Exact Grading of Multiple Path Delay Faults *†

Saravanan Padmanaban **Spyros Tragoudas**

Abstract

The problem of fault grading for multiple path delay faults is studied and a method of obtaining the exact coverage is presented. The faults covered are represented and manipulated as sets by zero-suppressed binary decision diagrams (ZBDD), which are shown to be able to store a very large number of path delay faults. For the extreme case of memory problem, a method to estimate the coverage of the test set is also presented. The problem of fault grading is solved with a polynomial number of BDD operations. Experimental results on the ISCAS'85 benchmark include test sets from ATPG tools and specifically designed tests in order to investigate the limitations and properties of the proposed method.

1 Introduction

The main objective of delay fault testing is to check for the timing performance of a designed circuit. The problem of calculating the number of faults identified by a given test set under a specific fault model is termed as non-enumerative fault grading or fault coverage. Under the path delay fault (PDF) model, a circuit is considered faulty if the propagation delay of a path exceeds the pre-determined clock period of the circuit, the delay fault can be observed by propagating a rising or falling transition through the path. This requires that each of the test in the test set consists of 2 vectors.

Existing enumerative and non-enumerative [1], [2], [8] fault grading techniques only examine the single path delay faults (SPDF).It has been shown that multiple path delay faults can affect the timing performance of a circuit [3]. Even though there exist methods for generating test patterns for this category of delay faults, there is no existing technique to calculate the exact coverage for the test sets aimed at multiple paths. This paper introduces the first method to calculate the exact coverage of multiple path delay faults for a given test set using a polynomial number of basic ZBDD operations.

The paper is organized as follows. Section 2 gives preliminary definitions including the ZBDD data structure. Section 3 outlines our non-enumerative method to calculate the number of MPDFs covered by a given test set. Section 5 presents a more evolving ZBDD based methodology for calculating only those PDFs that are useful in delay fault testing, which we term as *non-redundant multiple path delay fault* (nr-PDF). nr-PDFs are basically the set of *primitive PDFs* [3] identified by a given test set. The proposed method to identify the nr-PDF behaves linear to the number of MPDFs identified by the test set, in terms of basic ZBDD operations. Section 5 shows how the approach of Section 3 and 5 can be modified to handle memory intensive instances with minimal loss of accuracy in the coverage. However, our experimental results show that memory problems arise only on artificially constructed instances, where the number of detected PDFs is enormous. Experimental results and conclusions are presented in Section 6 and 7 respectively.

*The authors are with the Department of Electrical and Computer Engineering, Southern Illinois University, Carbondale, IL 62901, USA

†Research supported by NSF grant CCR 0096119.

2 Preliminaries

A data structure to represent a combinational set in a compact and canonical form is the *zero suppressed binary decision diagram* (ZBDD) [6]. A combination of n objects can be represented by an *n* bit vector $(x_n, x_{n-1}, \ldots, x_2, x_1)$, where each bit $x_k \in 1, 0$ expresses whether the corresponding object is included in the combination or not. A set of combinations can be represented by a set of the *n* bit binary vectors. Such sets are called combination sets. A combination set is sparse if the number of elements is much smaller than the total number of elements that may appear in the set. Combinational sets can be represented by a boolean function, using *n* variables for each of the element of the set. These boolean functions are called *characteristic functions*. It has been shown in [6] that ZBDD's represents combinational sets effectively. The ZBDD does not depend on the number of the variables as long as the combination sets are the same. The number of variables need not be fixed before generating the graph, as the variables for objects which does not appear in any combination are automatically suppressed. This property gives the advantage for the ZBDD over BDD in the representation of sparse sets. The ZBDD representing the combination set $\{\bar{a}\bar{b}c\bar{d}, \bar{a}\bar{b}\bar{c}d\}$, is shown in Figure 1b.

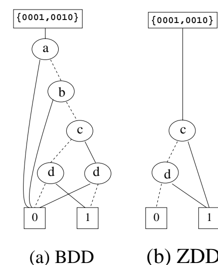


Figure 1: BDD and ZBDD for combination set $\{c, d\}$

Let each line and gate of a circuit be assigned an unique variable and a primary input with two variables - one for rising and one for falling transitions. Each and every PDF in a given circuit can be defined by a unique ordered combinational element. The presence of a variable v in a combinational element representing a PDF implies that $v = 1$ and the absence implies $v = 0$. The ability to represent a PDF as a combinational element reduces the problem of path delay fault coverage to represent the PDFs covered as a combinational set with each element of the set representing a covered PDF. The ability of the ZBDDs to store a

large number of PDFs has been already shown experimentally in [8].

3 Non-Enumerative Grading of MPDFs

In the first step the circuit C is simulated using a test vector i from the test set T and the set of PDFs tested by i (P_i) are identified. In the second step, the set of PDFs P_i are stored as a ZBDD and the set of total PDFs tested by $(i-1)$ vectors (P^T_{i-1}) are updated with the *new PDFs* (P^{new}_i) which is basically P_i or the subset of P_i.

Operation	Symbol	Description
Empty()	(1)	the empty set (0-terminal node).
Base()	(0)	the base set (1-terminal node).
Change(S, v)	!(S, v)	set v for all combinations in set S.
IntSec(S, Q)	$S \cap Q$	the intersection operation.
Union(S, Q)	$S \cup Q$	the union operation .
SubSet(S, v)	$\subset$(S, v)	factor of S by v.
Diff(S, Q)	$S - Q$	the set difference operation.
Prod(S, Q)	$S * Q$	the unate product operation.
Count(S)	$\|S\|$	the number of combinations in set S.

Table 1: Basic ZBDD Operations

The basic ZBDD operations used in the proposed method are listed in Table 1. More details about the operations can be found in [6] [7]. The pseudo-code of the basic algorithm is given in Table 2. The input circuit and test set are denoted by C and T. We define sets $P(C)$ and $P(g_i)$ to represent the tested PDFs for C, and the tested PDFs at the primary outputs for test pattern i. This phase requires two topological traversals of the input circuit. A forward traversal performs the actual simulation for the test patterns and a backward traversal marks the lines and nodes (gates and primary inputs) along the sensitized paths. Procedure Store_PDFs() constructs the ZBDD that represents the set of all PDFs tested under a single test pattern. The MPDFs are stored as a ZBDD using the *Unate Product Operator* as shown in Table 3.

```
PROCEDURE PDF_Grading(C, T)
  P(C) = Empty()
  FOR every test (T_{i,1}, T_{i,2}) ∈ T,  i = 1, 2, ..., |T|  DO
    Simulate(C, (T_{i,1}, T_{i,2}))
    P(g_i)= Store_PDFs(C, Type)
    FOR every Primary Output node g_i ∈ C DO
      P(C)= Union(P(C),P(g_i))
  Number of PDFs tested = Count(P(C))
END PDF_Grading()
```

Table 2: Procedure PDF_Grading

We define a *partial PDF* to be a PDF from some primary input to some internal gate. The ZBDD of an internal gate contains all the partial PDFs from the primary inputs to the gate. At the end of the traversal, the primary outputs are reached and, for every output, the ZBDD for all the PDFs from the primary inputs to the primary output is constructed. At this point, the union of all the ZBDDs for the primary outputs is computed and thus the set of all the PDFs tested by a single test is derived. The pseudo-code for Procedure Store_PDFs() is shown in Table 3.

The rising and falling transitions are denoted by R and F, respectively. For a primary input g, function VarR(g) (resp. VarF(g)) returns the variable that represents an R (resp. F) transition on the input. If g is a gate, Var(g) returns the variable for

```
PROCEDURE Store_PDFs(C, Type)
  DO
    Select node g_i ∈ C to be processed
    IF (g_i is marked)
      IF (g_i is a Primary Input)
        IF (transition on g_i is R) v = VarR(g_i)
        ELSE IF (transition on g_i is F) v = VarF(g_i)
        P(g_i) = Change(Base(),v)
      ELSE IF g_i is singly sensitized
        v = Var(g_i)
        P(g_i) = Empty()
        FOR every input node g_k of g_i
          IF (g_k is marked)
            P(g_i) = Union(P(g_i), Change(P(g_k), v))
      ELSE IF Type=" MPDF"
        IF g_i is co-sensitized
          v = Var(g_i)
          P(g_i) = Empty()
          FOR every input node g_k of g_i
            IF (g_k is marked)
              P(g_i) = Prod(P(g_i), Change(P(g_k), v))
    ELSE
      P(g_i) = Empty()
  UNTIL all nodes are processed
  Return P(g_k) ∀g_k ∈ Primary Output
END Store_PDFs()
```

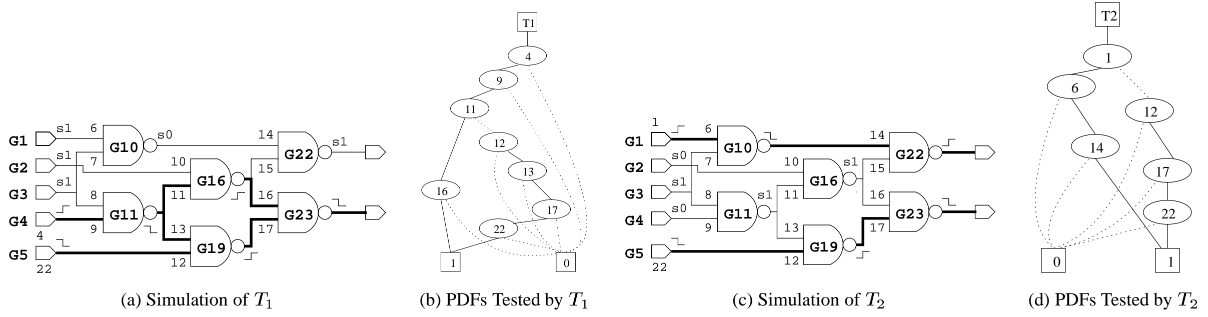
Table 3: Procedure Store_PDFs

the gate. $P(g)$ denotes the ZBDD at gate g that contains all the partial PDFs from the inputs to g.

The processing of nodes is done in a topological order. The marking of the nodes is performed during the first phase of the approach, in procedure Simulate(). The processing at each node differs depending on whether the node is a primary input or a gate. If the node is a primary input, an appropriate variable is assigned depending on the type of the transition on the input. If the node is a gate, the conditions for sensitization are checked. If the gate is singly sensitized, a union operation of the sets of the marked inputs, that represents the partial PDFs from the primary inputs to the respective particular lines is performed and the variable of that gate is added to the resulting set. If the gate is functionally sensitized, a unate product operation of the sets of the marked inputs, that represents the partial PDFs from the primary inputs to the respective particular lines is performed and the variable of that gate is added to the resulting set. At the end of the topological traversal, $P(g)$ of each output gate represents all the tested PDFs from the primary inputs to the particular primary output. The union of sets $P(g)$, of all the primary outputs results in the total PDFs tested a test vector i.

Example: The following example is used to illustrate the proposed method. Let the test vector $T_1 = \{11101, 11110\}$ be applied to the circuit C17 as shown in Figure 2a. The numbers marked on the lines are the variables used to denote that line. For the primary inputs variables 1-5 are used to represent the rising transitions on each input and variables 18-22 for the falling transitions. The bold lines in the circuit C17 indicate the sensitized lines.

In circuit shown in Figure 2a, the partial PDFs from primary inputs to line 12 and 13 are $\{4-9-13\}$ and $\{22-12\}$ respectively. Gate G19 is functionally sensitized, so a unate product operation is performed between the two partial products to represent the co-sensitization between them. The PDFs covered by the 1st test vector T_1 is P_1=$\{4-9-11-16, 4-9-12-13-17-22\}$. The same procedure is followed for the 2nd test vector $T_2 = \{00101, 10100\}$. The simulation of circuit C17 by T_2 is shown in Figure 2c. The PDFs covered by the test vector T_2 is



(a) Simulation of T_1 (b) PDFs Tested by T_1 (c) Simulation of T_2 (d) PDFs Tested by T_2

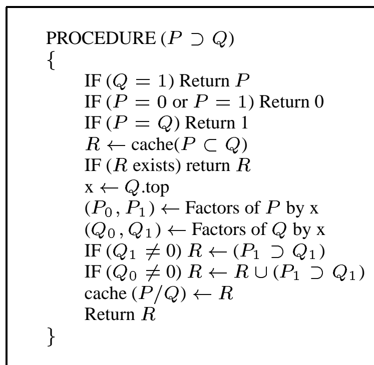
Figure 2: Output of the Iteration Process

P_2={$1-6-14, 12-17-22$}. So if the test set T comprises of 2 vectors T_1 and T_2, the total PDFs covered by test set T is $P_{total} = P_1 \cup P_2$ (i.e 4 PDFs are covered).

4 Elimination of Redundant MPDFs

Consider paths P_i, P_j and P_k to be single path delay faults (SPDF) and $P_iP_jP_K$ be a multiple path delay fault. If any one of the three SPDF is singly sensitized, then the multiple path is not needed to be tested [3]. Such MPDFs which need not be tested are termed as *redundant PDFs*, and should be eliminated from the set of PDFs tested which will result in the set of nr-PDFs.

The basic scheme proposed in Section 3 does not eliminate such redundant MPDFs from the graded fault set. The illustration provided for the basic scheme shows that the PDFs tested by test vectors 1 and 2 are P_1={$4-9-11-16, 4-9-12-13-17-22$} and P_2={$1-6-14, 12-17-22$}, the total PDFs tested as 4. But $12-17-22$ which is a SPDF, is a subset of the MPDF ($4-9-12-13-17-22$). It would be hence more appropriate to eliminate the MPDF ($4-9-12-13-17-22$) from the set of nr-PDFs tested by the test set P_{total} so that the coverage of the test set would be 3.



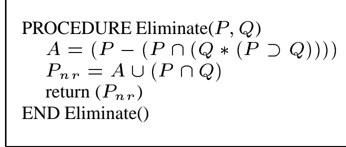
```
PROCEDURE (P ⊃ Q)
{
    IF (Q = 1) Return P
    IF (P = 0 or P = 1) Return 0
    IF (P = Q) Return 1
    R ← cache(P ⊂ Q)
    IF (R exists) return R
    x ← Q.top
    (P0, P1) ← Factors of P by x
    (Q0, Q1) ← Factors of Q by x
    IF (Q1 ≠ 0) R ← (P1 ⊃ Q1)
    IF (Q0 ≠ 0) R ← R ∪ (P1 ⊃ Q1)
    cache (P/Q) ← R
    Return R
}
```

Table 4: Containment Algorithm

In terms of delay fault testing, the coverage obtained by the basic scheme it is not very appropriate even though it is exact. A modification to the basic scheme is provided so as to eliminate the redundant MPDFs from the set of PDFs covered by the test set. A new operation, the **containment operation** (⊃) is introduced to identify the containment of elements of one set within another. Its implementation is similar to other existing ZBDD basic operators and is used to eliminate the redundant MPDFs. The example below illustrates the operation of the containment operator.

Example: Assume X_1 = {abd, abe, abg, cde, ceg, egh} be the set of partial PDFs tested by vector-1 and X_2 = {ab, ce} be the set of partial PDFs tested by vector-2. ($X_1 \supset X_2$) is obtained as:

($X_1 \supset X_2$) = ({abd, abe, abg, cde, ceg, egh}/{ab})
∪ ({abd, abe, abg, cde, ceg, egh}/{ce})
={d, e, g} ∪ {d, g}
={d, e, g}



```
PROCEDURE Eliminate(P, Q)
    A = (P − (P ∩ (Q ∗ (P ⊃ Q))))
    P_nr = A ∪ (P ∩ Q)
    return (P_nr)
END Eliminate()
```

Table 5: Procedure Eliminate

A basic algorithm describing the operation of the containment operator is described in Table 4. If P and Q are sets of PDFs then the non-redundant set of PDFs of P with respect to Q can be calculated using PROCEDURE Eliminate shown in Table 5. The procedure uses the containment operation to calculate the set of non-redundant PDFs of P (P_{nr}). The procedure to grade a test for nr-PDF is given in Table 6. The set of SPDF tested by the test set are stored as a ZBDD- $P(C_{g_i}^{SPDF})$, the set of MPDFs tested are stored as a ZBDD-$P(C_{g_i}^{MPDF})$. Performing the elimination process of $P(C_{g_i}^{MPDF})$ with respect to $P(C_{g_i}^{SPDF})$ will result in the set of nr-PDFs.

5 Overcoming Memory Problems

This section presents a technique to estimate the coverage of a given test set, in case the basic method fails to store the PDFs tested as a ZBDD. The method uses the framework proposed in [8], but the method is more complicated because the elimination of redundant MPDF is involved in the process of estimation of the coverage. An independent cut ζ is used to virtually partition the circuit into subcircuits [8]. Using arguments as in [8] that The number of new PDFs from the primary inputs to the primary output l_j and passes through the line l_i, tested by the nth vector is estimated as

$$PDF(l_i, l_j) = |C_i| \cdot |C_{i,j} \setminus P_{i,j}| + |C_i \setminus P_i| \cdot |C_{i,j} \cap P_{i,j}|$$

```
PROCEDURE nr-PDF_Grading(C, T)
    FOR every test (T_{i,1}, T_{i,2}) ∈ T,  i = 1, 2, ..., |T|   DO
        Simulate(C, (T_{i,1}, T_{i,2}))
        P(g_i) = Store_PDFs(C, Type, SPDF)
        FOR every Primary Output node g_i ∈ C DO
            P(C_{g_i}^{SPDF}) = Union(P(C_{g_i}^{SPDF}), P(g_i))
    FOR every test (T_{i,1}, T_{i,2}) ∈ T,  i = 1, 2, ..., |T|   DO
        Simulate(C, (T_{i,1}, T_{i,2}))
        P(g_i) = Store_PDFs(C, Type, MPDF)
        FOR every Primary Output node g_i ∈ C DO
            P(C_{g_i}^{MPDF}) = Union(P(C_{g_i}^{MPDF}), P(g_i))
    FOR every Primary Output node g_i ∈ C DO
        M = P(C_{g_i}^{MPDF})
        N = P(C_{g_i}^{SPDF})
        P(Non Redundant g_i) = Eliminate(M, N)
        Tested = Union(Tested, P(Non Redundant g_i))
END PDF_Grading()
```

Table 6: Procedure nr-PDF_Grading

Let C_i denote the non redundant set of partial PDFs from the primary input and terminating at the line l_i for the nth test vector. Let P_i be the non redundant set of partial PDFs originating from the primary inputs and terminating at the line l_i for the test vectors from 1 to $(n-1)$ respectively. $C_{i,j}$ and $P_{i,j}$ are the non-redundant sets of partial PDFs originating at line l_i and terminating at the line l_j for the nth test vectors and all the previous $(n-1)$ test vectors respectively.

Let $P_r^{P_i}$ be the set of redundant partial PDFs eliminated from P_i with respect to C_i. Let $P_r^{P_{i,j}}$ be the set of redundant partial PDFs eliminated from $P_{i,j}$ with respect to $C_{i,j}$. $P_r^{P_i}$ and $P_r^{P_{i,j}}$ have been used to estimate the coverage of the $(n-1)$ test vectors, which makes the method optimistic. The coverage of the $(n-1)$ vectors have to be re-calculated before the number of new PDFs covered by the nth test vector is added to the coverage of all the $(n-1)$ test vectors. Thus the number of new PDFs tested by the n test vectors is estimated as

$$PDF = \Sigma_{l_i \in \zeta_1} \Sigma_{l_j \in \zeta_2} PDF(l_i, l_j) + \Sigma_{i=1}^{n-1} PDF_{nr}$$

The last term $\Sigma_{i=1}^{n-1} PDF_{nr}$ is the coverage of the $(n-1)$ vectors calculated again by revisiting the first $(n-1)$ vectors to eliminate the PDF count by the $(n-1)$ test vectors. We use the redundant sets $P_r^{P_i}$ and $P_r^{P_{i,j}}$ identified by the nth test vectors is used for this purpose. The redundant set of partial PDFs $P_r^{P_i}$ and $P_r^{P_{i,j}}$ is eliminated from the sets C_i and $C_{i,j}$ before the calculation of the new PDFs identified by each of the $(n-1)$ vectors. The process of re-calculation ensures that the solution of the proposed method is always pessimistic.

6 Experimental Results

The experiments were run on a 750MHz SUN Blade-1000 workstation with 1GB RAM. Table 7 and Table 8 report the number of nr-PDFs covered by test sets generated using the methods proposed in [4] and [5]. The test sets from [4] and [5] were chosen for expermentation as they provably generate tests with very high coverage for path delay faults than most of the existing methods. Results shown in Table 7 is the coverage obtained for test set generated by [5]. Column 2 in the table represents the size of the test set, Column 3 represents the number of MPDFs covered by the test set, which also includes the redundant PDFs. Column 4 represents the number of non-redundant PDFs which is the sum of SPDFs and non-redundant MPDFs.

Circuit	Vectors	MPDF	nr-PDF	Time (min)
C1355	389,400	71,457	70,054	8.4
C1908	218,964	106,980	99,079	23.2
C2670	211,725	85,647	16,509	21.9
C3540	179,589	1,214,926	1,102,638	96.35
C5315	655,776	431,478	91,017	151.22
C6288	714,955	422,810	375,594	199.71
C7552	540,597	540,597	68,389	162.9

Table 7: Coverage using Test Sets from [5]

The methods of [1] and [2] failed (either due to memory overflow problem or internal data structure problem) for test sets with more than 25,000 vectors, even though it targets only SPDFs. The results presented here shows the superiority of the proposed method when compared to any existing method as the method does not depend on the number of vectors in the test set. The method is expected to fail only when the coverage of the test set is extremely high and makes the system run out of memory. The PDFs covered is not a monotonous function of the test sets unlike the technique in [8]. This behavior is caused by the elimination of the redundant MPDFs.

The results in Table 8, shows the results for the test patterns generated by the method proposed in [4]. It shows the comparison between the performance of the two method proposed here - coverage in the presence and in the absence of the cuts. Column 10 shows the number of nr-PDFs missed from being counted because of the pessimistic approach of the technique. Column 11 shows the reduction in memory (reduction in the number of nodes in the ZBDD) with respect to the method without the cuts. The method of using cuts was not experimented for the test set in Table 7 due to size of the test set. The results of Table 8 and 9 show the effectiveness of using the cuts, an average coverage loss of 5% is justified by an average memory reduction of 40%.

The proposed methods out performs any existing method even for coverage of SPDFs. But any existing ATPG tool does not have very high coverage to check for the memory limits of the proposed method. We use a fictitious setup to check the memory limits of the proposed method. In the fictitious setup, all PDFs are assumed to be sensitizable. So a transition at a primary input can be propagated through all sub graphs originating at that input. With such a setup, we generate a test set that can cover all the possible PDFs in the circuit.

Let I and O be the primary inputs and outputs of a combinational circuit C. The subgraph G for an I/O pair is derived and all the inputs of the internal gates are randomly assigned to be singly sensitized or co-sensitized together with some other input of the gate. The ZBDD representing all the PDFs of G is formed. The same procedure is performed for all possible I/O pairs, this defines an iteration of the approach. Similarly 100 iterations are performed for each benchmark with different random assignments for each iteration. After the 1st iteration, the number of PDFs covered are more than 100% of the total number of PDFs in the circuit due to the presence of the MPDFs. However with the increase in the number of the iterations, the number of PDFs reduces towards the total SPDFs of the circuit. The assignments on the input of the internal gates are done so that the ratio between the SPDFs to the MPDFs for a sub-graph for an iteration is 75:25. The results of the fictitious tests are shown in Table 9. The same setup with a ratio of 25:75 be-

Circuit	Vectors	MPDF	Basic Scheme			With Cuts				
			nr-PDFs Covered	ZBDD Nodes	Time (sec)	nr-PDFs Covered	ZBDD Nodes	nr-PDFs Lost(%)	Nodes Gained(%)	Time (sec)
C1355	2,501	3,236	3,204	1,905	5.6	3,034	1,508	5.3	20.8	125.6
C1908	1,443	5,254	4,730	2,124	17.23	4,537	1,746	4.08	17.8	370.34
C2670	1,883	6,263	4,867	2,218	46.19	4,697	1,791	3.49	19.25	1435.75
C3540	1,204	818,531	720,726	43,982	1323.44	691,806	23,562	4.01	46.42	7963.93
C5315	5,848	33,653	21,719	12,386	481.62	21,176	8,220	2.5	33.6	1678.22
C6288	1,548	67,686	46,019	20,889	1930.37	45,006	10,120	2.2	51.56	6734.53
C7552	6,842	54,671	37,409	14,781	1485.27	35,988	8,927	3.8	39.6	7732.18

Table 8: Coverage using Test Sets from [4]

Circuit	Vectors	MPDF	Without Cuts			With Cuts				
			nr-PDFs Covered	ZBDD Nodes	Time (min)	nr-PDFs Covered	ZBDD Nodes	nr-PDFs Lost(%)	Nodes Gained(%)	Time (min)
C1355	4,100	10,662,774	8,442,212	124,208	71.37	7,994,774	65,508	5.3	47.2	238.94
C1908	3,300	1,985,904	1,458,388	85,224	38.21	1,388,385	59,864	4.8	29.7	96.45
C2670	23,300	1,849,902	1,361,636	55,564	120.82	1,298,277	40,912	4.6	26.3	653.38
C3540	5,000	68,676,482	56,549,121	156,689	180.5	53,495,468	90,876	5.4	42.0	420.91
C5315	17,800	4,633,280	2,741,090	110,802	135.5	2,595,812	66,036	5.3	40.4	588.2
C6288	3,200	$1.98 * 10^{20}$	$1.98 * 10^{20}$	1,967,242	428.25	$1.98 * 10^{20}$	1,034,769	4.9	47.4	769.5
C7552	20,700	2,100,448	1,453,458	132,018	108.5	1,402,586	74,854	3.5	43.3	610.35

Table 9: Fictitious Test

tween the SPDFs and MPDFs was used and the system ran out of memory for the following benchmarks - C5225, C7552. Similar memory problems can be eliminated by using the method of cuts proposed in Section 5.

7 Conclusion

It has been shown that the fault coverage problem can be reduced to a set manipulation problem, which is equivalent to basic operations over zero suppressed binary decision diagrams. The experimental results and the results from [8] show the ability of the ZBDDs to store and manipulate large number of PDFs. The results also support the fact that the proposed method does not depend on the number of vectors in the test set unlike existing methods, and can store a very large number of PDFs.

The method can also be extended to diagnose delay failure in circuits. In the problem of diagnosis, two sets of PDFs are stored - one corresponding to the set that pass the delay test(termed as fault free set) and one for the failed test (termed as suspect set). The suspect set can be reduced by eliminating redundant PDFs. The method [9] uses a complicated data structure which is not efficient to store and represent MPDFs. The problem can be solved by using two ZBDDs - one to represent the fault free set of PDFs and one for the suspect set. The *eliminate* procedure proposed here can be used to eliminate the fault free PDFs from the suspect set. The resulting suspect set can be used to locate the cause of the delay fault.

References

[1] Deodhar J.V and Tragoudas .S, *Color Counting Technique for Fault Coverage*, Proc. International Symposium on Quality of Electronic Design, March 2001.

[2] Gharaybeh M. A., Bushnell M. L. and Agrawal V. D., *An exact non-enumerative fault simulator for path-delay faults*, Proc. of International Test Conference, 1996, pp. 276-285.

[3] Ke W. and Menon P.R., *Synthesis of Delay-Verifiable Combinational Circuits*, IEEE Trans. on Computers, vol. 44, pp.213-222, Feb. 1995.

[4] Michael M., and Tragoudas S., *ATPG for Path Delay Faults without Path Enumeration*, Proc. International Symposium on Quality of Electronic Design, March 2001.

[5] Michael M., and Tragoudas S., *ATPG for Path Delay Faults at Functional Level*, to appear in ACM Trans. on Design Automation of Electronic Systems, Vol. 7, No. 1, Jan. 2002.

[6] Minato S-I., *Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems*, Proc. of Design Automation Conference, 1993, pp. 272-277.

[7] Minato S-I., *Calculation of Unate Cube Set Algebra Using Zero-Suppressed BDDs*, Proc. of Design Automation Conference, 1994, pp. 420-424.

[8] Padmanaban S., Michael M. and Tragoudas S., *Exact Path Delay Grading with Fundamental BDD Operations*, Proc. of International Test Conference, 2001, pp.642-651.

[9] Pant P., Yuan .C.H., Gupta. S.K. and Chatterjee A., *Path Delay Fault Diagnosis in Combinational Cicuits with Implicit Fault Enumeration*, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, no. 10, Oct. 2001,pp. 1226-1235.

[10] Tragoudas S. and Denny N., *Testing of Path Delay Faults using Test Points*, Proc. of IEEE Symposium on Defect and Fault Tolerance in VLSI Systems, Nov. 1999, pp.86-94.