

# Diagnosing Multiple Transition Faults in the Absence of Timing Information

Jiang Brandon Liu, Magdy Abadir  
High Perf. Tools and Meth.  
Freescale Semiconductor  
Austin, TX

{brandon.liu, magdy.abadir}@freescale.com

Andreas Veneris, Sean Safarpour  
Dept. Elec. & Comp. Eng.  
University of Toronto  
Toronto, ON

{veneris, ssafarpo}@eecg.toronto.edu

## ABSTRACT

As timing requirements in today's advanced VLSI designs become more aggressive, the need for automated tools to diagnose timing failures increases. This work presents two such algorithms capable of diagnosing multiple delay faults. One method uses multiple transition fault models and the other reasons with ternary logic values, thus achieving model-independent diagnosis. Experiments are conducted on ISCAS'85 combinational and full-scan version of ISCAS'89 sequential circuits corrupted with multiple transition faults. The performance of both algorithms are evaluated and compared. The results show good efficiency and diagnostic resolution.

## Categories and Subject Descriptors

J.6 [Computer-Aided Engineering]: Computer-aided design (CAD)

## General Terms

Algorithms, Performance, Verification

## Keywords

Diagnosis, Incremental, Multiple Faults, Transition Faults, Delay Faults

## 1. INTRODUCTION

Advances in today's VLSI technology poses stringent requirements on the timing of modern designs. Due to their complexity and aggressive clocking strategies, designs that function properly at low clock frequencies may fail at the rated speed [1, 4]. Accurate diagnosis of delay faults is an important problem faced by the industry. Direct probing mechanisms such as E-Beam probing are slow and expensive. Automated diagnosis tools are needed to reduce the search space for physical probing.

Much work has been done on delay fault testing. A comprehensive review can be found in [4]. Comparatively less work has been done on delay fault diagnosis. Most of it deals

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GLSVLSI'05, April 17–19, 2005, Chicago, Illinois, USA.  
Copyright 2005 ACM 1-59593-057-4/05/0004 ...\$5.00.

with single delay faults [7, 10]. Dastidar and Touba [3] describe a diagnostic algorithm based on critical path-tracing under six value algebra. Their work can handle multiple delay faults by computing the union of suspect fault paths of single ones.

In this work, we describe two algorithms for *multiple* delay (transition) fault diagnosis under an incremental framework. They are effect-cause simulation based approaches that achieve good diagnostic resolution even in the absence of device timing information. Furthermore, the diagnosis is conducted purely upon logic simulation information.

The main contributions of this paper are:

1. It devises a modeled incremental diagnosis algorithm for multiple transition faults.
2. It develops theory to guarantee the capture of all transition faults while efficiently pruning the diagnostic space.
3. It describes a complementary model-free diagnosis algorithm.
4. It evaluates and compares the efficiency of the two methods by experiments.

The experimental results on ISCAS'85 and full-scan versions of the ISCAS'89 benchmark circuits demonstrate the effectiveness and resolution of the proposed algorithms.

The remaining of the paper is organized as follows. In Section 2, we describe a diagnostic algorithm based on transition fault models. Section 3 briefly describes a model-free algorithm based on ternary logic simulation. The experimental results are presented in Section 4. Finally, Section 5 concludes this work.

## 2. DIAGNOSIS WITH TRANSITION FAULT MODEL

The diagnosis methodology is formulated under an incremental framework, which has been experimentally shown to be well suited for identifying multiple faults/errors in past work [5, 12]. During *incremental diagnosis*, the algorithm iteratively identifies suspect faulty locations one at a time. Upon the discovery of each location, it selects a suitable fault/error model which brings the logic behavior of the faulty circuit and its specification closer. Subsequent diagnostic runs are conducted on the resulting circuit(s). Figure 1 outlines its overall flow.

In this work, a transition fault model is used in the fault injection step. The transition fault model utilized is a common delay fault model. Two transition faults are associated

with each gate: a slow-to-rise fault and a slow-to-fall fault. It is assumed that in the fault-free circuit each gate has some nominal delay. Under the transition fault model, the extra delay caused by the fault is assumed to be large enough to prevent the transition from reaching any primary output at the time of observation. In other words, the delay fault can be observed independently of whether the transition propagates through a long or a short path to any primary output. Therefore, this model is also called the gross delay fault model [8]. It should be noted, the topic of path delay fault diagnosis [6] is not treated here.

Other popular fault models include gate delay model and path delay model. A gate delay fault model assumes that the delay fault is lumped at one gate in the circuit. However, unlike the transition model, the gate delay fault model does not assume that the increased delay will affect the performance independent of the propagation path through the fault site. It is assumed that long paths through the fault site might cause performance degradation. Since timing information is not used in this work, performance degradation cannot be measured.

Under the path delay fault model a combinational circuit is considered faulty if the delay of any of its paths exceeds a specific limit. A delay defect on a path can be observed by propagating a transition through the path [9]. In the context of binary logic simulation, multiple transition faults can be used to model path delay faults. Because timing information is not used during diagnosis, the only observable faulty effect is the failure in achieving a transition. Logic behavior of single or multiple path delay faults can be emulated by the appropriate placement of these transition faults.

Our algorithm accepts a gate-level specification and the logic behavior of a faulty chip, and produces a list of probing sites for test engineers. An example is used to illustrate the diagnosis approach and annotate the proper theory which follows the four steps of the paradigm (loop) shown in Figure 1. Throughout this example, a line is identified by the name of the gate that drives it. Further, we refer to a line under consideration by diagnosis as a *suspect line*. Any set of lines returned by diagnosis is called a *candidate tuple*.

Figure 2 (a) contains a circuit with a slow-to-fall (STF) and a slow-to-rise (STR) fault located on lines  $G_8$  and  $G_9$  respectively. To detect a transition fault in a combinational circuit it is necessary to apply two input vectors,  $V = \langle V_1, V_2 \rangle$ . The first vector,  $V_1$ , initializes the circuit, while the second vector,  $V_2$ , activates the fault and propagates its effect to some primary output. Fault diagnosis occurs after the chip fails for some test vector stimuli. In the figure, each line also carries one simulated fault-free/faulty vector pair [1]. For instance, the values on line  $O1$  in Figure 2 (a) mean that in a fault-free simulation, a 0 to 1 transition should occur. However, this transition did not occur in the faulty chip.

In this example, it is seen that both primary outputs are erroneous. Recall that in fault diagnosis, faulty values are the “desired behavior” we try to capture by injecting fault models at appropriate locations in our netlist. Note that the internal values of the faulty chip are not visible to the algorithm.

The algorithm starts with the good circuit simulation values (Figure 2 (b)). It performs path-trace [1] to quickly grade the lines in the circuit. Path-trace works as follows. It starts from an Erroneous Primary Output (EPO) and

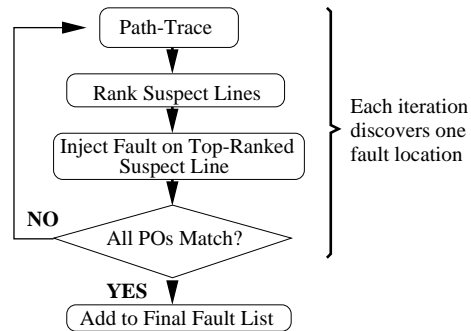


Figure 1: Algorithm flow.

pessimistically marks lines (non primary inputs) that may belong to a sensitized path (i.e. a path of lines with different logic values under the influence of some fault(s)). If the output of a gate has been marked and the gate has one or more fan-in(s) with controlling values, then the procedure marks all controlling fan-ins; if the gate contains no fan-ins with controlling values, then they are all marked as well; if a branch is marked, then the stem also gets marked. Note that multiple marking is possible.

In Figure 2 (b), path-trace marks lines with \*. It only reasons with vector  $V_2$  because this is the vector that excites the fault. Path-trace is conducted for multiple vectors. A line is considered *permanently marked* if it is marked by path-trace and a transition occurs. The lines that are permanently marked are compiled into a list. This list is sorted by how many times the line is marked.

This list is further shortened by the following theorem:

*Theorem:* In a circuit corrupted with  $N$  faults, suppose path-trace is conducted over  $V$  vector pairs that detect at least one fault. Then there is one wire that is permanently marked at least  $\frac{|V|}{N}$  times.

*Proof:* It has been shown that each application of path-trace marks at least one detected member from the actual and each equivalent solution set [11]. In other words, at least one of the  $N$  members from the actual fault tuple is permanently marked each time. Because there are  $V$  vectors and each vector has at least one EPO, one of the members from the actual tuple has to be permanently marked at least  $\frac{|V|}{N}$  times by the Pigeon Hole Principle.  $\diamond$

In practice, many wires qualify as this theorem only stipulates a requirement. So the candidate list in each iteration is ranked by the number of times a wire is permanently marked.

In Figure 2 (b),  $G_9$  is marked twice for the same vector. Hence, it is chosen as a top candidate. The algorithm examines the transition in the fault-free simulation. Since a rising transition is observed, it injects a slow-to-rise fault on  $G_9$ . For multiple failing vectors, the algorithm uses the theorem based on the number of rises and falls to decide which transition fault to inject. Next, the fanout cone is simulated as shown in Figure 2(c). Notice that the faulty effect propagates to  $O_2$ , which now has the same fault value as in Figure 2(a).

This concludes one iteration of the incremental diagnosis (Figure 1). Since not all outputs match, another iteration is called. Path-trace is executed again (Figure 2 (c)), this time using the injected fault information as well. As a result, only

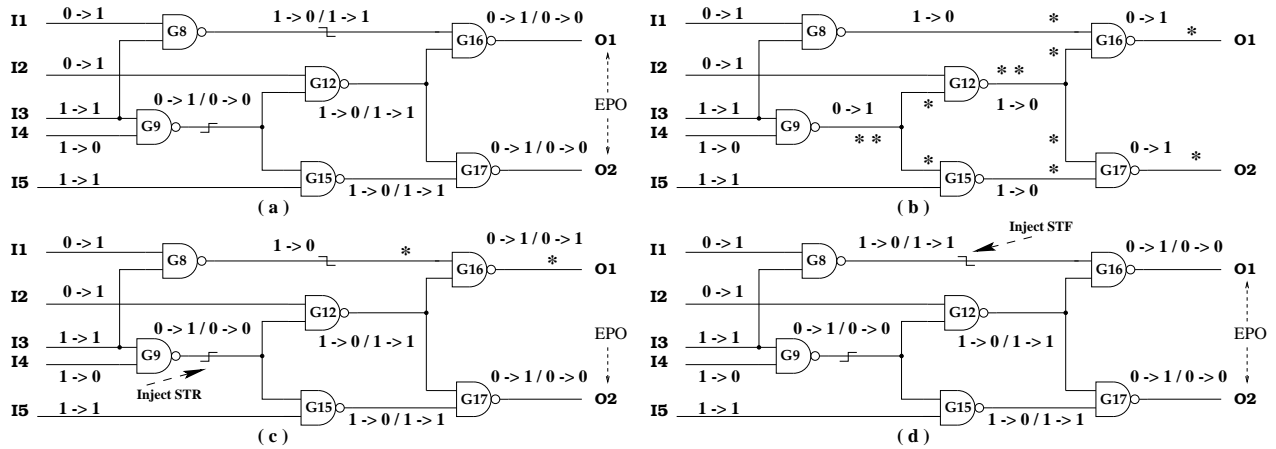


Figure 2: Diagnosis of two transition faults.

lines  $G8$  and  $G16$  get marked.  $G8$  is chosen and a slow-to-fall fault is injected at that location (Figure 2 (d)). Now the primary output matches that of the faulty circuit. The algorithm concludes and returns  $\{G8, G9\}$  as a candidate tuple.

This example, though illustrative of the conceptual approach, represents a simplified scenario. In a realistic diagnosis environment, structural circuit complexity often makes the accurate ranking of suspect lines difficult. Consequently, the truly faulty lines do not often rank at the top of the list. To improve the search process, the algorithm maintains a search-tree data structure, similar to that in [5, 12], to preserve intermediate information while pruning the diagnosis space.

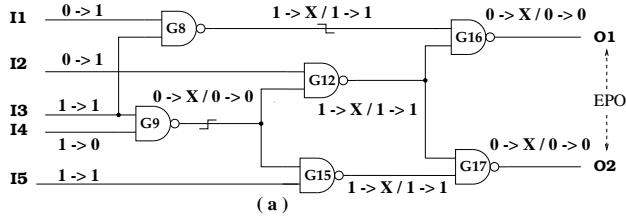


Figure 3: Example of model-free diagnosis.

### 3. MODEL-FREE ALGORITHM

The model-free algorithm also follows the incremental diagnosis framework from Figure 1. It differs from the modeled approach in that it discovers faults without relying on any specific fault models. Instead, it reasons with ternary logic and the effect of inverting local logic values upon the primary outputs. The model-free approach has the following advantages: 1) the accuracy of the diagnosis algorithm does not depend on the validity of the fault model; 2) it can capture any unmodeled effects that defects have on the logic behavior; 3) it reduces the fault modeling space; 4) it can be easily applied to diagnosis of any type of fault. To achieve these advantages, this approach moderately sacrifices on diagnostic resolution.

The model-free algorithm proceeds in two independent phases [5] as illustrated in Figure 3. The *first* phase performs incremental diagnosis by simulating *logic unknown* values on single lines to capture all possible fault effects [2]. This phase generates a somewhat pessimistic list of candi-

date tuples that may explain the erroneous circuit behavior. The *second* phase takes this list and conducts simulations of all combinations of logic values for the faulty scenarios for each tuple. Their ability to change primary output values is used as a criterion to reduce the length of the candidate list which is returned by diagnosis.

### 4. EXPERIMENTS

We run experiments on ISCAS'85 and on full-scan versions of ISCAS'89 circuits. For every circuit, we perform three sets of twenty experiments. Two, three and four transition faults are randomly injected in each set of experiments. In all cases, 500 random vector pairs are generated and all those that detect faults are used in diagnosis. Both modeled and model-free algorithms are applied to the same sets of circuits. Experiments are reported on a Sun Ultra 10 machine with 256 MB of memory.

Results for circuits with two, three, and four faults are summarized in the three major sections of Table 1. The first column contains the name of the benchmark circuits. For each section of Table 1, the first three columns contain the diagnosis results from the modeled algorithm and the following three columns contain the results from the model-free algorithm. Each set of results has the average number of transition fault sites returned by the algorithm, the time spent (in CPU seconds) and the hit ratio in identifying the actual fault. The transition fault sites are the distinct nodes that can be used by a test engineer or by an algorithm to construct suspect faulty paths. The *hit ratio* refers to the percentage of actual faults discovered if all sites are examined.

In general, the number of probe sites is small for the modeled approach. The probe sites are compiled by the union of original and equivalent fault tuples discovered. However, there are much fewer equivalent tuples (c.f. Table 1 in [12]) because the fault equivalence rules for transition faults are more restrictive than those for stuck-at faults as testing a transition fault requires more than one vector [4, 13]. In detail, for the single transition fault case, only two rules can be applied for equivalence: (i) if a gate has a single input, then the input transition faults are equivalent to the output transition faults, and (ii) if a gate has only one fanout, then the output transition faults are equivalent to the input transition faults on the fanout gate. The presence of multiple faults relaxes the equivalence rule, which leads to a

Ckt Name	Two Transition Faults						Three Transition Faults						Four Transition Faults					
	Modeled			Model-free			Modeled			Model-free			Modeled			Model-free		
	sites	time	h.r.	sites	time	h.r.	sites	time	h.r.	sites	time	h.r.	sites	time	h.r.	sites	time	h.r.
C432	4.0	3.7	100	11.0	2.5	90	4.0	9.4	100	7.2	4.8	80	5.4	28.8	100	12.5	11.2	69
C499	4.7	58.5	100	16.5	18.2	85	4.0	52.5	100	12.4	41.5	100	4.8	91.4	100	12.3	126.4	92
C880	2.8	6.1	100	11.8	4.3	100	4.0	19.7	100	13.0	9.3	100	4.8	29.4	95	14.0	22.3	94
C1355	3.8	18.3	100	16.8	12.5	100	4.0	52.6	100	13.6	41.9	87	4.4	214.2	100	24.3	70.2	94
C1908	9.5	78.9	100	11.3	12.7	100	4.5	35.2	100	11.7	33.3	78	4.4	214.2	100	24.3	70.2	94
C2670	3.6	10.6	100	10.0	9.2	100	4.2	34.1	100	14.8	63.8	94	5.5	55.0	100	15.0	36.1	88
C3540	3.3	57.6	100	7.0	8.3	100	3.3	144.7	100	7.3	189.2	94	5.0	36.9	100	18.7	95.9	83
C5315	2.8	50.8	100	14.2	71.6	100	3.5	143.2	100	10.2	221.9	94	6.0	40.4	75	9.5	343.6	75
C6288	2.4	180.1	100	7.0	170.9	20	3.0	716.8	100	5.2	432.6	60	5.0	340.3	100	17.0	519.7	92
C7552	5.2	180.6	100	12.0	61.2	100	3.3	322.2	100	12.5	324.7	83	4.0	112.1	100	13.5	477.5	38
S1196	12.2	20.9	100	11.2	20.6	100	3.8	11.8	100	11.3	25.4	100	4.2	249.6	100	11.8	610.3	56
S1494	10.5	9.9	100	15.3	16.9	100	5.8	22.5	94	11.3	22.4	83	5.0	30.2	88	10.5	16.2	63
S9234	9.8	158.7	100	12.3	21.1	100	4.0	49.8	67	10.0	100.3	83	6.0	212.3	88	17.0	31.1	75

Table 1: Results for two, three, and four transition faults.

relatively large number of equivalent fault tuples (e.g. S1196 and S1494 in Table 1).

Fault equivalence of the transition fault model does not impact the model-free diagnosis algorithm. Instead, equivalence is governed by that of logic unknowns irrespective of the type of fault it is diagnosing. Due to the pessimistic nature of logic unknown values, the model-free algorithm produces a larger list of transition fault sites. Its diagnostic resolution is further impacted by ignoring the information carried by two-vector stimulus (ie. whether a transition has occurred in the simulation of fault-free circuit). This information is intrinsic to the fault model used in modeled diagnosis. The model-free algorithm performs poorly on benchmark C6288, which is a multiplier with many multiple fanout gates. The high fanout count results in logic unknown values being propagated to many gates. Subsequent injections of an unknown logic value on many wires can recover all EPOs and thus qualify as candidate tuples.

It comes as no surprise that the modeled approach demonstrates a better performance. With an exact fault model, diagnosis is simplified in the sense that the fault model acts as an assumption, which narrows the possible set of logic behavior lines may have under the influence of a fault. For example, if a line is injected with a slow-to-rise fault, it assumes the value of logic 0 whenever a 0 to 1 transition is encountered. Therefore, given a suspect line in transition fault diagnosis, the cardinality of the fault model space equals to 3 (i.e. slow-to-rise, slow-to-fall, and fault-free).

Notice for some circuits in modeled diagnosis and many in model-free diagnosis, the hit ratio for original faults is not 100% percent. If the entire diagnostic space is traversed, diagnosis algorithms can guarantee the detection of actual and all equivalent fault tuples [5]. During these experiments, only an arbitrarily chosen portion of the diagnosis space is searched. All experiments return some solution tuples; however in some cases, they consist of equivalent candidate wire members. The simple ranking formula used produced hundred percent hit ratio in the majority of circuits. Investigation is under way to improve the ranking of the suspect wires during diagnosis.

Results from both modeled and model-free algorithms demonstrate good diagnostic resolution. The compact list of transition fault sites can be used to construct suspect delay fault path(s). Once identified, a behavioral simulation with timing information may be applied to further refine the list.

## 5. CONCLUSION

In this paper, two fault diagnosis algorithms for multiple delay faults are developed. One algorithm uses a transition fault model to capture the faulty effect(s); the other reasons with ternary logic values and is independent of any fault models. Both algorithms are developed under an incremental framework that has been shown to be efficient in diagnosing multiple faults. Moreover, neither algorithms require the knowledge of timing information. Their performance is evaluated and compared with ISCAS'85 and fullscan ISCAS'89 circuits. Both demonstrate good efficiency and resolution with the modeled approach exhibiting better resolution. Future work intends to improve the algorithm efficiency and apply it directly to multiple path-delay faults.

## 6. REFERENCES

- [1] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable Design*. IEEE Press, Piscataway, New Jersey, revised edition, 1994.
- [2] V. Boppana and M. Fujita. Modeling the unknown! towards model-independent fault and error diagnosis. *Proc. ITC*: 1094–1101, 1998.
- [3] J. Ghosh-Dastidar and N. A. Toubia. A systematic approach for diagnosing multiple delay faults. *IEEE Symposium on Defect and Fault Tolerance*: 492–499, 1998.
- [4] A. Krstic and K.-T. Cheng. *Delay Fault Testing for VLSI Circuits*. Kuwer Academic Publishers, Boston/Dordrecht/London, 1998.
- [5] J. B. Liu, A. Veneris. Incremental Fault Diagnosis. *IEEE Trans CAD*: 240–251, 2005.
- [6] S. Padmanaban and S. Tragoudas. An implicit path-delay fault diagnosis methodology. *IEEE Trans CAD*: 1399–1409, 2003.
- [7] P. Pant, Y.-C. Hsu, S. K. Gupta, and A. Chatterjee. Path delay fault diagnosis in combinational circuits with implicit fault enumeration. *IEEE Trans CAD*: 1226–1235, 2001.
- [8] E. S. Park and M. R. Mercer. Robust and nonrobust tests for path delay faults in a combinational circuits. *Proc. IEEE ITC*: 1027–1034, 1987.
- [9] G. L. Smith. Model for delay faults based upon paths. *Proc. IEEE ITC*: 342–349, 1985.
- [10] R. C. Tekumalla, S. Venkataraman, and J. Ghosh-Dastidar. On diagnosing path delay faults in an at-speed environment. *IEEE VTS*: 28–33, 2001.
- [11] A. Veneris and I. N. Hajj. Design error diagnosis and correction via test vector simulation. *IEEE Trans. CAD*: 1803–1816, December 1999.
- [12] A. Veneris, J. B. Liu, M. Amiri, and M. S. Abadir. Incremental diagnosis and debugging of multiple faults and errors. *Proc. IEEE DATE*: 716–721, 2002.
- [13] J. A. Waicukauski, E. Lindblood, B. Rosen, and V. Iyengar. Transition fault simulation. *IEEE Design & Test of Computers*, 4(2):32–38, April 1987.