Resistive Bridge Fault Modeling, Simulation and Test Generation[1]

Vijay R. Sar-Dessai
Intel Corporation, FM5-64
1900 Prairie City Road
Folsom CA 95630
Tel: (916) 356-1759
Fax: (916) (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

Resistive bridging faults in combinational CMOS circuits are studied in this work. Circuit-level models are abstracted to voltage behavior for use in voltage-level fault simulation and test generation. Fault simulation is done using different test sets in order to study their effectiveness. Test generation is done to detect the highest possible bridging resistance for each fault. Different test sets, power supply voltages, and fault models are studied on the ISCAS85 benchmark circuits.

I. Introduction

Shorts between circuit nodes are the predominant type of manufacturing defect [1]. These shorts can be of two types: *intra-gate shorts* between nodes within a logic gate and *inter-gate* or *external shorts* between outputs of different logic gates [2][3]. Inter-gate shorts, or *bridging faults*, account for about 90% of all shorts [3][4]. Thus in order to accurately estimate the quality of a chip, it is important to have a fault simulator for realistic bridging faults. It is also important to generate a test vector set that can achieve high fault coverage for bridging faults.

The *accuracy* of a bridging fault simulator and automatic test pattern generator (ATPG) strongly depends on the accuracy of the bridging fault model [5]. A bridging fault model should not only consider the behavior of the driving gates, but should also include the driven gate behavior. This is because the logical interpretation of the voltage at the bridged nodes depends on the logical threshold of the gate to which the bridged node is connected. In reality, not only do different gates have different thresholds, but each input of a gate has a different threshold [6][7].

It is well known that the stuck-at fault model is inadequate for modeling bridging faults [8][9]. Many models have been developed for bridging faults [6][10][11] [12][13][14][15][16]. Most of these fault models assume a zero ohm bridge resistance, but several models assume a resistive bridge [3][17][18][19][20]. As shown in [1], many bridges can have significant resistance. Figure 1 shows the bridging resistance distribution fit to the data in [1]. R_b is the bridging resistance and $P(R_b)$ is the bridging resistance distribution function.

Since a test for a zero-ohm bridge does not guarantee detection of a resistive bridge, ideally the fault model should be a resistive bridging fault model, instead of a zero-ohm bridging fault model. As noted in [21], since the resistive bridging fault model is only an approximation of defect behavior, we may need to use several different fault models to achieve high defect coverage. In this work we consider logic testing for resistive bridging faults, as compared to prior work on zero-ohm bridges [22].

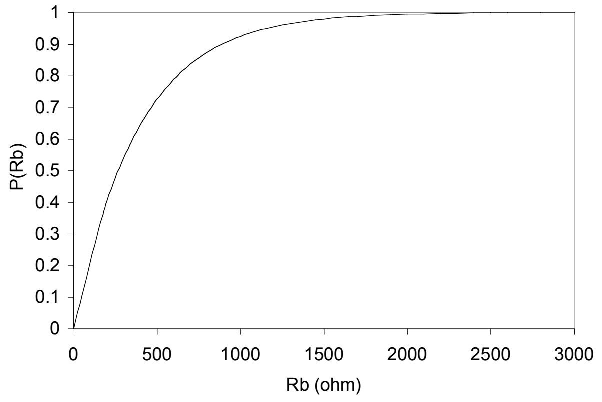


Figure 1. Bridging resistance distribution function.

It is well known that as the power supply voltage V_{DD} is decreased, higher bridging resistances are detected, or faults which escape detection at a higher value of V_{DD} are detected when V_{DD} is dropped [23][24][25][26][27]. Hence it is useful to do logic testing at several V_{DD} values.

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

II. Fault Model

To accurately model the behavior of bridging faults, we must determine the voltage at the bridged nodes for each vector that excites the bridging fault. Then, based on the logic threshold of the driven gates, we can determine whether the bridge is detectable at the driven gate output. We can also determine the maximum detectable resistance at the output of this gate, which gives us the detectable resistance interval. (A detectable resistance interval is the range of bridging fault resistance that can be detected).

[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.

Figure 2 shows a bridging resistance R_b between nodes X and Y. The fault is excited with logic 1 on node X and logic 0 on node Y (or vice-versa). The bridging fault is essentially a resistance between V_{DD} and GND, via pull-up devices in gate g1 and pull-down devices in gate g2. Voltages 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.

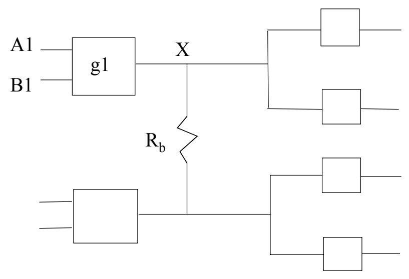


Figure 2. Resistive bridging fault.

Figure 3 shows the typical variation of voltages V_X and V_Y (for the excitation {X,Y} = {1,0}) as the bridging resistance increases from 0Ω. The interpretation of the voltages at nodes X and Y depends on the logic threshold of gates fed by these nodes. When the bridge has a 0Ω resistance, the voltages V_X and V_Y are equal to V_0. As the bridge resistance R_b increases, V_X increases from V_0 to V_{DD} and V_Y decreases from V_0 to 0. If any gate fed by node X had a logic threshold between V_0 and V_{DD}, the fault will be detected at the output of that gate, assuming that other inputs of this gate are at their non-controlling values. Likewise, if any gate fed by node Y has a logic threshold between V_0 and 0, the fault will be detected at the output of this gate. In both cases, the detectable resistance depends on the value of the logic threshold. As shown in Figure 3, the detectable resistance is R_{dp} at node P and R_{ds} at node S. (In the figure, 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 detectable resistance interval is [0 R_{dp}] at node P and [0 R_{ds}] at node S. The fault is undetectable at nodes Q and R.

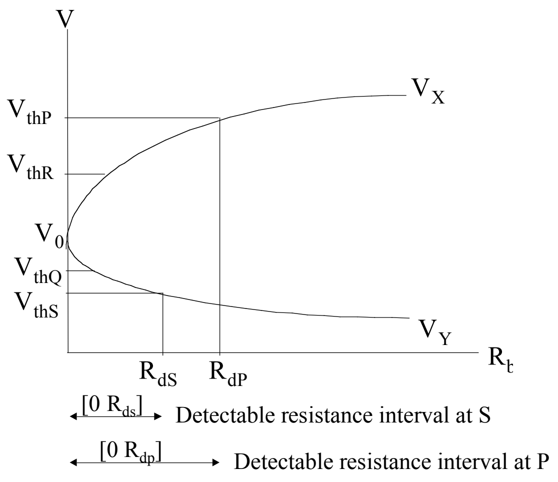


Figure 3. Variation of bridged node voltage with bridge resistance.

A. Description of Fault Model

In this work, bridging faults have been modeled by HSPICE [28] circuit simulation of almost all possible bridging fault configurations for all gates included in the gate-level description of the ISCAS85 benchmark circuits [29]. Each circuit is built using basic gates, and no complex gates are used. Each gate is implemented using complementary CMOS logic. We use the SPICE level 3 parameters for the HP CMOS14TB 0.5 μm process, running at a nominal V_{DD} of 3.3V. For the devices in this process, V_{tn}, the threshold of the n-device is 0.6684V and V_{tp}, the threshold of the p-device is -0.9352V. By simulating the different types of bridging faults that can occur in various combinations of these gates, we obtain a set of look-up tables that describe the logic-level behavior of the fault site during fault simulation and test generation.

1. Case 1: Bridge between two primary inputs

We define primary inputs (PIs) as sources of infinite current, so bridging faults between them are not logic testable. Hence this type of bridging fault is not modeled.

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

Figure 4 shows a bridging fault 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.

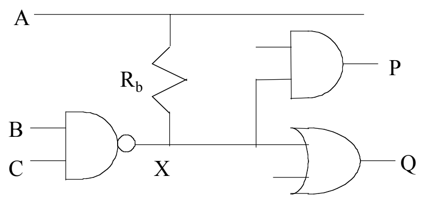


Figure 4. Bridging fault between PI and gate output.

For example, HSPICE simulation shows that if the applied vector is {A,B,C} = {0,0,1}, then we can detect a bridging resistance up to 1600Ω at node P and up to 1400Ω at node Q, assuming that other inputs of the AND2 and OR2 gates are held at their non-controlling values. The fault does not propagate along A. (In the simulation, the bridged node in the driven gate is the input node closest to the output of the driven gate).

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

Figure 5 illustrates a case in which the outputs of a NAND2 and a NOR2 gate are bridged, and the bridged nodes X and Y feed into different gates. The bridge resistance detectable at the outputs depends on the vector at A1, B1, A2, B2 and the logic thresholds of the driven gates.

For vector {A1,B1,A2,B2} = {1,0,1,1}, the bridging fault will propagate along node X, and the resistance detectable is up to 1000Ω at P and up to 1400Ω at Q. Due to the vector used and the thresholds of the NOT and the AND2 gates, the fault does not propagate along node Y, and is undetectable at nodes R and S.

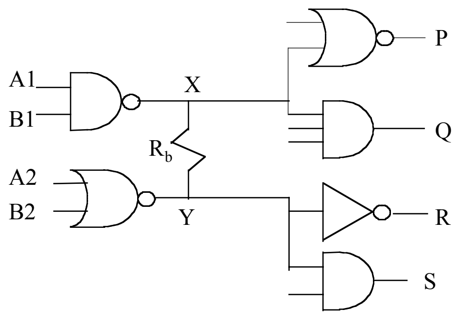


Figure 5. Bridging fault between nodes feeding different gates.

4. Case 4: Bridge between outputs of two gates (bridged nodes feeding into same gate)

Figure 6 illustrates the case in which the outputs of a NAND2 and NAND2 gate are bridged, and the bridged nodes feed into the same AND3 gate. (For the driven gate, the bridged inputs are the inputs that are closest to the gate output). The bridge resistance detectable at node P depends only on the vector at A1, B1, A2, B2 (assuming that the third input of the AND3 gate is at its non-controlling value). With a vector of {A1,B1,A2,B2} = {1,0,1,1}, HSPICE simulation of this circuit shows that a bridge resistance of up to 800Ω is detectable at node P.

5. Case 5: Bridge involving primary outputs

A primary output is assumed to be feeding a gate having a threshold of $V_{DD}/2$. Thus any bridge involving primary outputs can be classified as a case 2 or bridging fault.

The fault simulator and ATPG developed in this work is based on this accurate fault model. By doing HSPICE simulations of all possible gate combinations in all the above cases, we can accurately model the behavior of bridging faults, and insert the detectable resistance interval obtained from the simulations at the fault site.

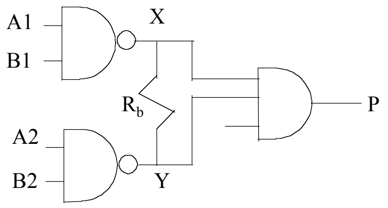


Figure 6. Bridging fault between nodes feeding the same gate.

B. Fault Coverage Metric

Metal bridging resistance mainly falls in the range from 0Ω to 1000Ω [1]. A geometric distribution was found to be a good fit to the data [17]. The cumulative distribution function (CDF) of the bridging resistance is:

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

where R_b is the bridging resistance and p = 0.00258 for the data in [1]. The normalized fault coverage $c(i)$ for the bridging fault configuration i can be computed using:

$$c(i) = \frac{P(R_{b\_upper}(i) - P(R_{b\_lower}(i))}{1-(1-p)^{R_{b\max}(i)}} \qquad (2)$$

where $R_{b\_upper}(i)$ and $R_{b\_lower}(i)$ are the upper and lower bounds respectively of the detectable resistance interval. $R_{b\max}(i)$ is the maximum detectable resistance at the fault site under any sensitization or propagation. The fault coverage of a test vector v is given by:

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

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

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

where $c_h(i)$ is the highest achieved normalized fault coverage for the bridging fault i. Since $c_h(i)$ is normalized, C_c is the coverage of all bridging faults potentially detectable by low-speed voltage test.

C. Construction Of Look-Up Tables

In order to obtain information about the behavior of the circuit at the fault site for fault simulation, a number of look-up tables must be built. First the logic threshold of each gate type in the ISCAS85 circuits is determined (we assume that for a given gate, all inputs have the same logic threshold, which is the threshold value of the input node closest to the output node). Look-up tables are constructed for case 2, case 3 and case 4 bridging faults. Case 5 faults are included under cases 2 and 3. The table entries containing the sensitizing vector, the propagation path, the logic threshold of the propagating gate, and the maximum detectable resistance.

For case 2 we built one table for each of the 22 gate types in the ISCAS85 benchmarks. For case 3, there are 253 combinations of bridged gates in the ISCAS85 circuits, many of which occur rarely. We generate 171 tables for all combinations of 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 bridged nodes are its inputs that are closest to its output. The bridging resistance at which the output voltage changes from its faulty value to its fault-free value is determined to be the maximum detectable bridging resistance. (We use $V_{DD}/2$ to separate faulty and fault-free values). 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 bridging fault falling under case 4 is one with two inputs of the driven gate being fed from the bridged nodes.. This type of bridge is rare in the ISCAS85 circuits, and is not modeled.

Some case 4 bridging faults exhibit anomalous behavior in terms of the maximum detectable resistance. An example is the circuit in Figure 7(a), which has the behavior depicted in Figure 7(b) when simulated at low voltage (2V) with a vector {A1,B1,A2,B2} = {0,0,0,1}.

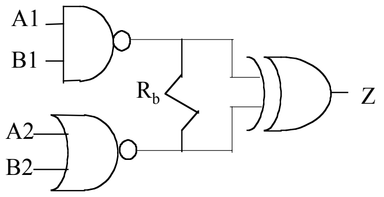
Instead of the detectable resistance being in the interval of 0Ω to the maximum detectable resistance R_{bmax}, it lies in the interval [R_{bmin}, R_{bmax}]. During table construction, the entry in the table corresponding to the maximum detectable resistance is replaced with a resistance interval. (In all other cases, it is implicitly assumed that the lower limit of detectable resistance is 0Ω).

The tables occupy about 3MB of space, and table construction time is considerable.

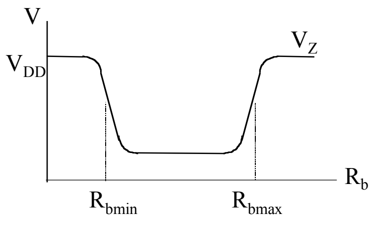
D. Fault Behavior At Decreased V_{DD}

Experience has shown that logic testing at decreased power supply voltage V_{DD} improves real fault coverage [23-27]. To do fault simulation and ATPG at different V_{DD} values using our modeling approach, separate look-up tables have been built for each V_{DD} value.

Most of this prior work suggests that reduced V_{DD} will always improve fault coverage, except in rare cases. Our simulation results show that there exist some instances of bridging faults in common circuit configurations in which the fault is detectable at a higher V_{DD} value but undetectable at decreased V_{DD} values. All these instances are case 4 type bridging. Figure 8 shows such a case, involving a NAND2 gate having bridged inputs.



(a)



(b)

Figure 7. XOR gate with inputs bridged.

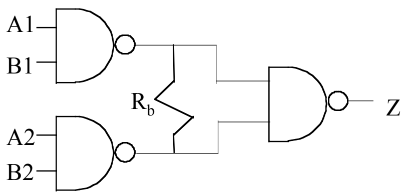


Figure 8. NAND2 gate with inputs bridged.

Table 1 shows the result of HSPICE simulation of this circuit for different test vectors that excite the fault, along with the maximum detectable resistance R_{bmax} at node Z for two different values of V_{DD}. We have chosen 1.2V for low-voltage simulation because it is 2←V_{tn}, (where V_{tn} is the device threshold of the NMOS device), in the VLV range [30]. (An X in the table means that the fault is undetectable). For this circuit, simulation was done for bridging resistance from 0 to 6000Ω.

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

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

The results indicate that at decreased V_{DD}, the bridging fault 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 [31] illustrate this behavior.

This anomalous behavior at different values of V_{DD} can have varying impacts on overall fault coverage. 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}. If these cases are relatively rare, then the fault coverage will improve with decreased V_{DD}.

III. Fault Simulation

Using the fault model described in the previous section, a fault simulator has been built. The fault simulator is a Single Pattern Single Fault Propagation (SPSFP) fault simulator. Implementing the bridging fault simulator involves generating the fault list and implementing the fault simulation algorithm.

We randomly choose a fraction of the all-pairs external bridges for fault simulation and ATPG. The number chosen is high enough to provide adequate fault dropping and fault coverage resolution. From the reduced fault list, bridging faults between primary inputs are eliminated, because logic test cannot detect them. Feedback faults are also discarded from the fault list. Unmodeled bridging faults are dropped.

The steps during fault simulation are as follows:

- For each fault in the set of logic-testable faults, we determine the maximum detectable resistance from the look-up table associated with that bridge. This maximum detectable bridging resistance depends only on the gates whose output nodes are bridged and the gates fed by the bridged nodes. Since this resistance is the resistance detectable at the fault site under the best possible excitation and propagation conditions, it is an upper bound on the coverage we can achieve for this fault.
- For each vector in the test set, fault-free logic simulation is performed. A list of excited faults is formed from the logic-testable fault list.
- For each excited fault, the look-up table is used to determine the bridging resistance detectable at the fault site. This resistance 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 detectable resistance cannot be placed at the fault site, even though all conditions in the corresponding entry in the look-up table are met. (The bridged node is represented by a thick line). 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. This situation is relatively rare in our implementation of the benchmarks and is not modeled, so we do not place the resistance at the output of the gate. 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 resistance interval is placed at the output of this gate.

- We follow the approach used in [3] to simulate the faulty circuit. In this convention, the resistance interval, which specifies the range of resistances that can be detected by that test vector, is placed at the output of the gate fed by the bridged node. For example, for the case of a primary input bridged to any other node illustrated in Figure 10, the interval [0,1600] is inserted at node P and the interval [0,1400] is inserted at node Q. The faulty value at each faulty node is also inserted. Thus, in the figure, 0/1 at node P indicates that within the specified resistance interval, the logic value is 0 (faulty value) and outside this interval the logic value is 1 (fault-free value).

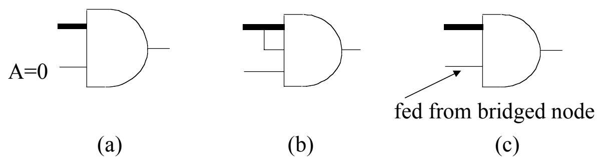


Figure 9. Cases in which resistance interval is not placed at fault site.

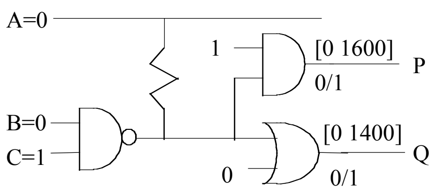


Figure 10. Inserting resistance interval at fault site.

- Fault simulation continues with propagation of the resistance interval from the fault site towards the primary outputs. Only those gates having input nodes with resistance intervals on them are evaluated to determine the resistance interval at the output of the gate. During this forward simulation, the resistance interval can get reduced if two or more nodes carrying resistance intervals feed into the same gate. The resistance interval at the output of such gates can be a union or an intersection of the intervals at the inputs of the gate. There are three ways by which a gate has a resistance interval at its input that either disappears or shrinks at the gate 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 the gate inputs have different resistance intervals [0 R1] and [0 R2] (with R1 < R2) associated with them, and the output of the gate had a resistance interval which is smaller than the interval at either input. In Figure 11(b) the two inputs have resistance intervals R1 and R2 (with R1 > R2) and the gate output has no resistance interval, making it fault-free.

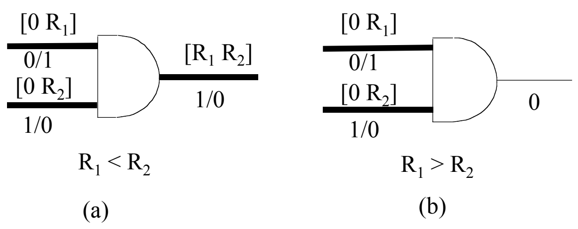


Figure 11. Cases of loss of coverage.

- Once the resistance intervals at the primary outputs are known, the normalized fault coverage *c(i)* is computed using equation (2), taking the union of all resistance intervals over all primary outputs. If *c(i)* is above our coverage threshold (100% here), the fault is dropped.
- The fault coverage C_v of this test vector is then computed using equation (3).
- The above procedure is repeated for each vector. For each fault, the best fault coverage obtained so far is noted. This is then used to compute the cumulative fault coverage C_c of the entire test vector set using equation (4).

If the bridging resistance distribution is unknown, then the decision on dropping a fault can be made by examining the detectable resistance intervals. If higher resistances are less probable, then a higher detectable resistance implies better fault coverage.

The fault coverage metric calculated is relative, in the sense that it is a ratio of achieved coverage to the maximum possible coverage. The maximum possible coverage is the coverage at the fault site. This coverage may be impossible to obtain, because sensitization or propagation constraints. Hence the coverage for non-dropped faults, and therefore overall coverage, is a lower bound of the true coverage.

IV. Automatic Test Pattern Generation

The ATPG principles for logic testing of bridging faults 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 resistive bridging faults, the search process is more complicated because it involves 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 bridging fault that can detect the highest possible bridging resistance. Consider the bridging fault in Figure 12. If we want to generate a test vector that can detect the bridging resistance R_b, 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)

Thus the test generation problem reduces to selecting logic values to be justified at A1, B1, A2 and B2 (to sensitize the fault) and selecting a propagation path (X or Y) to propagate the fault. However, if we want to generate a test vector that can detect *the maximum possible* value of R_b, then we have to make our selections carefully:

- excite {X,Y} to {1,0} and propagate along X: Since the fault is propagating along 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 detectable resistance at the fault site. To propagate this resistance to a primary output along X, we should choose that gate connected to X which has the *highest* logic threshold.

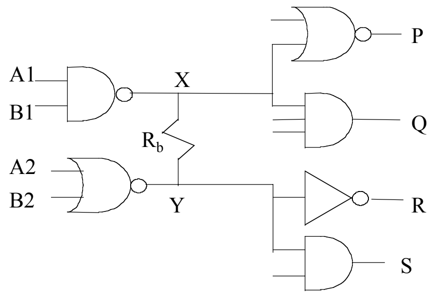


Figure 12. Test generation for bridging fault between outputs of two gates.

- excite {X,Y} to {1,0} and propagate along Y: The logic 1 on X should be the strongest possible, and the logic 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 which has the *lowest* logic threshold.
- excite X,Y to {0,1} and propagate along X: There is only one choice to excite the fault, which is {A1,B1,A2,B2} = {1,1,0,0}. However to ensure maximum detectable resistance, we should choose the gate connected to X which has the *lowest* logic threshold.
- excite X,Y to {0,1} and propagate along Y: The sensitization condition is the same as in (c) above. However, to propagate the fault, we should choose the gate connected to Y which has the *highest* logic threshold.

Each excitation and propagation choice leads to a different value for maximum detectable bridging resistance. The look-up tables constructed during pre-processing give the conditions necessary to detect the maximum detectable bridging resistance.

If the bridged nodes feed the same gate, as shown in Figure 13, then there may be several choices for exciting the fault, but propagation can take place only along the output node P.

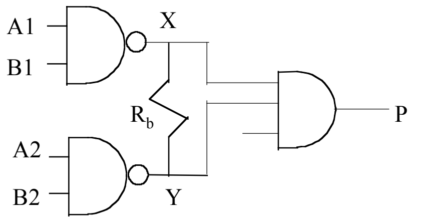


Figure 13. Test generation for bridging fault between nodes feeding same gate.

B. ATPG Algorithm

The goal during logic ATPG is to generate a test vector that either results in 100% fault coverage for each fault, or improves on the fault coverage already obtained during fault simulation prior to ATPG.

For the target fault, we examine the look-up table associated with that type of bridging fault. Starting with the first entry in the table, we try to justify the sensitization values at the inputs of the driving. If justification is not possible, we proceed to the next entry in the table.

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 primary output is then selected, and we try to justify the path. At each stage along this path, we check to see if the detectable resistance 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 fault for the present entry in the table, we proceed to the next entry.

If the fault coverage for the target fault is within the drop limit (100% here) we drop the fault. Fault simulation with fault dropping is then done with this vector. If the fault coverage for the target fault is less than 100%, the fault remains in the fault list, because a later test vector may achieve a higher fault coverage.

C. ATPG Implementation

The ATPG tool has been built on top of the fault simulator described in the previous section. The PODEM [32] algorithm is used, with modifications during justification of excitation values and propagation of the fault towards a primary output.

1. Fault excitation

The fault excitation stage involves setting the inputs of gates whose outputs are bridged to the logic values in the look-up table entry. Before attempting to justify these node values, the fault coverage at the fault site FC_FS(*i*) for fault (*i*) (calculated from the detectable resistance value in the same entry of the look-up table) is compared with the best coverage Best_Cov(*i*) already achieved by fault simulation. If FC_FS(*i*) for this entry is lower than Best_Cov(*i*), then the ATPG attempt for this fault is terminated, because FC_FS(*i*) is the upper bound on the coverage.

Justifying the nodes is done in a serial manner. The deepest node (the node that is furthest away from 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 detectable resistance. Therefore, the first successful sensitization without having reached the backtrack limit on earlier attempts at sensitization of this fault should give the maximum detectable resistance at the fault site. This value may or may not be the same as the maximum detectable resistance for this fault determined prior to fault simulation. Figure 14 shows a simple case in which the maximum detectable resistance determined prior to fault simulation turns out to be higher than the maximum detectable resistance determined during ATPG.

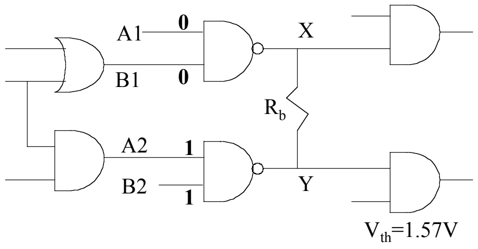


Figure 14. Bridging fault case in which R_{bmax} is lowered.

Prior to fault simulation, the maximum detectable resistance 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, during ATPG, it is discovered that this sensitization cannot be justified. When we finally get a successful sensitization the first time without reaching the backtrack limit for any previous sensitization, the resistance at that entry of the look-up table is the actual maximum detectable resistance, and may be lower than 1800Ω. (If the backtrack limit was reached at any time before the first successful attempt, the maximum detectable resistance value is not modified). The fault coverage already obtained during fault simulation with an earlier vector is modified accordingly, and the fault is dropped if the new fault coverage value 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 primary output while minimizing a reduction in the resistance interval along the path. Resistance intervals are inserted at the outputs of all driven gates and propagated along the shortest path.

The logic value and resistance interval at each node along the path is examined. If the resistance interval at a node is the same as the resistance interval at the fault site, we proceed to the next node along the propagation path, 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. In the figure, the chosen propagation node is P2, and the propagation path is {P2, Z}. First, J is justified to a 0, so that the fault site resistance interval appears at P2. Then we have to 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 a 1 on F, we will get a resistance interval on P1, thus making it a faulty node instead of a fault-free node. If this resistance interval is allowed to propagate, it may cause the resistance interval at Z to be smaller than the one at P2, thus causing loss of coverage. 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.

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 detectable resistance interval.

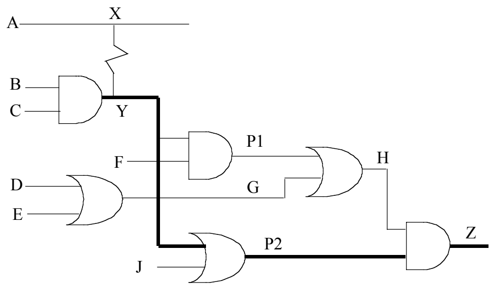


Figure 15. Backtracking during justification of side inputs.

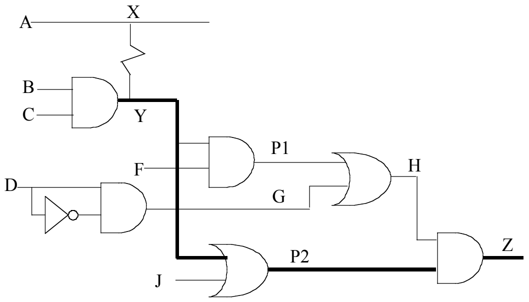


Figure 16. Saving best choice during justification of side inputs.

During fault propagation, if a node on the propagation path has a resistance interval less than the interval we are trying to propagate, we must backtrack. Figure 17 illustrates this case.

The desired resistance interval [0 2k] appears on node B, and the propagation path is {B,F,J}. We set A to 1, so that the interval 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 interval [0 2k]. To remedy this, we backtrack and set D to 1 and C to 0.

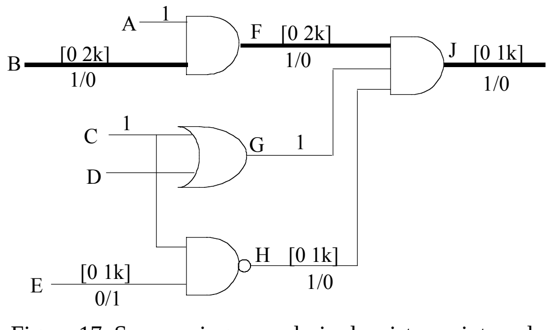


Figure 17. Suppressing an undesired resistance interval.

If the fault coverage at any node along the propagation path falls below the coverage already achieved for this fault, the path is dropped and the next shortest path from the faulty node to primary outputs is chosen. A limit has been set on 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 resistance interval to the primary outputs, then the next entry in the look-up table is chosen.

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

The fault coverage obtained by ATPG for a fault may not be the highest fault coverage 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% fault coverage, the fault is not dropped, because a later test vector for some other fault may achieve a higher fault coverage for this fault.

V. Results

A. Fault Simulation Results

The bridging fault simulator was run on the ISCAS85 benchmark circuits [29]. Table 2 gives some statistics of these circuits. Listed are the all-pair and 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. The test vectors were obtained from the ATALANTA stuck-at fault ATPG [33].

Table 2. Statistics for ISCAS85 circuits used.

Circuit	All-pair BFs	Reduced BFs	PI BFs	Feed-back BFs	Large case BFs	Other dropped BFs	Logic testable BFs	Applied vectors
c432	19,110	269	7	103	2	0	157	50
c499	28,403	191	8	47	0	1	135	53
c880	97,903	1086	27	110	0	1	948	48
c1355	171,991	1066	5	422	0	0	639	85
c1908	416,328	2206	7	521	16	0	1662	118
c2670	1,016,025	4572	110	168	0	0	4294	103
c3540	1,476,621	5315	4	780	100	0	4431	156
c5315	3,086,370	7407	38	237	11	0	7121	120
c6288	2,995,128	4492	1	1275	0	0	3216	34
c7552	6,913,621	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 faults completely detected (dropped) with at V_{DD}=3.3V and V_{DD}=1.2V, and for an average resistance distribution using equation (1) (realistic bridges) and a zero-ohm resistance distribution (zero-ohm bridges). Figure 19 shows the fault coverage. The following observations can be made from Figure 18 and Figure 19:

- At 3.3V, the realistic bridge coverage is high even though the drop rate is low. This is because most faults that were not dropped had a large detected resistance, though not equal to the maximum detectable resistance.
- Zero-ohm bridge coverage and drop rate is higher than for realistic bridges.

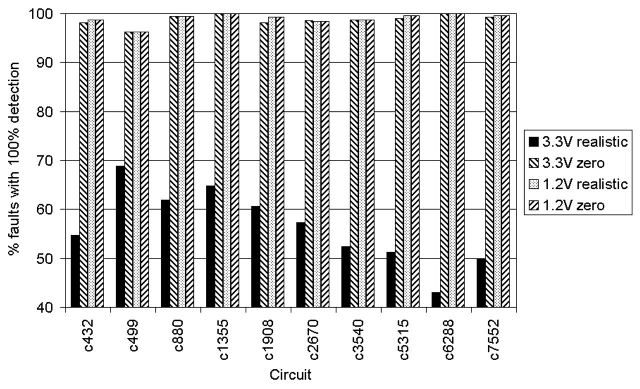


Figure 18. Faults dropped for each circuit.

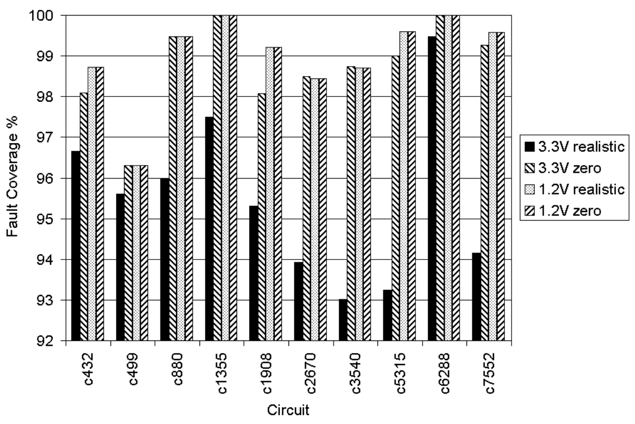


Figure 19. Fault coverage for each circuit.

- For realistic bridges, more faults are dropped and coverage is higher at 1.2V than 3.3V.
- For zero-ohm bridges, 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 those few faults which escape cause the overall fault coverage to drop. This anomaly occurs only for these two circuits because these circuits have a relatively high number of case 4 bridging faults.

B. ATPG Results

We applied our bridging fault ATPG to the ISCAS85 circuits. Since our current implementation is inefficient, we experimented on only the small ISCAS85 benchmarks.

As we discussed earlier, we may apply a test set prior to ATPG. We have used three different test vector sets for this purpose. 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 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 the circuits used for ATPG.

Table 3. Statistics for circuits used during ATPG.

Circuit	Nodes	Logic-testable faults	Stuck-at vectors	4-detect vectors	Random vectors
c432	196	1211	50	199	199
c499	243	1640	53	219	219
c880	443	2813	48	180	180

Figure 20 and Figure 21 shows the results of the four experiments performed on c432 for resistive bridges. Figure 20 shows the faults dropped (faults having 100% detection) and Figure 21 shows the fault coverage. Similar results were obtained for c499 and c880. The following observations can be made:

- NDET-ATPG, SA-ATPG, and RND-ATPG achieved better results than ATPG only, due to the fact that the ATPG only hit its abort limits on more faults.
- 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 bridging fault 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 bridging faults.
- In NDET-ATPG and RND-ATPG many faults are dropped in ATPG because it can be proven that they achieved 100% coverage during fault simulation.
- ATPG in NDET-ATPG and RND-ATPG detects only the targeted fault. During fault simulation with the generated vector, no faults are dropped, nor is there a significant improvement in fault coverage.

C. Resistive vs. Zero-Ohm Bridging Faults

In order to determine the usefulness of a zero-ohm bridge fault model, we compare the resistive bridge fault coverage of test sets developed using the zero-ohm bridge fault model. Figure 22 shows a comparison between the resistive bridge fault coverage of SA-ATPG on c432 done using the two models. As can be seen, the coverages are similar for the stuck-at vectors. But for the ATPG vectors the zero-ohm model has lower coverage, and also runs out of faults sooner due to its higher drop rate. Figure 23 shows the results of running ATPG only. The zero-ohm model has lower coverage for all vectors. A much larger fault list was used in these experiments to gain adequate resolution.

These results indicate that for a given test length, test vectors generated using the zero-ohm bridging fault model will have significantly lower resistive bridge fault coverage than vectors generated using the resistive bridge fault model. Similar results were obtained for the c499 and c880 benchmarks.

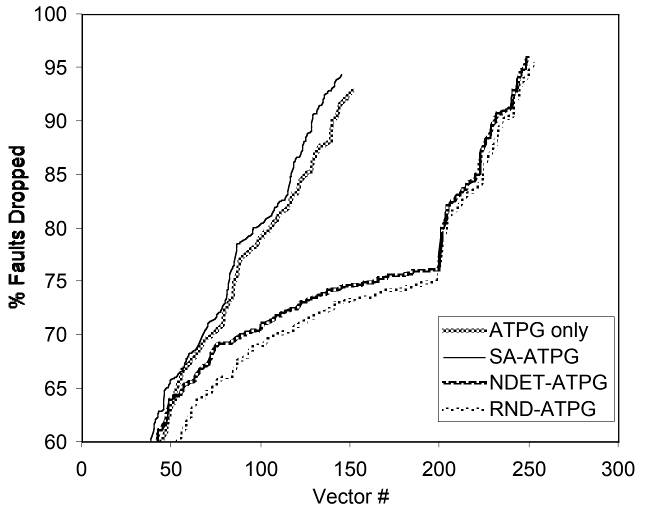


Figure 20. Faults dropped for c432 for resistive bridges.

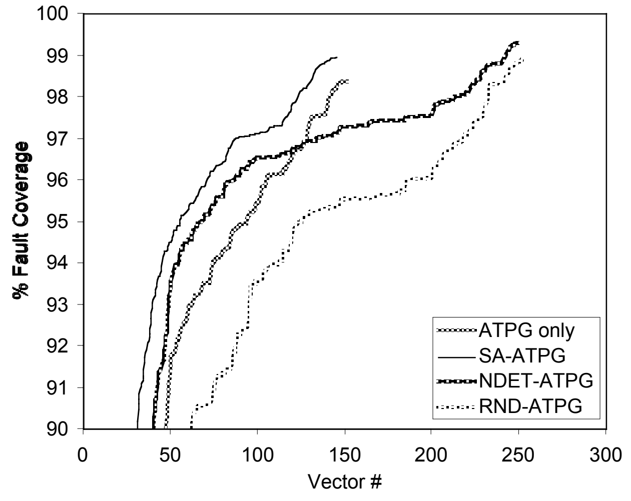


Figure 21. Fault coverage for c432 for resistive bridges.

VI. Conclusions

We have developed an accurate resistive bridging fault model and used it in fault simulation and ATPG on combinational circuits. Our fault simulation results confirm that reduced V_{DD} leads to an increase in overall resistive bridging fault coverage. Certain situations have been identified where reducing V_{DD} causes a fault to escape detection, despite being detected at higher V_{DD}.

ATPG results show that test vector generation targeting resistive bridging faults improves on the coverage obtained from fault simulation with a stuck-at or random fault test set. However, the best results are obtained when fault simulation with a stuck-at test vector set is followed by test generation for the remaining faults.

Fault simulation and ATPG results show that a test set generated for zero-ohm bridging faults does not perform as well for resistive bridging faults of similar size.

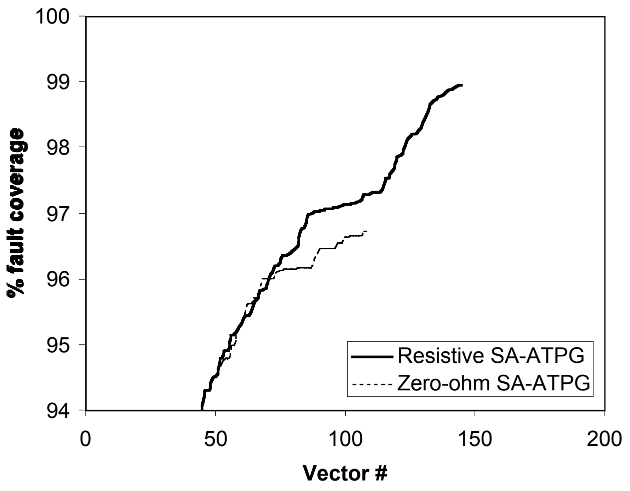


Figure 22. Resistive bridge fault coverage for SA-ATPG on c432 for resistive and zero-ohm fault models.

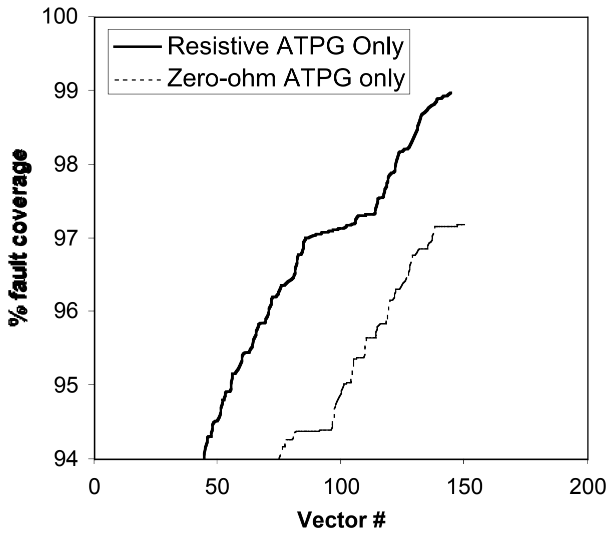


Figure 23. Resistive bridge fault coverage for ATPG only on c432 for resistive and zero-ohm fault models.

References

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

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

[3] 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.

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

[5] 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.

[6] Jeff Rearick and Janak H. Patel, “Fast and Accurate Bridging Fault Simulation,” Int. Test Conf., 1993, pp. 54-62.

[7] 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.

[8] T. M. Storey and W. Maly, “CMOS Bridging Fault Detection,” Int. Test Conf., 1990, pp. 842-851.

[9] 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.

[10] 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.

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

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

[13] 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.

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

[15] 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.

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

[17] Y. Liao and D. M. H. Walker, “Optimal Voltage Testing for Physically-Based Faults,” Proc. VLSI Test Symp., 1996, pp. 344-353.

[18] 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.

[19] 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.

[20] F. Peters and S. Oostdijk, “Realistic Defect Coverages of Voltage and Current Tests,” Proc. Int. Workshop on IDDQ Testing, 1996, pp. 4-8.

[21] Li-C. Wang, M. R. Mercer, and T. W. Williams, “Using Target Faults to Detect Non-Target Defects,” Int. Test Conf., 1996, pp. 629-638.

[22] 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.

[23] Yuyun 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.

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

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

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

[27] 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.

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

[29] 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.

[30] J. T.-Y. Chang and E. J. McCluskey, “Detecting Delay Flaws by Very-Low-Voltage Testing,” Proc. Int. Test Conf., 1996, pp. 367-376.

[31] 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.

[32] 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.

[33] 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.