Delay Fault Models Based on Robust Propagation of Signal-Transitions +

Irith Pomeranz and Sudhakar M. Reddy
Electrical and Computer Engineering Department
University of Iowa
Iowa City, IA 52242, U.S.A.

Abstract

We define a family of delay fault models to address several issues, including the number of delay faults, and the need to capture different defect mechanisms leading to delay faults. Under the kth model, we consider faults affecting subsets of k lines in the circuit. We describe a test generation procedure for the proposed models under robust propagation conditions using $k = 1$ and $k = 2$. We present experimental results to demonstrate the effectiveness of the procedure and the models. The models can be used alone or in addition to other delay fault models.

1. Introduction

Several delay fault models were proposed to model defects that change the timing behavior of a circuit [1]-[3]. The path delay fault model [3] models distributed as well as localized excessive delays. However, the number of paths (and therefore the number of path delay faults) in practical circuits may be very large [4]. In addition, the number of testable path delay faults may be very small [5]. Therefore, test sets to detect all the path delay faults may not exist, and test sets to detect all the detectable faults may not be computationally achievable.

To address the deficiencies of the models above, other models were proposed [6], [7]. In both models, as well as in the path delay fault model, tested lines are on the same path from the primary inputs to the primary outputs. However, defects that may increase circuit delay sometimes involve lines that do not drive each other. Crosstalk faults and high-resistance bridging faults [8] are examples of such faults. Such defects are not captured by the path delay fault model or its restrictions. Therefore, tests for these models may not detect such defects.

In this work, we define a family of delay fault models to address the issues above. We use a parameter k to describe a specific fault model. Under the kth model, we consider subsets of k lines in the circuit, each line associated with a signal-transition. Our goal is to ensure that for each subset of k lines and the associated signal-transitions, a fault involving all k lines will be detected. For example, when $k = 2$, we require that every fault involving a pair of lines and their signal-transitions will be detected. Under this model, detecting faults involving lines on the same path promotes the detection of path delay faults and of defects whose cumulative effects result in delay faults; while detecting faults involving lines on different paths promotes the detection of crosstalk faults and high-resistance bridging faults. Specifically, the faults whose detection is enabled are those where two lines carrying signal-transitions may interfere with each other causing one or both of the transitions to be delayed. Although the proposed models do not *guarantee* detection of path delay faults or crosstalk faults, tests generated for these models have the potential of showing improved coverage of these models. We demonstrate this point with respect to path delay faults.

We consider only robust tests in this work [9]. The importance of robust tests stems from the fact that they remain valid in the presence of arbitrary delays in the circuit. Thus, a robust test for a delay fault that increases the circuit delay by a sufficient amount is guaranteed to detect the fault. We concentrate on $k = 1$ and $k = 2$.

+ Research supported in part by NSF Grant No. MIP-9725053, and in part by SRC Grant No. 98-TJ-645.

2. Fault models and fault simulation procedures

In this section, we define fault models based on robust propagation of signal-transitions. In all the definitions, we consider only lines that are primary inputs, fanout branches or primary outputs. This is because these lines are sufficient to ensure robust propagation of signal-transitions through all other lines.

The *single line model* requires that a rising and a falling signal-transition would be robustly propagated from a primary input to a primary output through every line g in the circuit. We denote the set of tested single lines for a test t by $\Psi^1(t)$. For a test set T, we define $\Psi^1 = \bigcup \{\Psi^1(t): t \in T\}$.

The *line pair model* requires that for every pair of lines g_1, g_2 in the circuit, and for every combination of rising and falling transitions tr_1, tr_2 on g_1, g_2, respectively, tr_1 and tr_2 would be propagated through g_1 and g_2, respectively, by a test that robustly propagates signal-transitions from one or more primary inputs to one or more primary outputs through g_1 and g_2. We denote the set of tested line pairs for a test t by $\Psi^2(t)$. For a test set T, we define $\Psi^2 = \bigcup \{\Psi^2(t): t \in T\}$.

Simulation of the line k-subset model under a test set T proceeds as follows. Regardless of the value of k, the first step of the simulation procedure is to obtain, for every test $t \in T$, the set of lines $\Psi(t)$ through which signal-transitions are robustly propagated by t from the primary inputs to the primary outputs. Together with every line g in $\Psi(t)$, we maintain the signal-transition on g. The set $\Psi(t)$ can be obtained by performing one logic simulation pass over the circuit from the primary inputs to the primary outputs, and an additional pass from the primary outputs to the primary inputs.

In the second step of the simulation procedure, we omit from T tests such that the faults they detect are also detected by another test. This is done independent of k, as follows. Consider two tests t_1 and t_2 in T, such that $\Psi(t_2) \subseteq \Psi(t_1)$. In this case, every k-subset of lines detected by t_2 is also detected by t_1, and we omit t_2 from T.

The third step of the simulation procedure depends on the fault model being considered. For $k = 1$ (the single line model), we have $\Psi^1(t) = \Psi(t)$, and $\Psi^1 = \bigcup \{\Psi(t): t \in T\}$. For $k = 2$ (the line pair model), we find the set of line pairs $\Psi^2(t) = \Psi(t) \times \Psi(t)$ over every set $\Psi(t)$, and take the union of all the sets.

3. Test generation

From our experiments, it is possible to construct a test set T from random two-pattern tests, such that T would detect most of the single lines; however, relatively few line pairs are detected. The test generation procedure described in this section combines two tests out of the initial random test set, each detecting a different single line, into a test that detects the two lines together. This results in the detection of the line pair consisting of the two single lines. This procedure serves to demonstrate that it is possible to increase the line pair coverage at a relatively low computational cost.

Let t_1 be a test that detects a single line g_1 with a transition tr_1 on g_1. Let t_2 be a test that detects a single line g_2 with a transition tr_2 on g_2. Our goal is to combine t_1 and t_2 into a single test that detects both lines. The following input values are important for each one of the lines. For g_1, we have an output cone $O(g_1)$ that contains all the primary outputs driven by g_1. These primary outputs have an input cone denoted by $I(g_1)$ that contains all the primary inputs driving the outputs in $O(g_1)$. These primary inputs allow robust propagation of a signal-transition from a primary input through g_1 to a primary output. Similarly, we have the input cone $I(g_2)$ for g_2. If $I(g_1)$ and $I(g_2)$ are disjoint, we can take the values of $I(g_1)$ from t_1 and the values of $I(g_2)$ from t_2, and define a test that detects both faults. In general, $I(g_1)$ and $I(g_2)$ may overlap. We use heuristics to construct new tests based on t_1 and t_2 in this case.

4. Experimental results of test generation

We applied the proposed test generation procedure to the combinational logic of ISCAS-89 benchmark circuits. The results are reported in Table 1. We first show the number of tests in the initial random test set, followed by the number of tests obtained after test generation. We then show the total number of single lines, followed by the number of single lines detected by using the initial test set, and the number of single lines detected by the test set obtained after test generation. Finally, we show the total number of line pairs, the number of line pairs detected by using the initial test set, and the number of line pairs detected by the test set obtained after test generation.

Table 1: Results of test generation

	tests		single			pairs		
				detect			detect	
circuit	init	tg	total	init	tg	total	init	tg
s208	452	1248	244	81.56	100.00	29646	19.95	35.03
s344	1000	4214	402	88.06	96.77	80601	34.09	57.48
s420	692	6663	484	68.80	100.00	116886	13.41	44.95
s526	1000	21302	720	72.78	99.17	258840	17.99	58.70
s820	839	3522	1110	65.14	99.64	615495	3.19	6.82
s1196	1000	45890	1396	53.51	96.78	973710	6.47	37.45
s1423	1000	96822	1690	77.04	95.98	1427205	24.66	77.77
s1488	1000	7836	1720	69.36	98.60	1478340	3.56	9.19
s5378 *	1000	133049	1000	54.40	96.40	499500	8.78	82.33
s9234 *	1000	85565	1000	36.10	78.50	499500	7.36	50.29

* Only 1000 lines are considered

From Table 1, it can be seen that the proposed procedure can increase the number of single lines detected to 100% or close to it. It also increases significantly the number of line pairs detected. The percentage of detected line pairs is smaller than 100% partly because some line pairs are undetectable.

We also considered robust detection of path delay faults. We recorded the fault coverages with respect to all the paths and with respect to the longest paths after 1000 tests were applied, and after the complete test set was applied. In many circuits, 1000 tests correspond to the initial random test set. Thus, the data allows us to see the increase in path delay fault coverage due to test generation. In Table 2, we show the coverage of path delay faults after applying 1000 tests, and after applying the complete test set. Next, we show the coverage of longest paths after applying 1000 tests, and after applying the complete test set. Since for some circuits, the longest paths are untestable, we show in the last column of Table 2 the length of the longest tested path after applying 1000 tests, and after applying the complete test set.

Table 2: Path delay fault coverage

	fault coverage				longest	
	all paths		longest		detected	
circuit	1000	all.t	1000	all.t	1000	all.t
s208	89.31	99.31	50.00	100.00	14	14
s344	49.44	85.21	0.00	100.00	17	20
s420	50.00	95.66	0.00	50.00	16	28
s526	47.93	84.63	0.00	0.00	8	8
s820	67.99	99.59	4.55	95.45	10	10
s1196	10.60	57.04	0.00	0.00	18	23
s1423	1.34	7.71	0.00	0.00	18	33
s1488	61.17	97.45	0.00	100.00	15	17

From Table 2 it can be seen that the path delay fault coverage increases significantly as additional tests are generated by the proposed procedure. This includes an increase in the number of detected faults associated with longest paths in the circuit.

References

[1] Z. Barzilai and B. Rosen, "Comparison of AC Self-Testing Procedures", in Proc. 1983 Intl. Test Conf., pp. 89-94.

[2] J. D. Lesser and J. J. Schedletsky, "An experimental delay test generator for LSI logic", IEEE Trans. Computers, March 1980, pp. 235-248.

[3] G. L. Smith, "Model for delay faults based upon paths", in Proc. 1985 Intl. Test Conf., Nov. 1985, pp. 342-349.

[4] I. Pomeranz and S. M. Reddy, "An Efficient Non-Enumerative Method to Estimate Path Delay Fault Coverage", in Proc. Intl. Conf. on Computer-Aided Design, 1992, pp. 560-567.

[5] C. J. Lin and S. M. Reddy, "On delay fault testing in logic circuits," IEEE Trans. CAD, pp. 694-703, Sept. 1987.

[6] I. Pomeranz, S. M. Reddy and J. H. Patel "On Double Transition Faults as a Delay Fault Model", in Proc. 1996 Great Lakes Symp. on VLSI, March 1996, pp. 282-287.

[7] K. Heragu, J. H. Patel and V. D. Agrawal, "Segment Delay Faults: A New Fault Model", in Proc. 14th VLSI Test Symp., April 1996, pp. 32-39.

[8] S. Mandava, S. Chakravarty and S. Kundu, "On Detecting Bridges Causing Timing Failures", in Proc. Intl. Conf. on Computer Design, Oct. 1999, pp. 400-406.

[9] S. M. Reddy, M. K. Reddy and V. D. Agrawal, "Robust Tests for Stuck-Open Faults in CMOS Combinational Logic Circuits", in Proc. 14th Intl. Symp. on Fault-Tolerant Computing, June 1984, pp. 44-49.