A Critical Path Selection Method for Delay Testing

Saravanan Padmanaban[*]
Intel Corporation
Hillsboro, OR 97124
saravanan.padmanaban@intel.com

Spyros Tragoudas
ECE Department
Southern Illinois University
Carbondale, IL 62901
spyros@engr.siu.edu

Abstract

An approach for selecting critical paths along which testable path delay faults can exist is presented. The proposed method is particularly helpful on path intensive circuits. Critical paths are selected implicitly with the aid of a combination of decision diagrams. An implicit method to eliminate untestable faults along the selected paths is also presented. The effectiveness of the approach is demonstrated on path intensive ISCAS'85, ISCAS'89 and ITC'99 benchmarks.

1 Introduction

Delay fault testing is a test methodology to verify the temporal behavior of a manufactured circuit. The path delay fault (PDF) model is used because it is more suitable for testing defects in nanometer technology. Its main advantage over the transition fault and gate delay fault models is its ability to describe distributed delay defects. Its disadvantage is that even small circuits may contain an exponential number of faults in the worst case. It has been shown in [3] that a large fraction of PDFs in most circuits are functionally untestable. They cannot affect the timing behavior of the circuit and need not be tested. It was shown in [21] that multiple PDFs must also be tested in order to ensure the correct timing behavior of the circuit. A subset of all possible multiple PDFs are defined and termed as primitive PDFs. It is necessary and sufficient to test each primitive PDF to guarantee the temporal correctness of a circuit. However, the set of primitive PDFs is enormous, and hence for practical purposes delay testing is done using functionally sensitizable PDFs as defined in [3].

The number of functionally testable PDFs is extremely large to target. In practice, testing focuses on a subset of PDFs whose typical delay values are very large. A PDF is *c*ritical if it can affect the temporal behavior of a circuit and thus only critical PDFs must be targeted. It is common to test the longest (functionally) testable PDFs in a circuit but this definition is accurate only under a fixed delay typical value. It has been observed in [2] that process variation (inter die and intra die) induces a large path delay variation in submicron technology. Using the fixed delay model the set of testable longest paths from one die is not correlated to the set from another die. They may be completely disjoint sets. The bounded gate delay can be used to tackle such issues in nanometer technology. It can also take into consideration noise induced delays (like power supply noise, simultaneous switching noise, etc.,).

The problem of identifying the k longest paths in a directed acyclic graph with fixed weights on the nodes (or edges) has been shown to be intractable [17]. This clearly shows the intractability of problem of identifying the k longest paths in a directed acyclic graph with a range of weights on each of the node (or edge). This is exactly the problem required to be solved for identifying the set of critical paths in a circuit where the delay of the gates and interconnects vary due to process variations. We use bounded delays to model the range of delays of the gates and interconnects.

The problem of identifying the critical paths in a circuit not only deals with identifying the set of longest paths in a circuit, but deals with the problem of identifying the set of functionally sensitizable longest paths. This added constraint makes the problem more difficult and challenging to be handled in a reasonable time for path intensive circuits. The reason for this being that the long paths in a circuit can be functionally unsensitizable (timing independent false paths).

Techniques like [15] also study the problem of identifying the set of longest paths for testing purposes using fixed delay models. These paths do not reflect the actual set of critical paths for circuits fabricated using the submicron technology. Under the bounded delay model, critical PDFs are those testable PDFs whose delay may exceed a given delay threshold $\mathcal{D}_{th}$. The threshold delay $\mathcal{D}_{th}$ is a delay bound that can be defined as a fraction (say 90%) of the clock period (identified using static timing analysis). More precisely, the delay threshold is used to eliminate the set of all testable PDFs whose delay never exceeds the threshold. The remaining testable PDFs are then defined as the set of critical PDFs.

[*]The work was done when the author was with the ECE Department at the Southern Illinois University, Carbondale, IL 62901.

0-7803-8580-2/04 $20.00 Copyright 2004 IEEE

There are numerous problems in identifying critical PDFs. First, finding the set of potentially testable PDFs is an intractable problem. Graph-theoretic techniques as in [4] propose algorithms with exponential complexity and can only handle very small circuits. Secondly, the problem of identifying the critical PDFs from the set of potentially testable PDFs is extremely difficult because it may contain a large number of PDFs. [15] proposes a structural or PODEM like ATPG based method which is non scalable (being enumerative in nature) to handle many PDFs and is hence not efficient for this problem.

This paper proposes a decision diagram based time efficient approach to implicitly identify the set of critical PDFs for a given circuit. It is a three phase procedure. In the first phase untestable faults are identi- fied using static implications and are eliminated from further consideration resulting in the set of potentially testable faults. All the PDFs in the set of potentially testable faults are not guaranteed to be testable. The second phase is a non-enumerative function-based method to select a set that contains all potentially testable critical paths from within the set of potentially testable faults. However, a small percentage of the returned PDFs may be non-critical PDFs due to the pessimistic nature of the approach. The third phase uses a boolean function based ATPG framework that is capable of effectively classifying each examined PDF as either testable or untestable, and return the exact set of critical PDFs. During the ATPG process any non-critical PDF added to the set of potentially testable critical PDFs by the phase two is eliminated.

Time limits during testing enforce an upper bound on the number of PDFs that can be tested by an ATE. Given this bound the presented methodology can be used to identify $\mathcal{D}_{th}$ such that the number of critical PDFs does not exceed this upper bound. This process can be guided by a binary search on the values of $\mathcal{D}_{th}$. The smaller the threshold $\mathcal{D}_{th}$ the more confidence we have in the ATPG process. This confidence is guaranteed because the presented approach finds all critical PDFs with respect to $\mathcal{D}_{th}$. No such confidence can be provided by the existing techniques for critical path selection.

Section 2 introduces a non-enumerative approach to identify the set of untestable faults and to eliminate them from consideration. Section 3 outlines a nonenumerative approach for selecting potentially testable critical paths. Section 4 describes a boolean function based ATPG to classify a potentially testable critical path as either testable or not. Section 5 presents experimental results on path intensive benchmarks to demonstrate the effectiveness of the approach. The experiments show that graph theoretic methods fail in identifying the set of potentially critical PDFs. Section 6 concludes.

2 Identifying Untestable Faults using Static Implications

In this section, we describe a method to implicitly identify the set of potentially testable PDFs for a given circuit. This is done by using the Zero-Suppressed Binary Decision Diagram (ZBDD). The ZBDD is a canonical data structure which is suited towards efficiently storing sets of product terms rather than the sum of product terms as in the case of the binary decision diagrams (BDD). In a ZBDD, the absence of a variable v is interpreted as $v = 0$ unlike the BDDs where the variable v is interpreted as a don't-care. This property makes the ZBDD to effectively represent sets of PDFs as sets of minterms [16]. Figure 1(b) shows a ZBDD representing all the PDFs in the circuit shown in Figure 1(a)[1].

It is shown how ZBDDs are used to non-enumeratively identify a lower bound on the untestable faults in two phases. The first phase implicitly enumerates all the possible PDFs in the circuit as a single ZBDD. The second phase identifies the untestable PDFs using static implications and implicitly eliminates the untestable PDFs from the ZBDD representing the PDFs in the circuit. The resulting ZBDD represents the set of potentially testable PDFs.

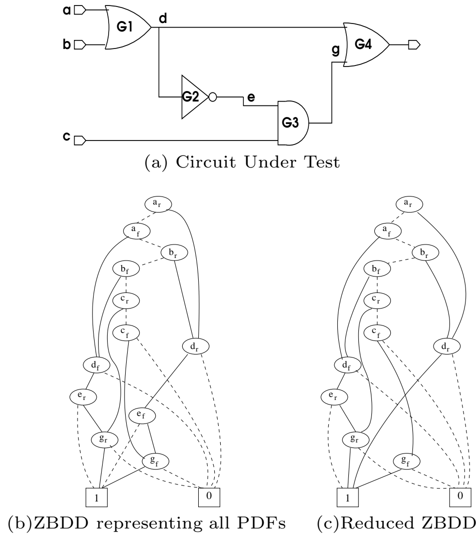
We list below static implication-based conditions that we use to quickly identify a subset of the robustly untestable faults (The conditions are restated from [6], [10], [19]). Each condition was found to have an impact in improving the lower bound. (Similar conditions have been implemented for other types of path sensitization.)

Condition 1 *Let $f_1, f_2, \cdots, f_m$ be the fan-ins of gate g. If line $l = 0$ (resp. 1) implies f_i be a controlling value for gate g, then all path delay faults through l with a falling transition on l and through $f_x (x \neq i)$ are robustly untestable.*

Condition 2 *Let $f_1, f_2, \cdots, f_m$ be the fan-ins of the gate f. Let $g_1, g_2, \cdots, g_n$ be the fan-ins of the gate g. If line $l = 0$ implies f_i to be the controlling value of f and $l = 1$ implies g_j to be the controlling value of g, then all physical paths through l, $f_x (x \neq i)$ and $g_y (y \neq j)$ are robustly untestable.*

It is demonstrated below how to implicitly implement the first condition using fundamental ZBDD operators introduced in [14]. Let the circuit shown in Figure 1(a) be the circuit under test. The circuit contains 10 PDFs. Each line is assigned an variable with a subscript r (resp. f) representing a rising transition on the line (resp. falling transition). All the PDFs in the circuit can be implicitly represented by single ZBDD (ξ), as shown in Figure 1(b). Each path in the ZBDD (from the root node to the terminal 1 node) corresponds to a PDF in the circuit. This ZBDD can be derived by a

[1] In the ZBDD, the solid lines corresponds to the 1-edges and the dotted lines correspond to the 0-edges.



(a) Circuit Under Test

(b)ZBDD representing all PDFs (c)Reduced ZBDD

Figure 1: Illustration of Untestable Fault Elimination

single topological traversal on the circuit. Let us now illustrate Condition 1 by means of an example and its implementation using ZBDDs. Assume that d is assigned logic 1. It can be observed that $d = 1$ is a controlling value of gate $G4$. It implies a logic 0 at gate $G3$, which is a non-controlling value for gate $G4$. According to Condition 1, all PDFs with a rising transition on line d and with a signal propagation through line g are robustly untestable.

We use two ZBDD variables l_r and l_f for each circuit line l in order to represent the signal transition (rising and falling, respectively) on each line l in the circuit. In our example, PDFs containing variables d_r and g_f are robustly untestable (by Condition 1) and can be eliminated from the ZBDD ξ. These PDFs are $\{\uparrow$ b.d.e.g$\}$ and $\{\uparrow$ a.d.e.g$\}$. Let $(S|_{v=1})$ denote the Subset operation, that identifies the subset of a set S whose elements contain variable v. The untestable PDFs are identified from ξ using this Subset operation. The process of identifying the untestable PDF is simply

$$(\xi|_{d_r=1})|_{g_f=1}.$$

For the example described in Figure 1, Fig. 1(c) shows the representation of the set of potential robustly testable PDFs. Similar operations have also been implemented to eliminate untestable faults through a set of lines or a sub-path rather than a pair of lines. In this case, the Subset operation has been invoked recursively.

It is also noted that for different types of path sensitization modified implication–based conditions have been implemented. For example, consider an AND gate g and let $f_1, f_2, \cdots, f_m$ be its fan-ins. If the assignment $l = 1$ on some line l implies the non-controlling value on f_{i-1} and a controlling on f_i, then all path delay faults with a rising transition on both lines d and f_{i-1} are functionally untestable. Such functionally untestable faults are then eliminated from ZBDD ξ a set difference operation to obtain a set of potentially testable PDFs ζ stored as a ZBDD. We have:

$$\zeta = \xi \ \setminus \ ((\xi|_{d_r=1})|_{g_f=1}).$$

The main advantage of the proposed method is its implicit nature in identifying the set of potentially testable faults. Let a circuit C contain P PDFs. Let n pairs of lines be identified by static implications such that all PDFs through the n pairs of lines are untestable. Even though P is exponential (in the worst case) to the number of nodes in the circuit, the proposed ZBDD based approach requires only $3 * n$ ZBDD operations (two operations to implement the subset and one operation to implement the set difference) in order to identify the set of potentially testable faults.

3 Identification of Testable Path Delay Faults

In this section we introduce an iterative approach to identify testable PDFs (which also includes testable PDFs that are non-critical). We start with the set of potentially testable PDFs identified using static implications described in Section 2. The procedure described here can also be used as a stand-alone ATPG or can be used together with the method described in Section 2 as a more efficient ATPG framework.

Given a PDF, methods proposed in [1] and [12] can be used for deriving boolean functions that describe all possible tests for the target fault. We use a very similar approach for deriving the boolean functions. However, we propose a method here to process segments rather than the entire paths.

Each sub-fault associated with a segment may be associated with any number of PDFs. If a sub-fault associated with a segment is untestable, then all PDFs through the segment are untestable. The two necessary and sufficient conditions to be checked to guarantee a PDF as untestable are:

- Nonexistence of a test for a segment l_i
- Existence of a test for the segment l_i and its successor l_{i+1}, and nonexistence of a common test for l_i and l_{i+1}.

This is used as the basis of the algorithm used for classifying the PDFs as either testable or untestable. The main feature behind using such an approach is from the observation from [3] that most of the PDFs in the practical circuits are robustly untestable. If a circuit contains a large percentage of untestable faults it tends to have a large number of untestable segments. This is the key idea of the proposed approach. Structural

ATPG techniques like [5] also process segments rather than paths to improve the computational efficiency. We note that structural techniques can store the implications derived for a given segment and the signal transitions on the on-input and off-input lines.

Algorithm 1 Identification of Testable Path Delay Faults

Require: $\mathcal{C}$, $\mathcal{S}$ensitization Criteria

1: Identify the Set of Potentially Testable PDFs (ζ)
2: **for all** $i \in$ Primary Inputs **do**
3: ζ_i = Subset of ζ originating at i
4: **while** $\zeta_i \neq \phi$ **do**
5: Pick PDF $\mathcal{P}$ by DFS
6: Split $\mathcal{P}$ into segments
7: **for all** segments p such that $p \in \mathcal{P}$ **do**
8: **if** $\mathcal{R}_r$ present in look-up array **then**
9: Extract $\mathcal{R}_r$ from look-up array
10: **else**
11: Generate $\mathcal{R}_r$
12: **end if**
13: **if** $\mathcal{R}_r$ is Satisfiable **then**
14: Add $\mathcal{R}_r$ to look-up array
15: $\mathcal{T}_R = \mathcal{T}_R \wedge \mathcal{R}_r$
16: **if** $\mathcal{T}_R$ is Not Satisfiable **then**
17: Eliminate PDFs in ζ_i through the segment from i to p
18: $\zeta_i = \zeta_i \setminus$ Tested PDFs
19: $\mathcal{T}_R = \phi$
20: **Break**
21: **end if**
22: **else**
23: Eliminate PDFs in ζ_i through p
24: $\zeta_i = \zeta_i \setminus$ Tested PDFs
25: $\mathcal{T}_R = \phi$
26: **Break** {Same procedure is followed for $\mathcal{R}_f$}
27: **end if**
28: **end for**
29: **if** $\mathcal{T}_R \neq \phi$ **then**
30: Tested PDFs = Tested PDFs $\cup \mathcal{P}$
31: Tested PDF Count = Tested PDF Count + 1
32: **end if**
33: **end while**
34: **end for**
35: Untestable PDFs = $\zeta \setminus$ Tested PDFs

Thus the implication procedure does not have to be repeated for different paths that pass through the same segment. However the justification phase which involves backtracking to assign values to all the primary inputs will have to be repeated for every path that passes through the segment. However the boolean function based method presented here does not have this drawback, which results in enormous saving of computational effort.

The basic methodology used to check whether a PDF is testable or untestable is described below:

Let ζ be the set of potentially testable PDFs identified from Section 2. The set of PDFs which is a subset of (ζ) and originating from an input p be denoted as ζ_p. The ZBDD representing ζ_p is traversed using depth first search algorithm. Traversing the ZBDD and not the circuit, improves the computation efficiency because we avoid traversing through unnecessary lines. A segment l_i is picked and its unique identification number is computed using the signal transitions t_{l_i} and t_p. The function $\mathcal{R}_r$ (respectively $\mathcal{R}_f$) is generated only if it is not present in the look-up array corresponding to the unique identification number of the segment l_i. Once $\mathcal{R}_r$ is obtained, the satisfiability of $\mathcal{R}_r$ can be checked in constant time (because the boolean functions are represented using BDDs). We distinguish between two cases:

- If $\mathcal{R}_r$ is not satisfiable, all PDFs in the ZBDD ζ_p through segment l_i are eliminated and the ZBDD is thus restructured. At this point, all PDFs that have been identified as testable by earlier processing is also eliminated from ζ_p. This is done so as to prevent processing the tested PDFs again after the ZBDD is restructured and reduced in size.
- If the functions corresponding to segments l_i and l_{i+1} are individually satisfiable but the conjunction of these two functions is not satisfiable then there exists no solution that can excite both segments together. Hence all PDFs in ζ_p through segments l_i and l_{i+1} are eliminated together with the PDFs identified as testable in earlier steps.

This iterative improvement process is performed till the ZBDD ζ_i becomes ϕ. The algorithm describing the procedure to identify testable and untestable faults is briefed by Algorithm 1.

4 Identification of Critical Path Delay Faults using the Bounded Delay Model

The procedure described here to identify the set of critical PDFs is a three-phase procedure whose first phase employs the method of Section 2 to identify potentially testable PDFs. This section concentrates on the second phase where a subset is selected implicitly. This set contains all potentially testable critical paths. A small percentage of them may be non critical. Phase three is the algorithm of Section 3 that identifies all critical PDFs. This algorithm is enhanced to explicitly eliminate any non-critical PDFs that Phase 2 has not eliminated. In particular, in addition to storing $\mathcal{F}_{l_i}$ for each segment l_i on the PDF, the minimum and maximum delay of the segment l_i is also stored in the look-up array.

Once an entire path delay fault is identified as testable, the PDF can then be classified either as critical or non-critical based on the minimum and maximum delay of the path. If the maximum delay of the path is less than $\mathcal{D}_{th}$, the PDF is classified as non-critical. However if the minimum delay of the path is greater than $\mathcal{D}_{th}$ or if $\mathcal{D}_{th}$ is a value in between the minimum and maximum delay of the path, the path is classified as critical. This check ensures the set of critical faults identified by the proposed approach is exact.

In the following, a block oriented procedure [7, 9] to identify the set of all potentially critical PDFs in a scalable manner is presented. The presented approach can be trivially modified to take care of wire delays by inserting buffers with a delay equivalent to the wire delay. It has been shown in [7] that block oriented techniques are fast, pessimistic and applicable to large scale designs. In contrast, it has been shown that path enumeration techniques as in [15] are exact, but slow, non scalable and cannot be used for path intensive circuits. This behavior in critical PDF selection for PDF testing has also been experimentally observed in [15].

Algorithm 2 Identification of Potentially Testable Critical Paths

```
Require: C, D_th
Require: d_{i_l}, d_{i_u} ∀i
Require: n {The max. length of segment}
 1: Compute d_i^{imin}, d_i^{imax}, d_i^{omin}, d_i^{omax} ∀i
 2: Compute ζ^C {Use Procedure from Section 2}
 3: for all i such that 1 ≤ i ≤ m do
 4:   if d_i^{imax}+d_i^{omax} < D_th then
 5:     ζ^C = ζ^C \ ζ^C_i
 6:   else if d_i^{imin}+d_i^{omin} > D_th then
 7:     L = L ∪ ζ^C_i
 8:     ζ^C = ζ^C \ ζ^C_i
 9:   else if d_i^{imin} + d_i^{omin} < D_th < d_i^{imax} + d_i^{omax}
      then
10:     for all P ∈ σ do
11:       for all k such that k ∈ P do
12:         if d_{x^1}^{imin} + Σ_{j=2}^{n} d_{x_l^j} + d_i^{omin} > D_th then
13:           L = L ∪ ζ^C_P
14:           ζ^C = ζ^C \ ζ^C_{iP}
15:         else if d_{x^1}^{imax} + Σ_{j=2}^{n} d_{x_u^j} + d_i^{omax} < D_th
            then
16:           ζ^C = ζ^C \ ζ^C_{iP}
17:         end if
18:       end for
19:     end for
20:   end if
21: end for
```

Let $\mathcal{C}$ be the circuit under investigation. Let the circuit $\mathcal{C}$ contain m gates. Let d_{i_l} and d_{i_u} be the lower and upper bound delay of a gate i. Let $\zeta^{\mathcal{C}}$ be the set of potentially testable faults in $\mathcal{C}$ identified by the procedure described in Section 2. Let d_i^{imin} be the delay of the shortest path from any primary input to gate i and d_i^{imax} be the delay of the longest path from any primary input to gate i. Similarly, let d_i^{omin} be the delay of the shortest path from gate i to any primary output and d_i^{omax} be the delay of the longest path from gate i to any primary output. The values of d_i^{imin}, d_i^{imax}, d_i^{omin} and d_i^{omax} can be calculated using topological traversals on the circuit.

Let $\mathcal{L}$ define the set constructed by the procedure described by Algorithm 2, which contains all potentially testable critical PDFs (but some non-critical PDFs as well). Set $\mathcal{L}$ is a subset of $\zeta^{\mathcal{C}}$, the set of potentially testable faults in $\mathcal{C}$. Manipulating the set of potentially testable faults (the output of Phase 1) as a ZBDD greatly helps to identify $\mathcal{L}$ from $\zeta^{\mathcal{C}}$ non-enumeratively.

For each gate i, if $d_i^{imax}+d_i^{omax} < \mathcal{D}_{th}$ then all the paths through i are classified as non-critical. Thus, they are eliminated from $\zeta^{\mathcal{C}}$ and do not participate in set $\mathcal{L}$. This operation is implemented on the ZBDD as $\zeta^{\mathcal{C}} = \zeta^{\mathcal{C}} \backslash \zeta_i{}^{\mathcal{C}}$.

However, if for gate i we have that $d_i^{imin}+d_i^{omin} > \mathcal{D}_{th}$ then the procedure classifies all paths through i as critical, and these paths are be added to $\mathcal{L}$. This operation is implemented on the ZBDD as $\mathcal{L} = \mathcal{L} \cup \zeta_i{}^{\mathcal{C}}$. By adding a path p to $\mathcal{L}$, we actually add the PDFs associated with path p to $\mathcal{L}$.

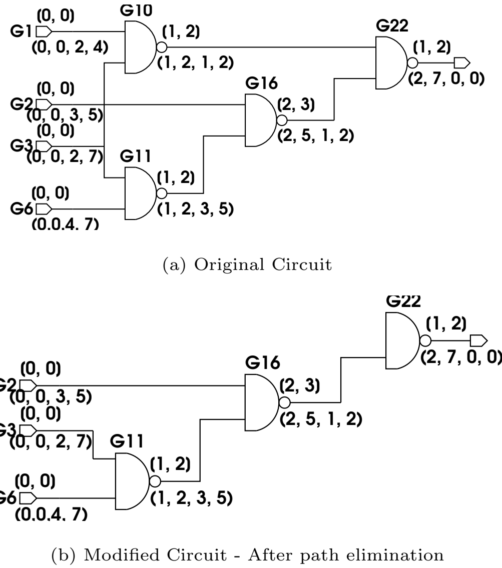
In the final case, when for a gate i we have that $d_i^{imin}+d_i^{omin} < \mathcal{D}_{th} < d_i^{imax}+d_i^{omax}$, some of the paths may be critical and some may be non-critical. If all the paths were added to $\mathcal{L}$ that would have made the approach extremely pessimistic. Instead, we use the heuristic approach described below to eliminate part of the uncertainty involved in this case.

Consider a gate i for which $d_i^{imin} + d_i^{omin} < \mathcal{D}_{th} < d_i^{imax} + d_i^{omax}$. All partial paths between i and the its predecessor nodes are obtained. Let the set of all partial paths be represented by σ. The length of partial path $\mathcal{P}$, $\mathcal{P} \in \sigma$, is defined as the number of gates on the partial path. Let the set of the gates on $\mathcal{P}$ be $\{x^1, x^2, \cdots, x^n\}$, labeled in a topological order. Thus, x^1 is the gate from which $\mathcal{P}$ originates and x^n is a fan-in of gate i. After identifying a partial $\mathcal{P}$ for gate i, all the paths through the partial path and through i need to be classified as either critical or non-critical. If $d_{x^1}^{imin} + \sum_{j=2}^{n} d_{x_l^j} + d_i^{omin} > \mathcal{D}_{th}$ then all paths through the partial path $\mathcal{P}$ and through i are classified as critical and are added to $\mathcal{L}$. This is implemented on the ZBDD as $\mathcal{L} = \mathcal{L} \cup \zeta^{\mathcal{C}}{}_{\mathcal{P}}$. If $d_{x^1}^{imax} + \sum_{j=2}^{n} d_{x_u^j} + d_i^{omax} < \mathcal{D}_{th}$ then all paths through the partial path $\mathcal{P}$ and through i are classified as non-critical and are eliminated from $\zeta^{\mathcal{C}}$. This operation can be stated as $\zeta^{\mathcal{C}} = \zeta^{\mathcal{C}} \backslash \zeta^{\mathcal{C}}{}_{\mathcal{P}}$.

It is noted here that the value of n needs to be kept as low as possible to maintain the non-enumerative behavior of the approach. As the value of n increases the approach will derive a solution close to the exact solution at the expense of execution time. The value of n can be ideally decided as small fraction of the length

of the topological longest path in $\mathcal{C}$.

The formal description of the procedure to identify the set of PDFs $\mathcal{L}$ is outlined by Algorithm 2. Note that this set is a superset of the set of potentially testable critical PDFs since a small number of PDFs may always have delay below $\mathcal{D}_{th}$. Nevertheless, all potentially critical PDFs are included in $\mathcal{L}$. The procedure described in Section 3 ensures that all non-critical PDFs from $\mathcal{L}$ are eliminated thereby calculating the exact set of critical PDFs. Algorithm 2 on ZBDDs results into a very time efficient implementation. A path tracing based implementation was implemented for comparison purpose and was shown to be inapplicable for path intensive circuits.



(a) Original Circuit

(b) Modified Circuit - After path elimination

Figure 2: Identification of $\mathcal{L}$

The operations in Algorithm 2 are demonstrated on the circuit $\mathcal{C}$ shown in Figure 2. In this figure, the delay of each gate is represented as a pair $[d_{i_l}, d_{i_u}]$ in Figure 2. The delay values of d_i^{imin}, d_i^{imax}, d_i^{omin} and d_i^{omax} for each gate are also represented by a quadruple $(d_i^{imin}, d_i^{imax}, d_i^{omin}, d_i^{omax})$. The topologically longest path of $\mathcal{C}$ is 7 units. We set $\mathcal{D}_{th}$ to be 6 units. In the following, we do not intend to simulate the algorithm on the algorithm, but rather to illustrate how it operates when (a) the conditions on lines 4 or line 6 (from Algorithm 2) are satisfied, and (b) when the condition on line 9 (from Algorithm 2) is satisfied.

In order to illustrate the operations in lines 4–8, assume that the first gate the algorithm operates on is $G10$ of the circuit in Figure 2(a). Then the condition in line 4 is satisfied since $d_i^{imax} + d_i^{omax} = 4$, where $i = G10$. This value is less than the preset value of $\mathcal{D}_{th} = 6$. Hence all PDFs through $G10$ are identified as non-critical and are removed from further consideration. The remaining PDFs will be stored in a ZBDD but in this example the sub-circuit of Figure 2(b) can represent all remaining PDFs and we chose, for the simplicity of the exposition, to store them with a sub-circuit.

Let us now demonstrate how the latter portion of the algorithm that uses partial paths of length $n = 1$ operates while at gate $G16$ of circuit Figure 2(b). Consider partial path $G2.G16$. The quadruple corresponding to gate $G16$ with respect to the partial path $G2.G16$ is $(2, 3, 1, 2)$. Hence $d_i^{imax} + d_i^{omax} = 5$, $i = G16$, with respect to the partial path $G2.G16$. This is less than the preset value of $\mathcal{D}_{th} = 6$. This implies that all paths through the partial path $G2.G16$ are also non-critical and are not considered any further. (The condition on line 15 is satisfied.)

Working that way, Algorithm 2 eliminates many non-critical paths. At the end, the PDF set $\mathcal{L}$ is formed, and the methodology described in Section 3 is invoked to identify the exact set of critical paths from $\mathcal{L}$.

5 Experimental Results

We implemented the proposed approach in C. We call the tool for identifying all untestable path delay faults OSIRIS. The performance of OSIRIS was experimented on the ISCAS'85 and ISCAS'89 benchmarks using a Sun Blade workstation. The results of the experimentation is presented in Table 1. We compare our results to the method of [20] due to its effectiveness in identifying all possible testable faults. Table 1 reports comparisons on the number of robustly untestable faults.

Column 2 shows the total number of PDFs in the corresponding circuit. Columns 3 shows the number of PDFs identified as robustly testable by [20] and Column 4 shows the number of aborted faults for that method. Furthermore, Column 5 shows the CPU execution time (interpolated) for the approach in [20].

Column 6 shows the number of PDFs identified as robustly untestable using the static implications discussed in Section 2. The difference between the total number of PDFs (Column 2) and the number of untestable faults in Column 6 is the cardinality of the set of potentially testable PDFs, and is reported in Column 7. This set of potentially testable PDFs indicates the number of faults targeted by the iterative procedure discussed in Section 3. Column 8 shows the total number of robustly testable PDFs identified by OSIRIS. Column 9 reports the number of PDFs aborted due to imposed bounds on the execution time (aborted faults). The total execution time for each circuit is reported in Column 10.

Columns 5 and 10 clearly indicate that the execution time required by OSIRIS to identify the number of robustly testable and untestable PDFs is only 50%, on an average, when compared to [20]. Furthermore observe that [20] aborts faults for circuits C1908, C3540, C5315 and C7552. Clearly, a major advantage of the proposed framework is the ability to process path intensive circuits without aborting any faults. This can

Circuit Name	Total Faults	TIP [20] Testable PDFs	Aborted	time (s) ◇	OSIRIS Untestable PDFs	Potentially Testable	Testable PDFs	Undetermined	time (s)
c880	17,284	16,083	0	2.94	163	17,121	16,083	0	2.8
c1355	8,346,432	22,784	0	29.92	1,086,222	371,892	22,784	0	18.2
c1908	1,458,114	97,588	51,942	3735.17	8,031,072	315,360	97,589	0	1281.4
c2670	1,359,920	15,370	0	11.8	1,146,841	213,079	15,370	0	11.6
c3540	57,353,342	88,408	481	4821.43	53,489,826	3,863,516	88,408	0	1456.0
c5315	2,682,610	81,435	926	4774.77	2,078,400	604,210	81,435	0	544.7
c6288	$1.98{\cdot}10^{20}$	*	*	*	$1.979{\cdot}10^{20}$	$1.09{\cdot}10^{17}$	40,323	$2.01{\cdot}10^{14}$	18,987.4
c7552	1,452,988	86,251	326	2860.29	1,003,938	449,050	86,251	0	680.1
s9234	489,708	21,389	0	13.96	443,926	45,782	21,389	0	9.83
s13207	2,690,738	27,603	0	80.24	2,302,098	388,640	27,603	0	46.77
s15850	329,476,092	182,673	0	510.85	322,624,671	6,851,421	182,673	0	357.21
s35932	394,282	21,783	0	360.18	355,201	39,081	21,783	0	147.76
s38417	2,783,158	598,062	0	2698.11	1,675,920	1,107,238	598,062	0	923.91
s38584	2,161,446	92,239	0	590.54	1,623,878	537,568	92,239	0	390.36

Table 1: Robustly Testable and Untestable Path Delay Faults

◇ CPU Time has been interpolated based on the hardware used, for comparison purposes.

be observed in Column 10 of Table 1. The only exception is the circuit C6288, which contains $2.01{\cdot}10^{14}$ undetermined faults. All aborted faults were present in fanout cones for which we were unable to build the BDDs.

We also implemented the approach using the non-canonical BED data-structure [8] which does not have any space limitation problems. When BEDs are used instead of BDDs the approach turned out to be significantly slower. (Approximately ten times slower.) For C6288 we set a time limit of 25,000 sec but within that limit we were not able to identify more robustly untestable PDFs. As an alternative to BEDs, one can use conjuctive normal form (CNF)-based formulations for which efficient solvers exist [11]. The method in [13] can be used to derive the required CNFs in a compact manner. However, preliminary experimentations in that direction resulted to similar results to the BED-based implementation.

Nevertheless, the proposed method identifies more robustly testable faults for C6288 than any existing technique. Observe that [20] fails to report any result for C6288 due to the difficulty in building the implication graph for this circuit. The only other technique to identify large number of testable faults for C6288 is [5]. Still [5] identifies only 12,592 robustly testable faults in 40 hours (interpolated CPU time) and aborts for 10^{18} faults. Our approach identifies 40,323 faults and only aborts 2.012^{14} faults. This clearly indicates efficiency of the proposed approach.

It can be observed from Column 6 that more than 90% of the PDFs are identified as untestable using the implemented static implications on ZBDDs. However it cannot be concluded that the static implications have more impact in identifying the untestable faults. It can be observed from Columns 7 and 8 that the number of robustly testable faults is only a small fraction of the number of potentially (robustly) testable faults determined by the approach in Section 2. Hence the iterative approach presented in Section 4 is very significant component of OSIRIS.

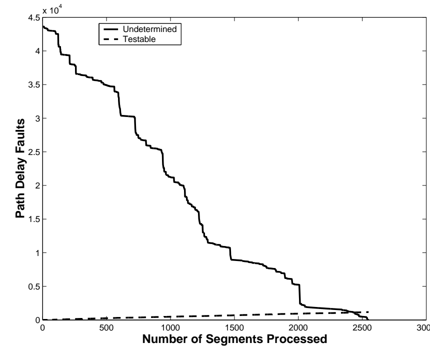


Figure 3: Performance of OSIRIS on s713

Figure 3 shows a plot depicting the performance of OSIRIS on the ISCAS'89 benchmark s713. The x-axis represents the number of segments processed and the y-axis represents the number of PDFs. The solid line in the plot shows the decrease in the number of undetermined faults (due to the elimination of untestable faults) as the number of segments increases. The dashed line shows the number of faults identified as testable as the number of processed segments increases. The implication based method of Section 2 was not activated in OSIRIS when these results were plotted. This was done in order to show the advantage of processing segments rather than processing the entire set of paths in the circuit. Circuit s713 contains a total of 43,624 faults among which only 1,184 are testable. However, the nature of the proposed algorithm makes it easy to identify all the testable and untestable faults by processing only 2,597 segments.

OSIRIS has been observed to perform better for circuits with more re-convergences, because it does not re-calculate functions $\mathcal{R}_r$ and $\mathcal{R}_f$ for the segments in-

volved. This can be clearly observed from the results listed for circuits C3540 and C6288. These circuits are very path intensive and contain a lot of re-convergences.

In the remaining of the section we discuss the performance of the three phase procedure for identifying critical testable PDFs using the bounded delay model. In order to demonstrate the scalability of the tool we provide with experimental results on all path intensive ISCAS'85, ISCAS'89 and ITC'99 benchmarks on a Sun Blade 1000 workstation. Only those benchmarks with more than 10^7 PDFs were chosen for this experimentation. In addition, functionally testable path delay faults are considered. This is a superset of the robustly testable path delay faults that we listed in the previous experiments. The information about the delay ranges for each gate was obtained from the TSMC 0.5μ technology files using the corner values for the nMOS and pMOS transistors. The value of n (the segment length) was set as 5% of the topological longest path in the corresponding circuit. The value of $\mathcal{D}_{th}$ (delay threshold) was set as 90% of the circuit delay obtained using static timing analysis.

Table 2 compares the proposed ZBDD-based method for identifying $\mathcal{L}$, the set of potentially testable critical PDFs, with a structural technique, similar to [15], which is based on branch and bound algorithms. The information about the exact set of critical faults identified from $\mathcal{L}$ is also provided. We consider in these experiments functionally sensitizable critical PDFs. Column 1 lists the circuits under consideration. Column 2 shows the total number faults in the circuit. Column 3 gives the number $|\mathcal{L}|$ of potentially testable critical PDFs identified after the second phase of the method in Section 5. (Note that $\mathcal{L}$ may contain some non critical PDFs). Column 4 shows the total CPU time required for identifying set $\mathcal{L}$. Columns 5 and 6 show the number of potentially testable critical PDFs identified using a method implemented using branch and bound algorithms and the CPU time required, respectively. Columns 5 and 6 provide no data for circuits where the CPU time exceeded the preset limit of 25,000 seconds.

Observe that the number of potentially testable PDFs using the proposed approach differs to that reported using structural methods. (See columns 3 and 5). This is because Phase 2 of our algorithm is pessimistic whereas the structural technique is exact. However, observe that the accuracy of the method falls short by only 10%, on average. Our approach is clearly very time efficient when compared to the branch and bound algorithm.

The time performance of the proposed method is observed to increase with the increase in the fault count in the circuit. Its implicit nature is shown in Column 4 and its performance does not depend on the number of PDFs in the circuit while computing $\mathcal{L}$. The results presented in the Columns 7 and 8 of Table 2 demonstrate the effectiveness of the approach in Section 3. Column 7 shows the exact number of critical PDFs. (They identified from set $\mathcal{L}$.) Column 8 shows the CPU time required for this process. In order for a PDFs to be identified as critical, a two phase procedure has been implemented. First, boolean functions were formed to check for the existence of a robust test. If a robust test does not exist, then we check for a non-robust. Also, any non critical PDFs in $\mathcal{L}$ were removed as explained at the beginning of Section 5.

To our knowledge, there exists no methodology to handle circuit instances with such a large path count with comparable CPU time. It is also noted that BDDs were used to represent the boolean functions during the ATPG process for all the benchmarks, except for b21_opt, b22_opt and c6288 where BEDs were used to represent the boolean functions.BEDs can also be used for all the other circuits, however a BED based implementation was observed to be slower than a BDD based implementation.

This is well expected due to the non-canonical nature of the BEDs. The only advantage of using a BED based implementation is memory saving. For path intensive circuits (like b21_opt, b22_opt and c6288), BEDs used 30-40% less memory, however the CPU time required increased by a factor of 15-20 times[2]. A behavior was also observed for a CNF based implementation also. For smaller circuits (like c3540, b_05, b_14, etc.,) the BED based implementation was 4-6 times slower than a BDD based implementation. Based on these observations, the overall framework has been implemented to use a BDD based approach whenever it is possible to represent the functionality of the circuit as a BDD and to use BEDs only if the functions cannot be represented by a BDD within the given system resources.

The experimentation described above using the gate delays from the TSMC 0.5μ library was repeated for the AMI 0.8μ library without changing the value of n. $\mathcal{D}_{th}$ was re-calculated based on the AMI 0.8μ library information. It was observed that the number of potentially testable critical faults was 823846, 346197 and 1208991 for the benchmarks b14, b14_1 and b15, respectively. The number of critical PDFs identified was 107354, 67466 and 390745, respectively. This shows a big increase in the number of critical PDFs with technology scaling (from 0.8μ to 0.5μ)[3]. This is because the process variation increases with technology scaling, causing a wider range for the gate delays. This wider range for the gate delays allows for more faults to be critical. This fact is also supported by the presented experimental results. The increase in number of critical faults with technology scaling supports our claim that the proposed non-enumerative method on the bounded

[2] The slower performance was observed by comparing the response of the BDD and BED based implementation only on the paths terminating on certain primary outputs in the above mentioned circuits, where we were able to represent functions as a BDD

[3] Similar behavior was observed for all the benchmarks listed in Table 2 for various processing technologies, however the results are not provided here for all benchmarks due to space limitations

Circuit Name	Total Faults	ZBDD Based Potentially Testable	ZBDD Based time (s)	Structural Based Potentially Testable	Structural Based time (s)	Critical Faults	ATPG Time (s)
b05	398,724,610	16,235,622	772.1	13,920,368	15,266	1,010,928	498.2
b14	186,784,982	9,981,266	929.2	8,201,928	13,929	680,992	466.7
b14_1	95,513,816	4,995,231	637.2	4,109,364	7,922	428,193	410.6
b15	96,511,691,200	1,191,344,982	791.9	*	*	5,242,925	1,879.3
b17	790,677,089,462	1,456,982,101	1,198.3	*	*	1,923,882	1,422.9
b17_opt	545,242,452,572	947,575,823	1,682.1	*	*	893,109	1,992.1
b20_opt	1,124,411,428	39,548,273	1,722.5	*	*	401,857	733.7
b21_opt	905,448,842	37,232,934	1,271.3	30,008,798	17,364	582,573	1,231.3
b22_opt	1,746,841,526	39,992,248	1,425.8	*	*	970,366	1,522.0
c3540	57,353,342	2,123,456	292.1	1,892,221	4,623	703,280	130.9
c6288	$1.98 \cdot 10^{20}$	708,242,991,082	792.4	*	*	14,783,347	17,098.0
s4863	2,636,114,122	52,994,304	385.7	*	*	208,303	387.6
s6669	431,685,738,673,270	4,691,062,272	527.8	*	*	545,920	4,293.0
s15850	329,476,092	10,929,556	203.1	8,278,662	12,786	908,344	308.9

Table 2: Identification of Critical Paths Delay Faults

delay model is increasingly more accurate in identifying the set of critical faults over branch and bound based, path enumerative techniques as in [15] which identify the longest sensitizable critical faults using the fixed delay model.

Column 7 of Table 2 indicates that for the listed circuits the number of critical functionally testable faults identified is mostly in the order of 10^5-10^6 faults. However, the number of critical paths selected is a direct reflection of the confidence level with which the circuit under test can be guaranteed for temporal correctness. If less critical testable PDFs are to be tested due to time limitations in the testing process then the value of the delay threshold $\mathcal{D}_{th}$ must be increased. Alternatively, one can focus only on robustly testable critical PDFs. Either decision reduces the confidence level with which the circuit under test can be guaranteed for temporal correctness.

6 Conclusions

We present a novel and efficient framework to classify the PDFs in a circuit as either testable or untestable. We use a combination of data structures, the ZBDDs and BDDs to classify the PDFs. We introduce an iterative approach to perform the classification. The framework performs effectively even for large and path intensive circuits. The experimental results demonstrate the effectiveness of the approach when compared to existing methods. We also modified the framework to identify all critical sensitizable PDFs using the bounded delay model which provides with more accuracy in the process of defining critical PDFs for deep submicron technology. The bounded delay model introduces significant computational complications. Nevertheless, it is shown that this method can handle all existing path intensive benchmarks in the ISCAS'85, ISCAS'89 and ITC'99 collections very fast. A future direction is to modify the proposed techniques to generate compact tests after identifying the set of testable faults.

References

[1] Bhattacharya D., Agrawal P. and Agrawal V.D., *Test Generation for Path Delay Faults using Binary Decision Diagrams*, IEEE Trans. on Computers, vol. 44, no. 3, Mar. 1995, pp. 434-447.

[2] Borkar S., et al., *Parameter variations and impact on circuits and microarchitecture*, Design Automation Conference, June 2003, pp. 338-342.

[3] Cheng K.T and Chen H.C., *Classification and Identification of Nonrobust Untestable Path Delay Faults*, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 15, Aug. 1996, pp. 845-853.

[4] Eppstein D., *Finding the k Shortest Paths*, IEEE Symp. Foundations of Computer Science, Santa Fe, 1994, pp. 154-165.

[5] Fuchs K., Pabst M. and Rossel T., *RESIST: A Recursive TestPattern Generation Algorithm for Path Delay Faults considering Various Test Classes*, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, no. 12, Dec. 1994, pp. 1550-1561.

[6] Heragu K., Patel J.H. and Agrawal V.D., *Fast identification of untestable delay faults using implications*, Proc. International Conference on CAD, 1997, pp. 642-647.

[7] Hitchcock R.B., Smith G.L. and Cheng D.D., *Timing Analysis of Computer Hardware*, IBM journal of Research and Development, January 1982, pp. 100-105.

[8] Hulgaard H., Williams P.F. and Andersen H.R., *Boolean Equivalence Checking of Combinational Circuits using Boolean Expression Diagrams*, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 18, no. 7, Jul. 1999, pp. 903-917.

[9] Kirkpatrick T.I., and Clark N.R., *PERT as an Aid to Logic Design*, IBM journal of Research and Development, March 1966, pp. 135-141.

[10] Li Z.C. , Brayton R.K., Min Y., *Efficient Identification of Non-Robustly Untestable Path Delay Faults*, Proc. Internation Test Conference, 1997, pp. 992-997.

[11] Marques Silva J.P., Sakallah K.A, *Grasp: A search algorithm for propositional satisfiability*, IEEE Transactions on Computers, vol. 48, no. 5, pp. 506–521.

[12] McGeer P.C., Saldanha A., Stephan P.R., Brayton R.K and Sangiovanni-Vincentelli A.L., *Timing Analysis and Delay Fault Test Generation using Path Recursive Functions*, Proc. of International Conference on Computer Aided Design, 1991, pp.180-183.

[13] Michael M., Tragoudas S., *ATPG Tools for Path Delay Faults at the Functional Level*, ACM Transactions on Design Automation of Electronic Systems, vol. 7, no. 1, pp. 35–37, 2002.

[14] Minato S.I, *Zero-Suppressed Binary Decision Diagrams*, Proc. Design Automation Conference, 1995.

[15] Murakami A., Kajihara S., Sasao,T., Pomeranz I., Reddy S.M., *Selection of Potentially Testable Path Delay Faults for Test Generation* Proc. International Test Conference, 2000, pp. 376 -384.

[16] Padmanaban S., Michael M. and Tragoudas S., *Exact Path Delay Fault Coverage with Fundamental Zero-Suppressed BDD Operations*, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Mar. 2003, pp. 305-316.

[17] Papadimitriou C.H and Steiglitz K., *Combinatorial Optimization : Algorithms and Complexity*, Dover Publishers, July 1998.

[18] Pomeranz I. and Reddy S.M., *SPACES-ACE: Simulator for Path Delay Faults*, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, no.2, Feb. 1994, pp. 254-263.

[19] Shao Y., Reddy S.M., Kajihara S. and Pomeranz I., *An Efficient Method to Identify Untestable Path Delay Faults*, Proc. Asian Test Symposium, 2001.

[20] Tafertshofer P., Ganz A., Antreich K.J., *IGRAINE - An Implication GRaph bAsed engINE for Fast Implication, Justification, and Propagation*, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, August 2000.

[21] Tekumalla, R.C. and Menon, P.R., *Identification of primitive faults in combinational and sequential circuits*, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 12, no. 12, December 2001, pp. 1426-1442.