On Test Coverage of Path Delay Faults

Ananta K. Majhi and James Jacob
Dept. of Elec. and Comm. Eng.
Indian Institute of Science
Bangalore 560 012, India
majhi,james@ece.iisc.ernet.in

Lalit M. Patnaik
Microprocessor Applications Lab.
Indian Institute of Science
Bangalore 560 012, India
lalit@micro.iisc.ernet.in

Vishwani D. Agrawal
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974
va@research.att.com

Abstract

We propose a coverage metric and a two-pass test generation method for path delay faults in combinational logic circuits. The coverage is measured for each line with a rising and a falling transition. However, the test criterion is different from that of the slow-to-rise and slow-to-fall transition faults. The test, called "line delay test", is a path delay test for the longest sensitizable path producing a given transition on the target line. The maximum number of tests (and faults) is limited to twice the number of lines. However, the line delay test criterion resembles path delay test and not the gate or transition delay test. Using a two-pass test generation procedure, we begin with a minimal set of longest paths covering all lines and generate tests for them. Fault simulation is used to determine the coverage metric. For uncovered lines, in the second pass, several paths of decreasing length are targeted. We present a theorem stating that a redundant stuck-at fault makes all path delay faults involving the faulty line untestable for either a rising or falling transition depending on the type of the stuck-at fault. The use of this theorem considerably reduces the effort of delay test generation. We give results on benchmark circuits.

1 Introduction

At least three types of fault models have been used to represent delay defects. *Transition fault* model [11] is a *qualitative* model and assumes a lumped delay defect (i.e., slow-to-rise or slow-to-fall) at an input or output of a gate. Carter *et al* [1] have introduced a *quantitative gate delay* fault model in which the delay through a gate is represented by intervals. A fault in this model is an added delay of certain amount (referred to as the *size* of the fault) in the rising or falling transition at the gate input or output. In *path delay fault* model [10], the cumulative effect of gate delays along a path from PIs to POs of the combinational logic is considered. The number of transition or gate delay faults is linearly proportional to the number of gates in the circuit, but the number of path faults can be exponential. Despite efforts to classify path faults and identify a subset that must be tested, their number remains a problem [4].

In this paper, we define a *rising line delay test* that sensitizes the longest sensitizable path passing through the target line producing a rising transition on it. Similarly, a falling line delay test is defined. The definition of "longest" can be appropriately chosen. For example, in the simplest case, it can be the path with largest number of gates. Alternatively, gates can be weighted by their nominal delays. However, once the path is selected, the test generation is independent of gate delays. The criterion of delay test through the longest path has been used for diagnosis [5].

The coverage is measured for all lines with two possible transitions. Thus, the maximum number of faults (or tests) is twice the number of lines. Yet, test criterion is similar to path delay fault, and not like gate or transition delay fault. In general, a test will cover several lines. This coverage methodology can also be applied to the reported methods that extract sensitizable paths [2, 3].

An iterative approach for generating a robust test was first proposed by Park and Mercer [8]. They devised an *approximate method* where the search space of test generation process is biased to find a test along a path whose propagation delay is greater than or equal to a predefined threshold value. Our approach is to use an *exact method* for generating a test for the longest robustly testable path through each line.

2 Two-Pass Test Generation

Finding the longest *sensitizable* and robustly testable path through a given delay fault site is an NP-hard problem [8]. We first attempt to find a robust test for the longest structural path through a line. If the path is not sensitizable, then we try to find a robust test for the next longest structural path, and so forth, until a test for the longest sensitizable path is found. Given enough resources this method guarantees a test for the longest sensitizable path through the line if such a test exists.

The first pass of our two-pass test generation strat-

1063-9667/95 $04.00 © 1995 IEEE

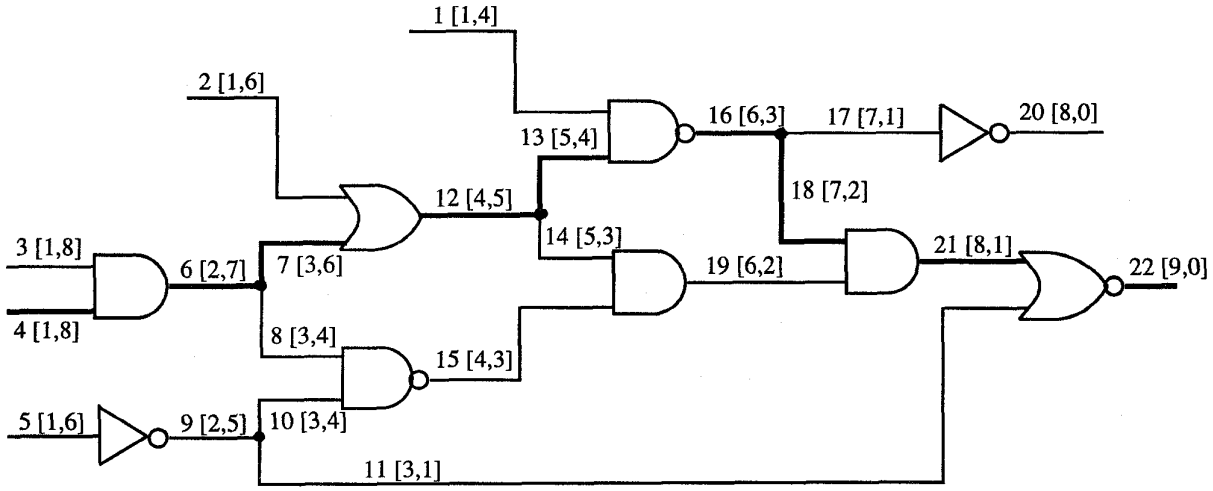

Figure 1: Test generation for longest path through line 4

egy is essentially the same as reported in [7]. Initially a simple path selection method is employed to obtain a list of paths that cover all signal lines by their respective longest structural paths. The multiple backtrace procedure employing a 9-value logic system [7] is used to derive robust tests for these targeted path faults. Once a robust test is generated, fault simulation is carried out to obtain information on the robust detection of other path faults. Whenever the fault simulator finds that a path is robustly tested by the generated test vector pair, each line on this path is examined to see if the vector pair satisfies the criterion of being a line delay test for any other line on that path. If the robustly tested path happens to be the longest structural path in the circuit through any line, then this line can be marked as *covered* since a line delay test has been obtained for the line with respect to a rising/falling transition. The fault coverage includes lines and transitions for which line delay tests were obtained.

The line delay fault coverage at the end of the first pass is generally low since many structural paths are not sensitizable. For each line that is not covered by the line delay tests of the first pass, we attempt a robust test for the second longest structural path. If a robust test exists for this path, we mark the line as covered. If a test is not possible for the second longest structural path, then we go for the third, fourth, etc., successively shorter paths till we get a robust test. Again fault simulation is employed after each vector pair is derived to determine the coverage metric. This strategy usually obtains significantly improved line delay fault coverage after the second pass.

Example 1: Consider the circuit given in Figure 1. Lines are numbered 1 through 22. The label of line l is $l[m,n]$, where m and n are *level* and *depth*, respectively. These are the maximum distances (in terms of the number of logic levels) from primary input and primary output. We generate a line delay test for falling transition on line 4. In the first pass the longest structural path through line 4 is enumerated as 4-6-7-12-13-16-18-21-22. However, our multiple backtrace procedure [7] will find that no robust test exists for this path and hence line 4 remains as *not covered* at the end of the first pass. In the second pass, we enumerate the second longest structural path through line 4 as 4-6-7-12-13-16-17-20. A robust test is obtained for this path using our test generator and hence this test will be a line delay test for line 4, which is now marked as *covered.* □

3 N-Longest Path Selection

A polynomial time algorithm is known for finding the minimal longest path cover for all lines [6]. However, in the absence of a simple algorithm for the N-longest path problem, we enumerate all possible paths through the target line L, order the paths according to decreasing length, and select the top N paths.

We first trace backward in a breadth first manner from line L towards PIs and mark all signal lines from which there is a path to L. We then trace forward from line L in a breadth first manner towards POs and mark all signal lines that can be reached from L. For each PI that has been marked in the backward trace from L, we enumerate all paths that start at the PI and pass through L, by traversing depth first along only the marked lines. As each path is enumerated, we store it in a linked list in decreasing order of path lengths. If the total number of possible paths through line L is greater than N, then

we insert the $(N+1)th$ path in the ordered list at the appropriate position and remove the last path of the list to maintain N longest paths through line L.

Example 2: Consider the circuit given in Figure 1. We illustrate the N-longest path selection procedure for line 4. Since line 4 is a primary input, we trace forward towards the POs for marking the lines that can be reached from PI 4. There are only 4 possible paths that can be enumerated from line 4. These paths are sorted with respect to the path lengths and stored in an ordered list. The longest path P_1(4-6-7-12-13-16-18-21-22) has a length 9. The next longest paths, P_2(4-6-7-12-13-16-17-20) and P_3(4-6-7-12-14-19-21-22), have lengths 8 and the last path P_4(4-6-8-15-19-21-22) has a length 7.

4 Elimination of redundant path faults

The following theorem relates to the identification of redundant path delay faults.

Theorem: Consider an untestable (redundant) stuck-at-0 (stuck-at-1) fault on line k in a logic circuit. Then all path delay faults through line k (and hence the line delay fault on line k) for which a rising (falling) transition reaches line k will be untestable.

Proof: We consider an untestable stuck-at-0 fault on line k. The proof for the opposite case is analogous. Since the stuck-at-0 fault on line k is untestable, the logic function realized by the circuit is unaltered when we replace the logic value on line k with a constant 0. Replacing the logic value to a constant 0 can also be viewed as a rising transition, due to arrive at line k, which is infinitely delayed (line k never attains the value 1 and hence is "stuck" at logic 0). Thus if the stuck-at-0 on line k does not alter the good circuit behavior (does not cause an incorrect logic value at the output), then an infinitely delayed rising transition on line k also can not cause an incorrect logic value at the circuit output. Hence all path delay faults through line k for which a rising transition arrives on line k will be untestable in the circuit. □

5 Experimental Results

We have implemented the two-pass test generation algorithm in the C language (about 4000 lines of code) on an IBM RS-6000/580 workstation. Table 1 gives the results for some ISCAS'85 and the scan-hold versions of ISCAS'89 benchmarks. *Total LDF* is the total number of line delay faults and is twice the number of lines in the circuit. *Red. Flts.* gives the number of redundant stuck-at faults obtained by COMPACTEST [9] which are used to avoid test generation for redundant path delay faults. Fourth column *Target Paths* gives the number of logical paths considered for test generation in the first pass. This is twice the number of the physical paths selected to cover each line via the longest path. Fifth column *Vec.* gives the number of robust tests generated in the first pass within a backtrack limit of 100. We have a fault simulator in the test generation system. Robustly detected paths are immediately marked in the targeted path list and hence not considered for further test generation. Sixth column, *Paths Tested*, gives the total number of path faults detected robustly from the *Target Paths* as well as from all other path faults, as reported by the fault simulator. *LDF Cov.* (seventh column) gives the number of line delay faults (LDF) detected in the first pass. The CPU time (in seconds) is given for the first pass in eighth column. Ninth column *Vec.* gives the number of additional robust test vectors generated in the second-pass of test generation. For the second pass, we have enumerated up to 100 longest paths through each line though more paths may exist for some lines. *Paths Tested* gives the number of additional paths tested robustly at the end of the second pass. The eleventh column *LDF Cov.* gives the number of new line delay faults detected in the second pass. The twelfth column *Final LDF Cov.%* gives the total line delay fault coverage obtained at the end of two-pass test generation. The CPU time (in seconds) in the last column is for the complete ATPG process including both passes.

For example, we have initially targeted 4404 longest paths (Target Paths) for circuit c5315. There is a total of 10630 line delay faults which is twice the number of lines in the circuit. In the first pass of the test generation process, 2341 robust test vectors are generated. After simulation with these vectors, we found that 4630 path delay faults in the circuit are detected robustly and 4775 line delay faults (LDF) are detected in the first pass of the ATPG process corresponding to a line delay fault (LDF) coverage of 45%. After the two-pass test generation process, we obtain another 1129 extra robust tests which in turn detect an additional 4278 path faults robustly and 3657 new line delay faults, giving a total coverage of 79.3%. The total time taken for the complete test generation process is 4779 seconds. The line delay fault coverage is less than 100% in many circuits primarily due to the backtrack limit employed by us in the test generation process to keep the time complexity manageable. Further, many circuits have a large number of untestable line delay faults.

There are 131 redundant stuck-at faults in c7552 circuit. By employing this information for eliminating redundant path delay faults, we completed the two-pass test generation method in 12073 seconds (CPU time on IBM RS-6000/580 workstation) as given in Table 1. It took 32457 seconds without incorporating the result of Section 4. COMPACTEST took only 85 seconds to identify redundancies in c7552.

Table 1: Two-pass test generation results for ISCAS benchmark circuits

Circuit	Total LDF	Red. Flts.	Pass I					Pass II			Final	Total
			Target Paths	Vec.	Paths Tested	LDF Cov.	CPU s *	Vec.	Paths Tested	LDF Cov.	LDF Cov. %	CPU s *
c880	1760	0	744	500	852	1243	8	92	391	499	98.9	18
c1355	2710	8	1100	0	0	0	48	25	42	131	4.8	5920
c1908	3816	9	1286	62	116	132	240	358	633	1657	46.9	2512
c2670	5340	117	2046	642	1175	1465	146	546	1221	1750	60.2	916
c3540	7080	141	2584	57	180	154	368	243	928	1064	17.2	2513
c5315	10630	59	4404	2341	4630	4775	1046	1129	4278	3657	79.3	4779
c6288	12576	34	4832	27	42	36	1444	137	363	379	3.3	2223
c7552	15104	131	5480	1182	2779	1052	3084	2483	9867	7181	54.5	12073
s27	54	0	24	14	43	49	0.1	2	4	6	100	0.1
s344	688	0	270	161	327	554	1.7	27	47	115	97.2	2.2
s349	698	2	276	157	324	546	1.6	29	49	127	96.4	2.2
s382	764	0	350	193	448	721	2.0	15	29	40	99.6	2.2
s400	800	6	370	187	440	720	2.3	13	22	57	97.1	2.6
s444	888	14	396	159	273	500	2.9	86	197	353	96.1	4.2
s526	1052	1	558	324	556	932	4.7	12	19	86	96.7	5.3
s641	1282	0	406	223	403	695	9.2	110	449	577	99.2	24.3
s953	1906	0	800	502	1742	1837	15.1	20	61	59	99.5	16.8
s1238	2476	69	1130	443	1370	1383	95.2	292	860	1043	97.9	229.9
s5378	10756	41	3322	1954	4240	7763	381	587	1989	2052	91.3	862.9
s13207	26414	**	8008	3020	5571	15201	3776	663	1790	3181	69.6	23709
s15850	31700	**	9114	2912	5172	13266	3700	1497	3852	5995	60.8	37152

* IBM RS-6000/580 ** COMPACTEST did not identify any redundant faults

6 Conclusion

The new coverage metric requires a pair of robust tests termed as *line delay tests* for each line in the circuit, one for the rising and the other for the falling transition on the line. The maximum number of faults (and tests) is limited to twice the total number of lines in the circuit. In the first pass of test generation process, we begin with a minimal set of longest paths covering all lines and generate robust tests for them. Fault simulation is used to determine the line delay fault coverage. The second pass considers those lines for which line delay tests could not be generated in the first pass, and attempts to generate robust tests for successively shorter paths through these lines, till a test for the longest sensitizable path is found. We have also employed information on redundant stuck faults in a circuit to avoid test generation for a large number of redundant path faults.

References

[1] J. L. Carter, V. S. Iyengar, and B. K. Rosen, "Efficient Test Coverage Determination for Delay Faults," in *Proc. Int'l Test Conf.*, 1987, pp. 418-427.

[2] H. Chang and J. A. Abraham, "VIPER: An Efficient Vigorously Sensitizable Path Extractor," *Proc. 30th Design Automation Conf.*, June 1993, pp. 112-117.

[3] H.-C. Chen and D. Du, "Path Sensitization in Critical Path Problem," *IEEE Trans. CAD*, vol. 12, pp. 196-207, February 1993.

[4] M. A. Gharaybeh, M. L. Bushnell, and V. D. Agrawal, "Classification and Test Generation for Path-Delay Faults Using Single Stuck-Fault Tests," in *Proc. Int'l Test Conf*, October 1995.

[5] P. Girard, C. Landrault, and S. Pravossoudovitch, "An Advanced Diagnostic Method for Delay Faults in Combinational Faulty Circuits," *Jour. Electronic Testing: Theory and Applic.*, vol. 6, No. 3, pp. 277-294, June 1995.

[6] W.-N. Li, S. M. Reddy, and S. Sahni, "On Path Selection in Combinational Logic Circuits," *IEEE Trans. CAD*, vol. 8, pp. 56-63, January 1989.

[7] A. K. Majhi, J. Jacob, L. M. Patnaik, and V.D. Agrawal, "An Efficient Automatic Test Generation System for Path Delay Faults in Combinational Circuits," in *Proc. 8th Int'l Conf. on VLSI Design*, January 1995, pp. 161-165.

[8] E. S. Park and M. R. Mercer, "An Efficient Delay Test Generation System for Combinational Logic Circuits," *IEEE Trans. on CAD*, vol. 11, pp. 926-938, July 1992.

[9] I. Pomeranz, L. N. Reddy, and S. M. Reddy, "COMPACTEST: A Method to Generate Compact Test Sets for Combinational Circuits," *IEEE Trans. on CAD*, vol. 12, pp. 1040-1049, July 1993.

[10] G. L. Smith, "Model for Path Delay Faults Based on Paths," *Proc. Int'l Test Conf*, 1985, pp. 342-349.

[11] 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, April 1987.