SIGMA: A Simulator for Segment Delay Faults*

Keerthi Heragu
Janak H. Patel
Center for Reliable and High-Performance Computing
University of Illinois at Urbana-Champaign, Urbana, IL 61801
heragu@crhc.uiuc.edu, patel@crhc.uiuc.edu

Vishwani D. Agrawal
Bell Labs, Lucent Technologies
700 Mountain Avenue
Murray Hill, NJ 07974
va@research.bell-labs.com

Abstract

We propose an efficient combinational circuit simulation technique for the recently proposed segment delay fault model. After simulation of a vector pair, activated segments are traced using a depth-first search. A segment numbering scheme finds the number of faults to be simulated. A labeling technique generates edge labels to compute a unique label for each segment fault. The use of labels avoids explicit storing of fault lists and allows efficient access to previously detected segment faults. Experimental results demonstrate several advantages of the segment delay fault model. First, the total number of faults remains manageable for small segment lengths. Second, many segments, not included in any robustly testable path fault, may have robust segment delay fault tests. Generating tests for such segments may increase the delay defect coverage.

1 Introduction

Physical defects that cause timing violations are modeled by delay faults at the logic level. *Transition delay faults* [1] and *gate delay faults* [2, 3] model slow-to-rise and slow-to-fall delay defects at gate inputs and outputs. *Path delay faults* [4, 5, 6, 7] model defects that cause cumulative propagation delays along paths to exceed specified limits. A more recently proposed model is the *segment delay fault model* [8]. Segment delay faults can model all general delay defects ranging from spot defects to distributed defects. The segment length, L, is an input parameter that can be chosen from defect statistics. L can be as small as 1 or as large as the maximum logic depth (path delay faults). Once L is chosen, the fault list will comprise of all segments of length L and all paths whose entire length is less than L. Both rising and falling transitions at the origins of segments are considered.

*This research was supported in part by Semiconductor Research Corporation under contract SRC 95-DP-109, in part by ARPA under contract DABT63-95-C-0069, and in part by Hewlett-Packard under an equipment grant.

Several path delay fault simulation algorithms have been proposed in the literature [9, 10, 11]. In this work, we propose a segment delay fault simulation technique that accepts an input parameter L, the length of segments to be simulated. We use a numbering technique [8] to compute $N(S_q^k)$, the number of segments of length k ending at a node q in a graph representation of a circuit. We propose a labeling technique that uses $N(S_q^k)$ values and computes a set of edge labels. These labels are used during simulation to assign unique numbers to segments. The use of labels avoids explicit storing of fault lists.

We use a 10-valued algebra [8] for robust simulation of segment delay faults. During the simulation of a vector pair, we record those inputs of gates that propagate robustly to their outputs. Following the simulation of a vector pair, activated segments are traced using a backward depth-first search.

We explore another important advantage of the segment delay fault model, i.e., some robustly testable segment faults may not be part of any robustly testable path fault. Figure 1 gives an example. Assume that gate G is implemented as a stuck-fault testable XOR gate. All stuck-faults in the circuit are testable. Also, all gates are irredundant. *It can be verified that no robustly or non-robustly testable path fault includes the segment P-Q-R-S with a rising transition on P.* The path faults *L-P-Q-R-S* and *M-P-Q-R-S* with a falling transition at their origins are testable together as a multiple path fault [7]. However, dealing with multiple path faults may be computationally infeasible. A complete test set for all robustly or non-robustly testable single path delay faults will not test for a delay fault on this segment for the corresponding transition. The test shown is not a robust (or non-robust) test for any path fault. However, it is a robust test [4] for sub-path *P-Q-R-S* with a rising transition on P. We expect that many path faults in real circuits may be robustly (or non-robustly) untestable, but there may be sub-paths that are robustly testable and it is useful to test them.

ICCAD '96
1063-6757/96 $5.00 © 1996 IEEE

2 Unique Representation of Segments

Pomeranz and Reddy have proposed a linear time path delay fault coverage estimation algorithm [12]. Efficient path representation techniques have also been proposed [10, 11]. We develop algorithms for segment numbering and labeling based on these ideas. Consider a directed acyclic graph (DAG) model for the circuit where the nodes of the graph represent gates, primary inputs (PIs), and primary outputs (POs). Edges represent signal lines. First, we outline an algorithm, from our earlier work [8], to compute the number of segments of a given length. The algorithm which traverses from PO to PI is modified to traverse from PI to PO to facilitate simulation, and is included here for completeness. Next, we propose a new edge labeling algorithm used to assign unique labels to detected segment faults.

2.1 Computing number of segments

We use the following notation:
S_q^k : set of segments of length k ending at node q
$N(S_q^k)$: cardinality of set S_q^k
$NP(q)$: number of immediate predecessor nodes of q
$p_q[1]$, ..., $p_q[NP(q)]$: immediate predecessor nodes of q

Assume that we have computed $N(S^k_{p_q[1]})$, ..., $N(S^k_{p_q[NP(q)]})$. We use this information to compute $N(S_q^{k+1})$ as $\Sigma_{j=1}^{NP(q)} N(S^k_{p_q[j]})$. The following algorithm computes the number of segments of lengths 1, ..., L, where L is an input parameter. We assume that the circuit graph G has M nodes and that nodes are processed in a topological order from PIs to POs.

Algorithm:
for $length = 1$ to L
  $total(length) = 0$
for $node = 1$ to M
  for $length = 1$ to L
    if $node$ is a PI
      $N(S_{node}^{length}) = 0$
    else if length equals 1
      $N(S_{node}^{length}) = NP(node)$
    else
      $N(S_{node}^{length}) = \Sigma_{j=1}^{NP(node)} N(S_{p_{node}[j]}^{length-1})$
    $total(length) = total(length) + N(S_{node}^{length})$

The number of segments of length k in G is $total(k)$.

Figure 2 illustrates this algorithm for the graph corresponding to $c17$ (with $L = 3$). For a given L, the fault list contains all segments of length L and paths whose entire length is less than L. If $N(P_q^k)$ denotes the number of sub-paths of length k originating from a PI and ending at a node q, the total number of paths whose entire length is less than L can be computed by using the $N(P_q^k)$ values ($k < L$) for POs.

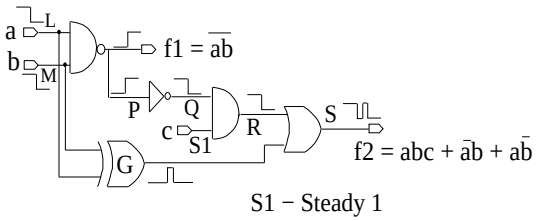

Figure 1: Robust test for segment fault P-Q-R-S

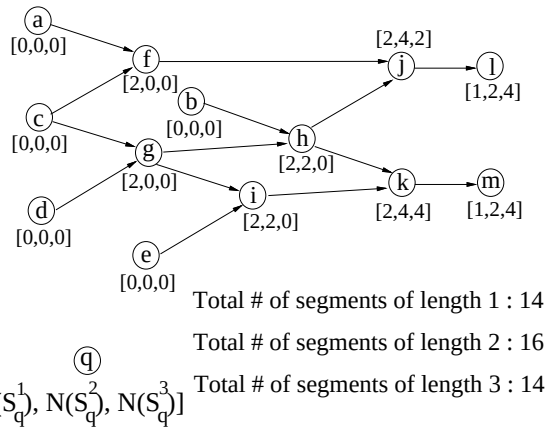

Figure 2: Computing number of segments in $c17$

2.2 Edge Labeling

We use the following additional notation:
B_d^k : k^{th} label of edge d
$e_q[1]$, ..., $e_q[NP(q)]$: immediate predecessor edges of node q

Assume that we have computed $N(S^k_{p_q[1]})$, ..., $N(S^k_{p_q[NP(q)]})$. We use this information to compute $B^{k+1}_{e_q[1]}$, $B^{k+1}_{e_q[2]}$, ..., $B^{k+1}_{e_q[NP(q)]}$. The following algorithm assigns labels for each edge in the graph.

Algorithm:
for $node = 1$ to M
  if $NP(node)$ equals 1
    for $length = 1$ to L
      $B^{length}_{e_{node}[1]} = 0$
  else
    for $j = 1$ to $NP(node)$
      $B^1_{e_{node}[j]} = j - 1$
    for $length = 2$ to L
      $label = 0$
      for $j = 1$ to $NP(node)$
        if ($label < N(S_{node}^{length})$)
          $B^{length}_{e_{node}[j]}$ = label
        else
          $B^{length}_{e_{node}[j]} = 0$
        $label = label + N(S^{length-1}_{p_{node}[j]})$

Figure 3 illustrates this algorithm for labeling edges in the graph corresponding to $c17$ (with $L = 3$).

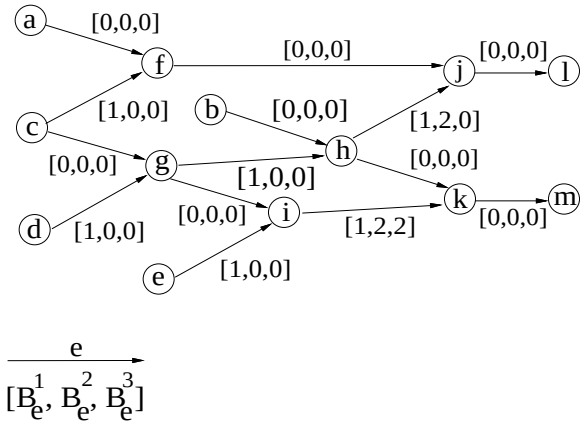

Figure 3: Generating edge labels in $c17$

2.3 Segment Labeling

For a given segment length L and a specific node q, our aim is to label the segments ending at q by 0, 1, ..., $N(S_q^L)$. The label for a segment is obtained by adding specific labels along its edges. For the ith edge along a segment, the $(L - i + 1)$th label of the edge is chosen. Consider the labeling of segments of length 2 that terminate at node j in the graph. Since $N(S_j^2)$ equals 4, the corresponding segment labels should be 0, 1, 2, and 3. This is obtained by adding the appropriate edge labels along a segment. Labeling of segment *j-h-b* is shown below:

Segment	Edge labels	Segment Label
j-h-b	$B^2_{e_j[2]} + B^1_{e_h[1]} = 2 + 0$	2

We make one backward pass over G to compute all edge labels. The total number of operations at each node is L. The complexity of the algorithm is thus $O(M \times L)$. Suppose, in practice, L were not to exceed a small constant, the run-time of the algorithm will be $O(M)$. Note that the uniqueness of segment labels is with respect to a node and segments ending at different nodes can have the same label.

3 Robust Segment Fault Simulation

Definition [8]: *A test for a segment delay fault is said to be robust if it guarantees detection of the fault irrespective of delays of all other signals in the circuit.*

We make the assumption that a segment delay fault, when present, has an increase in delay, enough to cause a delay fault on all paths that include the segment. Using a DAG model for a circuit, Figure 4 illustrates the procedure for generating a robust test for a fault associated with segment S (*e-f-g-h*). Proofs associated with robust tests can be found in our earlier work [8]. We divide the graph into three parts: part A consists

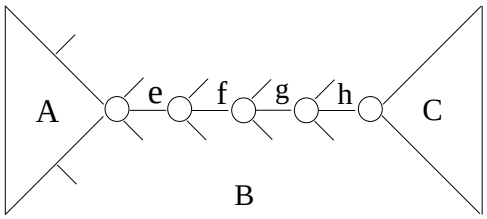

Figure 4: Generating a test for a fault associated with segment S (*e-f-g-h*)

of all nodes (and associated edges) that have a path to e, part C consists of all nodes (and associated edges) that are reachable from h, and part B consists of the remaining nodes and edges. Robust tests can be of two types:

1. *Standard-robust (S-R) tests*: The phases involved in generating a S-R test for a fault associated with S are: (a) *Transition launching:* requires e, the origin of S, to have the desired transition (with or without a hazard). (b) *Fault excitation and fault effect propagation:* requires robust propagation of the transition along a subpath originating at e, passing through S, and ending at a PO. Conditions for robust propagation have been defined by Smith [4] and Lin and Reddy [5].
2. *Reconvergent-robust (R-R) tests*: The transition launching phase for R-R tests is the same as that for S-R tests. Fault excitation requires robust propagation, as defined by Smith, of the transition along S. Fault effect propagation requires propagating the fault effect in part C of the graph using rules that are more relaxed than Smith's rules. We illustrate with an example. Consider an AND gate in part C of the graph with inputs y and z and output w. Assume that y has a falling transition and is carrying the robustly propagated fault effect of a segment fault S. Similarly, if z is also carrying the robustly propagated fault effect and has a falling transition, then the fault effect is robustly propagated to w. This is because there is at least one partial path ending at y and at least one ending at z that must have passed through S. Since, by our assumption, both these paths are delayed if S has a delay fault, the transition at w is guaranteed to be delayed. Propagation of two falling transitions at the inputs of an AND gate to its output is not considered robust [4].

We use a 10-valued algebra [8] for simulation with the following notation for a pair of values on a line:
S00 : Steady 0 with no static hazard
B00 : 0 on both vectors (can have a static hazard)

F10 : Falling transition that has passed through the fault site and carrying the fault effect
P10 : Robustly propagated [4] falling transition
T10 : Falling transition that has not been robustly propagated
The values S11, B11, F01, P01, and T01 are defined in a similar manner with 1's replacing 0's and the word "rising" replacing "falling" in the above notation. The values F10 and F01 are useful only when applying reconvergent-robust conditions. Propagation tables for gates can be found in our earlier work [8]. During simulation, at each gate, we keep track of its inputs, whose signals, propagate robustly [4] to its output. After simulation of each vector pair, we first determine all segment faults which are detected by the standard-robust rules. We perform a depth-first backward search starting at POs. For each gate, we trace back along its inputs whose signals have robustly propagated to its output for that vector pair. For each gate whose output signal has propagated robustly to a PO, we mark the gate as *detected* and determine detected segment faults of length L (an input parameter) ending at that gate. During the back trace, edge labels that are generated at the start of the program are added up and if a segment fault is detected, the flag corresponding to the segment label is marked. RF_q^k denotes the flag corresponding to the kth segment fault ending at a node q with a rising transition on q. The corresponding flag for a falling transition is denoted as FF_q^k. The flags provide easy access to previously detected segment faults. Figure 5 illustrates the simulation procedure using standard-robust rules. The nodes represent PIs, NAND gates, and POs. The nodes *detected* during our

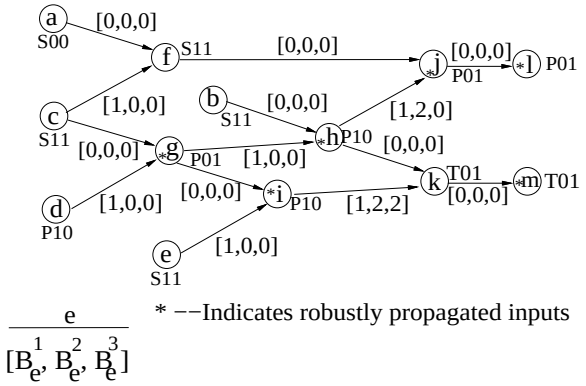

Figure 5: Standard-robust simulation of $c17$

backtrace are l, j, h, g, and d. Assuming L to be 3, the detected segments are:

Segment	Segment Label	Flag marked
l-j-h-g	3 (ending at l)	RF_l^3
j-h-g-d	1 (ending at j)	RF_j^1

Note that different segment flags are maintained with respect to each node. We also keep track of detected path faults whose lengths are less than L by using path labels [10].

Next, we determine additional detections, if any, of segment faults by the reconvergent-robust rules. We first simulate segment faults of length 1. We identify gates that satisfy two conditions: (1) they have a transition propagated (not necessarily robust) to a PO. (2) they have not been *detected* for the vector pair being simulated. Faults associated with segments that end at such gates are possible candidates for detection. We simulate the corresponding segment fault of length 1 associated with these gates, one at a time. If the value of the gate is P10 (P01) or T10 (T01), it is changed to F10 (F01). *If any primary output has a F10 or a F01, the simulated fault is considered to be robustly tested.* We mark the gate as *detected* and determine detected segment faults of length L ending at that gate. Figure 6 illustrates the simulation procedure. Assume that L is 3. Note that, for this vector pair, no segment fault of

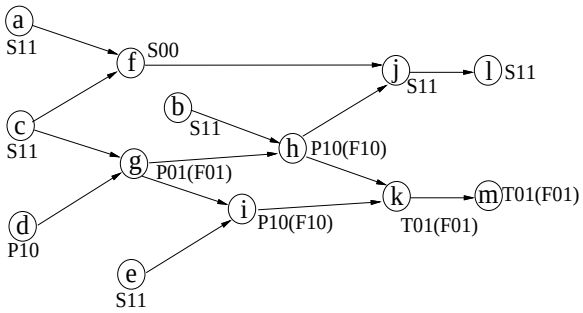

Figure 6: Reconvergent-robust simulation of $c17$ (values in parentheses)

length 3 is detected by the standard-robust rules. The gates which satisfy the above two conditions in this case are: h, i, g, and d. Simulation for gate g is shown. New simulation values are shown in parentheses. Since m, a PO, has the value $F01$, the simulated fault is considered *detected*. However, there is no segment fault of length 3 that ends at g, and hence, no fault is detected on this vector pair.

4 Results

We carried out experiments on ISCAS-85 and the combinational parts of ISCAS-89 benchmark circuits. The program, $SIGMA$, was implemented in C++ and run on a HP 9000 J200 workstation.

Table 1: Number of robustly detected segment faults (50,000 random vectors)

Circuit Name	# of robustly detected faults for segment length L					
	$L = 1$	$L = 2$	$L = 3$	$L = 4$	$L = 5$	$L = 10$
c880	**1424**	**1858**	**2428**	**2978**	**3634**	**4006**
	1388	*1793*	*2305*	*2791*	*3383*	*3725*
c1355	**740**	**927**	**1197**	**1370**	**1568**	**605**
	241	*193*	*145*	*96*	*92*	*72*
c1908	**2072**	**2732**	**3504**	**4326**	**5199**	**6703**
	1863	*2487*	*3216*	*3972*	*4720*	*5990*
c2670	**2405**	**3023**	**3843**	**4525**	**4914**	**4527**
	2365	*2978*	*3792*	*4469*	*4858*	*4501*
c3540	**4044**	**5333**	**6650**	**7805**	**8703**	**8421**
	2930	*3765*	*4629*	*5421*	*6095*	*6464*
c5315	**6124**	**8724**	**11622**	**13541**	**14860**	**12626**
	5955	*8487*	*11322*	*13133*	*14370*	*12421*
c6288	**715**	**706**	**658**	**598**	**528**	**237**
	428	*447*	*445*	*422*	*399*	*194*
c7552	**5844**	**7834**	**10252**	**12699**	**14865**	**10537**
	5587	*7496*	*9836*	*12221*	*14364*	*10392*
s15850	**21338**	**23986**	**26809**	**29609**	**31508**	**32520**
	20805	*23291*	*25961*	*28675*	*30505*	*31504*
s35932	**44821**	**47023**	**50647**	**49298**	**48434**	**39854**
	44821	*47023*	*50647*	*49298*	*48434*	*39726*
s38417	**55232**	**64570**	**75248**	**87290**	**100550**	**134964**
	53843	*62869*	*73186*	*84841*	*97707*	*131300*
s38584	**56923**	**72172**	**86010**	**95716**	**100363**	**79426**
	55772	*70620*	*84138*	*93581*	*98096*	*78512*

†Numbers in **bold** show the total # of robustly detected segment faults. Numbers in *italics* show the total # of robustly detected segment faults that are part of robustly detected path faults.

Table 1 shows results of two simulation experiments for varying segment length L (an input parameter to the program). For conciseness, results are presented for a subset of all possible values that L can take. A set of 50,000 random vectors (49,999 vector pairs) was used in our experiments. The first set of numbers in **bold** shows the total number of robustly detected segment faults. Simulation was performed using standard-robust and reconvergent-robust rules. The second set of numbers in *italics* show the total number of robustly detected segment faults that are part of robustly detected path faults. We expect that the second set of numbers will closely match the number of robustly detected segment faults had the tests for the detected path faults been generated by a path fault ATPG.

For most circuits, a large number of robustly detected segment faults are not part of any robustly detected path fault for the simulated vector set (refer to Figure 1 for illustration). This number is very large for $c3540$. In contrast, over 99% of all robustly detected segment faults in $s35932$ are part of robustly detected path faults. However, in absolute terms, almost all circuits had many robustly detected segment faults that were not part of any robustly detected path fault.

Table 2 shows more results for the same set of random vectors. There are three rows corresponding to each circuit. The first row of numbers indicates the number of robustly detected segment faults using standard-robust rules only, i.e., reconvergent-robust rules were not applied. Note that in most circuits, these numbers closely match the numbers shown in Table 1 (shown in bold). This indicates that the number of additional detections obtained by applying reconvergent-robust rules was not significant in most circuits. However, in $c3540$, the difference in these numbers appear to be significant. This indicates that the reconvergent-robust rules may be useful in some circuits. The numbers in the second row corresponding to each circuit indicate the corresponding segment fault coverages (standard robust rules). The third row of numbers indicates the CPU time in seconds taken for simulation. Results for path delay faults are also shown in the last column of the table. The path delay simulator used was our implementation of a technique that was reported by Pomeranz et al. [10]. The segment fault coverages

Table 2: # of Robustly detected Segment Faults–50,000 random vectors (Standard-robust rules)

Circuit	# of robustly detected faults for segment length L						Path faults
Name	$L=1$	$L=2$	$L=3$	$L=4$	$L=5$	$L=10$	
c880	1423	1851	2405	2948	3597	4005	1760
	94.2%	90.3%	83.1%	73.7%	61.6%	18.5%	10.1%
	53s	55s	57s	58s	59s	60s	193s
c1355	740	927	1197	1370	1568	605	54
	33.7%	25.3%	18.3%	11.1%	6.7%	0.1%	0.0006%
	59s	61s	61s	62s	62s	62s	212s
c1908	2071	2732	3504	4326	5199	6703	1689
	67.9%	55.2%	44.2%	33.4%	24.9%	6.2%	0.1%
	79s	82s	86s	84s	84s	85s	333s
c2670	2403	3022	3843	4525	4914	4527	2275
	54.2%	47.4%	39.4%	31.9%	25.1%	5.67%	0.1%
	190s	194s	200s	198s	199s	204s	669s
c3540	3167	4058	5003	5869	6604	6979	2631
	53.4%	44.8%	35.2%	27.0%	20.2%	2.9%	0.005%
	143s	140s	142s	144s	143s	145s	577s
c5315	6124	8724	11622	13541	14860	12626	7184
	67.9%	62.7%	54.2%	44.7%	36.0%	7.8%	0.2%
	306s	314s	317s	320s	330s	330s	1085
c6288	676	671	629	577	517	236	116
	6.9%	3.9%	2.0%	1.1%	0.57%	0.01%	6×10^{-17}%
	240s	235s	244s	237s	236s	238s	780s
c7552	5844	7834	10252	12699	14865	10537	4804
	46.7%	37.8%	28.1%	21.6%	16.3%	2.2%	0.3%
	438s	440s	436s	438s	448s	442s	1506s
s15850	21322	23976	26797	29590	31491	32511	9862
	74.4%	65.9%	56.9%	48.4%	39.0%	8.8%	0.003%
	2018s	2150s	2290s	2340s	2469s	2758s	4596s
s35932	44821	47023	50647	49298	48434	39854	21743
	73.9%	61.6%	53.4%	44.0%	37.5%	15.2%	5.5%
	2966s	3041s	3113s	3139s	3175s	3352s	8577s
s38417	55193	64530	75195	87203	10447	134948	54613
	81.7%	75.0%	68.0%	59.2%	50.5%	18.5%	1.9%
	6018s	5368s	6477s	5600s	6818s	7703s	11220s
s38584	56920	72120	86009	95714	100361	79422	41051
	82.5%	75.8%	69.1%	61.0%	52.0%	19.2%	1.8%
	4482s	4044s	5102s	4280s	5129s	6603s	10203s

†First row of numbers corresponding to each circuit shows the total # of robustly detected faults. Second row of numbers shows the fault coverages. Third row of numbers shows the CPU time in seconds.

Table 3: Robust fault coverages using transition fault test sets

Circuit	Robust segment fault coverages for segment length L						Path fault coverages	
Name	$L=1$	$L=2$	$L=3$	$L=4$	$L=5$	$L=10$	Robust	Non-robust
c880	40.86%	29.15%	19.07%	12.63%	8.03%	0.88%	0.88%	8.9%
c1355	23.08%	14.35%	7.52%	3.36%	1.50%	0.01%	0.0003%	0.46%
c2670	28.06%	20.99%	14.86%	10.03%	6.66%	1.00%	0.05%	0.78%

for some ISCAS-85 circuits is low while the path fault coverages is extremely low. This may be due to the presence of untestable faults and also due to the use of random vectors for simulation. The fault coverages tend to drop as the segment length L is increased. A segment fault ATPG may obtain higher coverages, or else, non-robust segment testing can be considered.

We examine segment faults of length 10 in $c3540$:
(1) Robustly detected segment faults (S-R + R-R rules): 8421
(2) Robustly detected segment faults (S-R rules only): 6979
(3) Robustly detected segment faults that are part of robustly detected path faults: 6464
A difference of 515 detected faults between (2) and (3) is a result of relaxed conditions for transition launching (refer to Section 3). For a segment fault S in this set, we require a transition at the origin of S. Also, a sub-path passing through S and ending at a PO should be robustly tested [4]. There is no restriction on the propagation conditions up to the origin of S. In contrast, for S to be covered under the path delay fault model, an entire path passing through S has to be robustly detected. A difference of 1442 detected faults between (1) and (2) is a result of relaxed propagation conditions for reconvergent fault effects.

We also carried out an experiment with transition fault tests defined by Waicukauski et al. [1]. The vector set used for $c880$ had a 100% transition fault coverage. The vector set used for $c1355$ and $c2670$ had a 100% coverage of all testable transition faults. The maximum achievable robust path delay fault coverage is 93.05% for $c880$ and 1.11% for $c2670$ [6] (the maximum achievable coverage for $c1355$ is not known). Coverages for the segment delay model for varying segment length L and the path delay model are given in Table 3.

The difference in the transition fault coverage and the robust coverage for segment faults of length 1 is a result of imposing robust propagation requirements in the segment delay model. The transition fault model traditionally does not impose a robust propagation requirement. Results seem to indicate that tests for transition faults may not be very good for detecting faults of a distributed nature. It may thus be useful to consider segment faults when the number of path faults is very large or when a large number of segment faults are not covered by testable path faults.

5 Conclusion

The simulator for segment delay faults, $SIGMA$, accepts L, the segment length to be simulated, as an input parameter and efficiently computes robust coverages. Labels generated for each segment avoid storage of fault lists and allow efficient access to previously detected segment faults. Results seem to confirm the claims we had made about the advantages of the segment delay fault model. The number of segments which are not covered by tested path faults are significant and it is useful to test them robustly. The target fault list was manageable for most of the benchmarks. For practical purposes, we feel that the segment length L should be chosen carefully such that it is large enough to cover distributed defects, but small enough to avoid an explosion of the number of faults.

References

[1] J. A. Waicukauski, E. Lindbloom, B. K. Rosen, and V. S. Iyengar, "Transition Fault Simulation," *IEEE Design & Test of Computers*, vol. 4, pp. 32–38, Apr. 1987.

[2] V. S. Iyengar, B. K. Rosen, and J. A. Waicukauski, "On Computing the Sizes of Detected Delay Faults," *IEEE Trans. on CAD*, vol. 9, pp. 299–312, Mar. 1990.

[3] A. K. Pramanick and S. M. Reddy, "On the Detection of Delay Faults," in *Proc. International. Test Conf.*, pp. 845–856, Sept. 1988.

[4] G. L. Smith, "Model for Delay Faults Based Upon Paths," in *Proc. International. Test Conf.*, pp. 342–349, Nov. 1985.

[5] C. J. Lin and S. M. Reddy, "On Delay Fault Testing in Logic Circuits," *IEEE Trans. on CAD*, vol. 6, pp. 694–703, Sept. 1987.

[6] K. Fuchs, M. Pabst, and T. Rossel, "RESIST: A Recursive Test Pattern Generation Algorithm for Path Delay Faults Considering Various Test Classes," *IEEE Trans. on CAD*, vol. 13, pp. 1550–1561, Dec. 1994.

[7] W. Ke and P. R. Menon, "Synthesis of Delay-verifiable Combinational Circuits," *IEEE Trans. on CAD*, vol. 44, pp. 213–222, Feb. 1995.

[8] K. Heragu, J. H. Patel, and V. D. Agrawal, "Segment Delay Faults: A New Fault Model," in *Proc. VLSI Test Symp.*, Apr. 1996.

[9] M. Gharaybeh, M. Bushnell, and V. D. Agrawal, "An Exact Non-Enumerative Fault Simulator for Path-Delay Faults," in *Proc. International. Test Conf.*, Oct. 1996.

[10] I. Pomeranz, L. N. Reddy, and S. M. Reddy, "SPADES: A Simulator for Path Delay Faults in Sequential Circuits," in *Proceedings EURO-DAC*, pp. 428–435, Sept. 1992.

[11] S. Bose, P. Agrawal, and V. D. Agrawal, "Path Delay Fault Simulation of Sequential Circuits," *Trans. VLSI Systems*, vol. 1, pp. 453–461, Dec. 1993.

[12] I. Pomeranz and S. M. Reddy, "An Efficient Non-Enumerative Method to Estimate Path Delay Fault Coverage," in *Proc. International. Conf. CAD*, pp. 560–566, Nov. 1992.