Operations on Multiple Transition Faults without Enumeration

Theodoros Toulas[1,a] and Spyros Tragoudas[2]

[1]*Synopsys Inc, Mountain View, USA*
[2]*Southern Illinois University, Carbondale, USA*

Abstract. The multiple transition fault model has been used to represent alternative defective gate combinations in the circuit. However, the number of faults is very large even of modest size circuits and therefore the defective configuration may not be considered. It is shown that multiple transition faults can be stored compactly in Binary Decision Diagrams. Furthermore, important operations for identifying the location of failures are implemented without fault enumeration. Experimental results on some of the largest ISCAS'85, ISCAS'89. and ITC'99 benchmarks demonstrate the scalability of the proposed method.

1 Introduction

With the technology moving to nanometer regime, the task of testing and diagnosing failures in integrated circuits (ICs) has become very demanding. Reductions on the area of the chip combined with increase in the number of transistors can cause delay defects.

The process of identifying the location of failures in an IC is guided by a set of possible fault sites [1]. This process is guided by a set of possible fault sites in the IC called suspects. Methods like [2]-[3] are restricted to a single defect location. This is very unlikely to be the case in deep submicron.

The path delay fault model is used to delay defects in deep submicron since it allows for many small delay defects to be distributed along a failing path [4]-[5]. Although effective in testing, its use in identifying the defective locations is limited [6]-[8]. The transition fault model [9] is used, but the authors assume that only one fault is being generated at each node. This is rarely the case in deep submicron. This may turn out to be a restriction that may misguide the identification of the location of failures.

This paper uses the multiple transition fault model (MTF) along sensitized paths to represent alternative defective gate combinations in the circuit. The number of MTFs is much higher than the number of PDFs. Transition fault based approaches in [10]-[11] do not scale well to the complexity of this model.

Fault implicit algorithms on appropriate data structure are used in this paper, to cope with the such scalability challenges. In this paper, we use the term test vector to denote a pair of input test patterns. The main contributions of this paper are:

1. Algorithms to compute and store in a non-fault enumeratively manner the number of MTFs that are excited by a bad vector (the initial suspect set) as well as by a set of good vectors (the good set).
2. Algorithms to prune the suspect set using the good set and guide the identification of defective locations.
3. A thorough experimental evaluation to demonstrate the impact of the proposed method.

This paper is organized as follows. Section 2 gives an overview of the proposed method. Section 3 describes the proposed method. Section 4 gives experimental results and finally Section 5 concludes.

2 Overview of the proposed method

We define the hit rate as the frequency of a gate in the MTFs that are sensitized by a bad vector (also called the bad set). The proposed tool will produce the hit rate of all gates. Gates with higher hit rates are most likely to have a defect and the silicon debugger should first examine those gates.

Consider the circuit in Figure 1. For simplicity in the exposition, we assume that all nodes have unit delay. Furthermore, we assume that the total delay defect observed (along any path) is at most one unit. Therefore, we assume that defective nodes reside along paths with the longest number of nodes.

In this example there are no inverting nodes, and therefore, all transition faults are rising. For brevity, in the following we do not explicitly indicate that the faults are rising. In Figure 1, the bad vector T1 = {10100,11101} sensitizes rising transitions along paths P1 = {1-5-7-10-12} and P2 = {4-6-9-11-12}. MTFs occur along these paths and a delay of unit 1 is observed at gate 12. A possible MTF is {1,5}, where each number indicates the gate id. This MTF may occur because both gates belong to the same path and the sum of potential

[a] theodoros.toulas@synopsys.com

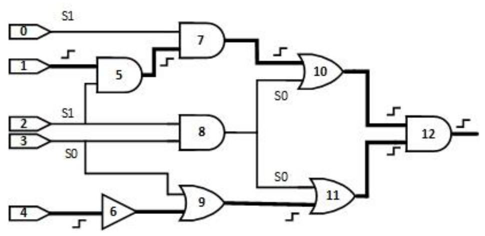
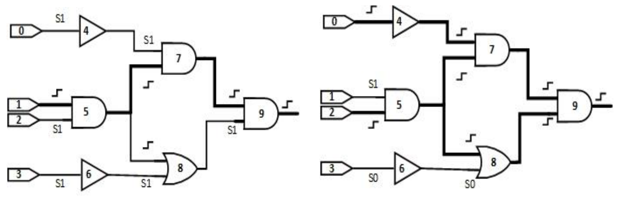


Figure 1. A long sensitized pdf along which MTFs are generated.



(a) Bad Vector (b) Good Vector

Figure 2. Pruning MTFs using good vectors.

defects associated to these gates may sum up to 1 unit. Another MTF is {1,4,5,7,9,10}. In this MTF possible delays on nodes 4 and 9 (which are on the same path) may sum up to 1 unit and possible delays on nodes {1,5,7,10} (which are on another path) may also sum up to 1 unit of defect.

In this example, MTF {1,4} may occur if one unit of defect is lumped on gate 1 and also one unit of defect is on gate 4. (Two gates are in different paths). The total number of MTF in this example is 511. In general, the total number of MTFs grows super-exponentially to the number of gates on the sensitized PDFs. It is a huge number even for a small number of paths. This paper presents algorithms that utilize the data structure in [12]. It is shown that is capable of storing very compactly a huge number of MTFs.

In Figure 1, the rising transitions faults at gates 1,4,5,6,7,9,10,11 and 12 have 256 hit rate in the total MTF initial suspect set $MTF_{initial}$. Therefore, there is no indication on which gate should be examined first for defects. The process may be assisted by good vectors. We will show a methodology on how to remove MTF from a suspect set and change the hit rate of fault sites in $MTF_{initial}$.

The example in Figure 2 illustrates the pruning process. The bad vector T_{bad} = {1011, 1111} sensitizes failing path Pf (1-5-7-9). Figure 2(b) shows a good vector T_{good} = {0100, 1110} that sensitizes paths P1 = {0-4-7-9}, P2 = {2-5-7-9} and P3 = {2-5-8-9}.

We define set $MTF_{initial}$ to be the union of all possible suspect MTFs. In the example of Fig 2 $MTF_{initial}$ = {{7}, {5}, {5,7}, {4}, {4,7}, {4,5}, {4,5,7}, {1}, {1,7}, {1,5}, {1,5,7}, {1,4}, {1,4,7}, {1,4,5}, {1,4,5,7}}. Each gate has same number of appearances in the $MTF_{initial}$.

The good vector sensitizes two long PDFs. Therefore, we include all possible MTF combinations along these PDFs. The union of all MTFs generated from the good vector in Figure 2(b) is the set MTF_{good}. Set MTF_{good} = {{0}, {4}, {0,4}, {2}, {5}, {2,5}, {0,2}, {0,5}, {0,2,5}, {2,4}, {4,5}, {2,4,5}, {0,2,4}, {0,4}, {0,2,4,5}, {2,8}, {5,8}, {2,5,8}, {0,7}, {4,7}, {0,4,7}, {2,7}, {5,7}, {2,5,7}, {0,2,7}, {0,5,7}, {0,2,5,7}, {2,4,7}, {4,5,7}, {2,4,5,7}, {0,2,4,7}, {0,4,7}), {0,2,4,5,7}, {7}, {0,8}, {4,8},{0,4,8}, {0,2,8}, {0,5,8}, {0,2,5,8}, {2,4,8}, {4,5,8}, {2,4,5,8}, {0,2,4,8}, {0,4,8}, {0,2,4,5,8}, {0,7,8}, {4,7,8}, {0,4,7,8}, {2,7,8}, {5,7,8}, {2,5,7,8}, {0,2,7,8}, {0,5,7,8}, {0,2,5,7,8}, {2,4,7,8}, {4,5,7,8}, {2,4,5,7,8}, {0,2,4,7,8}, {0,4,7,8}, {0,2,4,5,7,8}, {7,8}, {8}, {0,8,9}, {4,8,9}, {0,4,8,9}, {2,9}, {5,9}, {2,5,9}, {0,2,8,9}, {0,5,8,9}, {0,2,5,8,9}, {2,4,8,9}, {4,5,8,9}, {2,4,5,8,9}, {0,2,4,8,9}, {0,4,8,9}, {0,2,4,5,8,9}, {0,7,8,9}, {4,7,8,9}, {0,4,7,8,9}, {2,7,8,9}, {5,7,8,9}, {2,5,7,8,9}, {0,2,7,8,9}, {0,5,7,8,9}, {0,2,5,7,8,9}, {2,4,7,8,9}, {4,5,7,8,9}, {2,4,5,7,8,9}, {0,2,4,7,8,9}, {0,4,7,8,9}, {0,2,4,5,7,8,9}, {7,8,9}, {8,9}, {9}}.

We prune from the $MTF_{initial}$ all the MTFs that are also present in the MTF_{good}. In that way we also modify the number of gate hit rate in the set $MTF_{initial}$. The initial suspect set $MTF_{initial}$ had 15 MTFs. After pruning the suspect set deduced to MTF_{final} = {{1}, {1,7}, {1,5}, {1,5,7}, {1,4}, {1,4,7}, {1,4,5}, {1,4,5,7}} and the total number of MTFs reduced from 15 to 8. The hit rate of gate 1 is four, which is the highest frequency. The proposed tool guides silicon debug by considering this fault site first.

3 Generation, storage and manipulation of faults without enumeration

The advantage of the deductive method [16] is that all sensitized MTF by a single test vector, can be generated with a single topological traversal of the circuit netlist. Figure 3 shows an example of MTF generation using the deductive method. Let a test vector be T = {01, 11}. At each gate, all possible MTF that may occur until that gate are stored in ZBDD data structure [12]-[14].

Our method constructs two main MTF sets. One set is generated for the good set and one for the bad set. For the bad set we generate MTF along robust and non-robust PDFs. For the good set only long robust and validatable non-robust PDFs are considered [5]. The generation rules at each gate are the same for bad and good sets. A gate it is marked if it belongs to an appropriate PDF. The MTF generation takes into consideration only marked gates.

MTFs are generated non-enumeratively using basic operations in ZBDDs. Table 1 shows some basic ZBDDs operations being used in our approach. (More details about ZBDDs operators can be found in [12] and [13].) We represent the MTFs as combinational sets in the ZBDD. In Figure 4 we give an example of the representation of the MTFs on the sensitized circuit path of Figure 4(a) to its ZBDD shown in Figure 4(b). The ZBDD is a directed acyclic graph. Node 0 in Figure 4(b) is called its root. Nodes with the same label correspond to the same gate. There exist two terminal nodes, terminal 1 and terminal 0. In this example, there is only one sensitized PDF by test vector T = {01101, 11101}. The rising MTFs are stored in a ZBDD. Each MTF is a ZBDD path from root node 0 to terminal node 1. The doted lines indicate the absence of that particular gate in the MTF. For example, ZBDD path (solid line) 0-5-7-8 represent the MTF {0,5,7,8} and ZBDD path ¯0-5-7-¯8 (where ¯ is the doted line) represents MTF {5,7}.

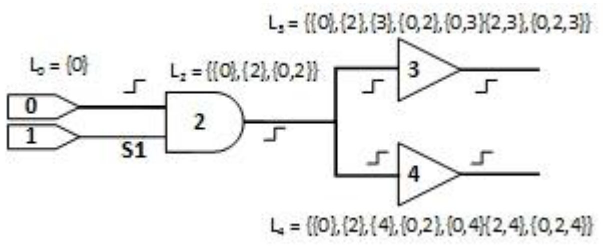

L₃ = {{0},{2},{3},{0,2},{0,3}{2,3},{0,2,3}}
L₀ = {0}
L₂ = {{0},{2},{0,2}}
0
1
S1
2
3
4
L₄ = {{0},{2},{4},{0,2},{0,4}{2,4},{0,2,4}}

Figure 3. List of multiple transition faults in deductive method

Table 1. Basic ZBDD Operations

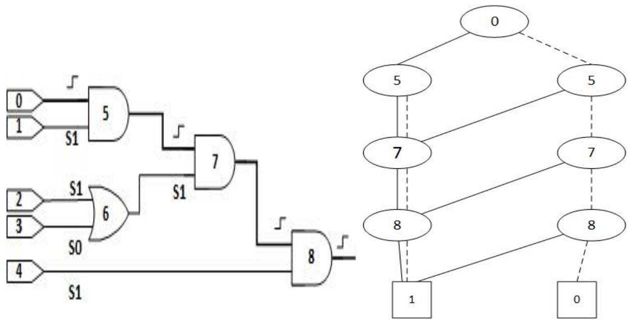
Operation	Symbol	Description
Union (A, B)	A ∪ B	The union operation
Prod (A, B)	A ∩ B	The unate product operation
Subset (A, B)	A ⊂ B	The subset operation
Diff (A, B)	A - B	The set difference operation

The generation of the MTF obeys the rules showed in Table 2 for the case of an AND gate. (We generate tables for all gates in the same manner.) At each simulation test vector T, we traverse the circuit in a topological manner. For each gate, rules from Table 2 are used to generate an MTF. Let us assume a 2-input AND gate whose inputs are labeled a, b and its outputs labeled c. When addressing the gate MTF that have propagated until inputs a and b are stored explicitly in a ZBDD. The table shows how to generate the MTFs at output c. They are generated in a non-fault enumerative manner using the operators in Table 1, and are kept as list MTF_C in the ZBDD.

We illustrate the method circuit shown in Figure 5. Let test set T consists of three test vectors T = {T1, T2, T3}. Assume that vector T1 = {1010, 1000} is a bad vector and PDF P1 = {2-6-8-9-10} is sensitized as shown in Figure 5(a). For brevity, we do not list the type of the transition at each MTF but we list the gate id. For the bad vector T1 the marked gates are 2, 6, 8, 9, 10. Therefore the initial suspect set $MTF_{initial}$ = {{2}, {8}, {2,8}, {9}, {2,9}, {8,9}, {2,8,9}, {10}, {2,10}, {8,10}, {2,8,10}, {9,10}, {2,9,10}, {8,9,10}, {2,8,9,10}, {6}, {{2,6}, {6,8}, {2,6,8}, {6,9}, {2,6,9}, {6,8,9}, {2,6,8,9}, {6,10}, {2,6,10}, {6,8,10}, {2,6,8,10}, {6,9,10}, {2,6,9,10}, {6,8,9,10}, {2,6,8,9,10}}.

Let test T2 and T3 be the good sets. Test vectors T2 = {1011, 0010} and T3 = {1011, 1110} sensitize PDFs P2 = {3-8-9-10} and P3 = {1-5-9-10}, respectively.

PDF P1 is sensitized robustly [15]-[16], and therefore has no defect. Gates 3, 8, 9, 10 are marked. The MTF set RMTF10 contains all the generated MTFs which guaranteed to be good (no defect). Set RMTF10 = {{3}, {8}, {3,8}, {9}, {3,9}, {8,9}, {3,8,9}, {10}, {3,10}, {8,10}, {3,8,10}, {9,10}, {3,9,10}, {8,9,10}, {3,8,9,10}}.



(a) A sensitized path delay fault (b) The MTF in ZBDD

Figure 4. MTF along a sensitized path stored in a ZBDD.

Table 2. Generating a list of MTF at the output of an AND gate using lists of MTF at its inputs

INPUTS		**OUTPUT**	**FAULT LIST**
a	b	c	MTF_C
0/1 or 1/0	S1	0/1 or 1/0	MTFa ∪ c ∪ (MTFa * c)
S1	0/1 or 1/0	0/1 or 1/0	MTFb ∪ c ∪ (MTFb * c)
0/1 or 1/0	1/0 or 0/1	1/0	MTFa ∪ MTFLb ∪ c ∪ (MTFa * c ∪ MTFb * ∪ MTFa * MTFb)
0/1	0/1	0/1	MTFa ∪ MTFb ∪ c ∪ (MTFa * c ∪ MTFb * c ∪ MTFa * MTFb)
1/0	1/0	1/0	MTFa * MTFb ∪ c ∪ ((MTFa * MTFb) * c)

Test vector T3 sensitizes a non-robust PDF P3 [15], [16]. We cannot guarantee that this PDF has no defect. However, P2 and P3 together they form a VNR test [5] and therefore, it is guaranteed that P3 has no defect. Gates 1, 3, 5, 8, 9, 10 are marked. MTF set RMTF10 contains all MTFs generated until that gate. RMTF10 = { {3}, {8}, {3,8}, {9}, {3,9}, {8,9}, {3,8,9}, {1}, {5}, {1,5}, {9}, {1,9}, {1,9}, {1,5,9}, {1,3}, {1,8}, {1,3,8}, {1,9}, {1,3,9}, {1,8,9}, {1,3,8,9}, {3,5}, {8,5}, {3,5,8}, {5,9}, {3,5,9}, {5,8,9}, {3,5,8,9}, {1,3,5}, {1,5,8}, {1,3,5,8}, {1,5,9}, {1,3,5,9}, {1,5,8,9}, {1,3,5,8,9}, {1,3}, {1,8}, {1,3,8}, {1,9}, {1,3,9}, {1,8,9}, {1,3,8,9},{1,3,5,8,9}, {10}, {3,10}, {8,10}, {3,8,10}, {9,10}, {3,9,10}, {8,9,10}, {3,8,9,10}, {1,10}, {5,10}, {1,5,10}, {9,10}, {1,9,10}, {1,9,10}, {1,5,9,10}, {1,3,10}, {1,8,10}, {1,3,8,10}, {1,9,10}, {1,3,9,10}, {1,8,9,10}, {1,3,8,9,10}, {3,5,10}, {5,8,10}, {3,5,8,10}, {5,9,10}, {3,5,9,10}, {5,8,9,10}, {3,5,8,9,10}, {1,3,5,10}, {1,5,8,10}, {1,3,5,8,10}, {1,5,9,10}, {1,3,5,9,10}, {1,5,8,9,10}, {1,3,5,8,9,10}, {1,3,10}, {1,8,10}, {1,3,8,10}, {1,9,10}, {1,3,9,10}, {1,8,9,10},{1,3,8,9,10}, {1,3,5,8,9,10}}. The good set MTF_{good} is the union of all MTFs generated by robust and VNR tests. Therefore, MTF_{good} = {VMTF10 ∪ RMTF10}.

Once both MTFs for bad and good tests are generated, a simple set difference ZBDD operation takes place. Namely, $MTF_{final} = MTF_{initial} - MTF_{good}$. With this operation the initial suspect set $MTF_{initial}$ reduces to MTF set MTF_{final} = {{2}, {2,8}, {9}, {2,9}, {2,8,9}, {2,10}, {2,8,10}, {2,9,10}, {2,8,9,10}, {6}, {{2,6}, {6,8}, {2,6,8}, {6,9}, {2,6,9}, {6,8,9}, {2,6,8,9}, {6,10}, {2,6,10}, {6,8,10}, {2,6,8,10}, {6,9,10}, {2,6,9,10}, {6,8,9,10}, {2,6,8,9,10} }. It contains all the possible defect sites for silicon debug.

For a single vector our method runs one traversal to generate all the possible MTFs. Moreover, simple ZBDD operations and algorithms, that are not described in this paper, can produce the transition fault sites with the highest ranking rate. The frequency of appearance of a gate inside the set MTF_{final} determines the order of inspecting during silicon debug.

The frequency of all gates can be identified by traversals on MTF_{final} [17]. The gate with the highest hit rate that is used for silicon debug.

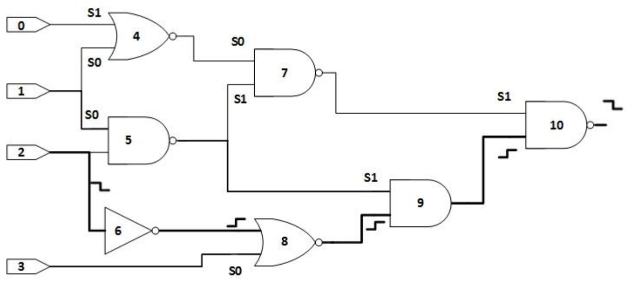
4 Experimental results

We experimented on a Unix machine with 8 GB memory and 2.40 GHz. The algorithms were implemented in C++. The ISCAS '85, ISCAS '89 and ITC '99 benchmarks were used as the circuits under test. In our experimentation, we had 3 bad vectors and 40,000 good vectors, for each benchmark. All gates were assumed to have unit delay. It was asserted that the total delay defect did not exceed two units of delay, and, therefore, MTFs generated only along the longest and second longest paths in each benchmark.

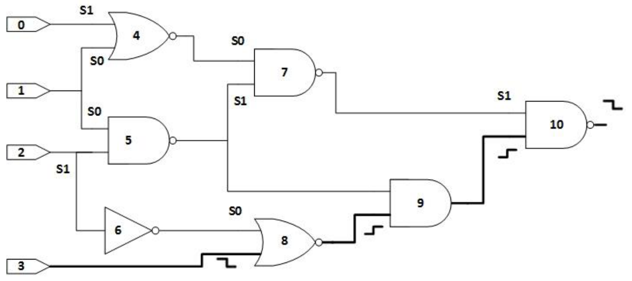
Table 3 reports our experimental results. Column 2 shows the size of $MTF_{initial}$, i.e., the number of MTF sensitized by bad vectors. Column 3 reports the total number of good MTF, i.e., the size of MTF_{good} consisting of MTFs from robustly and validatable non-robustly sensitized PDFs by the 40.000 good test vectors.

Column 4 shows the reduction in the suspect set using set MTF_{good}. In benchmarks such as s38584 and s9234 the number of total MTFs was huge and we could not enumerate them. Therefore, data structures such as arrays could not be used to store them efficiently. However, our method uses the ZBDD data structure and stored all the MTFs. The results for benchmarks s38584 and s9234 show the effectiveness of ZBDD data structure. Column 5 shows the percentage reduction in the suspect set MTF_{initia}l. Observe that the average reduction of the suspect set is 83.41%. In circuit c6288, the reduction is only 0,5% but the number of eliminated suspect MTFs is 1.030e+16, which is a huge number.

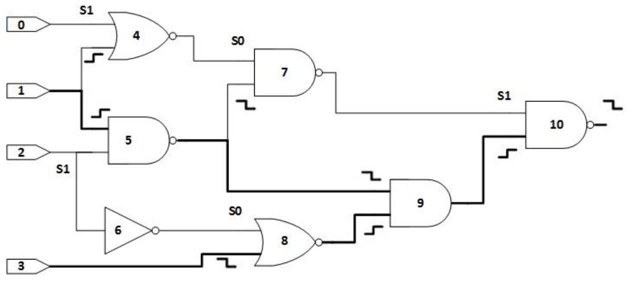
Columns 6 and 7 show the impact of the proposed approach in silicon debug. The numbers in the cells are the gate labels and the characters in parenthesis are the type of the transition ({r} for rising and {f} for failing). Column 6 reports for each benchmark the top two transition fault sites in order that appear more often in the set $MTF_{initial}$. Column 7 shows the same information for MTF_{final}. In bold fonts we list all gates that do not appear in column 6. The results show the impact of pruning for silicon debug.



(a) The bad vector



(b) The first good vector



(c)The second good vector

Figure 5. Illustration of the proposed method with a bad vector and two good vectors

the transition ({r} for rising and {f} for failing). Column 6 reports for each benchmark the top two transition fault sites in order that appear more often in the set $MTF_{initial}$. Column 7 shows the same information for MTF_{final}. In bold fonts we list all gates that do not appear in column 6. The results show the impact of pruning for silicon debug.

Column 8 lists the total CPU time (in seconds) by the proposed method. Columns 2, 3 and 4 show the total number of MTFs generated. It is clearly shown that the approach is fault implicit.

5 Conclusion

A method to guide silicon debug for delay defects has been proposed in order to guide silicon debug for delay defects. The collection of suspect MTFs has been generated implicitly by considering appropriately sensitized path delay faults for the bad and the good vectors. A method to effectively prune the initial suspect collection of MTF faults has been proposed and its impact has been evaluated experimentally.

Table 3. Suspect Set Reduction and recommended defective

Ben/s	**Number of MTF in MTF$_{initial}$**	**Number of MTF in MTF$_{good}$**	**Number of reduced Suspect MTF$_{good}$**	**(%) Reduction in suspect MTF**	**Top two hit rate gates before pruning**	**Top two hit rate gates after pruning**	**Time(sec)**
c880	1.00664e+08	4.60744e+13	144	99.99	41(r),43(r)	20(r),**88(f)**	103.89
c3540	364190	4.58564e+25	987	99.72	13,62(r)	62(r),**152(f)**	1092.42
c5315	571305	2.02657e+21	7013	98.77	112(r),114(r)	**682(f),10(r)**	1420.58
c6288	1.89e+18	2.77873e+16	1.88e+18	0.54	16(r),11(r)	16(r),11(r)	7369.65
c7550	882760	1.24182e+26	25804	97.07	5(r),233(f)	**2728(f),640(f)**	2205.43
s1196	34138	2.689e+11	4600	86.52	5(r),4(r)	**18(r)**,4(r)	108.45
s1296	132649	3.34936e+13	203	99.84	26(f),27(f)	**553(f),579(f)**	84.26
s1493	640407	6.84848e+10	112	99.98	293(r),294(r)	**25(f),30(f)**	145.33
s1494	309201	1.85235e+11	785	99.74	10(r),17(r)	**9(r)**,10(r)	52.11
s3330	131427	1.64227e+16	8250	93.72	166(r),151(f)	166(r),**147(r)**	2952.35
s9234	6.2068e+15	3.29304e+40	5.16581e+15	16.77	25(r),1578(f)	**1638(r),1639(r)**	2788.71
s13207	3.868561e+17	7.82535e+36	3.2021e+20	17.22	1236(r),1273(r)	**8409(r),8410(r)**	9496.72
s38984	1.29549e+10	1.07208e+29	2.13368e+06	99.98	1759(f)1760(f)	**785(r),1599(r)**	18854.98
b09	420046	1.2011e+16	48283	88.50	4(f),3(f)	3(f),4(f)	338.57
b10	180520	1.97929e+06	32231	82.14	15(r),21(r)	21(r),15(r)	83.96
b11	1.05872e+06	1.54903e+13	886441	82.18	8(r),9(r)	**768(r),769(f)**	122.22
b13	207592	2.14798e+07	38929	81.24	80(r),81(r)	**95(f),84(r)**	120.30
b15	688366	3.45974e+08	389457	43.42	39(f),40(f)	**73(f),42(r)**	5571.10
b17	557434	1.70268e+11	262342	52.93	14088(r),14089(r)	**14199(f),14200(f)**	19612.10
b20	479130	5.04538e+14	54747	88.57	48(f),49(f)	**50(f),51(r)**	12897.27
b21	140898	2.26338e+16	16156	88.53	8343(r),20781(r)	**21020(r),20783(r)**	12928.26
Average				**83.41%**			

References

1. S. Venkataraman and S. B. Drummonds, POIROT: A logic fault diagnosis tool and its applications, in Proc. Int. Test Conf., (Oct. 2000), pp. (253 –262).
2. K. Yang and K.-T. Cheng, Timing-reasoning-based delay fault diagnosis, in Proc. Des. Autom. Test Eur., (Mar. 2006), **vol. 1**, pp. 418 –423.
3. Z. Wang, M. Marek-Sadowska, K. H. Tsai, and J. Rajski, "Delay-fault diagnosis using timing information,"IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., **vol. 24**, no. 9, pp. 1315–1325), Sep. (2005)
4. P. Pant, Y. C. Hsu, S. K. Gupta, and A. Chatterjee, Path delay fault diagnosis in combinational circuits with implicit fault enumeration, IEEE Trans. Computer-Aided Design, **vol. 20**, pp. (1226–1235, Oct. 2001).
5. S. Padmanaban and S. Tragoudas, "An Implicit Path-Delay Fault Diagnosis Methodology,“ IEEE Trans. Computer-Aided Design Of Integrated Circuits, **Vol. 22,** No. 10, (Oct 2003)
6. A.M. Somashekar, S. Tragoudas, "Diagnosis of small delay defects arising due to manufacturing imperfections using path delay measurements," 14th International Symposium on Quality Electronic Design. pp.481–486, **4–6** (March 2013).
7. E.J Jang, J. Chung, and J.A. Abraham. Delay defect Diagnosis methodology using path delay measurements, In Proceedings of the 13th International Symposium on Integrated Circuits (ISIC), (2011)
8. V. J. Mehta,M. Marek-Sadowska, K. Tsai, J.Rajski, Timing-Aware Multiple-Delay-Fault Diagnosis, IEEE Trans. Computer-Aided Design,**vol. 28**, no. 2, Feb (2009)
9. J. A. Waicukauski, E. Lindbloom, B. Rosen, and V. Iyengar, Transition fault simulation, IEEE Des. Test. Comput., **vol. 4**, no. 2, pp. 32–38, (Apr. 1987).
10. J. B. Liu, A. Veneris, and S. Safarpour, "Diagnosing multiple transition faults in the absence of timing information," in Proc. 15th ACM Great Lakes Symp. VLSI, (2005), pp. 193–196.
11. Y. C. Lin, F. Lu, and K.-T. Cheng, "Multiple-fault diagnosis based on single-fault activation and single output observation," in Proc. Des. Autom. Test Eur., (Mar. 2006), pp. 1–6.
12. S. Padmanaban, M. Michael, and S. Tragoudas, "Exact path delay fault coverage with fundamental zero suppressed BDD operations," IEEE Trans. Computer- Aided Design, pp. 305 –316, (Mar. 2003)
13. S.-I. Minato, "Zero-suppressed BDD's for set manipulation in combinatorial problems," in Proc. Design Automation Conf., (1993), pp. 272 –277
14. S.-I. Minato, "Calculation of Unate Cube Set Algebra Using Zero-Suppressed BDDs," in Proc. Design Automation Conf., (1993), pp. 272–277
15. K. T. Cheng and H. C. Chen, "Classification and identification of nonrobust untestable path delay faults," IEEE Trans. Computer-Aided Design, **vol. 15**, pp. 845–853, (Aug. 1996)
16. Bushnell, M., Agrawal, Vishwani, "*Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*"
17. D.Jaramayan, "Optimization Techniques for Performance and Power Dissipation in Test and Validation," Ph.D Dissertation , ECE , Southern Illinois University, (May 2012)