PROBE: A PPSFP Simulator for Resistive Bridging Faults

Chul Young Lee
Compaq Computer Corporation
334 South Street SHR3-2/S30
Shrewsbury MA 01545-4112
Tel: (508) 836-5556
http://www.cs.tamu.edu/people/chull
E-mail: chull@cs.tamu.edu

D. M. H. Walker
Dept. of Computer Science
Texas A&M University
College Station TX 77843-3112
Tel: (979) 862-4387
Fax: (979) 847-8578
E-mail: walker@cs.tamu.edu

Abstract

Bridging faults in CMOS circuits are usually modeled as a wired-OR, wired-AND, or small fixed resistance. Real bridging faults have a resistance distribution ranging from very small to quite large. The parametric model has been proposed to handle this resistance distribution, along with table-oriented approaches that are accurate and fast. Fault simulators and a test generator have been developed using these models. Prior approaches were too slow to simulate or generate large test sets, handle large circuits, or analyze a wide variety of different test sets. We have developed PROBE[1], a pseudo-PPSFP simulator for resistive bridging faults that is significantly faster while maintaining circuit-level accuracy. We have used PROBE to analyze several large test sets on the ISCAS85 circuits in an effort to gain insight into how existing test generation approaches detect resistive bridges.

1. Introduction

It is known that a large percentage of fabrication defects result in bridging faults in CMOS VLSI circuits and that most of these are external bridging faults [1][2][3]. Therefore, tests for external bridging faults cover one of the dominant fault types. It is also known that the stuck-at fault, and wired and voting fault models are inadequate for modeling realistic bridging faults [4][5]. This is mainly because realistic bridging faults are resistive [6][7][8][9] and have limited resistance intervals [6][10][11] where detection can be achieved for the specific test patterns. The parametric bridging fault model incorporates these effects [10][11][12][13]. Using pre-computed lookup tables permits fast yet accurate circuit modeling [14][15][16]. However our previous approach [14] is still much too slow to handle large circuits or test sets. In particular, we were unable to analyze the coverage of long random or weighted random patterns such as generated by BIST.

[1] This research was supported by the National Science Foundation under grant MIP-9406946. This research was performed while Chul Y. Lee was a Ph.D. student at Texas A&M University.

Parallel pattern single fault propagation (PPSFP) [17] has proven to be a powerful technique to speed up stuck-at fault simulation. It gains its speed through simpler data structures and the fact that most faults are detected and dropped in the first few vectors. We reasoned that since similar fallout patterns are observed on real defects during production test, PPSFP should also speed up resistive bridging fault simulation. We have developed a pseudo-PPSFP algorithm for resistive bridging faults using a normalized fault coverage metric [14] and implemented them in a fault simulator PROBE (PPSFP for Resistance-Oriented Bridge Evaluation). PROBE is fast enough that we have been able to analyze stuck-at, random, weighted random, DOREME [18], and N-detect test sets on the ISCAS85 benchmarks.

In the sections that follow we discuss our fault propagation scheme, fault simulation algorithm, applications to ISCAS85 benchmarks with different test sets, conclusions and future work.

2. Detection Condition Set Propagation

2.1. Detection Condition Set (DCS)

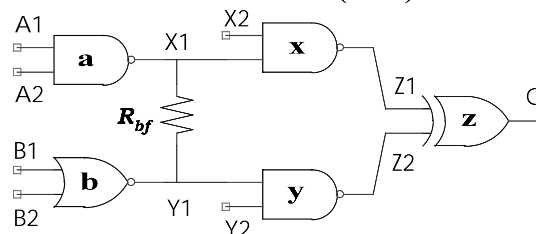

Figure 1. An external bridging fault.

Consider the external CMOS bridging fault in Figure 1. To sensitize the fault, nodes X1 and Y1 should be set to opposite fault free logic values, i.e. 1 and 0 respectively or vice versa. Assume that X2 and Y2 are fixed at logic 1. Then X1, Z2, and Q will have logic value D (faulty-off, 0/1), and Y1 and Z1 will have $\overline{D}$ (faulty-on, 1/0) for any of the test vectors <A1 A2 B1 B2> ∈ {0011, 0001, 0101, 0111}. The fault is detected by a low-speed Boolean test only when the resistance is in the interval [0, R_{upper}]. We

Table I. DCS propagation formula table

Inputs / Gate	$D : \overline{D}$	$D : D$	$\overline{D} : \overline{D}$	$1: D$	$1: \overline{D}$	$0 : D$	$0 : \overline{D}$
AND	$\overline{D}(S0-S1)$	$D(S1_i \cup S1_j)$	$\overline{D}(S0_i \cap S0_j)$	$D(S1)$	$\overline{D}(S0)$	$0(Null)$	$0(Null)$
NAND	$D(S0-S1)$	$\overline{D}(S1_i \cup S1_j)$	$D(S0_i \cap S0_j)$	$\overline{D}(S1)$	$D(S0)$	$1(Null)$	$1(Null)$
OR	$D(S1-S0)$	$D(S1_i \cap S1_j)$	$\overline{D}(S0_i \cup S0_j)$	$1(Null)$	$1(Null)$	$D(S1)$	$\overline{D}(S0)$
NOR	$\overline{D}(S1-S0)$	$\overline{D}(S1_i \cap S1_j)$	$D(S0_i \cup S0_j)$	$0(Null)$	$0(Null)$	$\overline{D}(S1)$	$D(S0)$
XOR	$D(\cup_{S_m \neq S_n}(S_m - S_n))$	$\overline{D}(\cup_{i \neq j}(S1_i - S1_j))$	$\overline{D}(\cup_{i \neq j}(S0_i - S0_j))$	$\overline{D}(S1)$	$D(S0)$	$D(S1)$	$\overline{D}(S0)$
XNOR	$\overline{D}(\cup_{S_m \neq S_n}(S_m - S_n))$	$D(\cup_{i \neq j}(S1_i - S1_j))$	$D(\cup_{i \neq j}(S0_i - S0_j))$	$D(S1)$	$\overline{D}(S0)$	$\overline{D}(S1)$	$D(S0)$

call this the *detectable resistance interval*. V_{th_d} is the logic threshold voltage of the driven gates, and R_{upper} is the maximum detectable resistance of the bridge. The fault is detected in the resistance interval where the input voltage of a driven gate causes a faulty logic value at the output. The detectable resistance interval depends on V_{th_d} because the maximum detectable bridging resistance occurs when the input voltage on the driven gate reaches the logical threshold. We assume that the gain of the driven gate is high enough that the driven gate output can be approximated as always a well-defined logic value. We call the set of detectable resistance intervals for a fault the *detection condition set* (DCS). At node Q, the DCS on the propagation path is represented as

$$S_Q = \{R_{bf} \mid R_{lower} \le R_{bf} \le R_{upper}\} .$$

A fault at node Q is represented as $D_Q(S_Q)$ where D_Q is the faulty logic value detected when the bridge resistance belongs to DCS S_Q. PROBE uses fault propagation and detection strategies that are based on the DCS operations described in the following sections. The reason for a DCS formulation is that as discussed in [11], resistance intervals can shrink or contain holes during propagation to a primary output, so using a single resistance value such as R_{upper} would result in loss of accuracy.

2.2. DCS Propagation Formula Table

Since the DCS are sets, propagation of them from a fault site through logic gates is done using set operations, as shown in Figure 2. The AND gate has two inputs which are both faulty and have their own DCS's. Let R_0 and R_1 be the maximum detectable bridging resistances via gates x and y in Figure 1 respectively. Then the DCS's at nodes Z1 and Z2 are $S0 = \{R_{bf} \mid 0 \le R_{bf} \le R_0\}$ and $S1 = \{R_{bf} \mid 0 \le R_{bf} \le R_1\}$ respectively. The output Q in Figure 2 has a faulty logic value (logic 1) only when the input A has a good value (logic 1, outside $S1$) and the input B has a faulty value (logic 1, inside $S0$). This is the shaded area of the Venn Diagram in Figure 2. In other words, the fault is only detected at Q for the set $S0$-$S1$. From the above detection through two-input AND/NAND gates requires

$$R_{bf} \in S0 - S1$$

and detection through two-input OR/NOR gates requires

$$R_{bf} \in S1 - S0 .$$

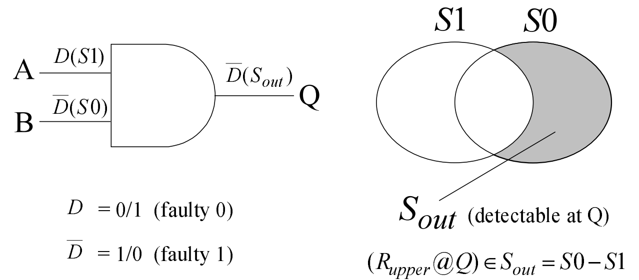

Figure 2. DCS propagation.

The set S0 is the DCS with logic value $\overline{D}$, and S1 is the DCS with logic value D. The DCS propagated at the output Q in Figure 1 is represented as

$$R_{bf} \in (S0 \cup S1) - (S0 \cap S1) = (S0 - S1) \cup (S1 - S0)$$

or simply

$$R_{bf} \in \cup_{S_m \neq S_n}(S_m - S_n)$$

where $S_m, S_n \in \{S0, S1\}$. This is the DCS at the output of an XOR gate. The DCS propagation formulae for other two-input gates is shown in Table I. For example, if a NAND gate has inputs $D(S1)$ and $\overline{D}(S0)$, the output is $D(S0$-$S1)$. The equations can be nested to handle n-input gates with more than two faulty inputs. If a gate input is independent of the fault, it does not have any DCS, which we represent as the *Null* set.

3. Pseudo-PPSFP using DCS

3.1. Worst Case Speedup

Regardless of the number of test vectors applied, there are only a limited numbers of sensitizations for each fault, and therefore a limited number of DCS computations that

must be performed. For example, there are only 6 possible sensitizations for an AND2-AND2 bridging fault (one choice for a one, and three choices for a zero, and vice-versa). Therefore, if we apply M test vectors in parallel, the redundancy in the DCS's is at least M/6, and M if there is only one unique sensitization. Most of the time, the redundancy will be higher than the minimum since multiple sensitization and propagation paths will be simulated. If we can eliminate all redundant computations, we can achieve a corresponding speedup in the computation. Table II shows the minimum speedup for several faults. Assuming M=32, the minimum speedup will be 2–5 over SPSFP simulation.

Table II. Minimum speedup for M bit PPSFP

Driving Gates	# Possible sensitizations	Worst case speedup
AND2-AND2	6	M/6
OR2-OR2	6	M/6
AND2-OR2	10	M/10
XOR2-XOR2	8	M/8
AND2-XOR2	8	M/8
OR2-XOR2	8	M/8
AND3-AND3	14	M/14
OR3-OR3	14	M/14

3.2. Categorized DCS

To eliminate the redundant DCS calculations, we categorize the DCS's and keep pointers to the corresponding *categorized DCS* (CDCS) for the different test patterns. For an individual bit j in the test pattern, we keep the logic values $D_{i,j}$ (identical for every fault configuration i) from the fault free logic simulation and get the *DCS pointer* $DCS^{(i,j)}$ for fault i and bit j in the pattern at every propagation stage starting from the fault site. (In a combinational circuit, the node values are fault free prior to the fault site). This pointer $DCS^{(i,j)}$ points to the categorized DCS $CDCS^{K}$. The DCS pointers $DCS^{(i,j)}$ will point to the same specific categorized DCS $CDCS^{K}$ if the bits with different bit index j in the parallel pattern have the same combination of logic values for all inputs of the logic gate (both at the fault site and during propagation), and the bits with different bit index j have the same DCS pointer (during fault propagation). For example, as shown in Figure 3, suppose bits j = 0, 2, 3, 6, M-2 of the input patterns to the AND3 gate q on the propagation path all have the same logic pattern, say '101'. Also suppose bits j = 0, 2, 3, 6, M-2 of a faulty input Q1 of this gate have a common DCS pointer and so does Q2. Then the pointers $DCS^{(i,j)}$ for bits j = 0, 2, 3, 6, M-2 of the output of gate q will point to the same CDCS, and this CDCS need only be computed once.

The number of distinct logic patterns for fault i at gate s in simulation pass t is $\mu_i(s,t)$, the *logic categorization factor.* A simulation pass is one application of the M patterns to all faults. $\theta_i(s,t)$ is the *sensitization reduction,* the number of vectors in a batch of M that either do not excite or do not propagate the fault. Then we define the *acceleration factor* for fault i as

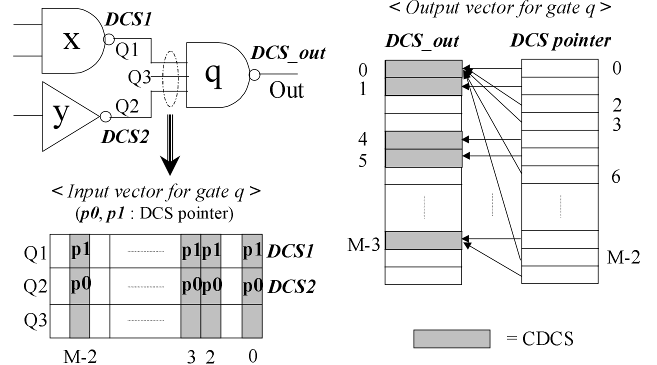

Figure 3. DCS propagation strategy,

$$\lambda_i(s,t) = \begin{cases} \dfrac{M - \theta_i(s,t)}{\mu_i(s,t)}, & 1 \le \mu_i(s,t) \le M \\ N/A, & \mu_i(s,t) = 0 \end{cases}$$

at gate s in simulation pass t. Memory and CPU time will be reduced more when the acceleration factor is larger. The acceleration factor is bounded below by the minimum speedup.

4. Fault Coverage

The detection probability (DP) for a single resistance interval is modeled as

$$c_{physical}(R_{upper}) = 1 - (1-p)^{R_{upper}}$$

fit to the data in [5][7] using $p=0.00258$ and R_{upper} is the upper bound of the resistance interval.

The *normalized detection probability* (NDP) for a single resistance interval is

$$c_{normal}(i,j) = \frac{1-(1-p)^{R_{upper}(i,j)}}{1-(1-p)^{R_{upper_max}(i)}}$$

where $R_{upper_max}(i)$ is the maximum possible detectable bridging resistance at the site of fault i, and $R_{upper}(i,j)$ is the upper bound of the resistance interval for fault i, test pattern j. For the whole circuit, we would have different DCS's and D values at different primary outputs. If $DCS_k^{(i,j)}$ is the DCS at primary output k with logic value $D_k^{(i,j)}$ for fault i, test pattern j, the DCS for the entire circuit for fault i, test pattern j in simulation pass t will be

$$S_t^{(i)} = \bigcup_j \bigcup_k (S_{k,t}^{(i,j)}).$$

In the general case, there might be resistance intervals whose lower bound is not 0 in $S_t^{(i)}$. We can generalize the formula for $c_{normal}(i,j)$ to handle this. Let $R_{lower}(i,j)$ and $R_{upper}(i,j)$ be the lower and upper bounds respectively of an

interval in the DCS. The DCS with maximum possible upper bound (thus maximum possible fault coverage) will be

$$S_{\max}^{(i)} = \{R_{bf} \mid 0 < R_{bf} < R_{upper_\max}\}.$$

Then the *normalized detection probability* $c_{normal}(i,t)$ for bridging fault configuration i in simulation pass t is

$$\frac{\sum_{n_{S_t^{(i)}}} [c_{physical}(R_{upper,n_{S_t^{(i)}}}(i,t)) - c_{physical}(R_{lower,n_{S_t^{(i)}}}(i,t))]}{c_{physical}(R_{upper_\max}(i))}.$$

We drop fault i from the fault simulation if $c_{normal}(i,t)$ is at least the user-defined threshold fault coverage (TC). *Overall fault coverage* (FC) for all faults in simulation pass t is

$$C_{overall}(t) = \sum_i \frac{c_{normal}(i,t)}{N}$$

where N is the total number of bridging faults, assuming all faults are equally likely.

5. Fault Simulation Algorithm

Figure 4 shows the flow chart for our fault simulation algorithm as implemented in the PROBE simulator. After parsing the circuit and getting the fault list, we get the first M test vectors for the first simulation pass (we currently use $M=32$). Fault free parallel logic simulation is performed and the fault free logic values are stored for all nodes. Then the faults in the fault list are simulated one by one for these M vectors. At the fault site, our simulator uses the pre-calculated lookup tables to find the detectable resistance interval of bridging fault i for all the M test vectors. These lookup tables are pre-computed with HSPICE simulations as described in [14][15].

When the detectable resistance interval is obtained from the tables for the first bit of the M-bit parallel test pattern, it is stored in the first categorized DCS which is labeled $CDCS^1$ and the DCS pointer for the first bit ($DCS^{(i,1)}$) will point to $CDCS^1$. Once the first CDCS is stored, the simulator checks the other bits of the same node and either creates a new CDCS (if the pattern is not seen in the prior bit indices) or points the DCS pointer to an existing CDCS (if the pattern is redundant). These pointers ($DCS^{(i,1)}$) and CDCS's ($CDCS^K$) are propagated to the primary outputs using our DCS propagation formulae. After propagating all these values to primary outputs, the normalized fault coverage $c_j(i)$ for fault i and bit j is calculated for all j, for distinct DCS's. The simulator checks whether any $c_j(i)$ has reached the threshold coverage and drops the fault i if this is the case. This is the inner loop of the algorithm in Figure 4. The simulation will terminate if all the faults are dropped. If not, the simulator will get the next M vectors and repeat the procedure. This is the outer loop of the algorithm. The simulation terminates either if the fault list is empty or it runs out of test patterns.

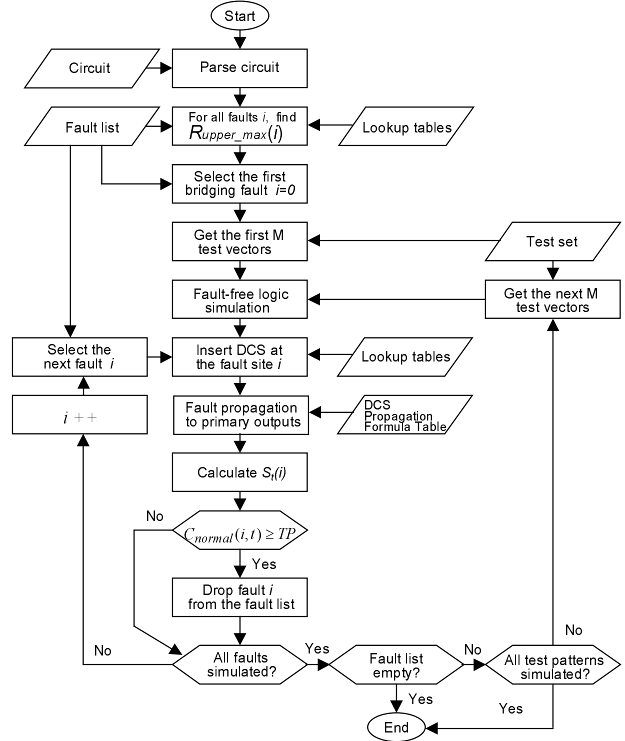

Figure 4. Fault simulation algorithm.

Before simulation starts, we do pre-processing to build *lookup tables* with circuit-level HSPICE simulation [15]. The execution time depends on the desired resistance interval resolution and cell library size, but typically takes several hours. These tables are used for DCS insertion at fault sites during logic-level simulation. Another important task done in the pre-processing step is *fault list preparation*. We are assuming that there are only 2-node bridging faults and we used random 2-node faults in our simulation. We currently exclude feedback faults in order to simplify the simulation, but will include them in a future version of the simulator. In prior resistive switch simulation work we found that feedback bridges rarely oscillator, so we do not except significantly different results than presented here.

6. Simulation Results

Table III shows the statistics of the ISCAS85 benchmark circuits simulated using 10,016 (32·313) random patterns on a Sun SPARC 5. Even after 10,000 random patterns, the fault coverage is often low and many faults are not dropped. In all but c6288 the acceleration factor is substantially higher than the lower bound of 2–5. The simulation time is orders of magnitude slower than stuck-at fault simulation using *fsim* [19]. This is primarily due to the lower drop rate, more complex fault model, and decision to keep lookup tables on disk to minimize memory requirements. In retrospect the lookup tables

could have easily been held in memory, since their size is a function of the cell library size, not the circuit size.

Table III. Benchmark simulation results for 10,016 random patterns

Circuit	Bridging Faults	Fault Coverage (%)	Faults Dropped (%)	Avg. Accel. Factor	Sim. Time (min)	*Fsim* Time (sec)
c432	157	98.02	78.34	13.7	13	0.97
c499	136	84.04	63.97	13.0	15	1.20
c880	949	95.70	65.96	11.2	116	1.68
c1355	639	96.47	69.32	12.3	216	2.63
c1908	1,662	97.31	73.41	12.5	628	3.98
c2670	4,294	87.96	61.34	10.1	2517	10.70
c3540	4,431	90.00	52.92	12.0	3381	10.45
c5315	7,121	94.77	60.78	10.5	6975	10.78
c6288	3,216	99.63	51.93	6.0	10,054	13.75
c7552	12,106	95.71	61.24	10.4	19,908	20.83

Table IV shows the results of applying different uncompacted test sets to 1,211 bridging faults for c432. All of the combinations use approximately 10,000 vectors. In most cases, DOREME vectors [18] are superior to other types. Figure 5 shows that even after applying 5,120 random vectors, DOREME vectors can quickly detect more faults. It supports the observation [20] that it is the vectors that detect the faults, not the fault model.

Figure 6 shows the fault coverage with different types of test sets. Applying DOREME tests achieves higher fault coverage (FC) than applying random vectors or vectors targeting stuck-at faults. The lower final fault coverage with random vectors is due to the random vector resistance of this circuit.

Table IV. Fault coverage and fault dropping for 1,211 faults in c432

Test Set	# Vectors	Coverage (%)	Drop (%)
Stuck-at	288	95.34	70.69
7-Detect	320	95.82	72.17
Weighted random	480	95.90	72.58
Random	10,016	95.95	72.83
DOREME	10,016	96.24	76.38
Rand + DOREME	5,120 + 5,120	96.24	76.30
SA + Random	288 + 9,728	95.98	74.73
SA + DOREME	288 + 9,728	96.24	76.38
7-Detect + Rand	320 + 9,696	95.98	75.14
7-Detect + DOREME	320 + 9,696	96.24	76.38
WRP + DOREME	480 + 9,536	96.24	76.38

Figure 7 is the number of escapes vs. number of vectors for DOREME vectors applied to c432 for 1,211 faults. The escapes are subdivided by DP level. Initially many escaped faults have low DP. By the end of the simulation very few faults have a DP below 90%. Almost all of the defect level is contributed by faults with 0% DP. This suggests that to achieve a higher FC and a lower defect level, the APTG should target the faults with 0% DP rather than trying to drop a fault with a 99% DP.

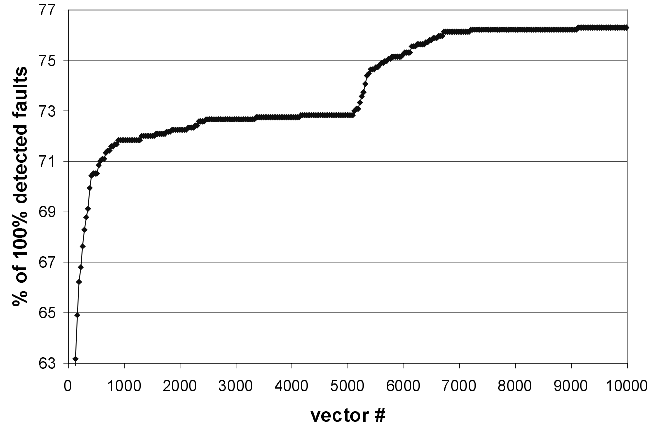

Figure 5. Fault dropping for c432 with 5,120 random and 5,120 DOREME vectors.

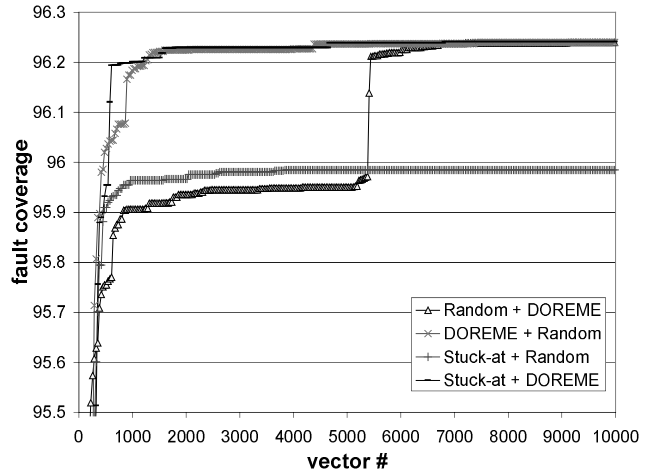

Figure 6. Fault coverage for mixed test sets with 10,016 vectors for c432.

7. Conclusions and Future Work

We have developed a pseudo-PPSFP algorithm for resistive bridging faults and implemented it in the PROBE simulator. We have used PROBE to study different test sets on ISCAS85 circuits. One of the purposes in developing this simulator was to make evaluation of these test sets on the complicated fault model feasible. The results show that while stuck-at, weighted random and N-detect test sets provide fairly good fault coverage. DOREME vectors are consistently superior. Combining DOREME vectors with other tests did not improve coverage. The DOREME vectors detect all faults detected by the other test sets.

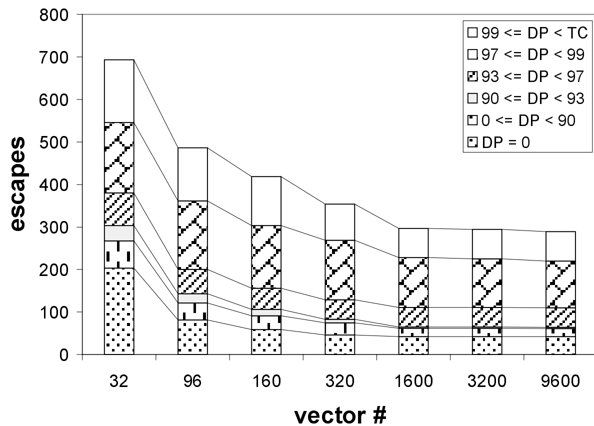

Figure 7. Detection probability distribution for c432.

Our experiments suggest that we can reduce simulation time by using a threshold detection probability (TP) below 100%, without much loss in fault coverage. We plan to perform additional experiments to determine the tradeoff between simulation time and coverage loss. Given that fault coverage quickly exceeds 90-95% during simulation, a TP in that range could greatly speed up simulation.

With physical design information it is possible to extract realistic bridging faults. These can be injected into the circuit using a gate-level bridging fault model such as a wired-OR or wired-AND, and tests generated for them. We would like to examine how well these tests compare to the optimal tests of [15] and the test sets analyzed here.

The pseudo-PPSFP algorithm in PROBE greatly accelerates simulation over SPSFP. We are developing a true PPSFP algorithm that promises several orders of magnitude speedup. Our goal is to achieve simulation times competitive with stuck-at simulation while accounting for the lower drop rate of resistive bridges.

Acknowledgements

Thanks to M. R. Grimaila for providing DOREME test vectors.

References

[1] J. Shen, W. Maly, and F. Ferguson, "Inductive Fault Analysis of MOS Intergated Circuits", *IEEE Design and Test Magazine*, Dec. 1985, pp. 53-56.

[2] K. J. Lee, J. J. Tang, and T. C. Huang, "BIFEST: A Built-in Intermediate Fault Effect Sensing and Test Generation System for CMOS Bridging Faults", *ACM Trans. on Design Automation of Electronic Systems*, Vol. 4, No. 2, April 1999, pp. 194-218.

[3] J. J. Sousa, F. M. Gonclves and J. P. Teixeira, "IC Defects-Based Testability Analysis", *Int'l Test Conf.*, Oct. 1991, pp. 500-509.

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

[5] R. Rodriguez-Montanes, E. M. J. H. Bruls and J. Figueras, "Bridging Defects Resistance Measurements in a CMOS Process", *Int'l Test Conf.*, 1992, pp. 892-899.

[6] M. Dalpasso *et al.*, "Parametric Bridging Fault Characterization for the Fault Simulation of Library-Based ICs", *Int'l Test Conf.,* 1992, pp. 486-495.

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

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

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

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

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

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

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

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

[15] V. Sar-Dessai and D. M. H. Walker, "Resistive Bridge Fault Modeling, Simulation and Test Generation", *Int'l Test Conf.*, Sept. 1999, pp. 596-605.

[16] C. Di and J. A. G. Jess, "An Efficient CMOS Bridging Fault Simulator: With SPICE Accuracy", *IEEE Trans. on CAD*, Vol. 15, No. 9, Sept. 1996, pp. 1071-1080.

[17] J. A. Waicukauski, E. B. Eichelberger, D. O. Forlenza, E. Lindbloom and T. McCarthy, "Fault Simulation for Structured VLSI", *VLSI Sys. Design*, Vol. 6, No. 12, Dec. 1985, pp. 20-32.

[18] M. R. Grimaila, S. Lee, J. Dworak, K. M. Butler, B. Stewart, H. Balachandran, B. Houchins, V. Mathur, J. Park, L.-C. Wang, and M. R. Mercer, "REDO – Random Excitation and Deterministic Observation – First Commercial Experiment", *Int'l Test Conf.*, 1999, pp. 268-274.

[19] H. K. Lee and D. S. Ha, "An Efficient Forward Fault Simulation Algorithm Based on the Parallel Pattern Single Fault Propagation", *Int'l Test Conf.*, Oct. 1991, pp. 946-955.

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