Using BDDs and ZBDDs for Efficient Identification of Testable Path Delay Faults

Saravanan Padmanaban
CSEE Department
University of Maryland Baltimore County
Baltimore, MD 21250
padmanab@umbc.edu

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

ABSTRACT

We present a novel framework to identify all the robustly testable and untestable path delay faults in a circuit. The method uses a combination of decision diagrams for manipulating path delay faults and boolean functions. The approach benefits from processing partial paths or fanout free segments in the circuit rather than the entire path. The effectiveness of the proposed framework is demonstrated experimentally. It is observed that the methodology identifies 350% more testable faults in the ISCAS'85 benchmark C6288 than any existing technique by utilizing only a fraction of the time compared to earlier work.

1 INTRODUCTION

One of the most fundamental problem studied in the field of testability and verification is automatic test pattern generation (ATPG) under a given fault model. Here we study the problem of identifying testable and untestable for path delay faults (PDF). There are several sub-problems involved with any test generation (TG) problem, such as identifying untestable or redundant faults, fault simulation, etc., All the above said problems have been studied in detail as isolated problems and enumerative techniques are employed to integrate the techniques together into a common framework. There exists no technique where the information about the untestable faults could be *non-enumeratively* used for TG so as to enhance its performance.

The main problem faced by the test generation techniques under the path delay fault model, is the exponential number of faults. However it has been observed in [2] that more than 90% of the PDFs in most of the practical circuits are robustly untestable. The terms robustly untestable faults, untestable faults and untestable PDFs are used interchangeably in this paper. All the explanations provided in this paper are with respect to robustly testable and robustly untestable path delay faults. However all the concepts presented are directly applicable for the other classifications of the path delay fault model. The untestable faults do not have to be targeted for TG which can greatly enhance the time performance of the test generator. There exists various techniques to identify a *l*ower bound on the untestable faults using static implications. See [4], [5], [11] for more details. The existing techniques identify pairs of lines in the circuit or segments of a path, through which all the PDFs are untestable. They also use a linear time algorithm to count the approximate number of untestable PDFs after identifying all the pairs of lines or segments.

The approach proposed here to deal with this problem is to exactly identify the set of faults (non-enumeratively) that constitute the lower bound on the untestable faults using static implication and eliminate it from the set of all possible faults in the circuit. The method takes advantage of the ability of the zero-suppressed binary decision diagram (ZBDD) [7] to represent PDFs and also to identify the set of untestable PDFs (lower bound) using static implication techniques. The remaining faults are potentially testable and needs to be targeted by the test generator. The test generator is built to process partial paths or fanout free segments iteratively using reduced ordered binary decision diagrams (BDD) thereby identifying all the testable faults in the circuit. A similar idea has been proposed in [8], however the approach only selects a subset of the potentially testable faults (based on the longest path criteria) due to the lack of efficient data structures for path representation and manipulation.

Section 2 outlines certain conditions based on static implications, to identify a lower bound to the set of untestable PDFs. We also show how these conditions can be implemented non-enumeratively using ZBDDs. Section 3 discusses certain key features about fanout free segments in circuits and their use in identifying testable faults. A method to represent each fanout free segment in a circuit by a unique number is also introduced. Section 4 introduces an iterative approach to identify robustly testable PDFs from the set of potentially testable PDFs and implicitly identify more robustly untestable PDFs. The experimental results and conclusions are presented in Sections 5 and 6 respectively.

2 IDENTIFYING UNTESTABLE FAULTS USING STATIC IMPLICATIONS

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 vari-

1530-1591/04 $20.00 (c) 2004 IEEE

able v is interpreted as a don't-care. This property makes the ZBDD to effectively represent sets of PDFs as sets of minterms [9]. Figure 1(b) shows a ZBDD representing all the PDFs in the circuit shown in Figure 1(a)[2].

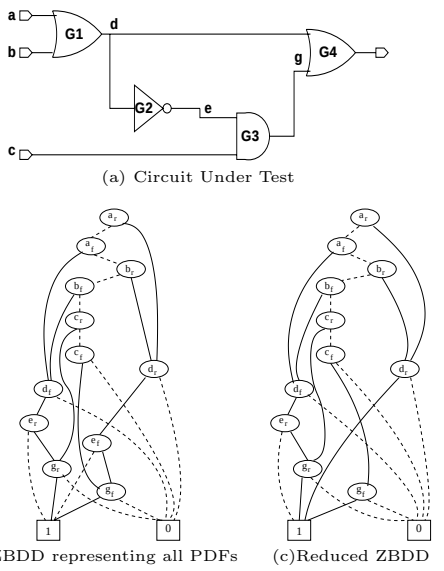
Next we show how ZBDDs are used to non-enumeratively identify a lower bound on the untestable faults in two phases. The first phase implicitly enumerates all the possible PDFs in the circuit as a single ZBDD. The second phase identifies the untestable PDFs using static implications and eliminates the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

We list below the conditions based on static implications that we use to identify a lower bound on the untestable faults [4], [5], [11]. Each of the condition was found to have an impact in improving the former quantity.

Condition 1 *Let $f_1, f_2, \cdots, f_m$ be the fanins 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 fanins of the gate f. Let $g_1, g_2, \cdots, g_n$ be the fanins 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.*

Next we discuss how the static implication based conditions described earlier can be implemented non-enumeratively using fundamental ZBDD operators introduced in [7]. An illustration describing the implementation of the Condition 1 is outlined by the following example.



(a) Circuit Under Test

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

Figure 1: Illustration of Untestable Fault Elimination

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 single topological traversal on the circuit. Let us now illustrate Condition 1 by means of an example and its implementation using ZBDDs. Assume, d be assigned logic 1. It can be observed that when $d = 1$, it is a controlling value of gate $G4$. It also implies a logic 0 at gate $G3$, which is a non-controlling value for gate $G4$. As discussed in Condition 1, it can be observed that all PDFs originating from any primary input, that induces a rising transition on line d and also propagates the signal through line g are robustly untestable.

For this purpose of representing the signal transition on each line in the circuit, we use two ZBDD variables for each line in the circuit. Each variable corresponding to a line l either corresponds to the odd or even parity of PDFs originating at the primary inputs and passing through line l. In the example, all PDFs containing both the variables, d_r and g_f are robustly untestable (by Condition 1) and can be eliminated from the ZBDD ξ. These PDFs are {↑ b.d.e.g} and {↑ 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

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

Once the untestable faults are identified, they can be eliminated from ξ using a set difference operation to obtain the set of potentially testable PDFs (ζ).

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

Similar operation can also be used to eliminate untestable faults through a set of lines or a sub-path rather than a pair of lines, by invoking the Subset operation recursively. For the example described, Figure 1(c) shows the representation of the set of potentially testable PDFs.

The main advantage of the proposed approach to identify the set of potentially testable faults is its non-enumerative nature. 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 can be exponential (in the worst case) to the number of nodes in the circuit, the proposed ZBDD based approach requires only $3 * n$ ZBDD operations (2 operations for the subset and 1 operation for the set difference) to identify the set of potentially testable faults.

The Conditions 1 and 2 are only a subset of conditions that can be used to identify untestable PDFs. Additional conditions to identify untestable PDFs and their proofs can be found in [4]. However it has to be noted that, only a lower bound on the robustly untestable PDFs are identified using the static implication based techniques and further processing is required to identify the set of all testable and untestable PDFs. We discuss such a procedure in Section 4.

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

3 FANOUT FREE SEGMENTS

In this section we discuss certain key properties of fanout free segments in a circuit and their use in the process of identifying testable faults. We also present a method to represent each fanout free segments in a circuit by an unique number and the need for such a representation. Though the algorithm describing the framework to identify testable faults is presented in Section 4, one of the key component to the algorithm is the representation and manipulation of fanout free segments which is discussed in this section. A fanout free segment can be defined as follow:

Definition 1 (Fanout Free Segment [11]) *A partial path or a fanout free segment is a sub-path in the circuit between a primary input and a fanout stem or between two fanout stems or between a fanout stem and a primary output.*

Here in this paper, without loss of generality we refer to the fanout free segment as a *segment*. The number of segments in a circuit is linear to the number of nodes in a circuit. Each segment is associated with two *s*ub-faults, one for a rising transition at the start of the segment and one for a falling transition at the start of the segment. Let $\mathcal{R}_l$ be a boolean function [3] whose solutions are robust tests for a given segment l.

Theorem 1 *Let P be a PDF constituted by n sub-faults corresponding to the fanout free segments $l_1, l_2, \cdots, l_n$ that makes the physical path corresponding to P. Let $\mathcal{R}_P$ and $\mathcal{R}_{l_i}$ be the functions whose solutions correspond to the robust tests for the PDF P and the fanout free segment l_i, respectively. Then*

$$\mathcal{R}_P = \bigwedge_{i=1}^{n} \mathcal{R}_{l_i}$$

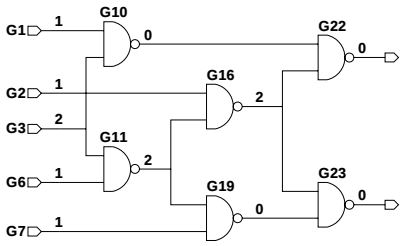
The Thoerem 1 physically signifies that $\mathcal{R}_P$ contains cubes that are robust tests for each of the n sub-faults $l_1, l_2, \cdots, l_n$ that make up P. This condition ensures that the transition at p is propagated to the primary output.

Theorem 2 *Let p be a primary input. Let l be a fanout free segment and $\mathcal{R}_l$ be the function whose solutions correspond to the robust tests for l. Let t_l and t_p be the transitions at the start of the segment l and the primary input p respectively. $\mathcal{R}_l$ is fixed for all path delay faults through l, originating from p with a transition t_p and propagating with a transition t_l at the start of l.*

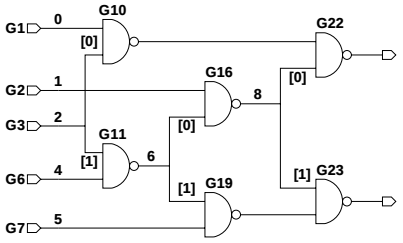
From Theorem 2 it can be observed that for a given segment l and a primary input p, there are only four possible values for $\mathcal{R}_l$ based on the values of the transition t_l at the start of the segment and the transition t_p at the primary input p. This is true for all the PDFs that originate at p and pass through l. Hence the maximum number of functions needed to generate tests for all PDFs in the circuit is

$$4 * \#\ of\ Primary\ Inputs * Number\ of\ Segments$$

which is strictly a polynomial quantity for any possible circuit. This property makes the proposed technique very effective even for large and path intensive circuits. Based on this observation, the functions $\mathcal{R}_l$ derived for a segment l for a given value of t_l and t_p can be stored and be used for all the paths originating from p and passing through l. This prevents the function from being recalculated for every PDF through l, there by improving the computational efficiency. For this purpose we need an efficient way to represent the segments and their associated signal transition. In this section, we show a segment representation scheme that assigns a unique integer to each segment for the purpose of storing and retrieving the function corresponding to the segment. The unique number can be used as the index to store $\mathcal{R}_l$ in a lookup array or hash table.



(a) Segment Counting



(b) Segment Labelling

Segment	ID	Segment	ID
G1.G10.G22	0	G7.G19.G23	5
G2.G16	1	G11.G16	6
G3.G10.G22	2	G11.G19.G23	7
G3.G11	3	G16.G22	8
G6.G11	4	G16.G23	9

(c) Unique Segment Representation

Figure 2: Segment - Unique Representation

The segment representation scheme is based on a labeling of the circuit. Every node x_i is assigned a label $N_p(x_i)$, that represents the number of segments from node x_i to all the primary outputs. We use the following procedure to compute the labels $N_p(x_i)$ and thereby computing the total number of segments in the circuit. A similar procedure to label paths is used in [10].

Procedure 1: Computing the label $N_p(x_i)$
(1) For each nodes x_i set $N_p(x_i) = 0$.
(2) If node x_i is a primary input of the circuit, set $N_p(x_i)$ = Number of fanout branches of x_i.
(3) If node x_i is an internal gate, set $N_p(x_i)$ = Number of fanout branches of x_i if the number of fanout branches of x_i is greater than 1.

[3] In this paper when we refer to boolean or algebraic functions, they are internally represented by BDDs

If the nodes in the circuit are $x_1, x_2, \cdots, x_n$, the total number of segments in the circuit is $\Sigma_{i=1}^{n} N_p(x_i)$. An illustrated of Procedure 1 is shown using circuit c17 in Figure 2(a). The $N_p(x_i)$ labels are given next to the nodes. It can also be observed that the total number of segments in the circuit is 10. Given the label $N_p(x_i)$ for every node x_i, we compute a second label for each node x_i with $N_p(x_i) > 0$, denoted as $N_l(x_i)$. We use the following procedure to compute the label $N_l(x_i)$ and thereby computing a unique number to identify each segment.

Procedure 2: Computing the labels $N_l(x_i)$
(1) For all the nodes x_i set $N_l(x_i) = 0$.
(2) If a node x_i has fanout branches $b_1, b_2, \cdots, b_k$, each branch is assigned a label $0, 1, \cdots, k-1$.
(3) Set j = 0.
(4) For each node x_i
(4.1) If $N_p(x_i) > 0$, set $B(j) = N_p(x_i)$.
(4.2) Increment j.
(5) For each primary input node $x_1, x_2, \cdots, x_n$ set $N_l(x_1) = 0, N_l(x_2) = B(1), N_l(x_3) = B(1) + B(2), \cdots$.
(6) Set $Count = N_l(x_n)$ and $j = n$.
(7) For each internal node x_i with $N_p(x_i) > 0$
(7.1) Set $Count = Count + B(j)$, Increment j.
(7.2) Set $N_l(x_i) = Count$.

This is also illustrated for the circuit c17 in Figure 2(b). The label $N_l(x_i)$ denotes the unique number of the first segment that originates from node x_i. It can be observed from Figures 2(a) and 2(b) that node $G3$ contains the labels $N_p(x_i)$ and $N_l(x_i)$ labeled as 2 and 2, this implies that there are 2 segments originating from node $G3$ and that the first of the two segments has a unique identification number 2. The unique number for any segment can be identified by summing up the $N_l(x_i)$ of the constituent nodes and their respective fanout branch labels if present. For example, the unique number corresponding to the segment G2.G16 is calculated as 1 ($N_l(G2) = 1$ and it does not have any fanout branch labels). The unique number for the segment G3.G11 is computed as 2+1 = 3 ($N_l(G3) = 1$ and its fanout branch contains a label 1). The information about the segments and their unique numbers for circuit c17 is shown in Figure 2(c).

If a circuit has L segments, then the two sub-faults associated with a segment l can be assigned the unique numbers i and $i + L$. The same concept can be extended to assign four unique numbers for each segment l based on the transitions t_l and t_p as per the discussion presented in Theorem 2. They can be assigned the unique numbers - i, $i + L$, $i + 2L$ and $i + 3L$. These four indices are used to store the corresponding function $\mathcal{R}_l$ in a lookup array during the process of identifying the testable faults. This process is explained in detail in Section 4.

4 IDENTIFICATION OF TESTABLE PATH DELAY FAULTS

In this section we introduce an iterative approach to identify robustly testable PDFs. We start with the set of potentially testable PDFs identified using static implications described in Section 2. Given a PDF, methods proposed in [1] and [6] can be used for deriving boolean functions that describe all possible robust tests for the target fault. We use a very similar approach for deriving the boolean functions.

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 [2] 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.

```
Identify the Set of Potentially Testable PDFs (ζ)
For Each i ∈ Primary Inputs
  ζ_i = Subset of ζ originating at i
  While ζ_i ≠ φ
    Pick PDF P by DFS
    Split P into segments
    For each segment p ∈ P
      If R_r present in lookup array
        Extract R_r from lookup array
      Else
        Generate R_r
      If R_r is Satisfiable
        Add R_r to lookup array
        T_R = T_R ∧ R_r
        If T_R is Not Satisfiable
          Eliminate PDFs in ζ_i through the segment
                                              from i to p
          ζ_i = ζ_i \ Tested PDFs
          T_R = φ
          Break
      Else
        Eliminate PDFs in ζ_i through p
        ζ_i = ζ_i \ Tested PDFs
        T_R = φ
        Break
      /*Same procedure is followed for R_f*/
    If T_R ≠ φ
      Tested PDFs = Tested PDFs ∪P
      Tested PDF Count = Tested PDF Count + 1
Untestable PDFs = ζ  \ Tested PDFs
```

Table 1: Identification of Testable Path Delay Faults

Structural ATPG techniques like [3] 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. 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 if 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 traversing through unnecessary lines are avoided. 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 lookup 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, however if the conjunction of the two functions is not satisfiable it implies that there exists no solution that can excite both the 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 in Table 1.

5 EXPERIMENTAL RESULTS

We implemented the proposed approach in C. We call the proposed tool OSIRIS. The performance of the tool was experimented on the ISCAS'85 and ISCAS'89 benchmarks using a Sun Blade workstation. The results of the experimentation is presented in Table 2. We compare our results to the method of [12] due to its effectiveness in identifying all possible testable faults. [12] has another similarity with the proposed technique in terms of the graph based data structures used for the problem studied. [12] uses an implication graph for the purpose of test generation and the proposed approach uses boolean functions implemented by BDDs, which also implicitly contains all possible implications. Hence both techniques implicitly identifies all possible solutions for a given fault.

Column 2 shows the total number of PDFs in the corresponding circuit. Columns 3 shows the number of PDFs identified as robustly testable by [12]. Column 4 shows the number of aborted faults for the method. Column 5 shows the execution time for approach [12].

Column 6 shows the number of PDFs identified as untestable during the preprocessing step, using the static implications discussed in Section 2. The difference between the total number of PDFs (Column 2) and the lower bound of untestable faults (Column 6) is 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 4. Column 8 shows the total number robustly testable PDFs identified by OSIRIS. Column 9 reports the number of PDFs yet to be processed. The total execution time for each circuit is reported in Column 10.

It can be observed from Columns 5 and 10 that the time required for execution is only 50% on an average when compared to [12], to identify the number of robustly testable and untestable PDFs. [12] contains aborted faults for the circuits C1908, C3540, C5315 and C7552. We note that a major advantage of the proposed framework is the ability to process path intensive circuits without aborting any faults. This can be observed in Column 10 of Table 2. The only exception is the circuit C6288, which contains $2.01 \cdot 10^{14}$ undetermined faults. However these are the faults present in the fanout cones for which we were unable to build the BDDs.

The proposed method identifies more testable faults for C6288 than any existing technique. [12] fails to report any result for C6288 due to the difficulty in building the implication graph for the circuit. The only other technique to identify large number of testable faults for C6288 is [3]. Still [3] identifies only 12,592 faults as testable in 40 hours(interpolated CPU time) and aborts for 10^{18} faults. It is observed here that the number of potentially testable faults identified for C6288 by the proposed technique is much less when compared to the number of faults aborted in [3]. 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 static implications. 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 testable faults are only a small fraction of the upper bound of testable faults. Hence the iterative approach is an equally important step in the identification of untestable faults.

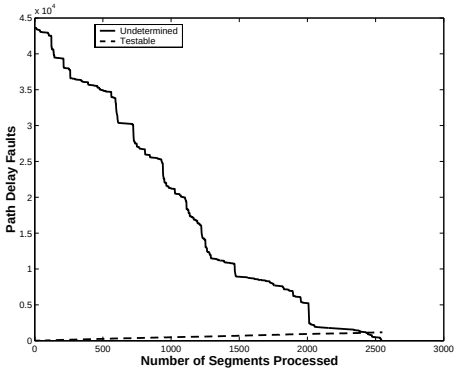


Figure 3: Performance of OSIRIS on s713

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) with the increase in the

Circuit Name	Total Faults	TIP [12] Testable PDFs	TIP [12] Aborted	TIP [12] time (s) ◇	OSIRIS Untestable PDFs	OSIRIS Potentially Testable	OSIRIS Testable PDFs	OSIRIS Undetermined	OSIRIS 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	$1.98{\cdot}10^{20}$	*	*	*	$1.98{\cdot}10^{20}$	$1.09{\cdot}10^{17}$	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 2: Robustly Testable and Untestable Path Delay Faults

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

number of segments and the dashed line shows the number of faults identified as testable with the increase in the number of segments. The pre-processing step where untestable faults are identified using static implications is not performed for this circuit. This is done to show the advantage of processing segments rather than processing the entire set of paths in the circuit. The 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.

The method has been observed to perform better for circuits with more reconvergences, because of the property of not having to recalculate the functions $\mathcal{R}_r$ and $\mathcal{R}_f$ for the segments involved. This can be clearly observed from the results presented for the circuits C3540 and C6288 which are quite path intensive and contain a lot of reconvergences in them.

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 demonstrates the effectiveness of the approach when compared to existing methods. The future direction is to modify this framework is 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] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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 Design, 1991, pp.180-183.

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

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.