On Generating High Quality Tests for Transition Faults

Yun Shao[1], Irith Pomeranz[2], and Sudhakar M. Reddy[1]

1: Electrical and Computer Engineering Department
University of Iowa, Iowa City, IA 52242, U.S.A.

2: School of Electrical and Computer Engineering
Purdue University, West Lafayette, IN 47907, U.S.A.

Abstract

In this work we propose a path-oriented test generation procedure called POTENT to generate high quality tests for transition faults. Both weak non-robust and strong non-robust tests can be generated by POTENT. We classify transition fault tests into six types according to their activation and propagation methods. The basic idea of POTENT is to test a transition fault along a longest testable path passing through the fault site. For transition faults that are activated or propagated through multi-paths, heuristics are proposed to maximize the propagation delay of the target fault. We also propose an efficient method to evaluate the quality of a given transition fault test set. Experimental results show that POTENT generates higher quality transition fault test sets than the conventional test generation method.

1. Introduction

Physical defects that cause the signal propagation delays in the circuit to increase can be modeled as delay faults. Two general types of delay fault models, the *gate delay fault model* [1-4] and the *path delay fault model* [5-6] have been used for modeling delay defects. Although the path delay fault model is generally considered to be more realistic and effective in modeling physical delay defects, it is often difficult to use in practice due to the huge number of paths in the circuit. Therefore, the gate delay fault model is more feasible for large circuits. The most commonly used gate delay fault model is the *transition fault* model [1]. According to this model, every line in the circuit is associated with two transition faults: a *slow-to-rise fault* (rising fault) and a *slow-to-fall fault* (falling fault). To simplify the analysis of transition faults, it is often assumed that the extra delay caused by a transition fault on a line is sufficiently large such that the delay of every path passing through this line exceeds the maximum allowed value, which is usually the system clock period for synchronous sequential circuits. We call this assumption the *gross delay assumption*. Based on this assumption it can be concluded that the following two requirements are necessary and sufficient for a two-pattern test (V_1, V_2) to be a test of a transition fault f on line l. Tests that meet these conditions are referred to as *conventional transition fault tests*.

(1) When the circuit inputs are changed from V_1 to V_2, the rising (falling) transition fault f is activated by a rising (falling) logic transition on line l.
(2) V_2 is a test for a stuck-at-0 (stuck-at-1) fault on l.

Generation of a test to detect a transition fault includes two steps. The first step is to activate the target fault by creating a transition on the faulty line through single or multiple paths. The second step is to propagate the fault effect to an output by sensitizing single or multiple paths from the faulty line to an output. Although the gross delay assumption simplifies test generation for transition faults, in reality a test generated for a given fault f by satisfying the above two conditions may not be a high quality test for detecting delay defects on the faulty line due to two reasons. First, the target fault can be activated and propagated along subpaths with arbitrary delays. As a result, small size delay defects, which only cause faulty behaviors on paths with relatively large delays, may not be detected. Second, a test satisfying the conventional requirements is generally a weak non-robust test, thus it can be invalidated due to signal hazards and other factors [7]. We refer to the first issue as the *criticality* problem and the second issue as the *robustness* problem. The criticality of a test indicates its capability of detecting delay defects of small size, while the robustness of a test indicates the possibility that the test is effective in the presence of factors that can invalidate a test. Both of them need to be addressed to achieve high quality transition fault tests. While our focus in this work is on improving the criticality of tests for transition faults, we also investigate the use of strong non-robust tests to improve the robustness of the tests.

This paper is organized as follows. In Section 2, background on transition fault testing and review of earlier works are presented. In Section 3, we propose a taxonomy of transition fault tests. In Section 4, test generation methods, which aim at generating high quality tests, are described. Section 5 presents a simple procedure to evaluate the criticality of a given test set. Experimental results for ISCAS89 benchmark circuits are given in Section 6. Section 7 summarizes the paper.

2. Preliminaries

2.1 Fault models and terminology

For gate delay faults, the amount of extra delay caused by a delay fault on the faulty line is called the *size* of the fault. To

1. Research supported in part by NSF Grant No. CCR-0097905 and in part by SRC Grant No. 2001-TJ-949.
2. Research supported in part by NSF Grant No. CCR-0098091 and in part by SRC Grant No. 2001-TJ-950.

Proceedings of the 11 th Asian Test Symposium(ATS'02)
1081-7735/02 $17.00 © 2002 IEEE

take the sizes of gate delay faults into consideration, the *small gate delay fault model* is used in [2-4]. In general a test for a given gate delay fault f can only detect delay defects on f with sufficiently large sizes. A commonly used metric called the *detection threshold* [2-4] is defined as follows to measure the criticality of a test for detecting a gate delay fault f.

Definition 2.1: Let T be a two-pattern test that detects a gate delay fault f. The *detection threshold* is the minimum size ε such that if the size of f is greater than ε, f is guaranteed to be detected by T. The detection threshold of test T for fault f is denoted as $\varepsilon(T, f)$.

For any given pair of tests T_1 and T_2, T_1 is considered to have *higher criticality* than T_2 for a fault f if $\varepsilon(T_1, f) < \varepsilon(T_2, f)$. It is desirable to generate a test set such that the detection threshold of every transition fault is minimized. Rigorous calculation of detection thresholds is described in [2-4]. For a given test, the detection threshold of each fault is calculated separately through timing aware simulation. However, to generate tests with minimal detection thresholds using timing aware simulation could be impractical for large circuits. In general, to reduce the detection threshold of a target fault f, the transition on the faulty line should be launched as late as possible and the fault effect should be propagated through paths with maximum delays. Based on this heuristic, we propose the use of a simple measure called the *nominal fault detection delay* to measure the criticality of a test. It is defined as the sum of the delay of the activation path(s) and the delay of the fault propagation path(s) for a given fault. The computation of this value is simple and it can be directly used in test generation to search for a test with a maximal nominal fault detection delay. The formal definition of nominal fault detection delay will be given in Section 5.

The *segment delay fault model* proposed in [8] is an extension of the gate delay fault model. A segment delay fault models the delay defects on a segment, which is a subpath of L consecutive lines, $L \geq 1$. It is assumed that every path containing a segment delay fault has a delay greater than the maximum allowed value. Tests to detect segment delay faults should propagate a transition launched at the beginning of a segment through the segment to an observed output, possibly through a multi-path. When L is selected to be 1, segment delay faults are essentially transition faults. Although segment delay faults alleviate part of the criticality problem (i.e., the capability to detect small size defects) of transition faults by testing longer segments instead of lines, a segment delay fault can still be activated and propagated through any path passing through it and hence a segment fault test may not detect small delay faults. Additionally, the number of segment delay faults increases with increasing segment lengths.

In this work, we use gate delay faults to model delay defects. However, the sizes of the gate delay faults are not explicitly considered. Instead, the activation path delay and the propagation path delay of every targeted fault are maximized to enhance the criticality of the test. In addition, each fault can be activated and propagated through multiple paths. In other words, we use the transition fault model together with constraints on the delays of the subpaths used to activate and propagate the fault. This will help achieve the goal of generating delay fault tests to detect gate delay faults of small sizes.

Another metric of transition fault test quality is *robustness*. A cursory classification of transition fault tests based on robustness is described in [2], which resembles the classification of path delay fault tests [9-10]. In general, there are two categories of tests for transition faults: *non-robust tests* and *robust tests*. Non-robust tests can be further classified into *weak non-robust (WNRB) tests* and *strong non-robust (SNRB) tests* [2]. A robust test for a transition fault guarantees to detect the fault regardless of signals not on the paths through which the fault effect is propagated, while a non-robust test can be invalidated [7]. Conventional transition fault tests described earlier are weak non-robust tests.

Next we define some terms used in this paper. A *single physical path* (*SP*) is a consecutive sequence of gates and connections starting from an input and ending at an output. A *multi-path* (*MP*) [12] is a set of single paths that end at the same line. The *slack* of a path P is defined as follows for synchronous sequential circuits.

Definition 2.2: Let $D(P)$ be the delay of path P and let the system clock period be T_c. The *slack* of P, denoted as $PathSlack(P)$, is $T_c - D(P)$.

The slack of path P is the difference between the system clock period and the delay of P. For a given transition fault f, the *slack* of f, which is called the *fault slack*, is defined as follows.

Definition 2.3: Let $FuncPath(f)$ be the set of all the functionally sensitizable paths passing through the fault site of f. The slack of f, denoted as $FaultSlack(f)$, is equal to $min(PathSlack(P))$, $P \in FuncPath(f)$.

Since only functionally sensitizable paths in a circuit can affect the circuit timing performance [9], $FaultSlack(f)$ is the minimum size of f that can cause a faulty behavior of the circuit when f is the only fault in the circuit.

There are three test application methods for scan designs: *enhanced scan, functional justification* (or *broadside testing*) and *scan-shift justification* (or *skewed-load*) [10]. In this work we focus on standard scan designs, and assume that functional justification is used. However, the methods described here can be straightforwardly extended to skewed-load and enhanced scan test methods.

2.2 Review of previous works

Pramanick and Reddy [2] proposed to generate a robust or non-robust test for a longest path (therefore with minimum slack) passing through the target fault. This method does not take testability into consideration when selecting a longest path containing the fault site. For many transition faults no test can be found for a selected longest structural path passing through the fault site. Therefore, the fault coverage achieved is typically low. Furthermore, requiring fault activation and propagation through single paths reduces the set of potential tests for a fault and their criticality.

One way to overcome the drawbacks of the method in [2] is to generate tests for a longest testable path containing the target fault. Majihi et. al. proposed a procedure in [12] to cover every line in the circuit with a longest robustly testable path. When a path is robustly tested, all the transition faults on the path are

robustly detected. There are two drawbacks to the procedure used in [12]. First, to find a longest testable path containing a line, all the paths passing through the line are enumerated and the longest N (N is a specified limit) paths are targeted for test generation until a testable path is found. This procedure may not be able to find any testable path due to the limited number of paths tried for each line, even if there is one. According to the results in [12], it takes long run times to cover a high percentage of the lines in the circuit for large circuits. Second, only single robustly testable paths are used to cover lines in [12]. Consequently, the transition fault coverage that can be achieved using the method in [12] is much lower than that of the conventional transition fault tests, which allow the use of multi-paths and non-robust tests to detect transition faults.

Mao et. al. [13] proposed a procedure to enhance the robustness and criticality of gate delay tests. A robustness measure is defined for each gate according to the probability of other delay faults blocking the fault propagation through the gate. Fault propagation during test generation is guided towards the gates with the larger robustness estimation values. Since the proposed robustness measure does not reflect the delays of fault propagation paths, the fault propagation delay is not effectively maximized.

Techniques to enhance the robustness of non-robust tests for path delay faults and gate delay faults have also been investigated in [13, 14]. They can be classified into two categories. The methods in the first category add more restrictions on fault propagation conditions and make non-robust tests more like robust tests. The second type of techniques minimize the delays associated with certain off-path inputs, such that the transitions on the off-path inputs will not mask the effects of the transitions on the on-path inputs.

2.3 Overview of the proposed procedure

We propose new techniques to efficiently generate high quality tests for transition faults. A path-oriented test generator called POTENT is developed for this purpose. In order to facilitate the generation of high quality tests, transition fault tests are classified into six categories according to how activation and propagation are achieved. Techniques to improve the criticality of tests for each class of faults are discussed in detail. In most cases a transition fault can be detected by testing a single path containing the fault. For transition faults that are detected through multi-path propagation, several heuristics are proposed to maximize the delay of the multi-path used to propagate the target fault. We also propose the use of strong non-robust tests to improve the robustness of conventional transition fault tests, which are weak non-robust. Techniques for strong non-robust test generation are presented.

To evaluate the quality of a given transition fault test set with respect to its criticality, we propose an efficient method to calculate the activation and propagation delay for every detected fault. A transition fault simulator is implemented based on the proposed method.

3. A Taxonomy of transition fault tests

In this section we first investigate the relations between path delay fault tests and transition fault tests. We then classify transition fault tests based on their activation and propagation methods.

3.1 Relations between path delay fault tests and transition fault tests

To test a single path delay fault, the desired transition must be launched at the input of the path and the off-path inputs of the path must satisfy certain conditions to *sensitize* the path. The path sensitization conditions for different types of tests are described in [9].

A robust test for a single path is a robust test for the transition faults on the on-path signal lines. This is because a robust test for a path delay fault guarantees the detection of any delay defects along the path. However, a non-robust test for a path passing through a specified transition fault may not be a valid transition fault test (i.e., a test that satisfies the two conditions described in Section 1) because of reconverging fanout.

We use the example in Figure 3.1 to illustrate this phenomenon. When a two-pattern test $<V_1, V_2>$ is used to test a delay fault, the expected logic value of a line after V_1 is applied is called the *initial value*, and the logic value that is expected after V_2 is applied is called the *final value*. In Figure 3.1 (a) and (b) the initial and final values of each line are shown in the form (v_1, v_2). Assume that there is a rising transition fault R on line a and R is contained in path P which consists of lines (a, d, f, h). We show in Figure 3.1 (a) the transitions obtained under the two-pattern test applied. It can be seen that the highlighted path P is strong non-robustly sensitized and hence the given two-pattern test is a non-robust test for path P. Next, we consider the same test as a transition fault test for fault R in Figure 3.1 (b). To facilitate the discussion of fault propagation for transition faults, the five-valued logic system {0, 1, X, D, D'}, which was originally proposed for stuck-at fault test generation [19], is adopted to represent the initial and final values on a circuit line. A D (D') value on a line represents that the logic value of the line in the fault-free circuit is 1 (0) and the logic value in the faulty circuit is 0 (1). When R is activated by launching a rising transition on line a, the initial and final values on line a are (0, D). The fault effect of R (represented by D and D' values) is propagated to both inputs of the AND gate with output h. The fault effect on line g (the D value) and the fault effect on line f (the D' value) cancel each other and the output waveform on line h is fault-free, represented by (1, 0). Thus, the strong non-robust test for P in Figure 3.1 (a) is not a valid transition fault test for the transition fault on line a.

A non-robust test for a path containing a transition fault can be invalided only if the fault effect reconverges at some gates along the path. Therefore, if the reconvergence of fault effects is prohibited, a non-robust test for a path passing through a transition fault is always a valid test for the transition fault.

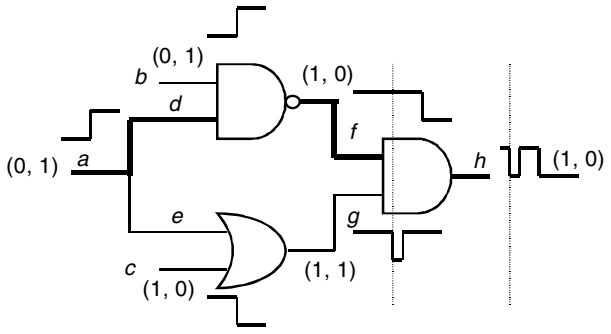

Figure 3.1 (a) Testing a Path Delay Fault

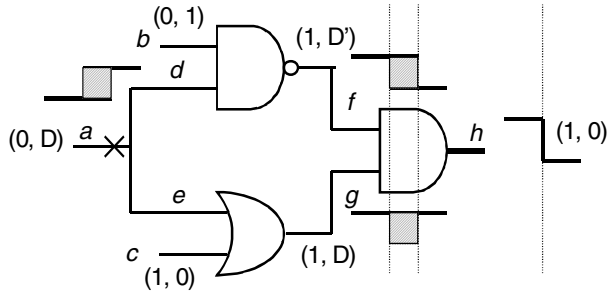

Figure 3.1 (b) Testing a Transition Fault

3.2 Classification of transition fault tests

Testing a transition fault involves two steps: *fault activation* and *fault propagation*. Both fault activation and fault propagation can be achieved through a single or a multi-path. The fault activation path is the path along which transitions are propagated from the circuit inputs to the fault site. The fault propagation path is the path along which the fault effect is propagated to the circuit outputs. Activating a transition fault requires propagation of a transition or transitions from one or more primary inputs to the fault site through a single or multi-path. These paths may satisfy conditions for robust, strong non-robust or functional sensitization. The fault effect can be propagated from the fault site through a single or multi-path using fault propagation conditions for robust, strong or weak non-robust propagation. Combining different activation and propagation conditions, one can classify tests for transition faults into 30 different types.

In this paper we consider only two types of fault propagation, namely *weak non-robust propagation* and *strong non-robust propagation*. Also, we consider only strong non-robust and functional activation. A conventional transition fault test uses the weak non-robust propagation and functional activation. In the weak non-robust propagation, the fault effect can be propagated as a signal glitch. In the strong non-robust propagation, the fault effect is always propagated with a rising or falling transition. In Figure 3.2, examples (a) and (b) illustrate the weak non-robust and the strong non-robust propagation of the fault effect on line *a* through the AND gate *c*, respectively.

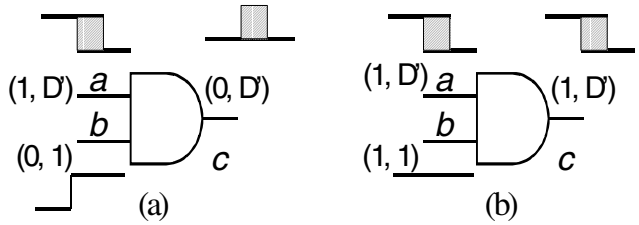

Figure 3.2 Weak and Strong Non-robust Fault Propagation

Restricting the activation and propagation to the above mentioned cases we considered six types of transition fault tests shown in Table 3.1. In Table 3.1, "SNRB" ("WNRB") represents strong (weak) non-robust. "Single" ("multi") means that a single path (multi-path) is used for fault activation or propagation. The proposed test generator that incorporates all six types of tests is called POTENT.

Table 3.1 Six Types of Transition Fault Tests

	activation		propagation	
	type of path	sensitization	type of path	sensitization
Type-I	single	SNRB	single	SNRB
Type-II	multiple	functional	single	SNRB
Type-III	single/multi	functional	multi	SNRB
Type-IV	single	SNRB	single	WNRB
Type-V	multi	functional	single	WNRB
Type-VI	single/multi	functional	multi	WNRB

Tests of types I-III strong non-robustly propagate fault effects to outputs. When a fault *f* is detected by a Type-I test *T*, *f* is tested along a path *P* passing through *f*, and *P* is strong non-robustly sensitized by *T*. For a Type-II test, the transition on the faulty line is launched through a multi-path P_M and none of the single paths contained in P_M is strong non-robustly sensitized (otherwise the test belongs to Type-I). A Type-III test activates the target fault through a single or multi-path but must strong non-robustly propagate the fault effect through a multi-path P_M to an output. None of the single paths contained in P_M is strong non-robustly sensitized. The multi-path propagation mechanism for transition faults is analogous to that for multi-path delay faults [11].

Tests of Type IV-VI weak non-robustly propagate fault effects to outputs. Type-IV, Type-V and Type-VI tests relax the SNRB propagation requirement of Type-I, Type-II and Type-III tests to WNRB propagation, respectively. It should be noticed that weak non-robustly sensitizing a single activation path from an input to a given fault site is not sufficient for creating a transition on the faulty line. Therefore, we do not consider weak non-robust activation.

For tests that use single path propagation (i.e., tests of Type I, II, IV, V), the fault effect is not allowed to reconverge along the propagation path.

4. Test generation for transition faults

Our objective is to generate high criticality tests that activate and propagate transition faults along paths with maximal delays. We divide our discussion of POTENT into two parts. The first part describes test generation techniques for single path propagation tests (i.e., Types I, II, IV and V tests). The second part discusses test generation for multi-path propagation tests and describes methods used to maximize the delay of the fault propagation path. Methods to maximize the activation path delay when a multi-path is used for activation are not discussed in this paper.

In order to generate a Type I or Type IV test for a transition fault, a testable path through the fault site needs to be selected. For Type I tests, such a path is strong non-robustly testable. For Type IV tests, the activation part of the selected path, which is the partial path from an input to the fault site, is strong non-robustly testable, while the propagation part of the path, which is the partial path from the fault site to an output, is weak non-robustly testable. The path expansion procedure described in [18] can be used to select a longest testable path through a given fault site. When generating a test for a selected path, one additional requirement is that the fault effect cannot reconverge along the selected path. This restriction is imposed to guarantee that a test generated for the selected path passing through a fault

is a valid transition fault test. A technique called *D-value sensitization* is proposed later to achieve this objective.

For Type II and Type V tests, the transition on the fault site can be launched through a multi-path but the fault effect is propagated through a single path. Only the propagation path of a fault is explicitly selected for generating Type II and V tests.

Unlike the test generation procedures for single path propagation tests, the activation path and propagation path of a Type III or Type VI test are not selected before generating a test but dynamically determined during test generation through multiple steps. In each step the fault effect is further propagated towards the circuit outputs until it reaches an output. We use several heuristics within the generic test generation procedure to improve the delay of the fault propagation path(s).

4.1 Test generation for single path propagation faults

We start with test generation of Type-I tests. The most critical issue for generating a Type-I test is to find a testable path passing through the target fault. We first briefly review the stepwise path expansion procedure in [17], which is used to select a longest testable path through a line. In order to minimize the probability of selecting untestable paths it is important to identify as many untestable paths as possible before path selection. The method we use is based on a non-enumerative method [15] introduced by Kajihara et. al. [16], which was further improved in [17]. The procedures in [16, 17] search for the so called *b-f* pairs in the circuit. A pair of logical lines (l_i , l_j) is called a *b-f* pair if every logical path passing through both l_i and l_j is untestable. The set of untestable paths identified can be compactly represented by *b-f* pairs. Experimental results show that most of the untestable paths can be identified using *b-f* pairs [16, 17]. Therefore, a path that does not contain a *b-f* pair has a high probability of being truly testable. We call such paths *potentially testable*. After finding the b-f pairs in a circuit, a potentially testable path can be expanded through a given line l by stepwise concatenation of circuit lines. During the expansion, all identified *b-f* pairs are avoided to ensure that the expanded path is potentially testable. For each expansion step of the procedure, a greedy approach is used to maximize the delay of the constructed potentially testable path. However, the final expanded path is not guaranteed to be a longest potentially testable path because of the local greedy heuristic used. To overcome this deficiency, we use a branch-and-bound method to iteratively search for potentially testable paths with longer delays than the longest potentially testable path already found. The delay of the currently identified longest potentially testable path is used as a lower bound to limit the search to the area where longer potentially testable paths can be found. This procedure stops when no potentially testable paths with longer delays can be found. By incorporating ATPG into the procedure, a longest testable path can be found for a given line in a short time if there exists one [18].

There is one difference between the path expansion procedure used for Type-I test generation and the one used in [18]. This is that the fault effect is not allowed to reconverge along the selected path when generating tests for it. A new sensitization method called *D-value sensitization* is used to achieve this goal. The five-valued logic system {0, 1, X, D, D'} is used to represent initial and final values of circuit lines. D and D' values are used to represent the propagated fault effect as in Figure 3.1. When a path is D-value sensitized, only the on-path inputs of the path have D or D', and off-path inputs cannot have D or D' values. Therefore, the fault effect cannot reconverge along the path. When ATPG is used in the path expansion procedure to verify the testability of selected paths, D-value sensitization is used instead of conventional path sensitization as in [18]. It should be noticed that a path that can be sensitized using standard path sensitization may not be sensitizable using D-value sensitization because of the additional requirement that prohibits the reconvergence of fault effects.

For a given fault f that is detectable by Type-I tests, we call a Type-I test *optimal* if it is a test for a longest path passing through f that is D-value sensitized. Such a path has the minimum slack among all D-value sensitizable paths passing through the fault.

Test generation of Type IV tests is similar to that of Type-I tests. The only difference is that the propagation path of a Type IV test is weak non-robustly sensitized. For both Type-I and Type-IV tests, the sum of the delays of the activation path and the propagation path is maximized using the path expansion procedure.

For Type-II and Type-IV tests, only the propagation path is selected using path expansion. Thus, Types II and IV tests for a fault f maximize the delay of the fault propagation path only.

4.2 Improving propagation delays for multi-path propagation faults

To generate a multi-path for propagation under tests of Types III and VI, we use a generic test generation procedure for transition faults based on the PODEM [19] algorithm. This procedure generates tests that use multi-path activation and propagation. The set of gates, for which the fault effect has been propagated to at least one of their inputs but not to their outputs, is called the *P-frontier* in the following discussion. The test generation procedure iteratively assigns logic values on circuit inputs to propagate the fault effect through the gates in the P-frontier. Every gate in the P-frontier for weak (strong) non-robust test generation satisfies the following two conditions.

(1) The fault effect has been weak (strong) non-robustly propagated to at least one input of the gate but has not been propagated to the output of the gate.
(2) It is possible to further weak (strong) non-robustly propagate the fault effect to the gate output.

We use several heuristics to improve fault propagation delays as explained later using the definition of nominal propagation delay.

The *nominal propagation delay* is the propagation delay of a given single path or a multi-path with one input. For a single path the nominal propagation delay is the sum of the expected delays of all the lines along the path. For a multi-path, there are some gates with multiple on-path inputs. To facilitate the discussion, we use the following notation. Let the controlling value and the non-controlling value of a gate g be $CV(g)$ and $NCV(g)$, respectively. Let the transition that settles to value v be T_v. For the sake of convenience $T_{CV(g)}$ ($T_{NCV(g)}$) is called the (*controlling*) *non-controlling transition*. The nominal propagation delay from the input of a path to an on-path input line l is denoted by $NPD(l)$. The following formulae are used to

compute *NPD* for a gate output *g* with multiple on-path inputs g_i, ($i = 1, \ldots, m$).

$NPD(g) = min(NPD(g_i)+D(g_i))$, $i = 1, \ldots, m$, if g_i has transition $T_{CV(g)}$. **(4.1)**

$NPD(g) = max(NPD(g_i)+D(g_i))$, $i = 1, \ldots, m$, if g_i has transition $T_{NCV(g)}$. **(4.2)**

We use the following examples to explain the above equations. An AND gate whose two inputs have falling transitions is shown in Figure 4.1 (a). It can be seen that the signal transition time on the output of *g* is determined by the earliest falling transition on the inputs. Thus, Equation 4.1 is applied in this situation. In Figure 4.1 (b) the output transition time is determined by the latest non-controlling transition. Therefore, Equation 4.2 should be applied.

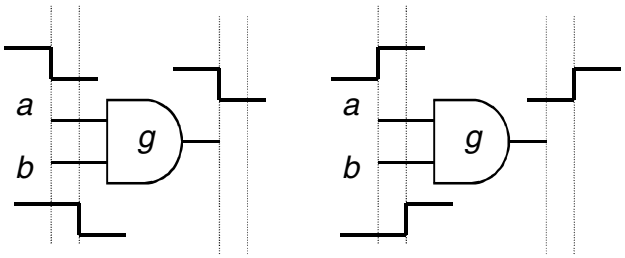

Figure 4.1 Output Transition Time for Reconvergent Gates

In real circuits, there are signal hazards on the circuit lines and a signal can have several momentary changes before it stabilizes when it is propagated along a path. However, analysis methods that take signal hazards into consideration are complex. Therefore, the nominal propagation delay is proposed as an approximate measure of the signal propagation delay from the input of a path to the output of the path. Based on the definition of nominal propagation delay, we define the nominal fault propagation delay as follows.

Definition 4.1: For a single or multi-path *p* along which a given fault *f* is propagated, the nominal propagation delay of *p* is called the *nominal fault propagation delay* of *f* along *p*.

Based on the above discussion, during test generation two delay values can be calculated for each gate *g* in the P-frontier. The first value, denoted as $NPD(g)$, is the nominal propagation delay from the target fault to the output of *g*. $NPD(g)$ can be computed by processing the gates along the fault propagation path (a single or multi-path) from the fault to *g* using Equations 4.1 and 4.2. The second value, denoted as $ND\_PO_{max}(g)$, is the maximum nominal propagation delay among all the paths from the output of *g* to the circuit outputs. This is the maximum possible propagation delay from *g* to an output. The sum of $NPD(g)$ and $ND\_PO_{max}(g)$, denoted as $NFPD_{max}(g)$, is the *maximum nominal fault propagation delay* that can be achieved by propagating the fault effect through gate *g*.

Figure 4.2 illustrates the meaning of $NPD(g)$ and $ND\_PO_{max}(g)$ for a P-frontier gate *g*. The gates in the P-frontier can be sorted according to their $NFPD_{max}$ values in decreasing order. When choosing a gate from the P-frontier for further fault propagation during test generation, the first gate in the P-frontier with the maximum $NFPD_{max}$ is selected. In this way, the longest candidate propagation path is always selected to propagate the fault effect and the nominal fault propagation delay of the fault is improved. We call this heuristic *P-frontier gate ordering*.

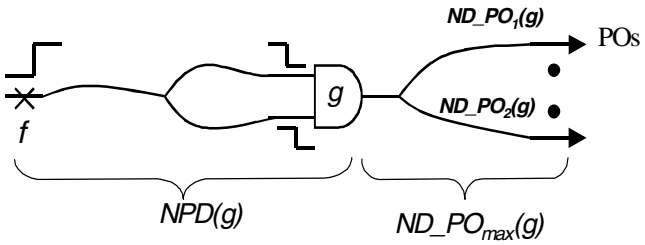

Figure 4.2 Illustrating $ND(g)$ and $ND\_PO_{max}(g)$

4.3 Overall flow of POTENT

The overall flow of POTENT includes three phases. When strong (weak) non-robust propagation is used, POTENT generates Type-I (Type IV), Type-II (Type V) and Type III (Type VI) tests in Phase I, Phase II and Phase III, respectively. For each fault targeted in Phase I, POTENT attempts to find a longest D-value sensitizable single path through the fault to test the target fault. In the second phase, POTENT searches for a longest D-value sensitizable single (partial) path to propagate the fault effect. All the remaining faults are targeted in the third phase, where multi-path activation and propagation are used to detect each fault. The heuristics described in Section 4.2 are implemented to improve fault propagation delays in the third phase.

To obtain a high quality transition fault test set using reasonable run times, initially a conventional transition fault test set *TS* can be generated and then augmented by tests from POTENT. The transition fault test set *TS* is fault simulated and all the faults that are detected with sufficient quality can be dropped. The remaining faults are targeted using POTENT. In Section 5 we will describe in detail how to evaluate the quality of a test set. Faults that are determined to be untestable using conventional test generation can also be dropped to avoid wasting test generation time for them. By reducing the set of target faults to a small subset of all the faults in the circuit, the run time of POTENT can be significantly reduced without compromising the quality of the final test set.

5. Evaluating the criticality of a test set

Various methods were proposed in earlier works to evaluate the quality of a given test in terms of detecting small size delay faults [2-4]. Although these methods can accurately determine the sizes of faults detected by a test set, they are complex and time consuming. In this paper we propose several simple metrics to evaluate the quality of a transition fault test. Generally, the minimum size of a detected fault is mainly determined by the arrival time of the transition on the faulty line and the maximum fault propagation delay from the faulty line to the outputs. We use the nominal fault propagation time defined in Section 4.2 to measure the fault propagation delay from the fault site to an output. We also define the *nominal transition arrival time* (*NTAT*) of a line as the time it takes for the transition to occur when all signal hazards are ignored. We denote the nominal transition time on line *l* as $NTAT(l)$. In real

circuits the transition on l may start before $NTAT(l)$ and stabilize after $NTAT(l)$ due to the existence of signal hazards. The advantage of using the $NTAT$ is that the calculation of $NTAT$ values for the circuit lines is simple, and the time complexity is linear in the size of the circuit. The $NTAT$ of a transition on the output of gate g can be calculated using the following formulae. The derivation of these formulae is similar to that of the nominal propagation delay equations (c.f. Equations 4.1 and 4.2).

$NTAT(g) = min(NTAT(g_i)+D(g_i))$, i = 1, …, m, if g_i has transition $T_{CV(g)}$. **(5.1)**

$NTAT(g) = max(NTAT(g_i)+D(g_i))$, i = 1, …, m, if g_i has transition $T_{NCV(g)}$. **(5.2)**

For a given fault f that is detected by a test T at multiple outputs, the *maximum nominal fault propagation delay* $NFPD_{max}(f, T)$ is defined as the maximum of the $NFPD$ values of all the outputs to which the fault effect is propagated. The *maximum nominal fault detection delay* of fault f is defined as the sum of $NTAT(f, T)$ and $NFPD_{max}(f, T)$. This value is denoted as $NFDD_{max}(f, T)$. Furthermore, we define the *test slack* of test T for fault f as follows. Here T_c is the system clock period.

$$TestSlack(f, T) = T_c - NFDD_{max}(f, T)$$
$$= T_c - (NTAT(f, T) + NFPD_{max}(f, T)) \quad \textbf{(5.3)}$$

The test slack $TestSlack(f, T)$ can be considered as an optimistic estimation of the detection threshold $\varepsilon(T, f)$, which is the minimum size of f that can be detected by T. This is because generally the existence of signal hazards will increase the uncertainty of fault detection (therefore reduce the detection threshold). $TestSlack(f, T)$ can be used to measure the criticality of test T for fault f. The smaller this value is, the higher the criticality of test T is for fault f. The fault slack of f is a lower bound of $TestSlack(f, T)$ because the fault slack is the minimum size of f that can affect the circuit performance. The difference between the fault slack and the test slack is called the *relative slack* and it is denoted as $\Delta TestSlack(f, T)$. If $\Delta TestSlack(f, T)$ is zero then test T achieves the highest possible criticality for fault f.

$$\Delta TestSlack(f, T) = TestSlack(f, T) - FaultSlack(f) \quad \textbf{(5.4)}$$

6. Experimental results

The proposed test generator was applied to ISCAS89 benchmark circuits using a Pentium IV 1.4 GHz PC with Linux operating system. All the experiments reported here are performed on standard scan designs using functional justification. The unit delay model is used throughout these experiments, i.e., every gate is assumed to have a unit delay. The backtrack limit of the ATPG was set to 100.

In Table 6.1 we give the results on the average relative slack obtained by conventional transition fault tests and by the weak non-robust tests derived by POTENT. After the circuit name we give the total number of transition faults followed by the number of faults detected, the number of faults found to be untestable and the number of faults aborted. Next we report the average relative slack over all the detected faults achieved by the conventional tests followed by the average relative slack achieved after augmenting the conventional tests to improve the criticality of the tests. In the next column the improvement of the relative slack by the application of POTENT is reported as a percentage of the average slack achieved by the conventional transition fault tests. That is, we report ($\Delta TestSlack(conv.)$ - ($\Delta TestSlack(POT.)$) / $\Delta TestSlack(conv.)$. It can be seen that the percentage improvements in the average relative slacks are high. In the last two columns we report the run times of the conventional and POTENT test generators.

In practice it is important to decrease the relative slack of the faults whose slack is small. Faults on such lines are more likely to lead to circuit malfunction. For this reason we next consider a set of "critical faults" defined as those faults whose slacks are no more than 20% of the system clock period. For this set of faults we give data in Table 6.2 similar to that in Table 6.1 where we consider all the faults. After the circuit name we give the number of critical faults followed by the number of critical faults detected. In the next two columns we report the average relative slack over the detected critical faults by the conventional transition fault tests and by POTENT. In the next column we give the percentage improvement in the average relative slack achieved by POTENT. Next we give the number of tests in the conventional transition fault test set. The test set augmented by POTENT is statically compacted using forward looking reverse-order fault simulation [20] without reducing its criticality. The number of tests in the compacted POTENT test set is reported in the last column. It can be seen that the percentage improvement in average relative slacks using POTENT are high for the critical faults also.

7. Concluding remarks

In this work we described a path-oriented test generation procedure POTENT that can generate high quality tests for transition faults. Each transition fault is tested along a longest single path passing through the fault site when it is possible. For transition faults that are tested through multi-path propagation, the propagation delay of the target fault is maximized using heuristics. A highly efficient method is also proposed to evaluate the criticality of a transition fault test set. Experimental results show that POTENT generates test sets of higher quality than conventional transition fault test sets.

Table 6.1 Results on All Faults using Weak Non-robust Tests

circuit	# flt.	# det.	#untst	#abt.	Δslack (conv.)	Δslack (POT.)	% red. Δslack	time (conv.) [sec]	time (POT.) [sec]
s1196	2110	2108	2	0	2.68	0.87	67.54	0.5	2.8
s1238	2316	2234	82	0	2.67	1.36	49.06	0.5	3.2
s1423	2512	2239	273	0	18.41	7.63	58.56	0.5	9.6
s1488	2770	2529	241	0	0.46	0.29	36.96	1.1	2.6
s1494	2810	2548	262	0	0.42	0.29	30.95	1.1	2.5
s5378	7040	6412	624	4	0.81	0.16	80.25	7.3	26.4
s9234	11328	9456	1694	178	5.59	4.09	26.83	107.0	174.7
s13207	15602	12489	3097	16	1.47	1.26	14.29	56.3	253.4
s15850	19046	13535	5435	76	4.54	1.82	59.91	120.9	215.7
s35932	63502	54599	8903	0	2.36	1.75	25.85	113.3	605.8
s38417	49738	48747	969	22	1.81	0.51	71.82	303.9	2011.1
s38584	61254	55122	6019	113	1.22	0.85	30.33	519.8	1401.5

Table 6.2 Results on Critical Faults using Weak Non-robust Tests

circuit	# crit. flt.	# det. crit. flt.	Δslack (conv.)	Δslack (POT.)	% red. Δslack	# org. tv	# tv aft add.
s1196	385	385	7.40	1.97	73.38	342	323
s1238	537	535	6.26	3.36	46.33	353	336
s1423	631	603	42.18	17.30	58.99	185	247
s1488	434	429	1.07	0.58	45.79	226	188
s1494	430	425	0.90	0.54	40.00	226	192
s5378	1189	1173	1.51	0.14	90.73	526	593
s9234	1105	1012	19.69	13.23	32.81	1023	1034
s13207	2286	2049	2.94	2.20	25.17	1103	968
s15850	1474	1402	20.05	6.12	69.48	832	927
s35932	28296	25088	3.92	3.02	22.96	143	510
s38417	6908	6876	4.86	1.01	79.22	2588	4112
s38584	1526	1361	7.50	4.94	34.13	2317	2617

References

[1] J. A. Waicukauski, E. Lindbloom, B. K. Rosen, and V. S. Iyengar, "Transition Fault Simulation", *IEEE Design and Test, pp. 32-38*, April 1987.

[2] A. K. Pramanick and S. M. Reddy, "On the Detection of Delay Faults", *Proc. ITC, pp. 845-856*, September 1988.

[3] V. S. Iyengar, B. K. Rosen, and J. A. Waicukauski, "On computing the sizes of detected delay faults", *IEEE TCAD, pp. 299-312*, March 1990.

[4] A. K. Pramanick and S.M. Reddy, "On the Computation of the Ranges of Detected Delay Fault Sizes," *Proc. ICCAD, pp. 126-129*, November 1989.

[5] G. L. Smith, "Model for Delay Faults Based Upon Paths", *Proc. ITC, pp. 342-349*, September 1985.

[6] C. J. Lin and S. M. Reddy, "On Delay Fault Testing in Logic Circuits", *IEEE TCAD, pp. 694-703*, September 1987.

[7] H. Konuk, "On Invalidation Mechanisms for Non-Robust Delay Tests", *Proc. IEEE ITC, pp. 393 -399*, October 2000.

[8] K. Heragu, J. H. Patel, and V. D. Agrawal, "Segment Delay Faults: A New Fault Model", *Proc. 14th IEEE VTS, pp. 32-39*, April 1996.

[9] K. T. Cheng and H.-C Chen, "Delay Testing For Non-Robust Untestable Circuits", *Proc. ITC, pp. 954-961*, October 1993.

[10] K. Fuchs, M. Pabst, and T. Rossel, "RESIST: A Recursive Test Pattern Generation Algorithm for Path Delay Faults Considering Various Test Classes", *IEEE TCAD, pp. 1550-1562*, Dec. 1994.

[11] W. Ke and P. R. Menon, "Delay-Verifiability of Combinational Circuits Based on Primitive Faults", *Proc. ICCD, pp. 86-90*, October 1994.

[12] A. K. Majhi, J. Jacob, L. M. Patnaik, and V. D. Agrawal, "On Test Coverage of Path Delay Faults", *Proc. 9th Int'l Conf. on VLSI Design, pp. 418-421*, January 1996.

[13] W. Mao, M. D. Ciletti, "Robustness Enhancement And Detection Threshold Reduction In ATPG For Gate Delay Faults", *Proc. ITC, pp. 588-597*, September 1992.

[14] K. T. Cheng, A. Krstic, and H.-C. Chen, "Generation of High Quality Tests for Robustly Untestable Path Delay Faults", *IEEE Trans. on Computers, pp. 1379-1392*, December 1996.

[15] I. Pomeranz and S. M. Reddy, "An Efficient Non-Enumerative Method to Estimate Path Delay Fault Coverage", *Proc. 29th ICCAD, pp. 560-566*, November 1992.

[16] S. Kajihara, K. Kinoshita, I. Pomeranz and S. M. Reddy, "A Method for Identifying Robust Dependent and Functionally Unsensitizable Paths", *Proc. 10th Int'l Conf. on VLSI Design, pp. 82-87*, January 1997.

[17] Y. Shao, S. Kajihara, I. Pomeranz, and S. M. Reddy, "An Efficient Method to Identify Untestable Path Delay Faults", *ATS, pp. 233-238*, November 2001.

[18] Y. Shao, S. M. Reddy, I. Pomeranz, and S. Kajihara "On Selecting Testable Paths in Scan Designs", ETW 2002.

[19] M. Abramovici, M. Breuer, and A. Friedman, Digital Systems Testing and Testable Design, IEEE Press, 1994.

[20] I. Pomeranz and S. M. Reddy, "Forward-Looking Fault Simulation for Improved Static Compaction", *IEEE TCAD, pp. 1262-1265*, October 2001.