Operations on Multiple Transition Faults without Enumeration

Theodoros Toulas1,a and Spyros Tragoudas2
1Synopsys Inc, Mountain View, USA
2Southern 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 $T_1 = \{10100,11101\}$ sensitizes rising transitions along paths $P_1 = \{1-5-7-10-12\}$ and $P_2 = \{4-6-9-11-12\}$. 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
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 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.

The example in Figure 2 illustrates the pruning process. The bad vector \(T_{\text{bad}} = \{01101, 11101\}\) sensitizes failing path Pf (1-5-7-9). Figure 2(b) shows a good vector \(T_{\text{good}} = \{0100, 1100\}\) that sensitizes paths P1 = \{0-4-7-9\}, P2 = \{2-3-5-7-9\} and P3 = \{2-5-7-9\}.

We define set MTF\(_{\text{initial}}\) to be the union of all possible suspect MTFs. In the example of Fig 2 MTF\(_{\text{initial}}\) = \{\{7\}, \{5\}, \{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\(_{\text{initial}}\).

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\(_{\text{good}}\). SET MTF\(_{\text{good}}\) = \{\{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\}, \{0,2,5,7,8\}, \{2,4,5,7,8\}, \{0,2,4,7,8\}, \{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\}, \{0,2,5,7,8,9\}, \{2,4,7,8,9\}, \{4,5,7,8,9\}, \{2,4,5,7,8,9\}, \{0,2,4,5,7,8,9\}, \{7,8,9\}, \{8,9\}, \{9\}\}.

We prune from the MTF\(_{\text{initial}}\) all the MTFs that are also present in the MTF\(_{\text{good}}\). In that way we also modify the number of gate hit rate in the set MTF\(_{\text{initial}}\). The initial suspect set MTF\(_{\text{initial}}\) had 15 MTFs. After pruning the suspect set deduced to MTF\(_{\text{final}}\) = \{\{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 \(T = \{011, 111\}\). 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 ZBDD 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 \(T = \{01101, 11001\}\). The rising MTFs are stored in a ZBDD. Each MTF is a ZBDD path from root node 0 to terminal node 1. The dotted 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 (dotted line) 0-5-7-8 (where \(\bar{z}\) is the dotted line) represents MTF \{5,7\}.

![Figure 1. A long sensitized pdf along which MTFs are generated.](image1)

![Figure 2. Pruning MTFs using good vectors.](image2)
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 MTFC in the ZBDD.

We illustrate the method circuit shown in Figure 5. Let test set $T$ consists of three test vectors $T = \{T_1, T_2, T_3\}$. Assume that vector $T_1 = \{1010, 1000\}$ is a bad vector and PDF $P_1 = \{2-6-8-9-10\}$ 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 $T_1$ the marked gates are $2, 6, 8, 9, 10$. Therefore, the initial suspect set $MTF_{init} = \{2, 6, 8, 9, 10\}$ is sensitized robustly [15]-[16]. We cannot guarantee that this PDF has no defect. However, $P_2$ and $P_3$ together they form a VNR test [5] and therefore, it is guaranteed that $P_3$ has no defect. Gates 1, 3, 5, 8, 9, 10 are marked. MTF set $RMTF_{10}$ contains all MTFs generated until that gate. $RMTF_{10} = \{1, 3, 5, 8, 9, 10\}$.

Let test set $T$ consists of three test vectors $T = \{T_1, T_2, T_3\}$. Assume that vector $T_1 = \{1010, 1000\}$ is a bad vector and PDF $P_1 = \{2-6-8-9-10\}$ 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 $T_1$ the marked gates are $2, 6, 8, 9, 10$. Therefore, the initial suspect set $MTF_{init} = \{2, 6, 8, 9, 10\}$ is sensitized robustly [15]-[16]. We cannot guarantee that this PDF has no defect. However, $P_2$ and $P_3$ together they form a VNR test [5] and therefore, it is guaranteed that $P_3$ has no defect. Gates 1, 3, 5, 8, 9, 10 are marked. MTF set $RMTF_{10}$ contains all MTFs generated until that gate. $RMTF_{10} = \{1, 3, 5, 8, 9, 10\}$.

Let test set $T$ consists of three test vectors $T = \{T_1, T_2, T_3\}$. Assume that vector $T_1 = \{1010, 1000\}$ is a bad vector and PDF $P_1 = \{2-6-8-9-10\}$ 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 $T_1$ the marked gates are $2, 6, 8, 9, 10$. Therefore, the initial suspect set $MTF_{init} = \{2, 6, 8, 9, 10\}$ is sensitized robustly [15]-[16]. We cannot guarantee that this PDF has no defect. However, $P_2$ and $P_3$ together they form a VNR test [5] and therefore, it is guaranteed that $P_3$ has no defect. Gates 1, 3, 5, 8, 9, 10 are marked. MTF set $RMTF_{10}$ contains all MTFs generated until that gate. $RMTF_{10} = \{1, 3, 5, 8, 9, 10\}$.
Once both MTFs for bad and good tests are generated, a simple set difference ZBDD operation takes place. Namely, $\text{MTF}_{\text{final}} = \text{MTF}_{\text{initial}} - \text{MTF}_{\text{good}}$. With this operation the initial suspect set $\text{MTF}_{\text{initial}}$ reduces to $\text{MTF}_{\text{final}} = \{\{2\}, \{2,8\}, \{9\}, \{2,9\}, \{2,8,9\}, \{10\}, \{2,9,10\}, \{2,8,9,10\}, \{6\}, \{2,6\}, \{6,8\}, \{2,6,8\}, \{6,8,9\}, \{6,9\}, \{2,6,8,9\}, \{6,9,10\}, \{2,6,9,10\}, \{6,8,9,10\}, \{2,6,8,9,10\}\}$. 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 $\text{MTF}_{\text{final}}$ determines the order of inspecting during silicon debug.

The frequency of all gates can be identified by traversals on $\text{MTF}_{\text{final}}$ [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 $\text{MTF}_{\text{initial}}$, i.e., the number of MTF sensitized by bad vectors. Column 3 reports the total number of good MTF, i.e., the size of $\text{MTF}_{\text{good}}$ 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 $\text{MTF}_{\text{good}}$. 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 $\text{MTF}_{\text{final}}$. 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 $\text{MTF}_{\text{initial}}$. Column 7 shows the same information for $\text{MTF}_{\text{final}}$. 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>
<thead>
<tr>
<th>Ben/s</th>
<th>Number of MTF in MTF_{initial}</th>
<th>Number of MTF in MTF_{good}</th>
<th>Number of reduced MTF_{good}</th>
<th>% Reduction in suspect MTF</th>
<th>Top two hit rate gaps before pruning</th>
<th>Top two hit rate gaps after pruning</th>
<th>Time(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>c880</td>
<td>1.00664e+08</td>
<td>4.60744e+13</td>
<td>144</td>
<td>99.99</td>
<td>41(r), 43(r)</td>
<td>20(r), 88(f)</td>
<td>103.89</td>
</tr>
<tr>
<td>c3540</td>
<td>364190</td>
<td>4.58564e+25</td>
<td>987</td>
<td>99.72</td>
<td>13,62(r)</td>
<td>62(r), 152(f)</td>
<td>1092.42</td>
</tr>
<tr>
<td>c5315</td>
<td>571305</td>
<td>2.02657e+21</td>
<td>7013</td>
<td>98.77</td>
<td>112(r), 114(r)</td>
<td>682(f), 10(r)</td>
<td>1420.58</td>
</tr>
<tr>
<td>c6288</td>
<td>1.89e+18</td>
<td>2.77873e+16</td>
<td>1.8e+18</td>
<td>0.54</td>
<td>16(r), 11(r)</td>
<td>16(r), 11(r)</td>
<td>7369.65</td>
</tr>
<tr>
<td>c7550</td>
<td>882760</td>
<td>1.24182e+26</td>
<td>25804</td>
<td>97.07</td>
<td>5(r), 233(f)</td>
<td>2728(f), 640(f)</td>
<td>2205.43</td>
</tr>
<tr>
<td>s1196</td>
<td>34138</td>
<td>2.689e+11</td>
<td>4600</td>
<td>86.52</td>
<td>5(r), 4(r)</td>
<td>18(r), 4(r)</td>
<td>108.45</td>
</tr>
<tr>
<td>s1296</td>
<td>132649</td>
<td>3.34936e+13</td>
<td>203</td>
<td>99.84</td>
<td>26(f), 27(f)</td>
<td>553(f), 579(f)</td>
<td>84.26</td>
</tr>
<tr>
<td>s1493</td>
<td>640407</td>
<td>6.84848e+10</td>
<td>112</td>
<td>99.98</td>
<td>293(r), 294(r)</td>
<td>25(f), 30(f)</td>
<td>145.33</td>
</tr>
<tr>
<td>s1494</td>
<td>309201</td>
<td>1.85235e+11</td>
<td>785</td>
<td>99.74</td>
<td>10(r), 17(r)</td>
<td>9(r), 10(r)</td>
<td>52.11</td>
</tr>
<tr>
<td>s3330</td>
<td>131427</td>
<td>1.64227e+16</td>
<td>8250</td>
<td>93.72</td>
<td>166(r), 151(f)</td>
<td>166(r), 147(r)</td>
<td>2952.35</td>
</tr>
<tr>
<td>s9234</td>
<td>6.2068e+15</td>
<td>3.29304e+40</td>
<td>5.16581e+15</td>
<td>16.77</td>
<td>25(r), 1578(f)</td>
<td>1638(r), 1639(f)</td>
<td>2788.71</td>
</tr>
<tr>
<td>s13207</td>
<td>3.868561e+17</td>
<td>7.82535e+36</td>
<td>3.2021e+20</td>
<td>17.22</td>
<td>1236(r), 1273(r)</td>
<td>8409(r), 8410(r)</td>
<td>9496.72</td>
</tr>
<tr>
<td>s38984</td>
<td>1.29549e+10</td>
<td>1.07280e+29</td>
<td>2.13368e+06</td>
<td>99.98</td>
<td>1759(r), 1760(f)</td>
<td>785(r), 1599(r)</td>
<td>18854.98</td>
</tr>
<tr>
<td>b09</td>
<td>420486</td>
<td>1.2011e+16</td>
<td>48283</td>
<td>88.50</td>
<td>4(f), 3(f)</td>
<td>3(f), 4(f)</td>
<td>338.57</td>
</tr>
<tr>
<td>b10</td>
<td>180520</td>
<td>1.97929e+06</td>
<td>32321</td>
<td>82.14</td>
<td>15(r), 21(r)</td>
<td>21(r), 15(r)</td>
<td>83.96</td>
</tr>
<tr>
<td>b11</td>
<td>1.05872e+06</td>
<td>1.54903e+13</td>
<td>886441</td>
<td>82.18</td>
<td>8(r), 9(r)</td>
<td>768(r), 769(f)</td>
<td>122.22</td>
</tr>
<tr>
<td>b13</td>
<td>207592</td>
<td>2.1479e+07</td>
<td>38929</td>
<td>81.24</td>
<td>80(r), 81(r)</td>
<td>95(f), 84(r)</td>
<td>120.30</td>
</tr>
<tr>
<td>b15</td>
<td>688366</td>
<td>3.45974e+08</td>
<td>389457</td>
<td>43.42</td>
<td>39(0), 40(f)</td>
<td>73(f), 42(r)</td>
<td>5571.10</td>
</tr>
<tr>
<td>b17</td>
<td>557434</td>
<td>1.70268e+11</td>
<td>262342</td>
<td>52.93</td>
<td>14088(r), 14089(r)</td>
<td>14199(f), 14200(f)</td>
<td>19612.10</td>
</tr>
<tr>
<td>b20</td>
<td>479130</td>
<td>5.04538e+14</td>
<td>54747</td>
<td>88.57</td>
<td>48(f), 49(f)</td>
<td>50(f), 51(r)</td>
<td>12897.27</td>
</tr>
<tr>
<td>b21</td>
<td>140898</td>
<td>2.26338e+16</td>
<td>16156</td>
<td>88.53</td>
<td>8343(r), 20781(r)</td>
<td>2120(r), 20783(r)</td>
<td>12928.26</td>
</tr>
</tbody>
</table>

Average 83.41%

References