An Efficient Scan Tree Design for Test Time Reduction

Y. Bonhomme¹ T. Yoneda¹ H. Fujiwara¹ P. Girard²

¹ Graduate School of Information Science, Nara Institute of Science and Technology
Email: {bonhomme, yoneda, fujiwara}@is.naist.jp

² LIRMM, UMR 5506 Université Montpellier II/CNRS, France
Email: girard@lirmm.fr

Abstract: We propose a new scan tree architecture for test application time reduction. This technique is based on a dynamic reconfiguration mode allowing one to reduce the dependence between the test set and the final scan tree architecture. The proposed method includes two different configuration modes: the scan tree mode and the single scan mode. The proposed method does not require any additional input or output. Experimental results show up to 95% of test application time saving and test data volume reduction in comparison with a single scan chain architecture.

Keywords: DfT, Scan Testing, Scan Tree

1. Introduction

The controllability and the observability of complex designs are nowadays guaranteed by DfT techniques. The full scan design is the most popular DfT technique [1] and it is very largely used in integrated circuits (ICs) or in System on chip (SoC) cores. Unfortunately, scan based architectures are expensive in test power consumption [2] and in test application time.

The test application time depends on the scan chain length. Indeed, the number of clock cycles required to scan in/out the test data is equal to the product of the number of test patterns by the scan chain length. The test sequence cannot be reduced without degrading the test quality. Thus, one solution to save test application time is to lessen the scan chain length.

At present, multiple scan chain design is often used to reduce the scan test shift time. This technique consists of splitting the scan chain in small scan sub-chains activated at the same time. The size of the different scan segments can be fixed [3,4] or variable [5,6]. Multiple scan chain design is now a standard [1] often used in ICs and in SoC cores (using the IEEE P1500 standard). The main drawback of this technique is the increase of the number of scan inputs and outputs dedicated to the test. Different techniques allow one to reduce the test application time with a low increase of the number of scan inputs and outputs [7-18].

Some solutions are based on test encoding [7-13]. The use of these techniques requires extra DfT logic. The hardware overhead of the decompressor is directly linked to the compression ratio.

Other solutions propose to reduce the test time without encoding process [14,15] and with a single scan input to drive several scan chains. The main drawback is a reduction of the circuit controllability. A dynamic reconfiguration can be used to improve the circuit controllability [15], but an additional test control input is then required.

Scan tree design is often used to deal with different scan DfT technique problems like test power [19] or test time [17,18]. The idea of scan tree design is that one scan cell drives other scan cells. This technique is interesting for test time reduction and it is a serial/parallel scan design with a single scan input. The same test data must be stored in different scan cells. The circuit controllability is reduced and this method requires a full compatibility between the parallel scan cells. So the effectiveness of a scan tree architecture depends on the correlation between the test data of the different scan cells. The first technique [17] proposes to reduce this dependence by increasing the number of don’t care values of the test sequence. But the solution effectiveness still depends on the circuit structure. The second technique [18] adopts a folding mode allowing one to increase the parallelism of the scan tree architecture. However, test sequence updating during the manufacturing phase but also the circuit time life, is impossible with both techniques.

In this paper, we propose an optimal scan tree architecture for test shift time reduction. The idea is to use a dynamic reconfiguration during the test application to switch from a scan tree mode to a single scan mode. A part of the test sequence is applied in scan tree mode to save test application time and the other part is applied in single scan mode. This procedure allows one to reduce the dependence between the scan tree architecture and the test sequence. With the proposed technique, it is possible to update the test sequence. This procedure
does not require additional test control input. A MISR is used for the circuit response compaction.

This paper is organized as follows. In section 2, a preliminary about full compatible scan tree architecture is presented and the proposed technique is described. In section 3, the scan tree generation of the proposed method is explained. Section 4 shows some experimental results for benchmark circuits and section 5 concludes the paper.

2. Proposed scan tree architecture

2.1 Preliminary (full compatible scan tree architecture)

Recently, two scan tree architectures [17,18] have been proposed to reduce the test application time. The basic principle of these techniques is described in figure 1. The idea is to store at the same time the similar test data in different scan cells. A computation of the test sequence is required to form groups of compatible scan cells that can receive the same test data.

Figure 1. Example of scan tree architecture generation

Figure 1.a presents an example of a single scan chain composed of 6 scan cells (FF₁ to FF₆) and the associated test set. From this data, groups of compatible scan cells are created (FF₁, FF₂ and FF₃), (FF₄ and FF₅) and (FF₆). Scan cells of the same group must receive the same test data. FF₂ and FF₃ can be placed in the same group as far as they receive the same test data, respectively 0X1 and 011. FF₆ is incompatible with all other scan cells. From these groups, scan tree architecture is generated and represented in figure 1.b. The test shift time saving corresponds to the ratio of the difference between L₁ and L₂ over L₁, where L₁ corresponds to the single scan chain length and L₂ to the scan tree length. Moreover, a reduction of the test data volume is obtained. This reduction is equivalent to the test shift time saving.

To generate scan tree architectures, full compatibility between the FFs of the same group is required. This full compatibility creates a high dependence between the scan tree architecture and the test sequence. To reduce this dependence, we propose a new scan tree architecture.

2.2 Proposed architecture

In this part, we present the proposed scan tree method. This method generates an optimal scan tree design with a dynamic reconfiguration during test application. The dynamic reconfiguration consists in switching between two modes. The first mode is the scan tree mode (ST mode) and the second one is the single scan mode (SS mode). The idea is to apply a part of the test sequence in ST mode and the other part in SS mode. This technique allows one to reduce the dependence between the scan tree architecture and the test sequence and to improve the test shift time reduction.

Figure 2. Proposed scan tree architecture

Figure 2 describes the proposed scan tree architecture. The switch functionality from ST mode to SS mode is carried out by a pattern recognition module, multiplexers and a flip-flop driven by the scan enable signal (in grey color in figure 2). The activation occurs when the last test pattern of the scan tree mode has been applied. This extra DFT logic has a low impact on the area overhead. A MISR is used for the response compression. Figure 3 presents the two configurations of the architecture: 3.a the ST mode and in 3.b the SS mode. In both cases, there is only one scan input and one scan output and a single test control input.

Figure 3. Two modes: a/ scan tree mode b/ single scan mode
Concerning design constraints (fanout limitation for the scan cells and routing), we shall try to consider them in further work. Clustering process already proposed in [20] may be used. The configuration flexibility of the proposed scan tree architecture should allow one to obtain a good trade-off between test time reduction and design constraints.

To design the scan tree architecture, it is necessary to find the optimal balance between the part of the test sequence to apply in ST mode and the other part to apply in SS mode. The next section presents the adequate scan tree architecture generation procedure.

3. Scan tree architecture generation

Scan tree architecture generation consists in finding the good balance between the part of the test sequence to apply in ST mode and the other part to apply in SS mode. The complexity of scan tree architecture generation to find the optimal solution depends on the number of test patterns and scan cells. This problem is NP-complete. The following section describes different heuristics to reduce the problem complexity. Section 3.1 presents the adopted solution of scan tree generation and section 3.2 details the solution to determine the part of the test sequence to apply in ST mode.

3.1 Scan tree generation

Scan tree generation consists of finding groups of compatible scan cells from a test set. We construct the incompatibility graph from the test set and adopt a vertex coloring process for the graph to solve the problem of finding optimal groups of compatible scan cells. In an incompatibility graph for a test, a vertex corresponds to a scan cell, and edge between two vertices exists if two scan cells corresponding to the vertices are incompatible in the test set.

Figure 4 presents two examples of scan tree generation. In both, the scan chain is composed of 6 scan cells. In figure 4.a, scan tree architecture is generated via a coloring graph from the test pattern \( t_1 \). The FF\(_i\) is incompatible with the FF\(_j\), FF\(_k\) and FF\(_l\). These incompatibilities are reported by edges between (FF\(_i\), FF\(_j\)), (FF\(_i\), FF\(_k\)) and (FF\(_j\), FF\(_k\)). The complexity of the coloring graph algorithm is \( O(n^2) \), \( n \) corresponding to the number of scan cells. Figure 4.b corresponds to the addition of the test pattern \( t_2 \) to the previous solution for \( t_1 \). The addition of the pattern \( t_2 \) increases the number of incompatibilities and coloring graph edges. In consequence, the length of the scan tree grows to 3 cells instead of 2. With this example, we show that the scan tree architecture has a high dependence with the test patterns. Test pattern selection must be achieved to determine the part of the test sequence to apply in ST mode.

![Figure 4. a/ scan tree generation from a test pattern and b/ scan tree generation from 2 test patterns](image)

Figure 5. Procedure of the test sequence reduction

3.2 Test pattern selection

The test pattern selection problem is reduced to Traveling Salesperson Problem (TSP [21]). It is not possible to find an exhaustive solution to determine the part of the test sequence that should be applied in ST mode. The complexity of the problem resolution depends on the number of test patterns. To reduce the complexity, a test sequence reduction procedure is applied. This procedure is described in figure 5. The first step is to compute the number of incompatibilities for each test pattern. The number of incompatibilities corresponds to the number of edges in the coloring graph. The second step consists in sorting the test sequence according to the number of incompatibilities. The last step
Table 1. Experimental results on ISCAS’89 benchmark circuits

<table>
<thead>
<tr>
<th>Benchmark reference</th>
<th># FFs</th>
<th># test patterns</th>
<th>FC (%)</th>
<th># clock cycles for single scan (1)</th>
<th># clock cycles for scan tree (2)</th>
<th># clock cycles for the proposed method (3)</th>
<th>test time saving (%) (1)(2)</th>
<th>test time saving (%) (1)(3)</th>
<th>computation time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1423</td>
<td>74</td>
<td>79</td>
<td>98.99</td>
<td>5,846</td>
<td>1,817</td>
<td>947</td>
<td>68.92</td>
<td>83.8</td>
<td>1</td>
</tr>
<tr>
<td>s5378</td>
<td>179</td>
<td>178</td>
<td>98.78</td>
<td>29,714</td>
<td>13,350</td>
<td>4,472</td>
<td>58.10</td>
<td>85.96</td>
<td>22</td>
</tr>
<tr>
<td>s9234</td>
<td>211</td>
<td>341</td>
<td>93.16</td>
<td>71,951</td>
<td>34,322</td>
<td>19,391</td>
<td>52.30</td>
<td>73.05</td>
<td>163</td>
</tr>
<tr>
<td>s13207</td>
<td>669</td>
<td>540</td>
<td>98.32</td>
<td>361,260</td>
<td>52,608</td>
<td>26,646</td>
<td>85.35</td>
<td>92.62</td>
<td>950</td>
</tr>
<tr>
<td>s15850</td>
<td>596</td>
<td>390</td>
<td>96.29</td>
<td>232,440</td>
<td>59,554</td>
<td>32,604</td>
<td>74.38</td>
<td>85.97</td>
<td>553</td>
</tr>
<tr>
<td>s38584</td>
<td>1426</td>
<td>1098</td>
<td>95.23</td>
<td>1,565,748</td>
<td>500,688</td>
<td>71,226</td>
<td>68.02</td>
<td>95.45</td>
<td>14,630</td>
</tr>
</tbody>
</table>

Table 2. Experimental results on ITC’99 benchmark circuits

<table>
<thead>
<tr>
<th>Benchmark reference</th>
<th># FFs</th>
<th># test patterns</th>
<th>FC (%)</th>
<th># clock cycles for single scan (1)</th>
<th># clock cycles for scan tree (2)</th>
<th># clock cycles for the proposed method (3)</th>
<th>test time saving (%) (1)(2)</th>
<th>test time saving (%) (1)(3)</th>
<th>computation time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b09</td>
<td>28</td>
<td>51</td>
<td>99.72</td>
<td>1,428</td>
<td>1,071</td>
<td>848</td>
<td>25</td>
<td>40.62</td>
<td>0.01</td>
</tr>
<tr>
<td>b10</td>
<td>17</td>
<td>102</td>
<td>100</td>
<td>1,734</td>
<td>918</td>
<td>712</td>
<td>47.06</td>
<td>58.94</td>
<td>0.01</td>
</tr>
<tr>
<td>b11</td>
<td>31</td>
<td>111</td>
<td>97.11</td>
<td>3,441</td>
<td>1,887</td>
<td>633</td>
<td>45.16</td>
<td>81.6</td>
<td>1</td>
</tr>
<tr>
<td>b12</td>
<td>121</td>
<td>183</td>
<td>99.96</td>
<td>22,143</td>
<td>6,954</td>
<td>4,441</td>
<td>68.6</td>
<td>79.94</td>
<td>15</td>
</tr>
<tr>
<td>b13</td>
<td>53</td>
<td>52</td>
<td>96.43</td>
<td>2,756</td>
<td>1,508</td>
<td>876</td>
<td>45.28</td>
<td>68.21</td>
<td>1</td>
</tr>
<tr>
<td>b14</td>
<td>245</td>
<td>906</td>
<td>97.07</td>
<td>22,1970</td>
<td>110,532</td>
<td>22,450</td>
<td>50.2</td>
<td>89.89</td>
<td>869</td>
</tr>
</tbody>
</table>

reduces the test sequence by removing the equivalent test patterns defined as follows.

**Definition:** a test pattern is equivalent to another test pattern if they have the same incompatibility graph. In the example of figure 5, the two patterns \( t_1 \) and \( t_7 \) can be applied with the same scan tree design.

![Figure 6. Test pattern selection procedure](image)

Finally, the complexity of the complete scan tree architecture generation is \( O(m^2n^2) \). In future work, we will try to find a solution to reduce the complexity in order to save computation time.

4. Experimental results

We have implemented the proposed method in C language on a Pentium M 1.6 GHz with 512 MB RAM and have applied it to ISCAS’89 and ITC’99 benchmark circuits. The test sets used are provided by Testgen tool [22] from Synopsys.

Table 1 and 2 give the results respectively for ISCAS’89 and ITC’99 benchmark circuits. The first four columns of the tables describe the circuits names, numbers of scan cells, lengths of test sequences (without any compaction) and corresponding fault coverages. The three following columns report the number of clock cycles required to apply the test sequence. The next two columns correspond to the test shift time savings (in percentage). The first one is the test shift time saving between the single scan chain and the full compatible scan tree architecture and the proposed method respectively. The second one is the test shift time saving between the full compatible scan tree architecture and the proposed method respectively. The next two columns correspond to the test shift time savings (in percentage). The first one is the test shift time saving between the single scan chain and the full compatible scan tree architecture and the proposed method respectively. The second one is the test shift time saving between the full compatible scan tree architecture and the proposed method respectively. The next two columns correspond to the test shift time savings (in percentage). The first one is the test shift time saving between the single scan chain and the full compatible scan tree architecture and the proposed method respectively. The second one is the test shift time saving between the full compatible scan tree architecture and the proposed method respectively.
architecture. The second one gives the test shift time savings between the single scan chain and the proposed method. The last column reports the corresponding computation time in seconds.

The number of clock cycles is computed considering the following expression:

$$\text{# clock \_cycles} = \left(\#t_{st} \times L_2\right) + \left(\#t_{st} \cdot \#t_{st} \times L_3\right)$$

where \(\#t_{st}\) is the number of test patterns applied in ST mode, \(L_2\) is the length of the scan tree architecture, \(#t\) is the test sequence length and \(L_1\) is the number of scan cells. For single scan chain, \(\#t_{st}\) is equal to 0. For full compatible scan tree, \(\#t_{st}\) is equivalent to \(\#t_{st}\). For the proposed solution, \(\#t_{st}\) is included between 0 and \(\#t\).

The test shift time saving using the full compatible scan tree is equal to 52.3\% in average and to 78.9\% at maximum in comparison with the single scan architecture. The proposed method improves significantly the test time reduction. Indeed, the average test shift time saving is equal to 78\% and reaches 95.45\% for s38584. The difference of time saving achievements clearly shows the effectiveness of the proposed solution. Moreover, an equivalent reduction of the test data volume is obtained. Concerning the computation time, it remains reasonable even though the results have not been provided by a workstation.

Figures 7 and 8 represent the evolution of the test shift time saving along the amount in percentage of patterns in the test sequence applied in ST mode. A linear evolution of test shift time saving can be observed in the first part of the graphics. This linear evolution reveals that the test patterns progressively included in the solution have a low impact on the current scan tree architecture, implying a beneficial reduction of the test time. Full compatible scan tree architecture corresponds to the last point on the curve (100\% of the test sequence is applied in scan tree mode).

Concerning the effectiveness of the proposed test sequence reduction to lessen the scan tree architecture generation complexity, the table 3 summarizes the achieved results. The first column reports the names of the circuits. The second and third columns correspond respectively to the test sequence length and to the reduced test sequence length considered for the architecture generation algorithm. The last column reports the corresponding percentage of reduction. The results show the effectiveness of this solution for some
circuit. In future work, we will try to improve the proposed solution to reduce the complexity of the scan tree architecture generation.

### Table 3. Optimization of the test sequence for the scan tree architecture generation on ISCAS’89 and ITC’99 benchmark circuits

<table>
<thead>
<tr>
<th>Benchmark reference</th>
<th>Test sequence length</th>
<th>Reduced test sequence length</th>
<th>Reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1423</td>
<td>79</td>
<td>68</td>
<td>13.92</td>
</tr>
<tr>
<td>s3578</td>
<td>166</td>
<td>166</td>
<td>-</td>
</tr>
<tr>
<td>s8234</td>
<td>341</td>
<td>341</td>
<td>-</td>
</tr>
<tr>
<td>s13207</td>
<td>540</td>
<td>538</td>
<td>0.03</td>
</tr>
<tr>
<td>s5850</td>
<td>390</td>
<td>390</td>
<td>-</td>
</tr>
<tr>
<td>s38584</td>
<td>1098</td>
<td>1088</td>
<td>-</td>
</tr>
<tr>
<td>b09</td>
<td>51</td>
<td>51</td>
<td>-</td>
</tr>
<tr>
<td>b10</td>
<td>102</td>
<td>87</td>
<td>14.7</td>
</tr>
<tr>
<td>b11</td>
<td>111</td>
<td>111</td>
<td>-</td>
</tr>
<tr>
<td>b12</td>
<td>183</td>
<td>173</td>
<td>5.78</td>
</tr>
<tr>
<td>b13</td>
<td>52</td>
<td>51</td>
<td>1.92</td>
</tr>
<tr>
<td>b14</td>
<td>906</td>
<td>893</td>
<td>1.43</td>
</tr>
</tbody>
</table>

### 5. Conclusion

In this paper, a new scan tree architecture is presented to reduce scan test shift time thanks to a dynamic reconfiguration during the test application. Experimental results for benchmark circuits show that our proposed method can reduce scan test shift time up to 95% of that for the single scan. In further work, we shall consider design constraints and reduce the scan tree generation complexity. Another perspective is to use several scan tree modes to improve the test time reduction.

### Acknowledgments

We would like to thank Prof. Michiko Inoue, Prof. Satoshi Ohtake, and Dr. Mariane Comte of Nara Institute of Science and Technology for their valuable comments. This work was supported in part by 21st Century COE (Center of Excellence) Program “Ubiquitous Networked Media Computing” and in part by JSPS (Japan Society for the Promotion of Science) under Grants-in-Aid for Scientific Research B(2) (No. 15300018).

### References