in [7], it is the vectors that detect the real defects, not the fault model. The purpose of this research is to study the fault coverage of several traditional test sets on realistic faults, in particular resistive bridging faults, and to build the first resistive bridging fault ATPG to provide the best possible results to serve as a comparison. Since even random vectors will quickly detect many resistive bridging faults, we also use the ATPG in a clean-up mode after applying traditional test sets, to see how much improvement can be achieved. The goals are to gain insight into the necessary requirements for achieving high resistive bridging fault coverage, whether targeting resistive bridging faults is necessary, the difficulty of doing so, and the benefits of resistive bridge clean-up vectors.

We chose to consider resistive bridging faults since a short between circuit nodes is the predominant type of manufacturing defect [8], and shorts between gate outputs or *bridging faults* (BF) account for about 90% of shorts [9][10][11]. Since the accuracy of fault simulation and ATPG is heavily dependent on the fault model [12], we developed an accurate resistive bridging fault (RBF) model that considers the behavior of both the driving and driven gates. The latter is necessary for the correct logical interpretation of the node voltages [13][14].

Many BF models have been developed [13][15][16][17][18][19][20][21]. Most assume a zero ohm resistance, but several assume a resistive bridge [22][23][24][25]. Figure 1 shows a distribution fit to the bridge resistance data in [8]. R_b is the bridging resistance and $P(R_b)$ is the cumulative probability. As can be seen, many BFs have significant resistance.

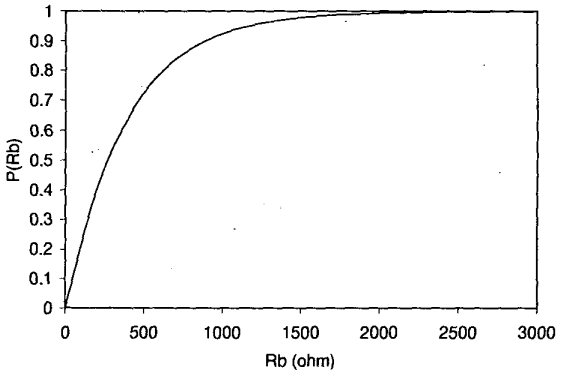

Figure 1. Bridging resistance distribution function.

[1] This research was supported by the National Science Foundation under grant MIP-9406946. This research was performed while Vijay Sar-Dessai was a M.S. student at Texas A&M University.

Since a test for a zero-ohm bridging fault (ZBF) does not guarantee detection of an RBF, ideally an RBF model should be used. In this work we consider logic testing for RBFs, as compared to prior work on ZBFs [26].

It is well known that as the power supply voltage V_{DD} is decreased, higher bridge resistance is detected, or faults which had escaped detection at a higher value of V_{DD} are detected [27][28][29][30][31]. Hence it is useful to do logic testing at several V_{DD} values.

In the sections that follow we first discuss our RBF model, then our fault simulation and ATPG algorithms, applications to ISCAS85 benchmarks, and conclusions.

II. Fault Model

To accurately model the behavior of BFs, we must determine the voltage at the bridged nodes for each vector that excites the BF. Then, based on the logic threshold of the driven gates, we can determine whether the BF is detectable at the driven gate output. We can also determine the maximum detectable resistance (R_{upper}) at the output of this gate, which gives us the *detectable resistance interval* (DRI), assuming all lower resistance is also detected. (A DRI is the range of BF resistance that can be detected at the fault site).

Figure 2 shows a bridging resistance between nodes X and Y. The fault is excited with logic 1 on X and logic 0 on Y (or vice-versa). The BF is essentially a resistance between V_{DD} and GND, via pull-up devices in gate g1 and pull-down devices in gate g2. V_X and V_Y, the voltages on X and Y, depend on the number of pull-up and pull-down devices involved in the bridge, and hence depend on the vector at A1, B1, A2 and B2 used to excite the fault.

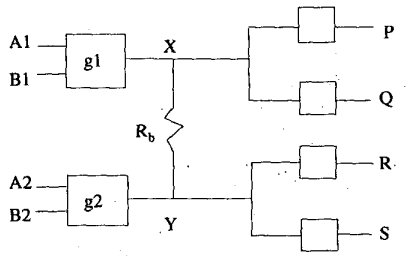

Figure 2. Resistive bridging fault.

Figure 3 shows the typical variation of voltages V_X and V_Y (for the excitation $\{X,Y\} = \{1,0\}$) as the bridge resistance increases from 0Ω. The interpretation of the voltages at X and Y depends on the logic threshold of the gates fed by these nodes. When the BF has a 0Ω resistance, voltages V_X and V_Y are equal to V_0. As R_b increases, V_X increases from V_0 to V_{DD} and V_Y decreases from V_0 to 0. If a gate fed by X has a threshold between V_0 and V_{DD}, the BF will be detected at the gate output, assuming the other gate inputs are at non-controlling values. Likewise, if a gate fed by Y has a threshold between V_0 and 0, the BF will be detected at the gate output. R_{upper} depends on the value of the logic threshold. As shown in Figure 3, R_{upper} is R_{dP} at P and R_{dS} at S. (V_{thP}, V_{thQ}, V_{thR} and V_{thS} are the logic thresholds of the driven gates at P, Q, R and S respectively). The DRI is [0 R_{dP}] at P and [0 R_{dS}] at S. The BF is undetectable at Q and R.

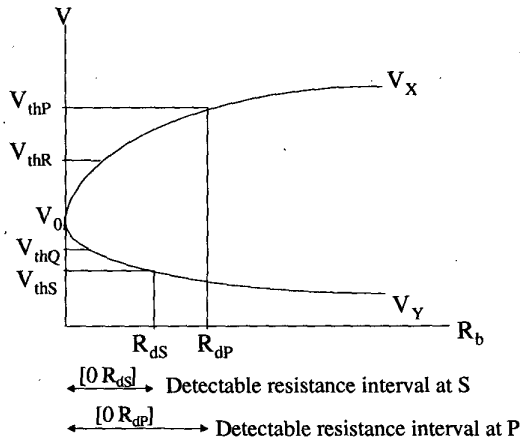

Figure 3. Detectable resistance intervals.

A. Fault Model Description

In this work, BFs have been modeled by HSPICE [32] circuit simulation for almost all-possible BF configurations in the gate-level description of the ISCAS85 benchmarks [33]. Each gate is implemented using complementary CMOS logic. We use a level 3 device model for the HP CMOS14TB 0.5 μm process, with nominal V_{DD} of 3.3V. The n-channel device threshold, V_{tn}, is 0.67V and the p-channel device threshold, V_{tp}, is -0.94V. The result is a set of look-up tables that describe the logic-level behavior of the fault site. Below we describe how each type of bridging configuration is analyzed.

1. Case 1: BF between two primary inputs

We define primary inputs (PIs) as voltage sources, so BFs between them are not logic testable. Hence this type of BF is not modeled.

2. Case 2: BF between a PI and gate output

Figure 4 shows a BF between a PI, A, and the output of a NAND2 gate, X. Node X feeds into two gates having different threshold voltages. The bridge resistance detectable at nodes P and Q depends on the test vector at A, B, C as well as on the logic threshold values of the two gates driven by node X.

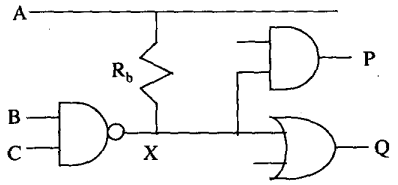

Figure 4. Bridging fault between PI and gate output.

For example, simulation shows that if the vector is $\{A,B,C\} = \{0,0,1\}$, R_{upper} is 1600Ω at P and 1400Ω at Q, assuming that other inputs of the AND2 and OR2 gates are

non-controlling. The fault does not propagate along A since it is a PI.

3. Case 3: BF between gate outputs (bridged nodes feeding into different gates)

Figure 5 illustrates a case in which NAND2 and NOR2 gate outputs are bridged, and the bridged nodes X and Y feed into different gates. R_{upper} depends on the vector {A1,B1,A2,B2} and the thresholds of the driven gates.

For {A1,B1,A2,B2} = {1,0,1,1}, the BF will propagate along X, R_{upper} is 1000Ω at P and 1400Ω at Q. For the vector used and thresholds of the NOT and AND2 gates, the fault does not propagate along Y, and is undetectable at R and S.

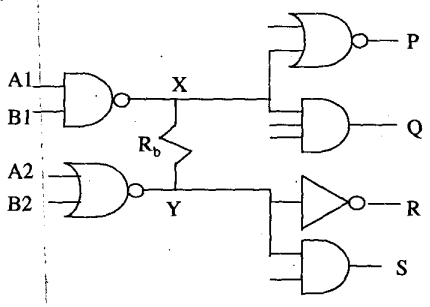

Figure 5. Bridging fault between nodes feeding different gates.

4. Case 4: BF between gate outputs (bridged nodes feeding into same gate)

Figure 6 illustrates the case in which two NAND gates are bridged, and feed into the same AND3 gate. The bridge resistance detectable at P depends only on {A1,B1,A2,B2} assuming C = 1. For {A1,B1,A2,B2} = {1,0,1,1}, R_{upper} is 800Ω at P.

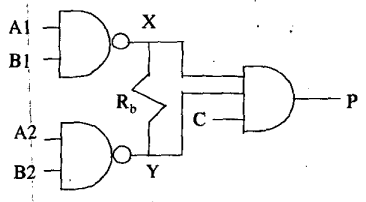

Figure 6. Bridging fault between nodes feeding the same gate.

5. Case 5: Bridge involving primary outputs

A primary output (PO) is assumed to be feeding a gate having a threshold of $V_{DD}/2$. Thus any bridge involving POs is treated as a case 2 or case 3 BF.

The fault simulator and ATPG developed in this work are based on this accurate fault model. By doing HSPICE simulations of all possible gate configurations for cases 2-5, we can accurately model the behavior of BFs, and insert the DRI obtained from the simulations at the fault site.

B. Fault Coverage Metric

Metal bridging resistance mainly falls in the range from 0Ω to 1000Ω [8]. This data can be described by a geometric distribution [22]. The cumulative distribution function of the bridging resistance is:

$$P(r \le R_b) = 1-(1-p)^{R_b} \quad (1)$$

where R_b is the bridging resistance and $p = 0.00258$ for the data in [8]. The normalized detection probability $c(i)$ for BF i is:

$$c(i) = \frac{P(R_{upper}(i)) - P(R_{lower}(i))}{1-(1-p)^{R_{max}(i)}} \quad (2)$$

where $R_{upper}(i)$ and $R_{lower}(i)$ are the upper and lower bounds respectively of the DRI. $R_{max}(i)$ is the maximum detectable resistance at the fault site under any sensitization or propagation. This value is updated as circuit constraints are determined. The fault coverage of a test vector v is given by:

$$C_v = \frac{1}{N}\sum_{i=1}^{N} c_v(i) \quad (3)$$

where $c_v(i)$ is the normalized detection probability for BF i using vector v and N is the total number of equally-likely logic-testable BFs in the circuit, which we refer to as *logic-testable bridging faults*. The fault coverage of a test set is given by:

$$C_c = \frac{1}{N}\sum_{i=1}^{N} c_h(i) \quad (4)$$

where $c_h(i)$ is the highest $c(i)$ achieved on the test set. Since $c_h(i)$ is normalized, C_c is the coverage of all BFs potentially detectable by low-speed voltage test.

C. Look-Up Table Construction

In this section we describe the procedure for building look-up tables based on the model described in Section II.A. First the logic threshold of each gate type is determined. In this work we use an output voltage of $V_{DD}/2$ to separate 0 and 1 values. Each gate input can have a different threshold, but to reduce our modeling effort we assume that all inputs have the same threshold, which is the one closest to the output node. Tables are constructed for cases 2-4. Case 5 faults are modeled as cases 2 and 3. The table entries contain the sensitizing vector, the propagation path, the logic threshold of the propagating gate, and the R_{max}, sorted in decreasing order of R_{max}.

For case 2 we built a table for each of the 22 gate types in the ISCAS85 benchmarks. For case 3, there are 253 combinations of bridged gates, many of which occur rarely. We generated tables for the 171 combinations involving gates having a fan-in of 5 or less.

For case 4 faults we must simulate a bridge between two gates, both feeding a third gate. For the driven gate, the inputs are chosen to be those closest to its output. The bridging resistance at which the output voltage changes from its faulty value to its fault-free value is defined as R_{max}. Since there are too many combinations of 3 gates, we generate tables only for those 20 combinations that occur in the ISCAS85 circuits. As in case 3, we do not model gates with a fan-in of more than 5. Another type of case 4 BF is one with two inputs of the driven gate being fed

from the same bridged node. This type of bridge is rare in the ISCAS85 circuits, and is not modeled.

Some case 4 BFs exhibit anomalous behavior in terms of the maximum detectable resistance. The circuit in Figure 7(a) has the behavior depicted in Figure 7(b) when simulated at 2V with vector {A1,B1,A2,B2} = {0,0,0,1}. Instead of a detectable resistance from 0Ω to R_{upper}, it lies in the DRI [R_{lower}, R_{upper}]. This behavior has been confirmed experimentally via focused-ion beam BF injection across XOR inputs in an actual chip [34]. For these cases, the entry in the table corresponding to the R_{upper} is replaced with a DRI. (In all other cases, R_{lower} is implicitly 0Ω).

The tables occupy about 3MB of space, and table construction time is several hours on a SPARC 4.

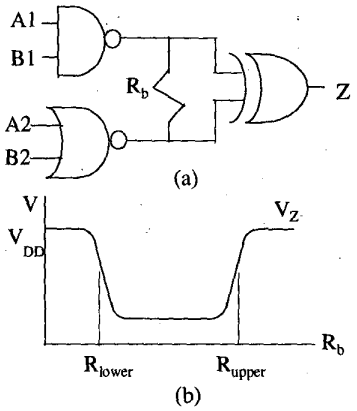

Figure 7. XOR gate with inputs bridged.

D. Fault Behavior At Decreased V_{DD}

Experience has shown that logic testing at decreased V_{DD} improves real fault coverage [27-31]. To do fault simulation and ATPG at different V_{DD} values, separate look-up tables have been built for each V_{DD} value.

Most prior work suggests that reduced V_{DD} will always improve fault coverage, except in rare and nonphysical cases. Our simulations show that some case 4 BFs in common circuit configurations are detectable at normal V_{DD} but undetectable at decreased V_{DD}. Figure 8 shows such a case, involving a NAND2 gate with bridged inputs.

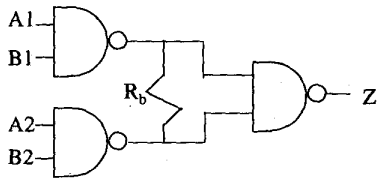

Figure 8. NAND2 gate with bridged inputs.

Table 1 shows the simulation results of this circuit for different test vectors that excite the fault, along with the R_{upper} at Z for two different V_{DD} values. We use 1.2V for low-voltage simulation because it is $2\times V_{tn}$, in the VLV range [35]. The BF resistance ranged from 0 to 6000Ω.

Table 1. Maximum detectable resistance vs. V_{DD} for Figure 8.

Test vector A1B1A2B2	R_{upper} at V_{DD}=3.3V	R_{upper} at V_{DD}=1.2V
0 0 1 1	2000Ω	> 6000Ω
1 1 0 0	1800Ω	> 6000Ω
1 0 1 1	1200Ω	undetectable
0 1 1 1	1200Ω	undetectable
1 1 0 1	1000Ω	undetectable
1 1 1 0	1000Ω	undetectable

The results show that at decreased V_{DD}, the BF is undetectable for some vectors, even though it is detectable at higher V_{DD}. But at low V_{DD} some vectors can detect a higher bridging resistance. The measured data in [36] illustrate this behavior.

This anomalous behavior at different values of V_{DD} can have varying impacts on overall fault coverage. As will be shown in Section V, if the circuit under test has several case 4 faults, and these faults exhibit the behavior described, then the overall fault coverage may drop at decreased V_{DD}.

III. Fault Simulation

Using the fault model described in the previous section, a prototype BF simulator has been built using the Tcl scripting language. The fault simulator uses single pattern single fault propagation. Implementing the BF simulator involves generating the fault list and implementing the fault simulation algorithm.

We randomly generate a set of BFs for fault simulation and ATPG. We chose to use random BFs rather than those extracted from a layout [21] in order to avoid biasing the results to a particular layout design, and to allow us to easily change the number of faults for different experiments. The number of BFs chosen is high enough to provide adequate fault dropping and fault coverage resolution. We refer to our BF set as the reduced fault list as compared to the all-pairs BF list. From the reduced BF list, case 1 and other unmodeled BFs are eliminated. We currently exclude feedback faults, but plan to analyze them in the future using techniques similar to those proposed for ZBFs.

The steps of the fault simulation algorithm are:

- For each BF, we determine R_{max} from the appropriate look-up table. For fault simulation R_{max} depends only on the gates connected to the bridged nodes.
- For each vector in the test set, fault-free logic simulation is performed and a list of excited faults is formed.
- For each excited fault, the look-up table is used to determine [R_{lower} R_{upper}] at the fault site. This DRI is then inserted at the outputs of the driven gates, provided other nodes feeding this gate are fault-free. Figure 9 shows three situations in which a DRI cannot be placed at the fault site, even though all conditions in the corresponding entry in the look-up table are met. (A thick line denotes the bridged node). In Figure 9(a), the fault-free node A

has a controlling value, so the fault is undetectable at the gate output. In Figure 9(b), the bridged node fans out and both branches feed into the same gate. We do not model this rare situation and so do not place the DRI at the gate output. In Figure 9(c), the gate input not involved in the bridging fault is fed from one of the bridged nodes, and hence is faulty too. No DRI is placed at the output of this gate.

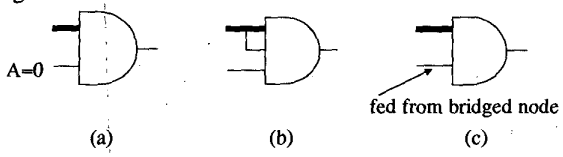

Figure 9. Cases in which interval is not placed at fault site.

- We simulate the faulty circuit using the approach in [12]. The DRI is placed at the output of the driven gates. For example, in Figure 10 the DRI [0,1600] is inserted at P and [0,1400] is inserted at Q. The faulty value at each faulty node is also inserted. Thus, in the figure, 0/1 at P indicates that within the DRI, the faulty logic value is 0 and outside the DRI the fault-free logic value is 1.

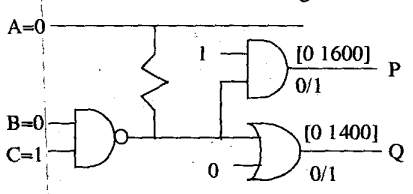

Figure 10. Inserting resistance interval at fault site.

- The DRIs are propagated towards the POs. Only those gates having inputs with DRIs are evaluated to determine the DRI at the gate output. During this forward simulation, the DRI can get reduced if two or more nodes carrying DRIs feed the same gate. The DRI at the output of such gates can be a union or an intersection of the DRIs at the gate inputs. There are three ways by which a gate has a DRI at its input that either disappears or shrinks at its output. The first occurs when the gate side input has a fault-free controlling value. Figure 11 shows the other two cases. In Figure 11(a) both gate inputs have different DRIs [0 R1] and [0 R2] (with R1 < R2), and the gate output has a DRI which is smaller than the DRI at either input. In Figure 11(b) the inputs have DRIs [0 R1] and [0 R2] (with R1 > R2) and the gate output has no DRI, making it fault-free.

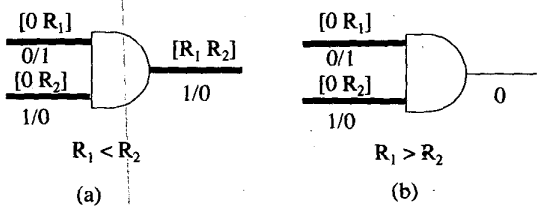

Figure 11. Cases of loss of coverage.

- Once the DRIs at the POs are known, the normalized detection probability $c(i)$ is computed using equation (2), taking the union of all DRIs over all POs. If $c(i)$ is 100%, the fault is dropped.
- The fault coverage C_v of the test vector is computed using equation (3).
- The above steps are repeated for each vector. For each fault, the best detection probability $c_h(i)$ obtained is noted. This is then used to compute the cumulative fault coverage C_c of the entire test set using equation (4).

If the BF resistance distribution is unknown, then the decision on dropping a fault can be made by examining the DRIs. If higher BF resistance is less probable, then a larger DRI implies better fault coverage.

The fault coverage metric is relative, in the sense that it is relative to what is possible independent of sensitization and propagation constraints. Hence the $c(i)$ for non-dropped faults, and therefore C_c, is a lower bound of the true fault coverage.

IV. Automatic Test Pattern Generation

The ATPG principles for logic testing of RBFs are similar to those of ATPG for single stuck-at faults. The primary difference is that for stuck-at faults, the *first* test vector that can satisfy the sensitization and propagation conditions is the required test vector. For RBFs, the search process is more complicated because it requires finding the *best* vector that can satisfy the sensitization and propagation conditions.

A. ATPG Approach

Our approach for ATPG is to generate a test vector for each BF that can detect the largest DRI. Consider the BF in Figure 12. If we want to generate a test vector that can detect the BF, there are several possible excitations and propagation paths:

(a) excite X=1, Y=0, propagate on X (through P or Q)
(b) excite X=1, Y=0, propagate on Y (through R or S)
(c) excite X=0, Y=1, propagate on X (through P or Q)
(d) excite X=0, Y=1, propagate on Y (through R or S)

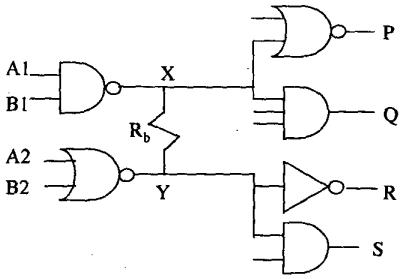

Figure 12. Test generation for BF between outputs of two gates.

Thus the test generation problem reduces to selecting logic values to be justified at {A1,B1,A2,B2} to sensitize the fault and selecting a path (X or Y) to propagate the fault. However, if we want to generate a test vector that can detect the largest DRI, then the selections must be made carefully:

- Case (a): Since the fault propagates on X, the logic 1 on X should be the *weakest* possible, and the logic 0 on Y should be the *strongest* possible, so that the 0 on Y overrides the 1 on X. In Figure 12, this can be achieved by justifying {A1,B1} = {0,1} or {1,0} and {A2,B2} = {1,1}. These sensitization values ensure the maximum DRI at the fault site. To propagate this DRI along X, we should choose the gate connected to X that has the *highest* logic threshold.
- Case (b): The 1 on X should be the strongest possible, and the 0 on Y should be the weakest possible. This can be achieved by justifying {A1,B1} = {0,0} and {A2,B2} = {0,1} or {1,0}. Since the fault is propagating along Y, we should choose the gate connected to Y that has the *lowest* logic threshold.
- Case (c): There is only one choice to excite the fault, {A1,B1,A2,B2} = {1,1,0,0}. To achieve maximum DRI, we should choose the gate connected to X that has the *lowest* logic threshold.
- Case (d): The sensitization condition is the same as in (c). To propagate the fault, we should choose the gate connected to Y that has the *highest* logic threshold.

Each excitation and propagation choice leads to a different value for maximum DRI. The look-up tables give the conditions necessary to detect the maximum DRI.

For a case 4 BF, as shown in Figure 13, there may be several choices for exciting the fault, but propagation can take place only along the output node P.

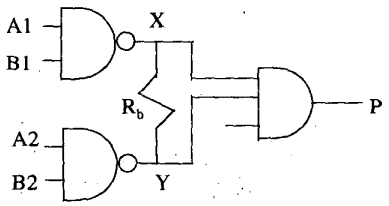

Figure 13. Test generation for a case 4 bridging fault.

B. ATPG Algorithm

The ATPG goal is to generate vectors that either result in 100% detection for each fault, or improves on the fault coverage obtained during fault simulation prior to ATPG.

For each target BF, we examine the look-up table associated with that type of BF. Starting with the first entry in the table (largest DRI), we try to justify the sensitization values at the inputs of the driving gate. If justification fails, we proceed to the next table entry (smaller DRI).

After justifying the sensitization values, we pick the propagation node indicated in the table entry. If the specified propagation node does not exist, we proceed to the next entry in the table.

The shortest path from the propagation node to a PO is then selected, and we try to justify the path. At each stage along this path, we check to see if the DRI has dropped from its value at the fault site, and if it has, we either backtrack or abort the present path. If it is not possible to propagate the DRI for the present entry in the table, we proceed to the next entry.

If the detection probability for the target BF is 100%, we drop the fault. Otherwise the BF remains on the fault list since due to aborts, a later vector may still achieve 100%. Fault simulation with fault dropping is then done with the generated vector.

C. ATPG Implementation

A prototype ATPG has been built on top of the BF simulator described in Section III, again using the Tcl scripting language. The PODEM algorithm [37] is used, with modifications during justification of excitation values and fault propagation.

1. Fault excitation

Fault excitation involves setting the inputs of bridged gates to the logic values in the look-up table entry. Before attempting to justify these values, the detection probability at the fault site DP_FS(i) for fault i (calculated from the DRI in the same entry of the look-up table in combination with the resistance distribution) is compared with the best detection Best_Det(i) already achieved by fault simulation. If DP_FS(i) is lower than Best_Det(i), then ATPG for this fault is terminated, because DP_FS(i) is the upper bound.

Justifying the nodes is done in a serial manner. The deepest node (the node that is furthest from the PIs) is attempted first, followed by the rest. If we fail to justify any node to its required logic value, then we move on to the next entry in the look-up table.

The look-up table entries are arranged in decreasing order of R_{upper}. Therefore, the first successful sensitization without having reached the backtrack limit on earlier sensitization attempts for this fault result in the maximum possible DRI at the fault site. This value may or may not be the same as the R_{max} for this fault determined prior to fault simulation. Figure 14 shows a simple case in which the R_{max} determined prior to fault simulation turns out to be higher than the R_{upper} determined during ATPG.

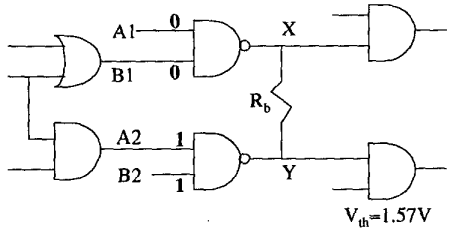

Figure 14. Bridging fault case in which R_{max} is lowered.

Prior to fault simulation, R_{max} was determined to be 1800Ω when propagating along Y through a gate with a logical threshold of 1.57V. The corresponding entry in the look-up table gives the sensitization at {A1,B1,A2,B2} to be {0,0,1,1}. However ATPG find that this cannot be justified, so a later table entry must be attempted, which may have a R_{upper} lower than 1800Ω. (If the backtrack limit was reached before the first successful attempt, R_{upper} is not modified). R_{max} and the detection probability already

obtained during fault simulation with an earlier vector are modified accordingly, and the fault is dropped if the probability is 100%. This situation also implies that the overall fault coverage obtained by fault simulation prior to ATPG is a lower bound on the actual fault coverage.

2. Fault propagation

During the fault propagation stage, we attempt to propagate the fault from the node specified in the entry in the look-up table towards a PO while minimizing a reduction in the DRI along the path. DRIs are inserted at the outputs of all driven gates and propagated along the shortest path. Logic values and DRIs are examined at each node along the path. If the DRI is not reduced from the value at the fault site, we proceed to the next node, justifying side inputs and backtracking as necessary.

Figure 15 shows a case in which in an attempt to justify a side input to a non-controlling value, we could only succeed in getting a faulty value at the node. The chosen propagation node is P2, and the propagation path is {P2, Z}. First, J is justified to 0, so that the DRI appears at P2. Then we must justify H to 1. To achieve this, we can either select P1 or G to be justified to 1. If we select P1, then by justifying F to 1, we will get a DRI on P1, thus making it a faulty node instead of a fault-free node. If this DRI is allowed to propagate, it may cause the DRI at Z to be smaller than the one at P2, thus causing loss of detection probability. We can avoid this problem by backtracking and selecting G instead of P1 to be justified to 1, which can be achieved by setting either D or E to 1.

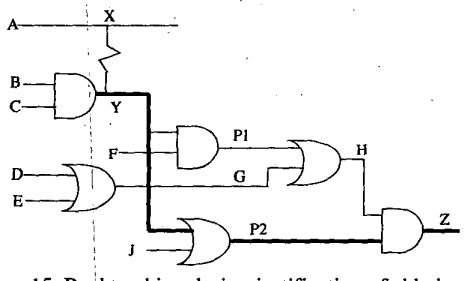

Figure 15. Backtracking during justification of side inputs.

Now consider Figure 16, which is a slight modification of Figure 15. We face the same situation as in Figure 15, the only difference being that when we backtrack and select G instead of P1 to justify H to 1, we discover that it is not possible to justify G to 1. We now have to revert back to justifying P1 to 1, even though this gives us a faulty value at P1. Therefore in such situations, before backtracking, we have to save the best solution we have achieved for the justification problem, even if it may lead to a reduction in the DRI.

During fault propagation, if a node on the propagation path has a DRI less than the DRI we are trying to propagate, we must backtrack. Figure 17 illustrates this case. The desired DRI [0 2k] appears on B, and the propagation path is {B,F,J}. We set A to 1, so that the DRI appears at F. If we set G and H to 1 by setting C to 1, this propagates [0 1k] on H and J, instead of the desired [0 2k]. To remedy this, we backtrack and set D to 1 and C to 0.

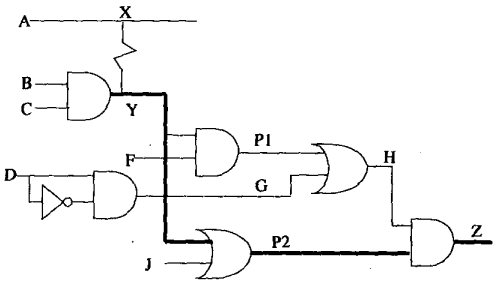

Figure 16. Saving best choice during justification of side inputs.

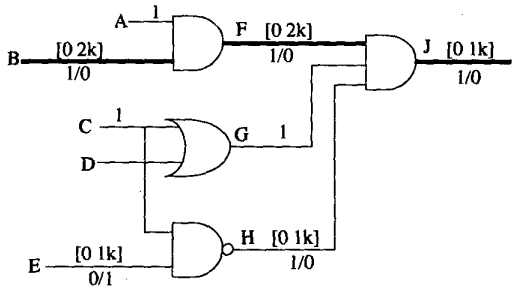

Figure 17. Suppressing an undesired resistance interval.

If the detection probability at any node along the propagation path falls below that already achieved for this fault, the path is dropped and the next shortest path from the faulty node to POs is chosen. We limit the number of propagation paths that are examined. If this limit is reached or if all propagation paths have been examined without propagating the fault-site DRI to the POs, then the next entry in the look-up table is chosen.

For the next entry in the look-up table, the fault-site detection probability DP_FS(i) for fault i for that entry is compared with the highest probability Best_Det(i) achieved for the fault. Since DP_FS(i) is an upper bound on the probability we can achieve for that entry, if it is lower than Best_Det(i), ATPG for this fault is stopped.

The detection probability obtained by ATPG for a fault may not be the highest obtainable for that fault, because of the limits set on backtracking during fault sensitization and fault propagation, and also the limit on the number of propagation paths examined. Therefore, if ATPG for a fault does not achieve 100% detection probability, the fault is not dropped, because a later test for some other fault may achieve a higher detection probability for this fault.

V. Results

A. Fault Simulation Results

The bridging fault simulator was run on the ISCAS85 benchmark circuits [33]. Table 2 gives some statistics of these circuits. Listed are the randomly selected faults, and then the PI, feedback, large fan-in, and unmodeled faults that are discarded, leaving the voltage-testable fault list.

The last column is the number of stuck-at test vectors, generated by the ATALANTA stuck-at fault ATPG [38].

Table 2. Statistics for ISCAS85 circuits used.

Circuit	Random BFs	PI BFs	Feed-back BFs	Large case BFs	Other dropped BFs	Logic testable BFs	Stuck-at vectors
c432	269	7	103	2	0	157	50
c499	191	8	47	0	1	135	53
c880	1086	27	110	0	1	948	48
c1355	1066	5	422	0	0	639	85
c1908	2206	7	521	16	0	1662	118
c2670	4572	110	168	0	0	4294	103
c3540	5315	4	780	100	0	4431	156
c5315	7407	38	237	11	0	7121	120
c6288	4492	1	1275	0	0	3216	34
c7552	12444	40	298	0	0	12106	204

Each circuit was simulated at V_{DD}=3.3V, 2.4V and 1.2V, and for different resistance distributions. The results of some simulations are shown in Figure 18 and Figure 19. Figure 18 shows the percentage of BFs completely detected (dropped) with V_{DD}=3.3V and V_{DD}=1.2V, and for RBFs using equation (1) and ZBFs. Figure 19 shows the fault coverage. The following observations can be made from Figure 18 and Figure 19:

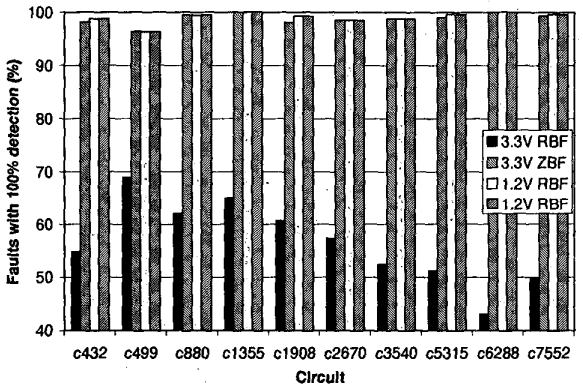

Figure 18. Faults dropped for each circuit.

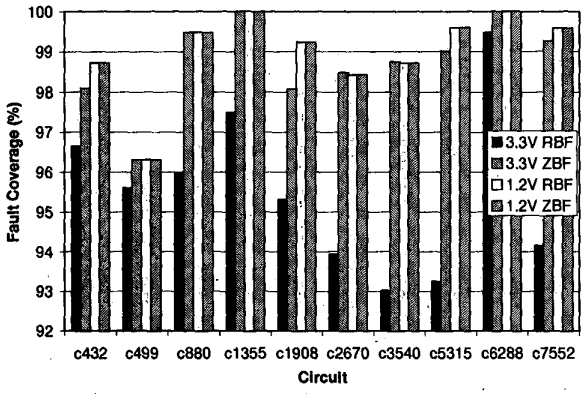

Figure 19. Fault coverage for each circuit.

- At 3.3V, the RBF fault coverage is high even though the percent dropped is low. This is because most remaining faults had a high detection probability, but not 100%.
- ZBF coverage and drop rate is higher than for RBFs.
- For RBFs, more faults are dropped and coverage is higher at 1.2V than 3.3V.
- For ZBFs, circuits c2670 and c3540 have *lower* fault coverage at 1.2V than at 3.3V. This is because the fault coverage is high at 3.3V, and at 1.2V the 2 BFs which escape cause the overall coverage to drop. This anomaly occurs only for these two circuits because they have a relatively high number of case 4 BFs.

B. ATPG Results

We applied our bridging fault ATPG to the ISCAS85 circuits. Since our Tcl implementation is several orders of magnitude slower than a C implementation, we experimented with only the small ISCAS85 benchmarks, and used low abort limits.

As discussed earlier, we experiment with both direct ATPG and ATPG used as clean-up following another test set. For this purpose we used three different test sets. The four different test sets generated were:

- *ATPG only*: Target each fault for test generation, and fault simulation is done with the generated vector.
- *Stuck-at fault simulation followed by ATPG (SA-ATPG)*: Apply the ATALANTA compacted single stuck-at test set prior to ATPG.
- *N-detect fault simulation followed by ATPG (NDET-ATPG)*: Apply an uncompacted 4-detect test set for fault simulation prior to ATPG.
- *Random fault simulation followed by ATPG (RND-ATPG)*: Apply a random test set equal in length to the 4-detect test set prior to ATPG.

Table 3 gives statistics for these test sets. Note that there are fewer ZBF ATPG vectors than RBF vectors due to the higher drop rate of ZBFs.

Table 3. Statistics for circuits used during ATPG.

Circuit	Logic-testable faults	RBF/ZBF ATPG only vectors	Stuck-at vectors	4-detect vectors	Random vectors
c432	1211	151/70	50	199	199
c499	1640	130/38	53	219	219
c880	2813	130/69	48	180	180

Figure 20 and Figure 21 shows the results of the four experiments performed on c432 for RBFs. Figure 20 shows the faults dropped and Figure 21 shows the fault coverage. Similar results were obtained for c499 and c880. The following observations can be made:

- The single stuck-at test achieved good RBF coverage. The longer (but uncompacted) N-detect test set achieved even better results. This suggests that intelligent ways of generating tests using the stuck-at fault model may be adequate to achieve high RBF coverage.

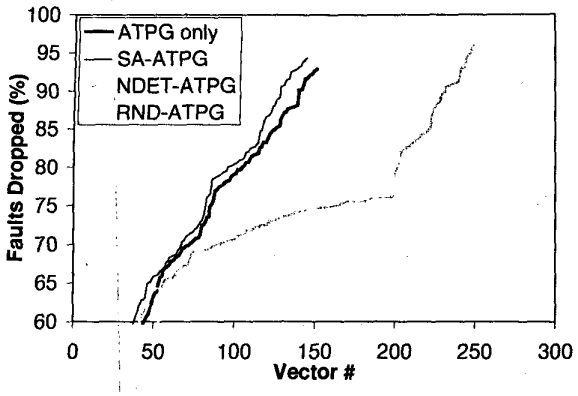

Figure 20. Faults dropped for c432 for resistive bridges.

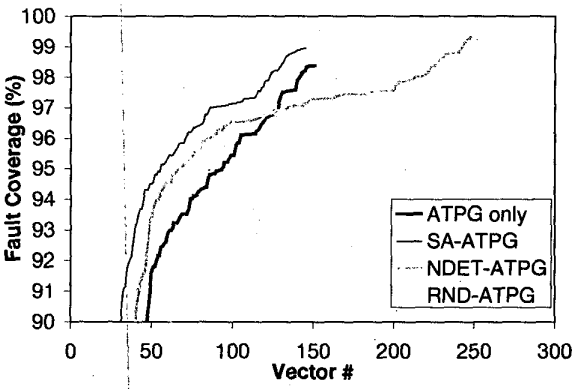

Figure 21. Fault coverage for c432 for resistive bridges.

- ATPG clean-up added 2-4% to the coverage, even after the 4-detect test. This suggests a clean-up targeting RBFs may be worth the cost and ATPG complexity.
- ATPG did better when used in a clean-up mode than when used by itself, due to the fact that the ATPG by itself reached its abort limits on more faults. This is primarily a function of the prototype implementation. We believe most of the unjustifiable cases can be quickly found with a learning-based search technique.
- The ATPG abort limits are too low. More than 4% of faults abort without achieving 100% detection. In C880 up to 30% of faults abort. A new implementation (now underway) is required to raise them to reasonable levels.
- Most fault coverage is lost due to undetected faults, not poorly detected faults. The majority of faults are dropped on their first detect. This suggests that rather than attempting to achieve the best detection probability on every fault, the ATPG time would be better spent by first targeting faults with low abort limits, and only increasing the limits on those few faults that abort on every table entry.
- The rate of fault dropping and improvement in fault coverage for ATPG only and SA-ATPG is higher than that for NDET-ATPG and RND-ATPG. This is because the stuck-at vectors are compacted. In the case of ATPG-only, the sharp rate of increase is because ATPG specifically targets each BF in the fault list and generates the best test for it. The N-detect and random test sets have many vectors that do not detect any BFs.
- In NDET-ATPG and RND-ATPG many faults are dropped when ATPG begins (at vector 200) because it can be proven that they had 100% detection during fault simulation (i.e. R_{max} is lowered to R_{upper}).
- ATPG in NDET-ATPG and RND-ATPG detects only the targeted fault. Fault simulation with the generated vector does not drop any faults. nor does it significantly improve the coverage of any faults.

C. Resistive vs. Zero-Ohm Bridging Faults

In order to determine the usefulness of a ZBF model, we compare the RBF coverage of test sets developed using the ZBF and RBF models. Figure 22 shows a comparison between the RBF coverage of SA-ATPG on c432 done using the two models. As can be seen, the coverage is similar for the stuck-at vectors (first 50 vectors). But for the ATPG vectors the ZBF model has lower coverage, and runs out of faults sooner due to its higher drop rate. Figure 23 shows the results of running ATPG only. In order to get a similar number of vectors for both models (145 for RBF, 150 for ZBF), several times more random ZBF faults were used than RBFs.

These results indicate that for a given test length, test vectors generated using the ZBF model will have significantly lower RBF coverage than vectors generated using the RBF model. Similar results were obtained for the c499 and c880 benchmarks.

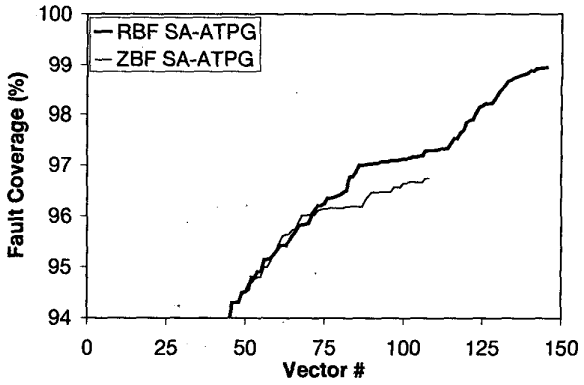

Figure 22. Resistive bridge fault coverage for SA-ATPG on c432 for resistive and zero-ohm fault models.

VI. Conclusions

We have developed an accurate resistive bridging fault model and used it in fault simulation and ATPG on the ISCAS85 circuits. Our fault simulation results show that stuck-at tests achieve relatively high RBF fault coverage. They also confirm that testing at reduced V_{DD} increases overall RBF coverage. Certain situations have been identified where reducing V_{DD} causes a loss in coverage.

ATPG results show that test vector generation targeting RBFs improves on the coverage obtained from fault simulation with stuck-at or random fault test sets. However the best results were obtained when ATPG is used as a clean up for a stuck-at test set. This is likely due to our simulator aborts, but given the costs of RBF ATPG, this combination of stuck-at and RBF ATPG is likely to result in the highest coverage at the lowest test generation cost.

Fault simulation and ATPG results show that a test set generated using the ZBF model does not perform as well as a test set of the same size generated using RBF.

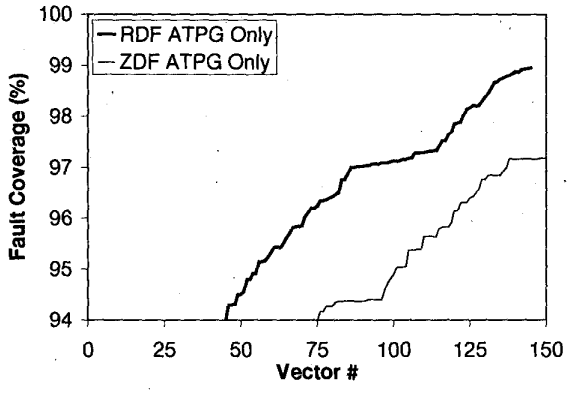

Figure 23. Resistive bridge fault coverage for ATPG only on c432 for resistive and zero-ohm bridging fault models.

Acknowledgements

We would like to thank Ray Mercer and his students for the N-detect test sets.

References

[1] S. Sengupta et al, "Defect-Based Test: A Key Enabler for Successful Migration to Structural Test", Intel Technical Journal, Q1,1999.

[2] W. Maly, "Realistic Fault Modeling for VLSI Testing", *ACM/IEEE 24th Design Automation Conf.*, June 1987, pp. 173-180.

[3] T. M. Storey and W. Maly, "CMOS Bridging Fault Detection," *Int. Test Conf.*, 1990, pp. 842-851.

[4] T. M. Storey, W. Maly, J. Andrews, and M. Miske, "Stuck Fault and Current Testing Comparison Using CMOS Chip Test," *Proc. Int. Test Conf.*, 1991, pp. 311-318.

[5] L-C. Wang, M. R. Mercer, and T. W. Williams, "Using Target Faults to Detect Non-Target Defects," *Int. Test Conf.*, 1996, pp. 629-638.

[6] S. C. Ma, P. Franco, and E. J. McCluskey, "An Experimental Chip to Evaluate Test Techniques Experiment Results," *Proc. Int. Test Conf.*, 1995, pp. 663-672.

[7] J. Patel et al, "Stuck-At Fault: The Fault Model of Choice for the Third Millennium!?", *Int'l Test Conf.*, Oct. 1998, pp. 1164-1167.

[8] R. Rodriguez-Montanes, E. Bruls, and J. Figueras, "Bridging Defect Resistance Measurements in a CMOS Process," *Int. Test Conf.*, 1992, pp. 892-899.

[9] M. Renovell, P. Huc, and Y. Bertrand, "CMOS Bridging Fault Modeling," *VLSI Test Symp.*, 1994, pp. 392-397.

[10] M. Renovell, P. Huc, and Y. Bertrand, "The Concept of Resistance Interval: A New Parametric Model for Realistic Resistive Bridging Fault," *VLSI Test Symp.*, 1995, pp. 184-189.

[11] J. J. T. Sousa, F. M. Goncalves, and J. P. Teixeira, "IC Defects-Based Testability Analysis," *Int. Test Conf.*, 1991, pp. 500-509.

[12] V. Sar-Dessai and D. M. H. Walker, "Accurate Fault Modeling and Fault Simulation of Resistive Bridges," *Int. Symp. Defect and Fault Tolerance in VLSI Systems*, 1998, pp. 102-107.

[13] J. Rearick and J. H. Patel, "Fast and Accurate Bridging Fault Simulation," *Int. Test Conf.*, 1993, pp. 54-62.

[14] J. M. Acken and S. D. Millman, "Fault Model Evolution for Diagnosis: Accuracy vs. Precision," *IEEE Custom Int. Circ. Conf.*, 1992, pp. 13.4.1-13.4.4.

[15] M. Abramovici and P. R. Menon, "A Practical Approach to Fault Simulation and Test Generation for Bridging Faults," IEEE Trans. Computers, vol. 34, no.7 pp. 658-662, July 1985.

[16] G. Freeman, "Development of Logic Level CMOS Bridging Fault Models,", Center for Reliable Computing Technical Report 86-10, Stanford University, 1986.

[17] J. M. Acken, "Deriving Accurate Fault Models,", CSL-TR-88-365, Computer Systems Laboratory, Stanford University, Oct. 1988.

[18] Chennian Di and Jochen A. G. Jess, "An Efficient CMOS Bridging Fault Simulator: With SPICE Accuracy," IEEE Trans. Computer-Aided Design, vol. 15, no. 9, pp.1071-1080, Sept. 1996.

[19] S. D. Millman and J. P. Garvey, Sr., "An Accurate Bridging Fault Test Pattern Generator," *Proc. Int. Test Conf.*, 1991, pp. 411-418.

[20] J. M. Acken and S. D. Millman, "Accurate Modeling and Simulation of Bridging Faults,". *Cust. Int. Circ. Conf.*, 1991, pp. 17.4.4-17.4.4.

[21] F. J. Ferguson and T. Larrabee, "Test Pattern Generation for Realistic Bridge Faults in CMOS ICs," *Int. Test Conf.*, 1991, pp. 492-499.

[22] Y. Liao and D. M. H. Walker, "Optimal Voltage Testing for Physically-Based Faults," *VLSI Test Symp.*, 1996, pp. 344-353.

[23] P. C. Maxwell and R. C. Aitken, "Biased Voting: A Method for Simulating CMOS Bridging Faults in the Presence of Variable Gate Logic Thresholds," *Proc Int. Test Conf.*, 1993, pp. 63-72.

[24] M. Dalpasso, M. Favalli, P. Olivio, and B. Ricco, "Parametric Bridging Fault Characterization for the Fault Simulation of Library-Based ICs," *Proc. Int. Test Conf.*, 1992, pp. 486-495.

[25] F. Peters and S. Oostdijk, "Realistic Defect Coverages of Voltage and Current Tests," *Int. Workshop on IDDQ Testing*, 1996, pp. 4-8.

[26] B. Chess and T. Larrabee, "Logic Testing of Bridging Faults in CMOS Integrated Circuits," IEEE Trans. Computers, vol. 47, no. 3, pp. 338-345, March 1988.

[27] Y. Liao and D. M. H. Walker, "Fault Coverage Analysis for Physically-Based CMOS Bridging Faults at Different Power Supply Voltages," *Proc. Int. Test Conf.*, 1996, pp. 767-775.

[28] H. Hao and E. J. McCluskey, "Very-Low-Voltage Testing for Weak CMOS Logic ICs," *Proc. Int. Test Conf.*, 1993, pp. 275-284.

[29] J. T.-Y. Chang and E. J. McCluskey, "Quantitative Analysis of Very-Low Voltage Testing," *Proc. VLSI Test Symp.*, 1996, pp. 332-337.

[30] M. Renovell, P. Huc, and Y. Bertrand, "Bridging Fault Coverage Improvement by Power Supply Control," *Proc. VLSI Test Symp.*, 1996, pp. 338-343.

[31] J. T. -Y. Chang, C.-W. Tseng, Y.-C. Chu, S. Wattal, M. Purtell, and E. J. McCluskey, "Experimental Results for IDDQ and VLV Testing," *Proc. VLSI Test Symp.*, 1998, pp. 118-123.

[32] HSPICE Manual, Campbell, CA: Meta-Software Inc. 1990.

[33] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinatorial Benchmark Circuits and a Target Translator in FORTRAN," *Proc. Int. Symp. On Circuits and Systems*, 1985, pp. 663-698.

[34] H. Balachandran, Private Communication, 1998.

[35] J. T.-Y. Chang and E. J. McCluskey, "Detecting Delay Flaws by Very-Low-Voltage Testing," *Int. Test Conf.*, 1996, pp. 367-376.

[36] S. C. Ma, P. Franco, and E. J. McCluskey, "An Experimental Chip to Evaluate Test Techniques Experiment Results," *Proc. Int. Test Conf.*, 1995, pp. 663-672.

[37] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," IEEE Trans. Computers, vol. C-30, no. 3, pp. 215-222, March 1981.

[38] H. K. Lee and D. S. Ha, "On the Generation of Test Patterns for Combinational Circuits," Technical Report No. 12_93, Dept. of Electrical Eng., Virginia Polytechnic Institute and State University.