A Deterministic Approach for Diagnosis Test Generation - Further Optimizations

Pavlinka Goranova Radoyska

Abstract: In the paper “A Deterministic Approach for Diagnosis Test Generation” was presented algorithm for diagnosis test pattern generation with polynomial complexity and deterministic nature. In those paper are given some optimizations and experimental results for the subalgorithms 3, 4 and 5, presented in that paper as well as the method for control the distinguishability.

Keywords: algorithm, digital circuits, test pattern generation, fault diagnosis, deterministic, z-set.

I. INTRODUCTION

The most of the circuit failure diagnostic methods are simulation based. These methods can be classified in two main classes: cause-effect [1], [2], [3], [4], [5] and effect-cause [6], [7], [8]. Well build test patterns are significant for methods efficiency. In the paper “A Deterministic Approach for Diagnosis Test Generation” has proposed an algorithm for diagnosis test pattern generation with deterministic nature. This algorithm is based on single stuck-at fault model. Its aims to build the better conditions for any of fault diagnosis methods. The algorithm consists of three main steps: (1) build the collection of all test patterns for every group of equivalent faults; (2) minimize the number of test patterns by merging the compatible test patterns; (3) minimize the number of test patterns by extracting the redundant test patterns. They are realized by five sub-algorithms.

The step (1) is performed by subalgorithm1 and subalgorithm2. Subalgorithm1 describes the steps for building the collections of input patterns \( L_v^i \), which can force the line \( i \) to level \( v \). Subalgorithm2 describes the steps for building the collections of test patterns \( T_a^v \), which can detect the fault \( f_a \). The test patterns in \( L_v^i \) and \( T_a^v \) are in 3-valent logic (‘0’, ‘1’ and ‘x’, which means “doesn’t matter”). Test pattern reduction is made after any calculation cycle and for every collection. The reduction is made by absorbing. If one test pattern becomes covering for every fault, which guarantee distinguishability and make the final diagnosis test set \( T_{res} \) by summarizing these collections and extracting the duplicated test patterns.

The rest of the paper is organized as follows. In Section 2, are given analysis, some experiment results and further optimizations for algorithm 3. In Section 3, are given analysis, some experiment results and further optimizations for algorithm 4. In Section 4, are given analysis, some experiment results and further optimizations for algorithm 5. In Section 4, are given dictionary based method for fault diagnosis. The experiments are performed on benchmark circuits: c17, 74182, 74283 and 74L85.

II. SUBALGORITHM3 ANALYSIS, EXPERIMENTS AND OPTIMIZATIONS

Subalgorithm2 builds the collection \( D = \{ \forall d_i = \leq f_i, t_i, FO_i > \} \), for every fault \( f_i \), test patterns \( t_i \), which detect this fault and corresponding fail output \( FO_i \). If fault effect for one test pattern and one fault can be observed on more then one output, for every output are made different \( d_i \).

Subalgorithm3. Improve distinguishability for the faults, observing on the same output and having the compatible test patterns. This algorithm follows the next steps:

1. For every primary output \( FO_i \) make D collections for every fault pairs \( f_a, f_b \),

\[
D_a = \{ \forall d_i = \leq f_i, t_i, FO_i > \} \quad \text{and} \quad D_b = \{ \forall d_i = \leq f_i, t_i, FO_i > \}
\]

2. If in \( D_a \) and \( D_b \) there is at least one incompatible test pattern, \( f_a \) and \( f_b \) are distinguish, take the other pair. Otherwise for the one of compatible pairs \( t_i, t_j \) change one
of ‘x’ levels to alternative value, so that \( t_i \) and \( t_j \) becomes incompatible.

There are two points for investigation and optimization at this algorithm: (1) test pattern election for compatible pairs and (2) a bit with ‘x’ value for change election. For fault election can be proposed three principles:

- pseudo-random (the first test pattern and the first fault in compatible pair);
- the test pattern with maximum number of ‘x’ values into two faults test pattern collections;
- the test pattern with minimum number of ‘x’ values into two faults test pattern collections.

For ‘x’ value for change election can be proposed three principles:

- Pseudo-random (the first ‘x’ with corresponding ‘0’ or ‘1’ level in the other test pattern in the pair).
- Reducing number of test patterns (the bit, which makes the test pattern suitable for absorption).
- Expanding number of test patterns (the bit, which makes the test pattern unique for the collection of test patterns, detecting this fault).

### Table 1. Experimental results for test pattern election

<table>
<thead>
<tr>
<th>c12</th>
<th>c14</th>
<th>c15</th>
<th>c17</th>
</tr>
</thead>
<tbody>
<tr>
<td>74182</td>
<td>74283</td>
<td>74L85</td>
<td></td>
</tr>
<tr>
<td>Number of Inputs</td>
<td>5</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Number of Faults</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Fault Dictionary size</td>
<td>22</td>
<td>83</td>
<td>128</td>
</tr>
<tr>
<td>Number of faults</td>
<td>240</td>
<td>1400</td>
<td>1600</td>
</tr>
<tr>
<td>Number of potentially undistinguished pairs</td>
<td>9</td>
<td>97</td>
<td>4</td>
</tr>
<tr>
<td>Undistinguished pairs percentage</td>
<td>3.75%</td>
<td>6.93%</td>
<td>0.25%</td>
</tr>
<tr>
<td>Number of diagnosis test patterns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- pseudo-random</td>
<td>15</td>
<td>48</td>
<td>49</td>
</tr>
<tr>
<td>- min</td>
<td>16</td>
<td>47</td>
<td>49</td>
</tr>
<tr>
<td>- max</td>
<td>16</td>
<td>49</td>
<td>49</td>
</tr>
</tbody>
</table>

In the table 1 are shown experimental results after applying the mentioned before three types of test pattern election: pseudo-random, maximum number of ‘x’ values and minimum number of ‘x’ values. The conclusion that can be made upon this experiment is: the order of test pattern election has not significant effect on the size of final diagnosis test pattern collection. It is due to the quite low percentage of the compatible pairs.

The experimental result on the methods for election the bit with ‘x’ value, give the similar results, due to the same considerations.

The effect of this algorithm is not so high but it is very important to guarantee the faults distinguishability. To keep low computation complexity, pseudo-random approach is preferred.

### III. Subalgorithm4 Analysis, Experiments and Optimizations

Subalgorithm 4. Test patterns for faults with different observation points merging. This algorithm follows the next steps:

1. For any primary output make the collection of unique test patterns \( T_a \) that can detect any fault on this observation point. If \( D \) collection for primary output \( a \) is \( D_a = \{ \forall d_i \in D : d_i = (t_i, F_O, t_i > F_O) \} \), then \( T_a = \{ \forall t_i \in D_a \} \), where \( a = 1 \div \text{outputs number} \).
2. Look for compatible test patterns \( t_i \in T_a \) and \( t_j \in T_b \) and replace them with common test pattern \( t_{com} \), according to the next rules:
   - \( 0 \& x = 0 \quad 0 \& x = 1 \quad 1 \& 0 = ? \quad (\text{conflict}) \)
   - \( x \& 0 = 0 \quad x \& 1 = 1 \quad 0 \& 1 = ? \quad (\text{conflict}) \)

If in any bit in the common test pattern there is a conflict, this pattern discards.

It is important how to choose the compatible patterns so that the resulting number of unique test patterns becomes minimal. Three different functions are written to optimize this sub-algorithm. In the first function are juxtaposed the collections with pseudo-random test pattern order (first come, first compared). In the second function the collections are ascending sort, based on filling (the number of non-‘x’ levels). In the third function the collections are also sorted, but the first collection is sorted in ascending and the second- in descending manner.

### Table 2. Experimental results for test pattern merging

<table>
<thead>
<tr>
<th>c12</th>
<th>c14</th>
<th>c15</th>
<th>c17</th>
</tr>
</thead>
<tbody>
<tr>
<td>74182</td>
<td>74283</td>
<td>74L85</td>
<td></td>
</tr>
<tr>
<td>Number of Inputs</td>
<td>5</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Number of Outputs</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Fault Dictionary size</td>
<td>22</td>
<td>83</td>
<td>128</td>
</tr>
<tr>
<td>Number of merged test patterns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- pseudo-random</td>
<td>149</td>
<td>1793</td>
<td>7261</td>
</tr>
<tr>
<td>- ascending sort</td>
<td>149</td>
<td>1891</td>
<td>7261</td>
</tr>
<tr>
<td>- ascending-descending sort</td>
<td>110</td>
<td>1469</td>
<td>7292</td>
</tr>
<tr>
<td>Number of diagnosis test patterns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- without merging</td>
<td>15</td>
<td>48</td>
<td>49</td>
</tr>
<tr>
<td>- pseudo-random</td>
<td>13</td>
<td>44</td>
<td>36</td>
</tr>
<tr>
<td>- ascending sort</td>
<td>12</td>
<td>40</td>
<td>35</td>
</tr>
<tr>
<td>- ascending-descending sort</td>
<td>15</td>
<td>43</td>
<td>36</td>
</tr>
</tbody>
</table>

In the table 2 are shown experimental results after applying the mentioned before three functions. Any of the function reduces the size of diagnosis test pattern collection. More over it reduce the number of unique \( c_1, FQ \) pairs, which reduce the operations in the subalgorithm5. The number of operations during the juxtaposition is proportional on \( n^2 \), where \( n \) is the average number of unique test patterns, which can detect any fault on given output.

From this table it can be seen that the test pattern merging is important procedure for reducing the size of the diagnosis test pattern collection. The best results are received while the incoming tests pattern collections are ascending sort. This is because the possibilities of merging are highest.
IV. SUBALGORITHM5 ANALYSIS, EXPERIMENTS AND OPTIMIZATIONS

Subalgorithm5. Build the final test pattern collection $T_{res}$ by collecting the minimal diagnosis test patterns for every fault. This algorithm follows the next steps:

1) For every $f_a \in F$ make

$$D_a = \{ \forall d \in D : d = < f_a, t, FO >, F_O >, f_a = f_a \}.$$ 

2) For every unique $< f_a, F_O >$ pair in the $D_a$ make the collections $F^a_i$ of detecting faults.

3) For every fault $f_a$ make an intersection $K = \bigcap F^a_i$, until in $K$ remains only $f_a$. For every $F^a_i$ add $t$ to the $T_{res}$.

4) Minimize the collection $T_{res}$ by extracting the duplicated test patterns.

The critical point of this algorithm is step 3) – the intersection making. The cardinality of final test patterns, that are possible to distinguish the fault $f_a$, is in strong dependence of the $F^a_i$ collections order.

Let have the fault $f_a$ and four $F^a_i$ collections with the same cardinality: $F^a_1 = \{ f_a, f_b, f_c \}$, $F^a_2 = \{ f_a, f_b, f_d \}$, $F^a_3 = \{ f_a, f_c, f_d \}$ and $F^a_4 = \{ f_a, f_c, f_e \}$. The intersection makes in three steps:

1) $K = F^a_1 \bigcap F^a_2 = \{ f_a, f_b, f_c \} \bigcap \{ f_a, f_b, f_d \} = \{ f_a, f_b \}$

2) $K = K \bigcap F^a_3 = \{ f_a, f_b \} \bigcap \{ f_a, f_c, f_d \} = \{ f_a \}$

3) $K = K \bigcap F^a_4 = \{ f_a \} \bigcap \{ f_a, f_c, f_e \} = \{ f_a \}$

It is seen that step 2) don’t change the members and cardinality of collection $K$. Hence it is unnecessary to add test pattern for $F^a_i$ collection in the $T_{res}$.

Let have the fault $f_a$ and three $F^a_i$ collections, respectively: $F^a_1 = \{ f_a, f_b \}$, $F^a_2 = \{ f_a, f_c \}$, $F^a_3 = \{ f_a, f_d \}$. If the intersection is made in the index order, in the process of intersection must take part all $F^a_i$ collections:

- $K = \bigcap F^a_i = \{ f_a, f_b \} \bigcap \{ f_a, f_c \} \bigcap \{ f_a, f_d \} = \{ f_a \}$

Respectively in the $T_{res}$ are added three test patterns. But if we change the order and start from collection $F^a_1$, only two test patterns will be added to the $T_{res}$:

- $K = \bigcap F^a_i = \{ f_a, f_b \} \bigcap \{ f_a, f_c \} = \{ f_a \}$

Hence it is suitable to order the $F^a_i$ collections in ascending order in respect of their cardinality and after that make the intersections.

This is the next question. If there is a test pattern $t_i$ with $F^a_i = \{ f_a \}$, when the final test pattern set will be the minimal: when includes the test patterns, such as $t_i$, which detect only one fault, or when includes more test patterns for every fault, which detect several faults.

In the Table 3 are shown experimental results after applying the mentioned before three functions. The best results are registered after descending sort the collections upon them size. For this approach if one test fails, it detects a lot of faults, but if one test passes, it excludes of fault candidate collection a lot of faults. This test pattern collection is suitable for manufacturing testing, because of its compactness and high detectable power.

<table>
<thead>
<tr>
<th>Number of diagnosis test patterns</th>
<th>c17</th>
<th>74182</th>
<th>74283</th>
<th>74L85</th>
</tr>
</thead>
<tbody>
<tr>
<td>- random</td>
<td>12</td>
<td>40</td>
<td>35</td>
<td>58</td>
</tr>
<tr>
<td>- sort – descending</td>
<td>11</td>
<td>29</td>
<td>34</td>
<td>53</td>
</tr>
<tr>
<td>- sort – ascending</td>
<td>12</td>
<td>35</td>
<td>37</td>
<td>53</td>
</tr>
</tbody>
</table>

V. MODULE FOR FAULT DIAGNOSIS

Test diagnosis dictionary $TDD = \{ p_0, p_1, ..., p_a \}$, where $p_i = < test \_ pattern, fail \_ output >$.

The module for fault diagnosis is written for controlling the effectiveness of generated diagnosis test pattern. This method is dictionary based and performs single stuck-at fault diagnosis. Its aims are to control the prerequisites for fault diagnostic in real processes. Diagnosis algorithm, realized in this module includes the next steps:

1) Build the collection of failing test pairs $Fail = \{ p_0, p_1, ..., p_a \}$.

2) Build the collection of pass test pairs $Pass = \{ p_0, p_1, ..., p_a \}$.

3) For every fault $f_i$ in fault dictionary do

   a) if flag_for_Adding = false and flag_for_Removing = false

   b) for every test pair $tp_j$ that detects $f_i$ do

      i) for every fail pair $p_j \in Fail$ do

         if $tp_j = p_j => flag_for_Adding = true$

         ii) for every fail pair $p_j \in Pass$ do

            if $tp_j = p_j => flag_for_Removing = true$

      c) if flag_for_Adding = true and flag_for_Removing = true then add $f_i$ to Candidates.

V. CONCLUSION AND FUTURE WORK

The next conclusions can be made about the subalgorithms after providing the optimizations and experiments.

The effect of subalgorithm3 (improve distinguishability between faults with the same observation point) on the size of final test pattern collection is not high, but it is important for improving distinguishability, checked in section 5. The pseudo-random approach for pattern and ‘x’ position selecting is preferred.
The test pattern merging, done in subalgorithm 4, is an important procedure for reducing the size of the diagnosis test pattern collection. The best results are received while the incoming test patterns are sorted in ascending direction.

For making the minimal diagnosis test pattern collection for every fault, done in subalgorithm 5, checking the collection, which are sorted in descending direction on their size, gave the best results.

The experimental results show that the fault distinguishability of the algorithm is in an acceptable level and make good circumstances for manufacturing tests and fault diagnosis.

The main disadvantages of this algorithm are two: it is memory huge and is single stuck-at fault oriented. The first disadvantage can be resolve by storing the temporary collections in the file or data base, which is in process. This algorithm is a first step for the most general algorithm, which aim is diagnosis test pattern generation for multiple faults with masking effect.

Acknowledgements

This work was supported by the TU-Sofia, project number № 091ni009-10.

REFERENCES