DELAY FAULT COVERAGE, TEST SET SIZE, AND PERFORMANCE TRADEOFFS

by

William K. Lam, Alexander Saldanha, Robert K. Brayton,
Alberto L. Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M92/119

29 October 1992

COVER PAGE

DELAY FAULT COVERAGE, TEST SET SIZE, AND PERFORMANCE TRADEOFFS

by

William K. Lam, Alexander Saldanha, Robert K. Brayton
Alberto L. Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M92/119

29 October 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

DELAY FAULT COVERAGE, TEST SET SIZE, AND PERFORMANCE TRADEOFFS

by

William K. Lam, Alexander Saldanha, Robert K. Brayton
Alberto L. Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M92/119

29 October 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Delay Fault Coverage, Test Set Size, and Performance Tradeoffs

William K. Lam* Alexander Saldanha† Robert K. Brayton
Alberto L. Sangiovanni-Vincentelli

University of California - Berkeley CA

Abstract

We prove that 100% delay-fault testability is not necessary to guarantee the speed of a combinational circuit. We show there exist path delay-faults which can never impact the circuit delay (computed using any correct timing analysis method) unless some other path delay-faults also affect it. These are termed robust dependent delay-faults and need not be considered in delay-fault testing. Necessary and sufficient conditions under which a set of path delay-faults is dependent are proved; this yields more accurate and larger delay-fault coverage estimates than previously used. Next, assuming only the existence of robust delay-fault tests for a very small set of paths (linear in circuit size), we show how the circuit speed (clock period) can be selected such that 100% robust delay-fault coverage is achieved. This leads to a quantitative tradeoff between the amount of testing effort (test set size) for a circuit and the verifiability of its performance. Finally, under a bounded delay model, we show that the test set size can be substantially reduced while maintaining the delay-fault coverage for the specified circuit speed. Examples and experimental results are given to show the effect of these three techniques on the amount of delay fault testing necessary to guarantee correct operation.

1 Introduction

The need for ensuring the temporal (dynamic) correctness of circuits has led to the development of delay-fault testing theory and methodologies [9]. Two fault models have been proposed for delay-fault testing, namely gate delay-faults and path delay-faults. Of these the path delay-fault model is more attractive since it models a larger set of defects and can give guarantees on circuit performance [16]. However, because there can be an exponential number of paths in a circuit, path delay-fault testing often requires very large test sets. This paper addresses the issue of guaranteeing circuit performance using only a small (possibly linear size) test set. For circuits where a 100% guarantee is not possible, we also address the question of providing a large fault coverage guarantee with small size test sets.

*Research support by Hertz Foundation and SRC
†Research support by Fujitsu Laboratories Ltd.

A **path delay-fault** occurs along a path P if the delay along P falls outside its specified limits (in general, either upper or lower bounds). Each path has two possible long path delay-faults associated with it, a rising delay-fault and a falling delay-fault. [1] Short path delay-faults can also occur and cause problems. Examples are in wave pipelined circuits, clocking schemes using transparent latches, and asynchronous circuits. The goal of delay-fault testing is to certify the absence of delay-faults that cause the manufactured circuit to fall outside its specified delay bounds.

In this paper we focus on upper bounds on circuit performance. We show that the path delay-faults in a circuit can be partitioned into two sets. For the first set, the occurrence of one or more delay-faults can cause an increase in circuit delay. This set must be tested for delay-faults to certify correct operation. It includes all robust testable path delay-faults (RPDF). The delay-faults in the second set are termed **robust dependent (RD) delay-faults.** The occurrence of these delay-faults cannot increase the circuit delay unless some non-RD delay-faults also occur. Thus, RD delay-faults need not be tested to ensure that a manufactured circuit operates at the desired speed.

We define the **delay-fault coverage** of a test set as the percentage of all non-RD delay-faults guaranteed not to occur if the circuit passes the test set. This is different from the percentage of all delay-faults that are robust testable. The latter is commonly called the **delay-fault testability** of the circuit. There may be delay-faults in a circuit for which no robust test exists, yet the absence of the delay-fault is guaranteed by the test set (since the circuit passed the delay tests). Thus, delay-fault coverage is a function not only of the test set size but also of the number of faults implied absent by the test set. The fault coverage of a test set can also depend on the delay bounds used in the testing procedure.

The approach taken in this paper is different from all previous work in both delay-fault test generation [8, 14, 13, 1] and synthesis [7, 12, 3, 4, 11, 6]. Up to now the ability to detect delay defects in a circuit has been measured by its delay-fault testability. Full testability requires that every path has a robust test. However, the goal of delay-fault testing is to be able to detect every delay defect that causes a late (early for short-path delay-faults) transition at some output of the circuit during normal operation. Thus delay-fault testing serves to guarantee the functional correctness (with respect to timing) of a circuit. In particular, paths along which delay-faults can never individually affect circuit functionality are of no interest both in test generation and synthesis. None of the previous work appears to have addressed this issue of delay testing. In this paper we demonstrate that this aspect of delay testing is crucial in providing a realistic estimate of the delay-fault coverage and also in reducing the delay testing effort for most circuits.

This view of delay-fault testing also differs from the philosophy of stuck-fault testing. The goal of stuck-fault testing is to determine a test for each fault or prove it redundant. Removal of redundant stuck-faults also yields smaller area and better performance reliability. This requirement of stuck-fault testing arises from a good correlation that exists between the absence of stuck-fault defects and the reliability of the circuit; that is, a circuit with fewer stuck-fault defects is more likely to operate correctly than one with more stuck-fault defects. This is not the case with delay faults where a certificate of correct timing functionality at the outputs of the circuit is desired. Hence, robust untestable paths are of no concern (and need not be isolated or removed) if they cannot impact the circuit delay.

[1] Each path delay-fault is associated with a path. Since we are dealing with the impact of delay-faults on circuit delay, we interchangeably refer to either the delay-fault or the associated path.

This paper contributes to the question of how much delay fault testing is necessary in three ways:

1. Necessary and sufficient conditions for a set of delay-faults to be RD are derived. We show that the delay-faults on any set of long false paths that can exist for any particular delay assignment to connections in the circuit forms an RD set. This does not affect the test set size since every RPDF still must be tested. However, it does give a more realistic and better estimate of the delay-fault coverage.

2. We show how to trade-off delay-fault coverage and circuit speed. By relaxing the operating speed for the circuit (this is the clock period in synchronous circuits), a larger delay-fault coverage can be achieved. We call the minimum clock rate for which the coverage is 100% the **robust delay** of the circuit. Using this concept, we present path delay-fault testing methodologies whose test set sizes are **linear** in the size of the circuit (as opposed to 100% RPDF testing which can be exponential in size).

3. Using a bounded delay model (the delay of a tested path is guaranteed to lie between both upper and lower bounds) we formulate a simple linear program to reduce the delay-fault test set size without decreasing the fault coverage.

Examples are given to illustrate these effects and the corresponding test set size.

The paper is organized as follows. The context for our work is provided by Section 2 which discusses the issues that arise in verifying circuit delays. Section 3 provides background information on robust delay-fault testing and delay analysis. The concept of robust dependent delay-faults is described in Section 4. Section 5 introduces the notion of robust delay and illustrates the tradeoffs between testing effort and performance verifiability in circuits. Test set reduction under a bounded delay model while maintaining the same fault coverage for the specified circuit speed is discussed in Section 6. Section 7 illustrates how each of these three aspects impacts the delay-fault coverage of a test set. Preliminary experimental results are presented in Section 8. Finally, Section 9 discusses how the theory described here can be used for the synthesis of circuits with 100% robust delay-fault coverage.

2 Verifiable delay analysis

A **connection** is just a wire or a pin to pin path through a gate. A **path** is denoted as a sequence of connections or points. A connection or point is denoted by a lower case letter, a path by an upper case letter, and a set of paths by an upper case script character. The delay of a connection $e = (a, b)$ between points a and b is denoted either $\delta(e)$ or $\delta(a, b)$, and the delay of path P is denoted $\delta(P)$. Delays may depend on whether a signal is rising or falling in which case a delay is denoted $\delta^r(e)$ or $\delta^f(a, b)$. For a path P, $\delta^r(P)$ means the delay of P when its output is rising; similarly for falling.

A circuit design C is a net list of gates. In analyzing a design, typically one assumes some delay model, for example a connection e has rising delay $\delta^r(e)$ such that $d^r_{\min} \leq \delta^r(e) \leq d^r_{\max}$ (called a **bounded delay model**). Usually an analysis of the performance of a circuit design derives properties about the design assuming that all connections meet their delay bounds. For example, in timing verification [10], an upper bound τ on the delay the circuit is computed, where τ is some complex function of the assumed delay bound parameters. Note that this

analysis is valid only for **acceptable** designs and manufactured circuits whose connections meet the delay bounds.

To guarantee that a **manufactured circuit** C_m operates at some specified delay we must make different assumptions since we know nothing a priori for the circuit under test. The following information is known if C_m passes the test. Let Q be the set of paths tested such that for each $P_i \in Q$,

$$\tau^i_{\min} \leq \sum_{e \in P_i} \delta(e) \leq \tau^i_{\max},$$

where $\tau^i_{\min}$ and $\tau^i_{\max}$ depend on the type of testing done. Note that there are no assumptions about delays on connections; only verifiable delays of paths are used. Only implicitly is there an assumption about connection delays, *i.e.* there exists some fixed but unknown delay $\delta(e)$ on each connection e. Thus the circuit under test (the manufactured circuit) is $C_m = (C, \Delta)$ where Δ is a vector of delay values, one value for each connection. We say Δ is a **delay assignment**. We assume $\Delta^e \geq 0$ for connection e but no upper bounds are assumed; just that certain sums are bounded:

$$\tau^i_{\min} \leq \sum_{e \in P_i} \Delta^e \leq \tau^i_{\max}.$$

The fact that we have no delay model assumptions on connections does not preclude delay analysis on C. We shall show that the following kind of analysis is possible. Let $D \subseteq \mathcal{P}$, where $\mathcal{P}$ is the set of all complete paths of C, *i.e.* paths from primary inputs to primary outputs (such as those tested). It is possible that the following can be determined through analysis: for any τ, if $\delta(P_i) \leq \tau, P_i \in \mathcal{P} - \mathcal{D}$, then $\delta(P_j) \leq f(\tau), P_j \in \mathcal{D}$. This implies that even before a manufactured circuit C_m is tested, we know that paths in $\mathcal{D}$ need not be tested for guaranteeing C_m as good or bad. In other words, if C_m passes a delay test at speed τ for paths in $\mathcal{P} - \mathcal{D}$, then this implies that C_m is guaranteed to operate correctly (by analysis) at speed $\max(\tau, f(\tau))$. Here $f(\tau)$ is some function determined by analysis from the net list C.

We show two types of this kind of analysis in the paper. The first is where $f(\tau) = \tau$; the set of paths $\mathcal{D}$ for which such a property holds is called **robust dependent**. The second type of analysis is where $f(\tau)$ is a complex function evaluated by solving a linear program. Both of these types of analysis are important in establishing an accurate estimate of the delay-fault coverage of a set of paths Q that is tested.

Delay analysis has been the subject of intense research in the past few years. There are two dimensions: the delay model used (unbounded[2], bounded, fixed, etc.), and the type of sequencing of input vectors assumed (single, double, multiple, periodic). Although our delay model for $C_m = (C, \Delta)$ is a fixed but unknown delay, the model for establishing $\delta(P_j) \leq f(\tau)$ can be obtained by any analysis that is a relaxed version of this. Thus even though $\delta(e) = \Delta^e$ for connection e of C_m, analysis methods assuming $\Delta^e_{\min} \leq \delta(e) \leq \Delta^e_{\max}$ also give legitimate bounds. Of course all the analysis methods used in this paper start only with the assumption that $\tau^i_{\min} \leq \sum_{e \in P_i} \Delta^e \leq \tau^i_{\max}$ for those paths P_i in the test set.

Note that the assumption that the circuit under test is $C_m = (C, \Delta)$ for some unknown fixed delay assignment Δ considers static variations in delays that occur from one manufactured circuit to another such as process variation, operating temperature, and circuit age. One concern with this assumption is whether it accounts for the small uncertainties in delays during circuit operation due to dynamic factors such as crosstalk, degraded signals, and slope factors.

[2]An unbounded delay model is one where $0 \leq \delta \leq d_{\max}$, *i.e.* $d_{\min} = 0$.

Since the analysis techniques used in computing the circuit delay are **robust** [10], *i.e.* the delay estimate is a valid upper bound for each delay assignment that lies within the tested lower and upper bounds. Thus, if C_m passes the delay tests for the worst case upper and lower bounds, every operation of C_m within these delays bounds is also guaranteed to be correct. It is also instructive to realize that delay analysis based on verifiable path delays is less likely to be impacted by dynamic delay variations than delay analysis based on gate delays. This is because it is likely that a small delay variation on a connection will not get reflected on the path delay. Of course, nothing is known if C_m operates outside these bounds; one is forced to explicitly test C_m in this case.

In summary, the delay analysis of this paper differs from the usual in that we perform delay analysis using only verifiable path delays instead of non-verifiable connection delay assumptions.

3 Definitions

3.1 Robust delay-fault testing

A brief definition for robust path delay-fault testing is provided in this section. The conditions for robust delay-fault testing are rephrased from [16].

A **controlling value** for a gate f is a value at its input that determines the value at the output independent of the other inputs, and is denoted $A(f)$. For example, 0 is a controlling value for an AND gate. A **non-controlling value** for a gate f is a value at its input which is not a controlling value for the gate, and is denoted $I(f)$. For example, 1 is a non-controlling value for an AND gate. A **simple gate** is any one of AND, OR, NAND, NOR, and NOT. All the results described here apply only to simple gates. Only these gates have controlling vs. non-controlling values for each input.

Let $P = \{f_0, f_1, ..., f_m\}$ be a path where each $f_i, i > 0$ is the output of a gate. The inputs of f_i other than f_{i-1} are called **side-inputs** of f_i along P and denoted $S(f_i, P)$. A path that starts at a primary input and ends at a side-input of P is a **side-path** of P. A **path delay-fault** occurs along path P if the delay along P falls outside its specified limits (in general, either upper or lower bounds).

Definition 3.1 *A test for a delay-fault is* **robust** *if and only if the test is valid independent of delays on all the connections in the circuit. A circuit has* **100% robust delay-fault testability** *if and only if every path delay-fault has a robust test.*

Definition 3.2 *Let* $P = \{f_0, f_1, ..., f_m\}$. *A delay-fault for the rising (falling) transition at* f_m *is said to be* **robust testable** *by the vector pair* $< v_1, v_2 >$ *if and only if at each node* f_i, $< v_1, v_2 >$ *yields the desired transition being tested, and for each* $g_j \in S(P, f_i)$*:*

1. $g_j(v_2) = I(f_i)$, *and*
2. *if* $f_{i-1}(v_1) = I(f_i)$, *then there is no transition on* g_j.

The vector v_2 *applied after* v_1, *delayed by an amount greater than the longest path in the circuit.*

3.2 Path sensitization

Definition 3.3 *A* **leaf-dag** *is a circuit composed of AND and OR gates with multiple fanout and inverters only permitted at the inputs. An inverter is not allowed multiple fanout.*

Every circuit (composed of simple gates) can be converted to a leaf-dag, albeit with possibly an exponential number of gate duplications.

Definition 3.4 *The* **I-edge** *of a path in a circuit refers to either the connection from the primary input if no inverter is there, or else the connection immediately after the inverter.*

Let η be the leaf-dag of C. Any delay assignment in C also exists in η; simply use the same delay range on each connection in η as the corresponding connection in C. However, the converse is not true.

The following definition is adapted from [2]. Given a delay assignment and assuming an initial unknown value on each connection of a circuit C, a path $P = \{f_0, f_1, ..., f_m\}$ in C is true under a single vector v if and only if the following conditions hold at each gate $f_i, i > 0$:

1. If all inputs of f_i are non-controlling, f_{i-1} must be the last side-input to present the non-controlling value.

2. If at least one input of f_i is controlling, f_{i-1} must be the first side-input to present the controlling value.

This definition extends naturally to sensitization under a multiple vector input denoted $v_0, v_1, ..., v_n$ with v_n applied last. The single vector **delay of a circuit** is determined by the longest true path under any input vector for the given delay assignment; similarly for multiple vector delay. The delay of a manufactured circuit refers to the actual operating delay of the circuit and is determined by the longest true path for the unknown delay assignment that exists in the manufactured circuit.

The following theorem relates the testability of a multiple stuck-0 (stuck-1) fault on the I-edges of each path of length at least L to the existence of a true path under a single vector of the primary inputs whose rising delay is of length at least L. Given a set M of I-edges, the multiple fault M stuck-0 (stuck-1) corresponds to a stuck-0 (stuck-1) fault on each I-edge in M.

Theorem 3.1 (adapted from [5]) *Let C be a given circuit and η its leaf-dag. Let $\mathcal{P} = \{P_1, P_2, ..., P_n\}$ be the set of all paths in C of length at least L, and let P_η be the corresponding paths in leaf-dag η. At least one $P_i \in \mathcal{P}$ is true under the vector v for the rising (falling) transition at an output in C if and only if v is a test in η for the multiple stuck-0 (stuck-1) fault on the I-edge of each path in $\mathcal{P}_\eta$.*

Theorem 3.1 describes the exact condition for computing the delay of a circuit under a single input vector. A tighter delay may be achieved by using a multiple vector criterion. However, the single vector criterion is an upper bound on the delay achieved by any correct multiple vector criterion. Thus, the existence of a test in η for the multiple stuck-fault on the I-edges of each path in $\mathcal{P}_\eta$ (in Theorem 3.1) is a necessary condition for a path in $\mathcal{P}$ in C to be true under multiple vector criterion. However, it is unknown if this is sufficient under a multiple vector criterion.

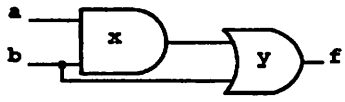

Figure 1: Redundant circuit with 100% robust delay-fault coverage

Corollary 3.1 *Let C be a given circuit and η its leaf-dag. Let $\mathcal{P} = \{P_1, P_2, ..., P_n\}$ be the set of all paths in C of length at least L, and let P_η be the corresponding paths in leaf-dag η. At least one $P_i \in \mathcal{P}$ is true under the vectors $v_0, v_1, ..., v_n$ for the rising (falling) transition at an output in C only if v_n is a test in η for the multiple stuck-0 (stuck-1) fault on the I-edge of each path in $\mathcal{P}_\eta$.*

4 Robust dependent delay-faults

4.1 Functional interactions between paths

We demonstrate with some examples the impact that the functional interaction between paths can have on the delay-fault coverage.

Example: Consider the simple circuit of Figure 1. There are six delay-faults (a rising and falling transition delay-fault for each path). The paths $\{b, y, f\}$ and $\{b, x, y, f\}$ are robust testable for falling transition delay-faults, but only path $\{b, y, f\}$ is robust testable for the rising transition delay-fault. Path $\{a, x, y, f\}$ is robust untestable for both transitions and $\{b, x, y, f\}$ is untestable for the rising transition.

First consider the case of a falling transition at the output f. We argue that whenever a falling transition propagates along $\{a, x, y, f\}$ a longer falling delay exists along $\{b, x, y, f\}$. Assume that a falling delay propagates along $\{a, x, y, f\}$. The output of x changes (after some delay) to 0 when any input switches to 0. Thus, b must be at 1 if the falling edge on a propagates through x. If the falling transition at x is the last event to propagate through y, input b of y must settle at 0 before the falling edge arrives at the input x of y. b must have a transition on it from either 0 to 1 or 1 to 0 in order for this to happen under any delay assignment. If b has a rising transition, the final value of f is 1, and the last event at f is a rising edge. Hence the falling transition along $\{a, x, y, f\}$ is not the last event at f. If b has a falling transition, b must be 1 at the moment a reaches x, since $\{a, x, y, f\}$ is the longest falling path. But since b eventually changes to 0, the path $\{b, x, y, f\}$ is longer than $\{a, x, y, f\}$ for the falling transition. Since a robust test exists for $\{b, x, y, f\}$, the occurrence of this longer delay-fault can be detected. Hence $\{a, x, y, f\}$ need not be tested for the falling delay-fault if $\{b, x, y, f\}$ is tested for a falling delay-fault.

Now consider the rising transition at the output of f. Assume that path $\{b, y, f\}$ is tested for the rising delay-fault. To propagate the last rising edge to f along $\{a, x, y, f\}$, b must have

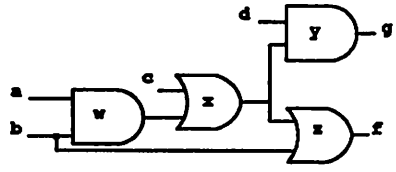

Figure 2: Irredundant circuit with 100% delay-fault coverage without 100% delay-fault testability

a rising transition. Thus, if $\{a, x, y, f\}$ is the last rising edge, then the rising transition from b through the output of y is later than that from x to the output of y. Hence $\{b, y, f\}$ is longer than $\{a, x, y, f\}$. Similarly, it can be shown that $\{b, x, y, f\}$ cannot propagate the last rising edge at f unless $\{b, y, f\}$ has a longer rising delay. Thus neither $\{a, x, y, f\}$ nor $\{b, x, y, f\}$ can affect the rising delay at f without implying that the rising delay of $\{b, y, f\}$ is longer. ■

There are several conclusions to be drawn from the example:

1. The requirement of robust delay-fault testing is related to the sensitization of the path. However, unlike path sensitization which is determined under a specific delay assignment to the connections of the circuit, the requirement for testing a delay-fault along a path is a delay-insensitive condition and must be valid for all possible delay assignments.

2. 100% robust delay-fault testability is not necessary for 100% robust delay-fault coverage.

3. Even 100% single stuck-fault testability is not necessary for 100% robust delay-fault coverage. In Figure 1 the stuck-0 fault on the AND gate is redundant. This is a particularly significant since it is widely believed that a circuits without complete stuck-fault coverage are not amenable to testing for delay-faults [3].

The question of whether circuits with 100% robust delay-fault coverage but without 100% delay-fault testability must always contain redundant stuck-faults is resolved negatively by the next example.

Example: The circuit shown in Figure 2 is 100% testable for single stuck-faults. Like the circuit of Figure 1, the path $\{a, w, x, z\}$ for rising and falling transitions, and $\{b, w, x, z\}$ for rising transition are not robust delay-fault testable. All other paths are are robust testable and the circuit has 100% delay-fault coverage; each untestable fault cannot affect the circuit delay without the occurrence of delay-faults on some of the testable paths. ■

Of course, 100% RPDF is a sufficient condition for 100% delay-fault coverage. However most synthesis techniques for 100% RPDF typically involve a large area penalty. Often the performance of the circuit is also degraded. We will show sufficient conditions under which a path need not be made robust testable. While we cannot claim that the condition is necessary, we show topological conditions under which it is necessary.

4.2 Conditions for RD paths

Definition 4.1 *Let $\mathcal{D}$ be the set of all path delay-faults in a circuit C and $\mathcal{R}$ a subset of $\mathcal{D}$. If for all τ, the absence of delay-faults ($> \tau$) in $\mathcal{D} - \mathcal{R}$ implies the delay of C is $\leq \tau$, R is said to be* **robust dependent (RD)**.

A RD set is insensitive to the delay assignment, *i.e.* $\mathcal{R}$ can be eliminated from consideration in delay-fault testing under every delay assignment. This is in contrast to some delay-faults whose absence is implied under only some and not all delay assignments; this is considered in Sections 5 and 6. Another significant point is that the implied delay of the circuit ("delay of C" in Definition 4.1) may be computed by any correct timing analysis method such as the single or multiple vector criteria referred to earlier. For example, an RD set using a single-vector criterion means that the implied circuit delay is computed according to Theorem 3.1.

With every rising (falling) path delay-fault is an associated path for the rising (falling) transition. A set of paths is called a **RD path-set** if the delay-faults on the paths is an RD set.

We now state a sufficient condition for RD paths. An internally non-inverting circuit is one that has inverters only at the primary input leads of the circuit. Any circuit can be converted to an internally non-inverting circuit with at most a single duplication of each gate of the circuit.

Lemma 4.1 *Let η be an internally non-inverting circuit and let M be a set of I-edges. If M stuck-0 (stuck-1) is redundant, then every subset of M is also stuck-0 (stuck-1) redundant.*

Proof: Express each output as a function of the literals corresponding to I-edges of η. Now each output may be viewed as a monotonic increasing Boolean function of these Boolean literals. A stuck-0 (stuck-1) fault on a subset of I-edges cannot increase (decrease) a monotonic increasing function. Since M is a redundant multiple stuck-0 (stuck-1) fault, it does not decrease (increase) any output function; if it did M would be testable. Thus, neither can any subset of M, which proves each stuck-0 (stuck-1) subset of M is redundant. ■

Since every leaf-dag is an internally non-inverting circuit, the above result holds for all leaf-dag's.

Theorem 4.1 *Let C be a given circuit and η its leaf-dag. Let $\mathcal{R}_\eta$ be the paths in η corresponding to a set of paths $\mathcal{R}$ in C. Let M_η be the I-edges of $\mathcal{R}_\eta$. If M_η stuck-0 (stuck-1) is redundant in η, then $\mathcal{R}$ is a rising (falling) RD path-set in C.*

Proof: Let M_η be stuck-0 redundant. Suppose path $P \in \mathcal{R}$ of length L is true for the rising transition for some delay assignment. By Corollary 3.1, the multiple fault corresponding to a stuck-0 fault on the set F of I-edges of all paths of length at least L (for the given delay assignment) is testable in η. F includes at least one edge $e \notin M_\eta$ since M_η and all its subsets are stuck-0 redundant (Lemma 4.1). Thus some path outside $\mathcal{R}$ is a true path of length at least L. By the definition of robust dependent delay-faults, this implies that $\mathcal{R}$ is an RD path-set in C.

The proof for the delay of the falling transition is similar. ■

Theorem 4.1 yields a sufficient condition under which a set of paths cannot affect the circuit delay under any delay assignment. Even though a path $P \in \mathcal{R}$ may become sensitizable under some delay assignment, it does not determine the circuit delay, since there exists another path $Q \notin \mathcal{R}$ at least as long as P. A delay-fault along P causes a delay at the circuit outputs if and only if a delay-fault exists along Q. Thus, delay-faults for paths in $\mathcal{R}$ need not be tested.

Figure 3: Sensitizable paths and RD delay-faults

The condition of Theorem 4.1 is sufficient. However it is necessary under the single vector criterion for delay analysis and the assumption that C has the following property: given any ordering of path lengths there exists a delay assignment to the connections of C that realizes the ordering. In general, all circuits do not meet this requirement (leaf-dag's do).

Theorem 4.2 *Assume that each ordering of path lengths in a given circuit C can be realized by some delay assignment. Let η be the leaf-dag corresponding to C, and $\mathcal{R}_\eta$ the paths in η corresponding to a set of paths $\mathcal{R}$ in C. Let M_η be the l-edges of $\mathcal{R}_\eta$. $\mathcal{R}$ is a rising (falling) RD path-set using the single vector criterion in C if and only if M_η stuck-0 (stuck-1) is redundant in η.*

Proof:
If part: Given by Theorem 4.1.
Only if part: Assume the multiple fault M_η stuck-0 is testable. Consider any delay assignment that makes the paths outside $\mathcal{R}_\eta$ shorter than all the paths in $\mathcal{R}_\eta$. By assumption, this partial order between the path lengths can also be satisfied in C by some delay assignment. By Theorem 3.1, the delay of C is at least the length of the shortest path in $\mathcal{R}$, since M_η is testable in η. Consequently, $\mathcal{R}$ is not an RD path-set for the single vector criterion in C. ■

It is possible to develop a tighter necessary condition which takes into consideration only those path orderings which exist for some delay assignments in a given circuit; however we do not have any application for it as yet. We do not know a necessary condition for RD delay-faults under multiple vector criteria for delay computation since the sufficient condition for a path to be true under any of these criteria is unknown.

Example: We use the circuit of Figure 3 to explain the difference between path sensitization and robust dependent delay-faults. Upper bounds on delays are shown within each gate; lower bounds are assumed to be 0. Since the output is constant at 0, the last transition under any input change is a falling edge. Thus the rising delay of the circuit is 0; hence we are not concerned with testing for rising transition delay-faults. None of the falling transition delay-faults along the three paths in the circuit (denoted P, Q, and R in the figure) is RPDF. The sets $\{P\}$ and $\{Q\}$ are both RD but the set $\{P, Q\}$ is not RD; neither is $\{R\}$ an RD set. However, under different delay assignments each path is responsible for the last falling transition at the output f. In the circuit shown on the left R is the longest sensitizable path; P and Q are the longest sensitizable paths in the circuit on the right. Thus, although a path may be sensitizable under some delay assignment it may still be part of an RD set. ■

We note that the statement of Theorem 4.1 is different from previous work in timing analysis [5]. The distinction is that the previous work provides a necessary condition under which a path is sensitizable. In contrast, Theorem 4.1 provides a sufficient condition (Theorem 4.2 shows when this condition is also necessary) under which a set of paths (even possibly sensitizable) cannot independently affect circuit delay. In the example of Figure 3 each of the paths P, Q, and R is a true path (and statically co-sensitizable) under different delay assignments.

4.3 False paths and RD delay-faults

We will refer to the *intrinsic delay-fault coverage* as the best fault coverage possible after the identification of a maximal RD set. It is doubtful that identification of a maximum RD set will be computationally tractable on most circuits. In practice, therefore, we are interested in an estimate of the RD set which can be used to compute a lower bound for fault coverage. This is termed the *reported delay-fault coverage*. A technique for identifying a large RD set is related to the identification of long false paths in timing analysis [10].

Lemma 4.2 *If $\mathcal{R}$ is an RD set, then every subset of $\mathcal{R}$ is also an RD set.*

Proof: Direct by using Lemma 4.1 on the condition of Theorem 4.1. ■

Theorem 4.3 *Let C be a circuit whose longest true path for all single vectors is less than L under some delay assignment. Then the set of delay-faults on all paths of length $\geq L$ is an RD set.*

Proof: Let η be the leaf-dag corresponding to C. Without loss in generality consider the rising transition delay of η. By Theorem 3.1 the multiple fault M corresponding to a stuck-0 fault on the I-edge of each path of length $\geq L$ is redundant. By Theorem 4.1, the delay-faults on these paths is an RD set in η. As stated earlier, since there is a one-to-one correspondence between delay-faults in η and C, the result follows. ■

There exist well-known circuits with many long false paths and the above result eliminates consideration of these paths in robust delay-fault testing. The theorem extends the RD property to every set of long false paths under any delay assignment to the wires of the circuit. Since a set of delay-faults is RD if the corresponding paths are long false paths, for any given delay assignment, we have the flexibility of choosing a delay assignment to try to maximize the number of long false paths. Heuristic algorithms based on this, such as the one described in the next section, are being explored.

4.4 Finding an RD set

Theorem 4.1 states the sufficient condition for an RD set in a given circuit C in terms of the testability of a multiple stuck-fault in the leaf-dag η corresponding to C. In practice, the identification of redundant multiple stuck-faults and the transformation to a leaf-dag are difficult to perform - the former requires large compute times, the latter usually requires enormous space.

Before describing an algorithm to identify a maximal RD set, we state a result that allows highly developed single stuck-fault redundancy identification methods to be exploited in determining a maximal RD set.

Theorem 4.4 *Let C be a non-inverting circuit and M a redundant multiple stuck-0 fault in C (faults considered on or after I-edges). Let C_M be the circuit obtained by replacing each connection in M by 0. If P is a rising (falling) RPDF path in C, then P is rising (falling) RPDF in C_M.*

Proof: Assume P is rising RPDF in C but is not rising RPDF in C_M. P cannot be in any rising RD set of C. There are two cases to consider since P is not rising RPDF in C_M. If some edge in P is set to 0 then the rising RD set (determined by M) includes P, thus contradicting. Thus P is non-RPDF in C_M and no edge of P is set to 0. Let $< v_1, v_2 >$ be the robust rising transition test for P in C. Consider any OR gate f in P. By Definition 3.2 each side-input to f is at 0 under both v_1 and v_2 in C and there is no transition on these side-inputs. Since C is non-inverting and a multiple stuck-0 fault is asserted, these side-inputs to f remain at 0 under v_1 and v_2 with no transition even in C_M. Hence propagation of the rising transition at any of the OR gates in P is not affected. Consider any AND gate in P. By Definition 3.2 each side-input to an AND gate is at 1 under v_2 in C. Since P is non-RPDF in C_M, at least one side-input to one or more AND gates in P is 0 under v_2 in C_M. In C, the output of P is 1 under v_2. In C_M the output of P is 0. Hence v_2 is a test for M, thus contradicting that M is redundant. ■

The converse is not true; if P is RPDF in C_M, it may not be RPDF in C. The theorem states an important result about robust delay-fault testability; a path P that is rising RPDF remains rising RPDF under removal of stuck-0 redundancies in a non-inverting circuit. This directly gives us a strategy for identifying a maximal RD set. This technique identifies redundant multiple stuck-faults by iteratively identifying redundant single stuck-faults. It eliminates the need to unfold a given circuit into a possibly exponential size leaf-dag.

The algorithm operates on a non-inverting circuit C' derived from a given circuit C by duplicating each gate at most once. The main loop of the algorithm iteratively performs two steps. First, redundant stuck-0 connections in C' are identified and replaced by 0. C' is irredundant at the termination of this step. In the second step selective duplication of gates is performed in C'. This possibly creates new redundant connections. These two steps are iterated until the resulting C' is fanout-free and irredundant. The paths in C' not passing through any constant connection form the non-RD set; the testability of these paths is determined in C to compute the delay-fault coverage.

Simple heuristics are used to select the order of redundancy removal and duplication. Redundant connections are determined in decreasing order of the number of paths passing through them. This aims at eliminating the largest set of paths possible at each step. Duplication is performed on a level by level basis from primary outputs to primary inputs. This allows redundant connections to be created close to the primary outputs, thus potentially allowing the elimination of large numbers of paths. While the resulting RD set is maximal, no claims are made on whether this strategy yields a set that is close to the maximum RD set.

We illustrate the working of the algorithm on the circuit of Figure 5. The given circuit (top-left in Figure) is non-inverting. Since there are no redundant single stuck-0 faults, we reduce the fanout of gate x by duplicating it (top-right in Figure). There is now one redundant stuck-0 fault in the circuit. On removing it, the resulting circuit has no internal fanout greater than one (bottom in Figure) and the algorithm completes. Thus the rising RD set for this circuit consists of the paths $\{a, w, x, z, f\}$ and $\{b, w, x, z, f\}$ which pass through the redundant connection. Note that if we had directly operated on the leaf-dag of the given circuit (shown in Figure 6) the multiple stuck-0 fault is identified on the I-edges of these same two paths.

```
Convert C to an internally non-inverting circuit C'
R = {}
While (C' has any gate with fanout > 1) {
   While (C' has a redundant stuck-0 connection e) {
      Replace e by constant 0
      R = R ∪ {e}
   }
   Find gate g with fanout > 1
   Duplicate g and move fanout connections
      so each duplicate has less fanout than g
}
All paths with some connection in R form rising RD set.
```

Figure 4: Algorithm to find a maximal rising RD set

5 Robust delay

In Section 4 it is shown that for 100% delay-fault coverage there should only be either RPDF or RD delay-faults. But what about circuits that do not meet this criterion? For these circuits, one would like to get as much information about the circuit performance as possible from the limited RPDF set. It is easy to see that testing only the RPDF set in such a circuit is not sufficient to guarantee the circuit operates at the specified speed; some untestable non-RPDF and non-RD delay-faults could cause the circuit delay to exceed specification. This raises the question of whether there is a minimum clock period τ such that the passing of the robust tests for a RPDF subset will guarantee the circuit delay will never exceed τ under all possible delay faults. If such a τ exists, then 100% delay-fault coverage (relative to τ) is achieved by testing a fraction of the robust testable delay-faults. Obviously, the existence of τ depends on the topological distribution of the subset of RPDF tested. Here, we give a mild condition that guarantees the existence of finite τ. A set of paths is called a **RPDF path-set** if every delay-fault on each path in the set is RPDF.

Definition 5.1 *The* **robust delay** τ_R *of a circuit is the minimum circuit delay* τ *that can be guaranteed by testing all the RPDF delay-faults.*

Theorem 5.1 *Let* $\mathcal{D}$ *denote the set of all paths in a circuit* C, $\mathcal{P} \subseteq \mathcal{D}$ *an RPDF path-set and* $\mathcal{R} \subseteq \mathcal{D}$ *a RD path-set. If every connection* $\mathcal{D} - \mathcal{R}$ *appears along some path in* $\mathcal{P}$, *then* C *has finite robust delay.*

Proof: Let d be the maximum delay of any path in $\mathcal{P}$ and let n denote the maximum number of edges in any path in C. Each path $P_i \in \mathcal{P}$ is tested to ensure that its delay does not exceed

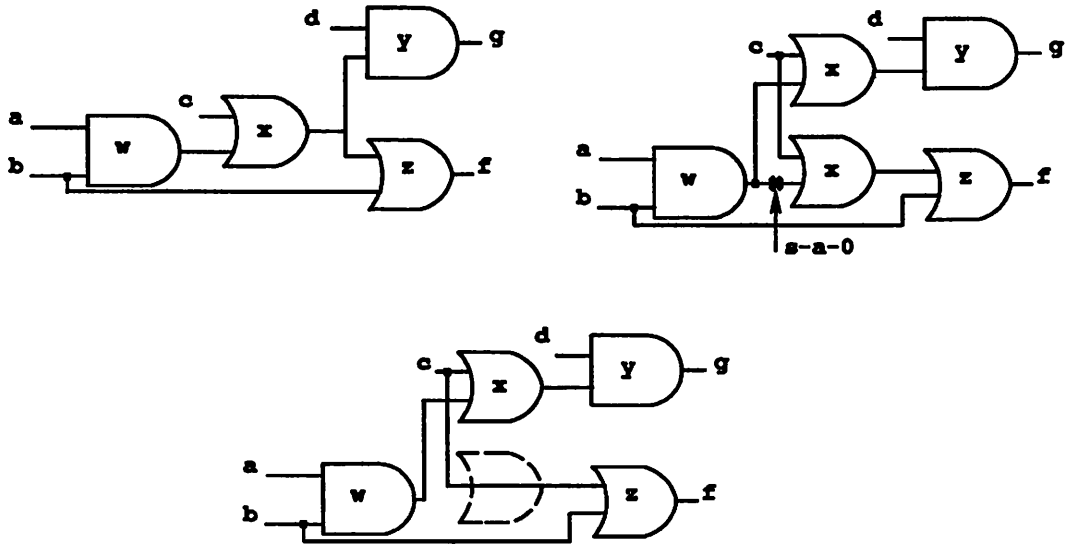

Figure 5: Heuristic algorithm on example circuit

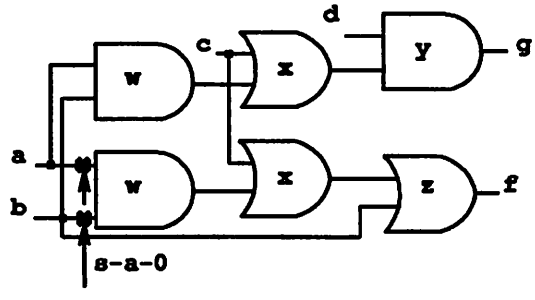

Figure 6: Leaf-dag of example circuit

d. Pick $\tau = n * d$. If the delay of some path Q in $\mathcal{D} - \mathcal{R}$ exceeds τ, at least one connection e in Q has delay exceeding d. But e is in some path $P_i \in \mathcal{P}$ (since e appears in some path in $\mathcal{P}$), and the delay of P_i would exceed d contradicting our assumption. Hence no path in C exceeds τ. Since the robust delay is no larger than τ, the result follows. ■

Theorem 5.1 relates the distribution of robust testable paths in a circuit and verifiability of the circuit's performance. If a circuit has a RPDF set that covers all the connections along non-RD paths, the robust delay of the circuit is finite. Otherwise, it may be infinite, since delay faults on some paths can cause the circuit's delay to be any value without being detected on RPDF paths. Hence, the robust delay is a measure of the verifiability of a circuit's performance under its available testability.

Although the robust delay may be too large to be useful, as will be explained below, upper bounds on robust delay can be easily computed. How close the upper bounds are to the best circuit performance will depend on the testing strategy used, *e.g.* fixed delay or variable testing using upper and lower bounds.

5.1 Computing robust delay

To obtain an upper bound on robust delay, one can select any RPDF path-set $\mathcal{P}$ and test the delays in $\mathcal{P}$. Assume the (fixed delay) testing scheme latches the circuit outputs at some time d. Let $e \in P$ refer to a connection in path P and $\mathcal{R}$ represent a RD set. An upper bound on robust delay is:

$$\max\left(\left(\sum_{e \in Q} \delta(e), \forall\, Q \notin (\mathcal{P} \cup \mathcal{R})\right), d\right)$$

subject to:

$$\sum_{e \in P} \delta(e) \leq d,\ \forall P \in \mathcal{P}$$

This is because a path can not have delay longer than the solution of the above linear program without violating a constraining inequality, which implies a delay-fault on a tested path. Note that the RD delay-faults are ignored.

In the **bounded delay model** the delay of each connection c is specified as $[d^c_{min}, d^c_{max}]$. The delay of a path P in this model is specified as $[d^P_{min}, d^P_{max}]$; these bounds could be determined by summing up the minimum and maximum delays of connections in the path respectively. Another issue of delay-fault testing is whether a variable or fixed clock speed is used during application of the tests. A fixed clock period means that each path delay is verified to be less than the (fixed) clock period of the circuit. However, a variable clock allows greater flexibility to the testing process. In particular, paths delays can be verified against their lower and upper bounds. We consider both situations; in particular, the added flexibility of a variable clock rate helps in reducing the test set size after the identification of a dependent set.

Definition 5.2 *Under the bounded model a* **short-path delay-fault** *occurs along P if and only if the actual delay of $P < d^P_{min}$. A* **long-path delay-fault** *occurs if and only if the actual delay of $P > d^P_{max}$.*

Although a long-path or short-path delay-fault causes a delay to fall outside specified bounds we are only interested in detecting those delay-faults that slow down the circuit. Long-path and

short-path delay-faults will be exploited only in reducing the size of the test set and implying bounds on the delays of other paths, thus increasing the delay-fault coverage. [3]

The upper bound on robust delay may be improved by using bounded delays for the paths in $\mathcal{P}$ and testing to check that the upper and lower bounds are met (variable delay testing).

Theorem 5.2 *An upper bound on the robust delay τ_R of a circuit is:*

$$\tau_R \leq \max((\sum_{e \in Q} \delta(e),\ \forall\, Q \notin (\mathcal{P} \cup \mathcal{R})), d)$$

where

$$d = \max(d^P_{\max})\ \forall P \in \mathcal{P}$$

and

$$d^P_{\min} \leq \sum_{e \in P} \delta(e) \leq d^P_{\max},\ \forall P \in \mathcal{P}$$

where $\mathcal{P}$ = set of all RPDF paths and $\mathcal{R}$ = a maximal RD set[4].

Proof: Let τ be the optimal value of the above linear program. Then the only paths longer than τ form a RD set. By definition, delay-faults in $\mathcal{R}$ can only be responsible for the delay if the delay on some non-RD paths is at least as much. Since the latter are accounted for in the linear program, the delay of the circuit is less than τ. Further, τ is the minimum delay that can be guaranteed by testing the RPDF paths given the selected RD set $\mathcal{R}$; so τ_R is no more than τ. ■

It can be seen that the closer $d^P_{\min}$ and $d^P_{\max}$ are, the tighter the bound will be, because the feasible space is more restricted. Another way to improve the bound is to use a larger RPDF set, so that more constraining inequalities are introduced to further reduce the feasible space, hence, tightening the upper bound. This will be illustrated in the examples in the next section.

An interesting case is when the circuit is 100% robust delay-fault testable and the entire RPDF set is tested. Then, the constraining inequalities are lower and upper bounds on all the paths. Obviously, the linear program yields the maximum path delay. Hence, in this case robust delay reduces to the usual notion of delay.

It is also interesting to note that the notion of the delay of a circuit is now tied to its testability. If 100% delay-fault coverage is achievable for a specified delay, it is "safe" to use, since it can be guaranteed. Thus, regardless of the true delay computed by accurate timing analysis, the delay value that can be verified is the "usable" delay; verifiability of this delay depends on the amount of available testability. With 100% robust delay-fault testability, the true delay can be verified and thus is usable. For circuits with less than 100% delay-fault testability, the verifiable delay is the robust delay, possibly greater than the true delay. Thus, the notion of robust delay captures the relationship between the amount of testability of a circuit and its "usable" delay. This usable delay varies with the amount of the circuit's delay-fault testability.

---

[3] A robust test for short-path delay-faults must be generated. It is known that a vector pair $< v_1, v_2 >$ tests for a long-path delay-fault for a rising or falling transition if and only if $< v_2, v_1 >$ tests for a short-path delay-fault for the opposite transition along the same path [17].

[4] In general, we need to replace each $\delta(a_i, a_j)$ in the inequalities by $\delta^r(a_i, a_j)$ or $\delta^f(a_i, a_j)$, as appropriate to each rising or falling path, if $\delta^r \neq \delta^f$.

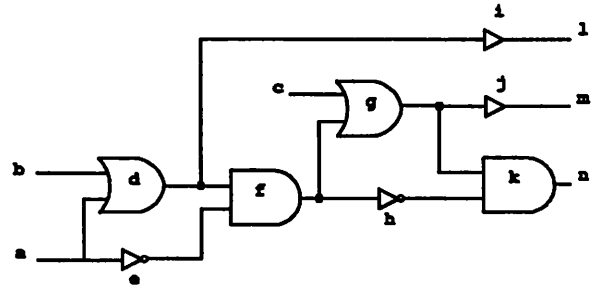

Figure 7: Computing robust delay

5.2 Examples

Here we give examples to demonstrate the computation of upper bounds for robust delays τ_R and illustrate their magnitudes relative to the delay d used to test the RPDF delay-faults.
Example: The circuit of Figure 7 is 100% single stuck-fault testable and but only 16 out of the 26 (rising and falling) path delay-faults are RPDF (about 62% testability). Although some paths are non-RPDF and non-RD, fortunately the set of testable paths form a path-cover, and the circuit has finite robust delay. Assume each connection has the bounded delay [0.9, 1.0]. The delay of each path is bounded by the sum of the lower and upper bounds of the connections in the path. Assume the RPDF faults pass the tests for short-path and long-path delay-faults.

The RPDF paths for both rising and falling transitions, denoted by $\mathcal{P}$ are: $\{a,e,f,h,k,n\}$, $\{a,e,f,g,j,m\}$, $\{b,d,f,h,k,n\}$, $\{b,d,f,g,j,m\}$, $\{c,g,k,n\}$, $\{c,g,j,m\}$, $\{b,d,i,l\}$, $\{a,d,i,l\}$. The robust delay is obtained as:

$$max\ (\sum_{x \in Q} \delta(x)),\ \forall\ Q \notin \mathcal{P}$$

such that

$$\begin{array}{lll}
3.6 & \leq \delta(a,e)+\delta(e,f)+\delta(f,h)+\delta(h,k)+\delta(k,n) & \leq 4 \\
3.6 & \leq \delta(a,e)+\delta(e,f)+\delta(f,g)+\delta(g,j)+\delta(j,m) & \leq 4 \\
3.6 & \leq \delta(b,d)+\delta(d,f)+\delta(f,h)+\delta(h,k)+\delta(k,n) & \leq 4 \\
3.6 & \leq \delta(b,d)+\delta(d,f)+\delta(f,g)+\delta(g,j)+\delta(j,m) & \leq 4 \\
1.8 & \leq \delta(c,g)+\delta(g,k)+\delta(k,n) & \leq 2 \\
1.8 & \leq \delta(c,g)+\delta(g,j)+\delta(j,m) & \leq 2 \\
1.8 & \leq \delta(b,d)+\delta(d,i)+\delta(i,l) & \leq 2 \\
1.8 & \leq \delta(a,d)+\delta(d,i)+\delta(i,l) & \leq 2
\end{array}$$

The linear programming result is 4.40. Therefore, with only 62% testability, 100% fault coverage can be achieved with the robust delay $\tau_R = 4.40$, only 10% more than the computed

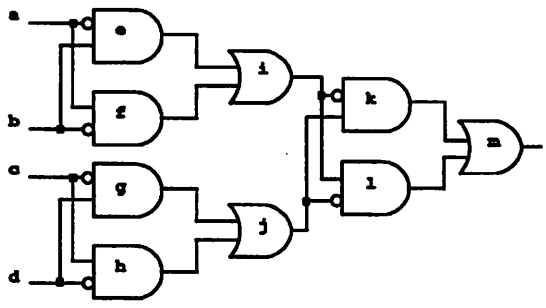

Figure 8: 4-bit parity checker

delay of 4.00. Note that this is exactly the robust delay of the circuit (not a bound) if the upper and lower bounds on each path were precise.

Under a floating delay model where only the upper bound is available, this scheme produces a higher robust delay:

$$max\ (\sum_{x \in Q} \delta(x)),\ \forall\ Q \notin \mathcal{P}$$

such that

$$\begin{aligned}
\delta(a,e) + \delta(e,f) + \delta(f,h) + \delta(h,k) + \delta(k,n) &\leq 4 \\
\delta(a,e) + \delta(e,f) + \delta(f,g) + \delta(g,j) + \delta(j,m) &\leq 4 \\
\delta(b,d) + \delta(d,f) + \delta(f,h) + \delta(h,k) + \delta(k,n) &\leq 4 \\
\delta(b,d) + \delta(d,f) + \delta(f,g) + \delta(g,j) + \delta(j,m) &\leq 4 \\
\delta(c,g) + \delta(g,k) + \delta(k,n) &\leq 2 \\
\delta(c,g) + \delta(g,j) + \delta(j,m) &\leq 2 \\
\delta(b,d) + \delta(d,i) + \delta(i,l) &\leq 2 \\
\delta(a,d) + \delta(d,i) + \delta(i,l) &\leq 2
\end{aligned}$$

The linear programming result is 8.00, twice as much as the computed delay. ■

Example: Figure 8 shows a more realistic circuit, a 4-bit parity checker in which all 32 delay-faults are RPDF. We assume each connection has delay [0.9, 1]; these are only used to derive bounds on path delays. If we test all delay-faults, we get the computed delay as the robust delay. By testing only a fraction of the delay-faults we can observe how the magnitude of robust delay varies with the percentage of the tested faults. In Figure 9 the first column in the table is the number of delay-faults (out of a total 32) that are tested; the second is the percentage tested; the third is the robust delay.

Test set size	% testability	Robust Delay
32	100	4.00
30	94	4.40
28	88	4.40
26	81	4.40
24	75	8.00
22	69	8.00
20	63	8.00
18	56	8.00
16	50	8.00
≤ 14	≤ 44	∞

Figure 9: Test set size vs. robust delay for parity checker

From the table, it can be seen that the computed delay of 4.00 can be verified by testing all RPDF faults; call this 100% testing. With 81% testing, 90% of the computed delay can be verified, i.e. 4.40 vs. 4.00; with as little as 50% testing, twice the computed delay can be verified. Below 44% testing, some delay faults may lengthen the circuit delay yet remain undetected in the testing, thus yielding infinite robust delay. Figure 10 plots the variation of robust delay versus the testing effort.

The test set of size 16 is linear in circuit size since each RPDF path includes some connection not covered by any other RPDF path. ■

The above examples illustrate that the amount of testing can be tailored to fit the required performance. Higher performance requires more testing. Fairly tight upper bounds on robust delay can be obtained with low testing effort. In particular, a test set linear in circuit size allows a remarkably high verifiable performance..

6 Delay-faults under bounded delays

In this section we illustrate the use of a bounded delay model in reducing the test set size for robust delay-fault testing.

Not all non-RD delay-faults need be tested under the bounded delay model. The fact that tested paths have delays within specified lower and upper bounds can be used to reduce the number of delay-faults tested. There are two aspects that deserve attention. First, the clock period that is applied while performing the delay-fault testing must be variable. Second, only bounded delays on complete paths are exploited in reducing the test set. This is because

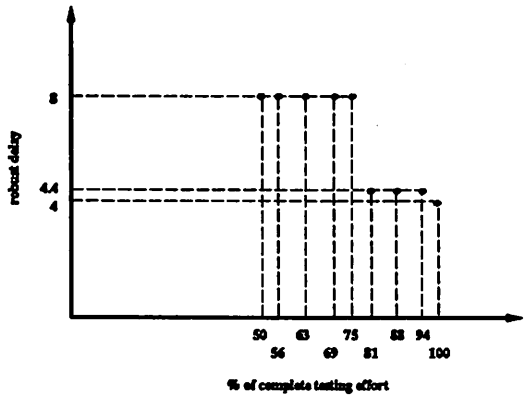

Figure 10: Robust delay vs. testing effort tradeoff

bounded delays on connections cannot be verified by testing; typically, bounds on connections are only used in determining bounds on path delays.

The goal is to perform two (one long-path and one short-path) tests on a few selected paths rather than a single long-path delay-fault test on a larger set of paths. This is possible if the occurrence of each of the untested delay-faults is prohibited by the absence of delay-faults on the tested paths. Under these conditions, this test strategy yields 100% delay-fault coverage.

Example: Consider the example shown in Figure 11. Each edge has bounded delay specified as $[1,2]$ except the edge f which has delay $[6,10]$. Let the specified clock period be 16. Assume that all but the bold path have been tested and each falls within its lower and upper bounds (taken as the sum of the lower and upper bounds, respectively, of each edge in the path). We argue that the bold path does not have to be delay-fault tested. This is done by proving that any assignment that slows down the chosen path (causing its delay to be greater than 16) causes some other tested path to also speed up or slow down outside its specified bounds. The proof is expressed as a linear program expressing the condition that the chosen path be the only one to exceed the clock period.

$$\begin{array}{lll} 9 & \leq \delta(a)+\delta(d)+\delta(f)+\delta(g) & \leq 16 \\ 9 & \leq \delta(b)b+\delta(d)+\delta(f)+\delta(g) & \leq 16 \\ 3 & \leq \delta(b)+\delta(c)+\delta(g) & \leq 6 \\ 8 & \leq \delta(e)+\delta(f)+\delta(g) & \leq 14 \\ & \delta(a)+\delta(c)+\delta(g) & > 16 \end{array}$$

This set of constraints is not feasible (under the additional constraints that the delay of each edge be non-negative); this demonstrates that the path $\{a,c,g\}$ cannot have the only delay-fault that slows down the circuit. Note that the bounds on the connections are not used, e.g.

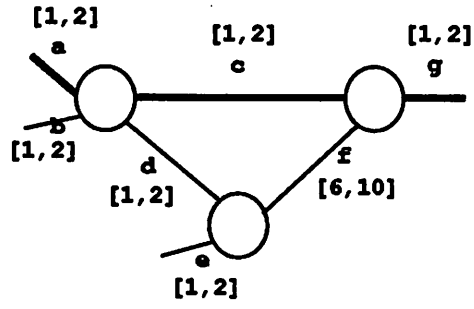

Figure 11: All paths need not be delay-fault tested under bounded delay model

$1 \leq \delta(a) \leq 2$ is not used since it is not verified. ■

Based on the above example, given a set of non-RD delay-faults, implications using the bounded delays may be used to reduce the test set size by formulating a series of linear programs. While an exact version of such an algorithm appears intractable, a heuristic algorithm that is intuitively promising is under development.

7 Computing delay-fault coverage

We take the position that the *fault coverage* should reflect the confidence that one can have after testing that the circuit is correct, *i.e.* in this case can be operated correctly at some speed τ. It should reflect the percentage of faults tested out of all faults that really need to be tested. Having seen three different dimensions of delay-fault testing namely (1) RD sets (2) bounded delay testing and (3) implied path delays, we show that each is essential in computing a realistic value for the delay-fault coverage. We propose the following formula for delay-fault coverage.

Proposition 7.1 *Let $\mathcal{F}$ denote the set of all path delay-faults in a circuit, and $\mathcal{R}$ denote a robust dependent (RD) set. Given a test set T, let $\mathcal{T}$ denote the delay-faults that are robust tested by T for bounded delays. For a specified circuit speed τ, let $I(\mathcal{T},\tau)$ denote the set of non-RD delay-faults whose absence (delay less than τ) is guaranteed by testing $\mathcal{T}$.* IFC(τ) *denotes the intrinsic (or maximum) delay-fault coverage achievable to guarantee correct circuit operation with delay at most τ, and* RFC(τ) *denotes the reported delay-fault coverage. Then:*

$$IFC(\tau) \geq RFC(\tau) = \frac{(|\mathcal{T}| + |I(\mathcal{T},\tau)|)}{(|\mathcal{F}| - |\mathcal{R}|)}.$$

It is apparent from the formula that not using one or more of the three aspects introduced in this paper will decrease the reported delay-fault coverage. For example, in the denominator

we find only those delay-faults that need to be tested, $\mathcal{F} - \mathcal{R}$; thus the RD set appears. In the numerator, the implied set $I(\mathcal{T}, \tau)$ appears accounting for the third effect. Bounded delay testing is implicit in the term $I(\mathcal{T}, \tau)$ since this kind of testing would allow more path delays to be implied. The effect of $\mathcal{R}$ is slightly similar to redundant faults in stuck-fault testing - since no test is necessary for a redundant fault, this should be subtracted from the set of all faults in computing coverage.

An open question is how far the reported delay-fault coverage is from the intrinsic delay-fault coverage. This depends on whether there exist any other effects among delay-faults not captured by the three phenomena described in this paper, *e.g.* are there other implied faults or are there other faults that need not be tested.

8 Results

We show results on MCNC benchmark circuits. Each circuit is first optimized using the standard script (called *script.rugged*) in the SIS system [15]. The lower bound on delay fault coverage, determined using the heuristic described in Section 4.4, is compared against the robust delay-fault testability in the optimized circuits. The first column is the circuit name; the second is the total number of delay-faults (rising and falling transitions); the third gives the size of the RD set found; the fourth lists the size of the RPDF set; the fifth gives the robust delay-fault testability. The column *FC-1* gives the fault coverage computed using only the RD set. Note the substantial increase in this compared to the testability. For example, 93% of the total delay-faults in *bw* are identified as an RD set resulting in an increase from 0.01 to 0.20. The column *FC-2* gives the fault coverage achieved by using a variable testing speed but only using upper bounds on the path delay (lower bounds assumed 0 for all paths). A unit delay is assumed on each connection in the circuit. In this case, any non-RD path whose delay is implied to be a finite number if the circuit passes the RPDF tests is considered tested. Except for the circuits with 100% testability, the circuits do not have finite robust delay since 100% fault coverage cannot be achieved for any finite delay. Despite this, all the circuits show very high fault coverage. For example, the coverage in *bw* is now raised from 0.21 to 0.66. Considering that the testability is 0.01, this is a significant increase. However, note that FC-2 ignores the speed of the circuit. To get the coverage reported, the circuit may have to be slowed down considerably.

Notice the large number of RD paths identified by the algorithm of Section 4.4 for every example, except those with 100% testability. We have also tried different heuristics from those described in Section 4.4. For example, the unfolding was performed from primary inputs towards primary outputs; in another case we tried unfolding in decreasing order of the number of paths through the gates in the circuit. While we have observed variations in the size of the maximal RD set identified by each heuristic, the technique of Section 4.4 yields the best quality results and also appears to have less memory requirements in the unfolding.

Compare the results for the un-optimized and optimized versions of a 32-bit carry-skip (or carry-bypass) adder, *cbp.32.4*. The circuit is composed of eight 4-bit carry-skip adders connected in cascade. Notice that Boolean optimization techniques cause a dramatic increase by a factor of 87 in the number of paths (delay-faults). Yet the fault coverage in the resulting circuit is almost the same as the initial circuit. The effect of the usefulness of RD sets is demonstrated by the size of the non-RDR set which is 0.08% of the total number of delay-

Name	# PDF	# RD	# RPDF	Testability	FC-1	FC-2
rd53	310	102	157	0.51	0.75	0.83
con1	444	0	44	1.00	1.00	1.00
z4ml	526	246	187	0.36	0.67	0.90
misex2	758	317	373	0.49	0.85	0.99
misex1	1144	744	254	0.22	0.64	0.96
9sym	1184	216	674	0.57	0.70	0.89
vg2	1576	0	1576	1.00	1.00	1.00
apex7	1960	98	1775	0.91	0.95	0.97
sao2	2706	1548	626	0.23	0.54	0.83
clip	3710	2865	612	0.16	0.72	0.89
rd73	3816	2837	624	0.16	0.64	0.91
e64	4290	0	4290	1.00	1.00	1.00
5xp1	5246	4346	447	0.09	0.50	0.79
rd84	6290	4573	911	0.15	0.53	0.83
duke2	8320	4466	3050	0.37	0.79	0.94
apex6	9898	3815	4743	0.48	0.78	0.96
alu4	25626	18637	3509	0.14	0.50	0.87
bw	49864	46873	627	0.01	0.21	0.66
rot	58734	39552	11917	0.20	0.62	0.97
des	240232	118827	94719	0.39	0.78	0.97
cbp.32.4†	256128	221176	11154	0.04	0.32	0.85
cbp.32.4	22278702	22260784	7474	0.0003	0.42	0.83

†: Initial un-optimized circuit
PDF = All path delay-faults in optimized circuit
RD = Chosen RD set
Non-RD = Path delay-faults which are not in chosen RD set
RPDF = Robust testable delay-faults
FC-1 = (# RPDF) / (# Non-RD)
FC-2 = (# Tested faults) / (# Non-RD)
Tested faults = Non-RD faults whose absence is guaranteed if circuit passes RPDF tests at finite speed

Figure 12: Lower bound on delay-fault coverage

faults in the circuit. [5]

The number of false paths for any delay assignment is a lower bound on the size of the RD set (Theorem ??). Some of the circuits shown above have few false paths under a unit gate delay model, yet have a large maximal RD set. While the maximum RD set size provides some indication of the number of false paths (it is an exact indicator assuming all possible path length orderings (Theorem 4.2)) that may exist for a specific delay assignment, an interesting problem for the future will be to better understand this relationship.

9 Conclusions

We have shown that the delay-fault coverage achievable by a test set is relative to the speed at which operation is desired. This observation helps address the issue of how much delay-fault testing is required to guarantee that a circuit operates correctly at its specified speed in three ways:

1. For a specified circuit speed, there exist path delay-faults which never need to be tested, since they cannot affect the circuit delay without some other delay-faults (which are included in the set to be tested) also impacting the circuit delay. Necessary and sufficient conditions for the existence of these delay-faults, called RD sets, are proved. We show that delay-faults on any set of long false paths that occurs, given any delay assignment to the connections of the circuit, forms a RD set. This yields an effective lower bound on the size of the RD set. This theory increases the fault coverage reported for robust delay-fault testing. Previous methods use testability (*i.e.* the percentage of delay-faults that are RPDF) in reporting coverage. However, the size of the test set is not affected since every RPDF still must be tested.

2. The next aspect describes the possible tradeoff that exists between the circuit speed and the delay-fault test set size required to guarantee that a manufactured circuit operates at that speed. This is termed the verifiable performance of the circuit and denotes the usable delay relative to the desired testing effort that can be guaranteed. The notion of robust delay of a circuit is used to represent the minimum useful delay of a circuit under a given bounded delay model. The quantitative tradeoff between the robust delay and the testing effort has been formulated using a linear program, while also accounting for the presence of RD delay-faults. In particular, linear (in circuit) size test sets are shown to provide 100% delay-fault coverage with a very small decrease in the verifiable performance.

3. The final aspect reduces the delay-fault test set size without decreasing the delay-fault coverage or changing the desired circuit speed. Under a bounded delay model, where the delay of a tested path is guaranteed to lie between some upper and lower bound, a simple linear program is formulated to identify delay-faults that do not require testing.

Each of these concepts were demonstrated on examples; however comprehensive experiments remain for the future. We are presently developing effective heuristics to obtain good

[5] Experimental results on even larger circuits will be reported in the final version of the paper; our results are presently restricted by the efficiency of the current ATPG program used.

solutions with low computational costs. Preliminary experiments were reported and show encouraging results.

A primary application of this theory will be the synthesis of circuits with 100% robust delay-fault coverage. The results of this paper prove that 100% testability is not necessary for 100% delay-fault coverage. Based on the techniques introduced to reduce the complexity of delay-fault testing, synthesis for circuit with 100% fault coverage becomes a two-step process. (1) A circuit may not have 100% fault coverage for a specified speed, yet a little reduction in speed may guarantee 100% fault coverage; this occurs without any modification to the circuit. This first step of the synthesis process brings the tradeoff between performance and coverage into consideration. (2) If the fault coverage or circuit performance is not satisfactory after the first step, we synthesize circuits which have delay-faults which are either RPDF or belong to a single RD set; this is a weaker condition that 100% robust delay-fault testability. An optimization step to reduce the size of the test set by exploiting accurate delay information on paths, derived from accurate gate modeling, is used. We are in the process of exploring techniques for synthesis of circuits for 100% fault coverage based on this approach.

References

[1] D. Bhattacharya, P. Agrawal, and V. Agrawal. Delay fault test generation for scan / hold circuits using Boolean expressions. In *Proceedings of the Design Automation Conference*, pages 159–164, June 1992.

[2] H-C. Chen and D. Du. Path sensitization in critical path problem. In *Proceedings of the International Conference on Computer-Aided Design*, pages 208–211, November 1991.

[3] S. Devadas and K. Keutzer. Necessary and sufficient conditions for robust delay-fault testability of combinational logic circuits. In *The Proceedings of the 6th MIT Conference on Advanced Research in VLSI*, pages 221–238, April 1990.

[4] S. Devadas and K. Keutzer. Synthesis and optimization procedures for robustly delay-fault testable combinational logic circuits. In *Proceedings of the Design Automation Conference*, pages 221–227, June 1990.

[5] S. Devadas, K. Keutzer, and S. Malik. Delay computation in combinational logic circuits: Theory and algorithms. In *Proceedings of the International Conference on Computer-Aided Design*, pages 176–179, November 1991.

[6] N. Jha, I. Pomerantz, S. Reddy, and R. Miller. Synthesis of multi-level combinational circuits for complete robust path delay fault testability. In *Proceedings of the International Fault Tolerant Computing Symposium*, pages 280–287, June 1992.

[7] S. Kundu and S. Reddy. On the design of robust testable CMOS combinational logic circuits. In *Proceedings of the International Fault Tolerant Computing Symposium*, pages 220–225, June 1988.

[8] C. Lin and S. Reddy. On delay fault testing in logic circuits. *IEEE Transactions on Computer-Aided Design*, C-6(5):694–703, September 1987.

[9] Y. Malaiya and R. Narayanswamy. Testing for timing faults in synchronous sequential integrated circuits. In *Proceedings of the International Test Conference*, pages 560–571, October 1983.

[10] P. McGeer and R. Brayton. Provably correct critical paths. In *The Proceedings of the Decennial Caltech VLSI Conference*, 1989.

[11] A. Pramanick and S. Reddy. On the design of path delay fault testable combinational circuits. In *Proceedings of the International Fault Tolerant Computing Symposium*, pages 374–381, June 1990.

[12] K. Roy, J. Abraham, K. De, and S. Lusky. Synthesis of delay fault testable combinational logic. In *Proceedings of the International Conference on Computer-Aided Design*, pages 418–421, November 1989.

[13] A. Saldanha, R. Brayton, and A. Sangiovanni-Vincentelli. Equivalence of robust delay-fault and single stuck-fault test generation. In *Proceedings of the Design Automation Conference*, pages 173–176, June 1992.

[14] M. Schulz, K. Fuchs, and F. Fink. Advanced automatic test pattern generation techniques for path delay faults. In *Proceedings of the International Fault Tolerant Computing Symposium*, pages 44–51, June 1989.

[15] E. Sentovich, K. Singh, C. Moon, H. Savoj, R. Brayton, and A. Sangiovanni-Vincentelli. Sequential circuit design using synthesis and optimization. In *Proceedings of the International Conference on Computer Design*, pages 328–333, October 1992.

[16] G. Smith. Model for delay faults based upon paths. In *Proceedings of the International Test Conference*, pages 342–349, August 1985.

[17] K. Wagner. The error latency of delay faults in combinational and sequential circuits. In *Proceedings of the International Test Conference*, pages 334–341, November 1985.