Abstract - We propose a dynamic test compaction procedure to generate high-quality test patterns for path delay faults. While the proposed procedure generates a compact two-pattern test set for paths selected by a path selection criterion, the generated test set would detect not only faults on the selected paths but also faults on many unselected paths. Hence both high test quality by detecting untargeted faults and test cost reduction by reducing test patterns can be achieved. Experimental results show that the proposed procedure could generate a compact test set that detect many untargeted path delay faults certainly, compared with the static test compaction method previously proposed in [15].

I. Introduction

Path delay fault model [1] is known as a powerful delay fault model to detect defects which lead to the timing violation. Since the path delay fault model localized as well as distributed excessive delays, test patterns generated for a path delay fault can detect most of other types of delay faults such as gate delay faults [2] on the path. However it is practically impossible to generate test patterns for all paths because there are a huge number of paths in a logic circuit. Therefore we need to select only a subset of paths and target it in test generation.

In path selection for test generation, it is important to select paths which are likely to be faulty, i.e. longer paths. Path selection criteria are categorized into two approaches, which are based on static timing analysis and statistical or dynamic timing analysis. The former selects structurally longest paths in the circuit, and are categorized into two approaches further. One approach is to select N longest paths in order of the path length. The length of any selected path is longer than the length of any unselected path. However since the selected paths may not be distributed all over the circuit and may be locally concentrated in a part of the circuit. The other approach is to select a set of paths which contains at least one of the longest paths through each line [3-8]. If we select paths based on this approach, the selected paths would be distributed all over the circuit. However the structurally longest paths may not be actual longest paths in a manufactured circuit due to process variation and/or circuit noise [9,10]. On the other hand, [11,12] tried to select actual longest paths by using statistical or dynamic timing analysis. However it is difficult to know exact delay distribution of manufactured circuits. In addition, the longest paths may be different for each manufactured circuit. Hence actual longest paths cannot be selected necessarily.

A test generation method proposed in [13,14] selects two subsets of paths. For paths in the primary set consisting of structurally longest paths, test patterns are guaranteed to be generated. For paths in the secondary set consisting of next longest paths, fault detection is not guaranteed, but it is considered so as to maximize accidental detection by the test patterns for paths in the primary set.

Recently an idea of test generation for a given set of path delay faults was proposed [15]. A test generation procedure based on the idea can bring an effective solution for two major problems in test generation for path delay faults, namely reducing the number of test patterns and achieving high fault coverage against process variation and noise. In test compaction, while each two-pattern test is generated for more than one fault in the given fault list as well as ordinary test compaction methods, the faults simultaneously detected are selected such that paths with the faults have many cross points. When crossing paths on which there is a common gate are tested simultaneously, non-target paths consisting of partial paths of the paths can be accidentally sensitized and faults on the non-targeted paths can be detected simultaneously too. Note that the accidentally detected faults may not be included in the fault list. Hence even if longer paths in a manufactured circuit are not structurally long paths and not included in the target fault list, the compact test set generated by the method would detect the longer untargeted faults. Although a test compaction procedure based on this idea was given in [15], it is a simple static test compaction [16] and hence the advantage of the idea has not been derived very much.

In this paper we propose a dynamic test compaction procedure based on the concept described in [15]. Dynamic compaction [16] is a well-known compaction procedure with a higher ability of compaction than static compaction. While the proposed procedure selects a primary target fault and secondary target faults from the fault list, the secondary targeted faults are selected such that there are many cross points with the primary fault or other secondary faults.
processed in the test pattern. Experimental results showed that the size of test sets generated by the proposed dynamic compaction procedure is about six times smaller than that of uncompacted test sets, and 1.5 times smaller than that generated by the static test compaction procedure described in [15]. Though final fault coverage of the test sets by the proposed procedure is lower a little due to the much smaller size of the test sets, certainly detected faults by specified inputs at test generation are increased.

This paper is organized as follows. In Section 2, we explain the method of test compaction for high-quality path delay testing described in [15]. In Section 3, we propose a dynamic test compaction procedure to detect many untargeted path delay faults. In Section 4, experimental results and discussions are given. Finally, we conclude this paper in Section 5.

II. Path Delay Tests with Process Variation Tolerance

In [15] an idea of test generation for path delay faults has been proposed. The compaction method tries to generate not only compact test patterns for a given set of path delay faults but also detect many path delay faults not included in the fault set. The basic idea of this method is to test path delay faults on crossing paths simultaneously. Fig. 1 illustrates a simple example. Assume that there are two paths, \( \text{PI}_1 \rightarrow g \rightarrow \text{PO}_2 \) and \( \text{PI}_2 \rightarrow g \rightarrow \text{PO}_1 \), which cross at a gate \( g \). If test patterns are generated for \( \text{PI}_1 \rightarrow g \rightarrow \text{PO}_2 \) and \( \text{PI}_2 \rightarrow g \rightarrow \text{PO}_1 \), separately, path delay faults on these paths may not be tested by the generated test patterns. On the other hand, if paths \( \text{PI}_1 \rightarrow g \rightarrow \text{PO}_2 \) and \( \text{PI}_2 \rightarrow g \rightarrow \text{PO}_1 \) are tested by a same test pattern \( t \), then \( t \) can test both \( \text{PI}_1 \rightarrow g \rightarrow \text{PO}_2 \) and \( \text{PI}_2 \rightarrow g \rightarrow \text{PO}_1 \) simultaneously. When faults on the two paths, \( \text{PI}_1 \rightarrow g \rightarrow \text{PO}_1 \) and \( \text{PI}_2 \rightarrow g \rightarrow \text{PO}_2 \), are included in the fault list, we have no need to generate additional test patterns for them. This situation would lead to efficient generation of a compact test set. Even when two paths, \( \text{PI}_1 \rightarrow g \rightarrow \text{PO}_1 \) and \( \text{PI}_2 \rightarrow g \rightarrow \text{PO}_2 \), are not included in the fault list, \( t \) would enhance fault coverage.

In order to simultaneously test crossing paths on which there is a common gate, the paths must be satisfied with following conditions:

1. The crossing paths have same transition at the common gate each other.
2. The transition at the common gate is from the controlling value [22] of the gate to the non-controlling value.

Since paths that are likely to be faulty should be tested, longer paths are selected according to a criterion. Test patterns generated would detect path delay faults on the selected paths certainly if they are testable. However, it is difficult to predict the delay size of a path in manufactured circuits because of process variation or noise. As a result, there remain paths that are more likely to be faulty than the selected ones and the generated test patterns might miss a fault on the paths. Test patterns generated by this method, however, would detect not only faults on the selected paths but also some faults on unselected paths. If the unselected paths whose faults are accidentally detected consist of parts of the selected paths, the length of the unselected paths is relatively long because the selected paths are long. Therefore the test patterns potentially compensate the detection of untargeted faults.

III. The proposed dynamic compaction

A. Outline of the Proposed Procedure

In this section we propose a dynamic compaction procedure to generate test patterns for a given set of path delay faults which are selected with a criterion of path selection.

First we pick up an undetected fault from the fault list and generate an initial test pattern with unspecified bits for the fault. We call the fault a primary fault. Next, another

\[ \text{PI}_1 \rightarrow g \rightarrow \text{PO}_2 \]

\[ \text{PI}_2 \rightarrow g \rightarrow \text{PO}_1 \]

Fig. 1: Two paths with one cross point.

\[ \text{XX} \]

\[ \text{X1} \]

\[ \text{XX} \]

\[ \text{X0} \]

\[ \text{I}_1 \]

\[ \text{I}_2 \]

\[ \text{I}_3 \]

\[ \text{I}_4 \]

\[ \text{O}_1 \]

\[ \text{O}_2 \]

\[ \text{O}_3 \]

\[ \text{O}_4 \]

\[ \text{O}_5 \]

(a) Initial path selection and test generation

\[ \text{XX} \]

\[ \text{X1} \]

\[ \text{XX} \]

\[ \text{X0} \]

\[ \text{g}_1 \]

\[ \text{I}_1 \]

\[ \text{I}_2 \]

\[ \text{I}_3 \]

\[ \text{I}_4 \]

\[ \text{O}_1 \]

\[ \text{O}_2 \]

\[ \text{O}_3 \]

\[ \text{O}_4 \]

\[ \text{O}_5 \]

(b) Dynamic compaction for \( p_2 \)

\[ \text{XX} \]

\[ \text{X1} \]

\[ \text{XX} \]

\[ \text{X0} \]

\[ \text{g}_1 \]

\[ \text{g}_2 \]

\[ \text{I}_1 \]

\[ \text{I}_2 \]

\[ \text{I}_3 \]

\[ \text{I}_4 \]

\[ \text{O}_1 \]

\[ \text{O}_2 \]

\[ \text{O}_3 \]

\[ \text{O}_4 \]

\[ \text{O}_5 \]

(c) Dynamic compaction for \( p_3 \)

Fig. 2: Outline of the proposed procedure
undetected fault, which is called a secondary fault, is picked up from the fault list. When the secondary fault is selected, we care whether the path of the secondary fault crosses the path of the primary fault or not. Test generation for the secondary fault tries assigning logic values to unspecified bits in the initial test pattern for the primary fault.

Fig. 2 illustrates the outline of the proposed dynamic compaction procedure. Suppose that path \( p_1 \), \( p_2 \) and \( p_3 \) is included in the given fault list. First, we pick up an undetected fault and generates an initial test pattern like Fig. 2 (a), where the path of the primary fault is path \( p_1 \), and the generated initial test pattern is \( (i_1, i_2, i_3, i_4, i_5) = (0, X, X, 0, 0) \), \( (1, 1, 0, 0, 0) \). Next we search an undetected fault in the fault list that satisfies the following conditions:

1. The fault is detected by filling unspecified bits of the initial test pattern.
2. The path of the fault produces more sensitized paths additionally by crossing the path of the primary fault than any other undetected fault in the fault list.

In Fig. 2(b), assume that the path \( p_2 \) is satisfied with above conditions, and that the new test pattern, \( (i_1, i_2, i_3, i_4, i_5) = (0, X, X, 0, X) \), \( (1, 1, 0, 0, 1) \), is generated by dynamic compaction. Two paths \( p_1 \) and \( p_2 \) are tested by the test pattern simultaneously, and more two additional paths, \( I_1-g_1-O_1 \), \( I_2-g_1-O_1 \) can be tested by the test pattern. After that we repeat to select secondary faults and to assign values for the detection of the secondary faults as long as there is a candidate of the secondary fault. Fig. 2(c) illustrates the repetition of the dynamic compaction. Path \( p_1 \) is processed after the dynamic compaction for \( p_2 \). In this case, more three additional paths can be tested by the test pattern, \( (i_1, i_2, i_3, i_4, i_5) = (0, X, X, 0, 1) \), \( (1, 1, 0, 0, 0) \). Finally, five paths in the given fault list can be tested by one test pattern, and five paths are tested by the test pattern additionally.

### B. Selection of a primary fault

In selecting a primary fault, it is important to select one which gives a chance to cross with other paths to be selected as secondary faults. Such a path as a primary fault would satisfy the conditions below.

1. The path has more off-inputs on each gate having the transition from the controlling value of the gate to the non-controlling value.
2. The path has more fan-out branches associated with the path.

![Fig. 3: Deference of the first selected path.](image)

Fig. 3 shows two examples. Assume that two sub-circuits in Fig. 3 are a part of a logic circuit, and the path \( p_1 \), \( (a-c-d) \) with rising transition at \( a \), and the path \( p_2 \), \( (g-i-k) \) with falling transition at \( g \) are included in the fault list. The path \( p_1 \) can cross a path including \( b-c-e \) with rising transition at \( b \) and a path \( b-c-f \) with rising transition at \( b \). If the three paths can be tested by the same test pattern, six paths can be tested additionally. On the other hand, the path \( p_2 \) can cross only a path \( h-i-k \) with falling transition at \( b \). No other paths can be crossed with the two paths. Therefore when we select a primary fault to generate an initial test pattern, we had better care the above two conditions.

### C. Detailed procedure

We implemented a dynamic test compaction procedure shown below. This procedure generates test patterns for faults in a given fault list with heuristics to detect as many untargeted path delay faults as possible by crossing paths in the fault list. Note that, in this procedure, \( P_{\text{init}} \) is the given fault list, and \( T_{\text{fix}} \) is the final test pattern set.

**Step1:** Select a primary fault \( p \) from \( P_{\text{init}} \), which is undetected and satisfied with two conditions described in the previous section. If \( P_{\text{init}} \) is empty, go to Step 8. Otherwise go to Step 2.

**Step2:** Remove \( p \) from \( P_{\text{init}} \), and go to Step 3.

**Step3:** If \( p \) is untestable, go to Step 1. Otherwise generates the initial test pattern \( t \) for \( p \), which has unspecified bits, and go to Step 4.

**Step4:** Search a fault \( q \) in \( P_{\text{init}} \), which is satisfied with below two conditions.

1. The test pattern for \( q \) can be generated by filling unspecified bits of \( t \).
2. The path of \( q \) produces most additional paths by crossing the path \( p \) and any other paths in \( P_{\text{init}} \).

If there is a path in \( P_{\text{init}} \), which are satisfied with these conditions, go to Step 5. Otherwise go to Step 7.

**Step5:** Generate a new test pattern \( t' \) for \( q \) by filling unspecified bits of \( t \). Remove \( q \) from \( P_{\text{init}} \), define a new test pattern \( t' \) as new \( t \), and go to step 6.

**Step6:** If \( P_{\text{init}} \) is not empty, return to Step 4. Otherwise go to Step 7.

**Step7:** Output the generated test pattern \( t \) to \( T_{\text{fix}} \), and go to Step 8.

**Step8:** If \( P_{\text{init}} \) is not empty, return to Step1. Otherwise this procedure is finished.
IV. Experimental Results and Discussions

We implemented the proposed procedure of dynamic test compaction using C programming language on a PC (Pentium III Xeon 2GHz, 4GB memory) and applied it to full scan version of ISCAS'89 benchmark circuits. We assume single path delay fault, and refer to untestable paths as non-robust untestable paths in the rest of this paper. However, discussions in this paper are valid for any other sensitizing conditions of path delay faults. We constructed a given fault list such that one of the longest potentially testable paths through each line of the circuit are included. In the construction of a given fault list, we use the partial path sensitization method to identify untestable paths. Note that the length of a path is determined by the number of logic gates on the path. In our experiments, we compared with uncompacted test sets, compacted test sets generated by static compaction procedure described in [15] and compacted test sets generated by the proposed dynamic compaction procedure.

Table 1 shows statistics of each circuit in terms of testable paths and selected paths. The columns of Table 1 give the total number of logical paths i.e. path delay faults, the number of testable paths which can be calculated by ATPG for all logical paths, and the number of selected paths and the number of testable paths out of the selected paths. The given fault list consists of the selected paths. In the selected paths some untestable paths existed except for s35932 because of the incompleteness of the partial path sensitization method in path selection. However most of potentially testable paths were testable.

Table 2 gives the number of generated test patterns. The first column shows circuit names. The second column shows the results of test generation without test compaction where each test pattern is generated for an undetected fault in the fault list and fault simulation is performed for the generated test pattern after random-filling for unspecified bits. The third column shows the result of static test compaction by ATPG for all logical paths, and the number of selected paths and the number of testable paths out of the selected paths. The fourth column shows the results of test generation without test compaction where each test pattern is generated for an undetected fault in the fault list and fault simulation is performed for the generated test pattern. The fifth column shows the result of static test compaction by ATPG for all logical paths, and the number of selected paths and the number of testable paths out of the selected paths. The sixth column shows the result of static test compaction by ATPG for all logical paths, and the number of testable paths out of the selected paths. The seventh column shows the number of detected path delay faults in the untargeted path delay faults for each test pattern. The eighth column shows the number of testable paths which can be calculated by ATPG for all logical paths, and the number of testable paths out of the selected paths. The third, fifth and seventh columns show the number of detected path delay faults in 10000 longest testable paths. This experiment brings out how generated test sets cover structurally longest paths. In this experiment, we calculated the number of detected path delay faults by ATPG for an uncompacted test set, a compacted test set generated by static compaction and a compacted test set generated by dynamic compaction. Accidentally detected path delay faults by random filling for unspecified bits are not included in these results. In Table 3, the second column shows the number of test pattern sets generated compared with the uncompacted test pattern sets. Therefore the generated test patterns can detect more path delay faults than that by static compaction. Hence we should generate test patterns using dynamic compaction method.

Table 4 shows the coverage of untargeted path delay faults in 10000 longest testable paths. This experiment brings out how generated test sets cover structurally longest paths. In this experiment, we calculated the number of detected path delay faults by ATPG for an uncompacted test set, a compacted test set generated by static compaction and a compacted test set generated by dynamic compaction. Accidentally detected path delay faults by random filling for unspecified bits are not included in these results. In Table 4, the second column shows the number of test pattern sets generated compared with the uncompacted test pattern sets. Therefore the generated test patterns can detect more path delay faults than that by static compaction. Hence we should generate test patterns using dynamic compaction method.

Table 1: The number of selected paths and testable paths.

<table>
<thead>
<tr>
<th>circuit</th>
<th>#total paths</th>
<th>#testable paths</th>
<th>#selected paths</th>
<th>#testable paths in selected paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>s5378</td>
<td>27,084</td>
<td>21,928</td>
<td>4,170</td>
<td>4,133</td>
</tr>
<tr>
<td>s9234</td>
<td>489,708</td>
<td>59,854</td>
<td>5,193</td>
<td>5,159</td>
</tr>
<tr>
<td>s13207</td>
<td>2,690,738</td>
<td>476,145</td>
<td>8,792</td>
<td>8,723</td>
</tr>
<tr>
<td>s15850</td>
<td>329,476,092</td>
<td>10,782,994</td>
<td>10,027</td>
<td>9,950</td>
</tr>
<tr>
<td>s35932</td>
<td>394,282</td>
<td>58,657</td>
<td>28,549</td>
<td>28,549</td>
</tr>
<tr>
<td>s38417</td>
<td>2,783,158</td>
<td>1,138,194</td>
<td>28,713</td>
<td>27,496</td>
</tr>
<tr>
<td>s38584</td>
<td>2,161,446</td>
<td>334,927</td>
<td>30,891</td>
<td>30,730</td>
</tr>
</tbody>
</table>

Table 2: The size of test pattern sets.

<table>
<thead>
<tr>
<th>circuit</th>
<th>uncompacted tests</th>
<th>compacted tests (static compaction)</th>
<th>compacted tests (dynamic compaction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>s5378</td>
<td>920</td>
<td>254</td>
<td>173</td>
</tr>
<tr>
<td>s9234</td>
<td>1287</td>
<td>429</td>
<td>296</td>
</tr>
<tr>
<td>s13207</td>
<td>1441</td>
<td>511</td>
<td>384</td>
</tr>
<tr>
<td>s15850</td>
<td>1855</td>
<td>922</td>
<td>456</td>
</tr>
<tr>
<td>s35932</td>
<td>257</td>
<td>32</td>
<td>24</td>
</tr>
<tr>
<td>s38417</td>
<td>6799</td>
<td>833</td>
<td>596</td>
</tr>
<tr>
<td>s38584</td>
<td>2803</td>
<td>522</td>
<td>347</td>
</tr>
</tbody>
</table>
uncompacted test patterns. Although the generated compacted test patterns by static compaction could certainly detect many untargeted path delay faults, the number of detected paths are less than that by dynamic compaction. From this result, since the compacted test patterns can detect many longer paths, even if critical paths are distributed by process variation or electrical noise, real critical path delay faults would detect by the compact test patterns. Therefore, to generate the compacted test patterns by the proposed method is to generate high quality compact test patterns for recent DSM circuits.

Table 5 shows the number of detected faults and fault efficiency of each test pattern set. In this experiment, unspecified bits of each test pattern are filled with 0 or 1 at random. Therefore detected faults include in faults accidentally detected by random-filling. From Table 5, we can observe that uncompacted test pattern sets have higher fault efficiency than the other compacted test sets. Since the number of uncompacted test pattern sets is larger than that of compacted test sets and uncompacted test sets would have many unspecified bits, by filling unspecified bits with 0 or 1 at random, accidentally detected faults would be increase.
On the other hand, compacted test pattern sets would have a small number of unspecified bits in itself. Even if certainly detected faults are increased by the proposed dynamic compaction method, the total number of detected faults is not increased so much. However these accidentally detected faults cannot guarantee the quality of generated test pattern, since they may not be detected by another random-filling for unspecified bits. To guarantee the quality of generated test pattern set, many faults should be detected certainly. Therefore generated test pattern sets by the proposed dynamic compaction have high quality to guarantee the circuit operation.

V. Conclusions

In this paper, we proposed a dynamic compaction procedure to test paths with cross points simultaneously so as to accidentally detect many faults which may not be included in the target fault list. The proposed procedure for path delay faults brought improvement of test quality in spite of reduction of test patterns. Experimental results showed that the proposed procedure could generate a compact two-pattern test set and it could detect many untargeted path delay faults efficiently. Our future work is to improve the heuristics algorithm to detect more certainly detected paths.

Acknowledgements

This work was supported by the New Energy and Industrial Technology Development Organization (NEDO).

References