Contents

Abstract .................................................................................................................................................... 2
1. Introduction ........................................................................................................................................ 2
2. Background ........................................................................................................................................ 2
3. Main Results ....................................................................................................................................... 4
4. Double-Tree Scan (DTS) Structure ................................................................................................. 5
5. Full DTS ........................................................................................................................................... 6
   5.1. Centralized Control .................................................................................................................. 6
   5.2. Distributed Control .................................................................................................................. 12
6. Partial DTS ....................................................................................................................................... 14
   6.1 Path Sequencer for Partial DTS ............................................................................................... 17
7. Power and Energy Savings ............................................................................................................... 19
8. Experimental Results ..................................................................................................................... 20
9. DTS Extensions ............................................................................................................................. 21
10. Conclusion and Future Work ....................................................................................................... 21
References ............................................................................................................................................ 22
A Novel Scan-Path Architecture for Low Test Power

Bhargab B. Bhattacharya  
Indian Statistical Institute  
Calcutta, India  
bbc@isical.ac.in

Sharad C. Seth  
University of Nebraska-Lincoln  
Lincoln, Nebraska  
seth@cse.unl.edu

Sheng Zhang  
Broadcom Corporation  
Santa Clara, California  
szhang1031@gmail.com

Abstract

In a scan-based system with a large number of flip-flops, a major component of power is consumed during scan-shift and clocking operations in the test mode. In this paper, a novel scan-path architecture is proposed, which drastically reduces the scan-shift and clock activity during testing. The inherent combinatorial properties of a double-tree structure are employed to design the scan architecture, gating logic, and a simple shift controller. The design is independent of the structure of the circuit-under-test (CUT) and its test set. It provides a significant reduction both in instantaneous and average power needed for clocking and scan-shifting. The architecture is suitable for built-in self-test (BIST) under random testing and deterministic test environment, as well as in external testing schemes for test-data and test-length compression.

1. Introduction

With the emergence of mobile devices, design of low-power VLSI systems has become a major concern in circuit synthesis. A significant component of the power consumed in CMOS circuits is caused by the switching activity (SA) at various circuit nodes during operation. The dynamic power consumed at a circuit node is proportional to the total number of switching transitions the logic signals undergo at that node multiplied by its capacitance and the frequency of operation.

Power/energy minimization during testing has become important in the context of deep sub-micron technology because of higher device densities and clock rates [11]. In a scan-based system, the amount of power is consumed in the scan-path and clock tree during the scan operations could far exceed the power consumed in the functional mode. Maximum sustained power over a specified limit may cause excessive heating of the device; instantaneous peak power may cause excessive (inductive) voltage drop in the power and ground lines because of current swing and erroneously change the logic states of circuit nodes. Reducing average power extends the battery life in mobile applications and makes it possible to employ BIST schemes that may otherwise consume excessive energy due to long test lengths.

2. Background

The existing power/energy minimization techniques that use the traditional linear scan-path architectures include test scheduling [2], toggle suppression and blocking of useless test patterns [3], test pattern generation for low test power [4, 5, 6, 7, 8, 9], and use of Golomb coding for scan testing [10]. For deterministic testing, power reduction can be achieved by reordering scan chains and test vectors [11]. Test compaction is another option considered for low power in a scan-based system [12]. For minimizing switching activity during scan-shift, multiple scan-path architectures with selective freezing have been proposed [13, 4, 14]. Circuit-specific scan-path architectures aimed at reducing test power, as well as test-data volume, and test application time, have
Figure 1: Three different scan-path architectures

also been proposed [[15],[16]]. Gated clock routing for low-power microprocessor design is studied in [[17]].

The DTS architecture is motivated by the observation that in typical designs, with a large number of scan flip flops, a major component of energy is consumed in scan-shifting and clocking of the scan chain. We focus attention on designs with a single serial scan input and output. The following properties are desirable in designing such a scan-path and its associated shift control mechanism:

1. It should be possible to load or unload the scan-path with \( f \) FFs in \( f \) shift clock cycles;
2. It should be possible to completely overlap the loading of a new test vector with the unloading of the previous response;
3. The scan control mechanism should be simple and it should not increase the shifting time;
4. The design should be independent of the structure of the CUT and its test set;
5. The input order of values shifted in, may also be preserved at the shifted output.

The simplest architecture with these properties is the classical linear scan [[18],[19]] (Fig. 1a), where all the flip-flops are configured as a chain in test mode. In a linear scan chain of length \( f \), the total worst-case switching activity during scan-shifting in terms of number of transitions is \( O(T \cdot f^2) \), where \( T \) is the number of test vectors. Similarly, the clock activity is \( O(T \cdot f^2) \). The other extreme example is the full parallel or random-access scan [[20, 21], Baik 04] (Fig. 1b) where, each flip-flop can be loaded and unloaded independently. Between these two options lie the multiple parallel scan-path architectures, where the scan-path is decomposed into several smaller and independent linear chains (Fig. 1c). Several such architectures have been reported for reducing the shift and clock activity [[13], [22],[14]] and for BIST applications [[23]]. During shifting, the clock signal is gated to selected scan...
chain, and blocked from reaching all other scan cells. If \( s \) is the number of linear chains, the switching activity reduces to \( O(s \cdot T \cdot (f/s)^2) = O(T \cdot f^2/s) \), i.e., by a constant factor. Since, the power loss (both for shifting and clocking) is quadratic in the length of the chain, a fully linear scan-path consumes maximum power, whereas, a fully parallel scan-path requires minimum power. Increasing the multiplicity of the scan-paths for a given number of flip-flops, reduces the length of the paths, and hence power demand. However, the control mechanism becomes overly complex as the multiplicity increases. For a large number of scan chains, the MUX block either introduces large fan-in/fanout, or delay. The former consumes more power and the latter increases the shifting time. Thus, the multiple-parallel scan-path schemes, beyond a certain value of multiplicity, are impractical. A multiple scan-path with only 3 linear chains was considered by Saxena, Butler, and Whetsel [[22]]. Nicolic and Al-Hashimi [[13]] used at most 7 chains, and Sinanoglu and Orailoglu [[24]] studied up to 24 linear chains. Furthermore, the last two design methods in are strongly dependent either on the structure of the CUT, or on the test set itself.

It may be noted that some of the other well known scan architectures that do not have power minimization as their goal allow multiple scan chains to operate in parallel, e.g. the STUMPS architecture [[25]] uses an LFSR/phase shifter (MISR) to load (unload) multiple chains in parallel, and the shared scan-in (Illinois scan) architecture allows parallel scan chains to be loaded with identical bit streams [[26], [27], 03]. These schemes are not directly comparable to DTS, although the scan chains they employ can be implemented with as DTS to reduce power consumption.

3. Main Results

This paper presents a novel scan-path architecture called double-tree scan (DTS) for low-power test applications. The structure resembles two trees glued at the leaf nodes. We report on the design methodology of DTS, clock gating logic, and shift controller. The architecture is independent of the CUT and its test set. The scheme reduces both the scan-shift and clock activity. The design is simple and hardware overhead is low. It provides very significant amount of power/energy savings over various existing schemes, and the benefit becomes more prominent as the number of scan flip-flops increases. Earlier results of this work were reported in [[28]].

The rest of the paper is organized as follows. We start with the definition of the basic structure and properties of the DTS in Section 4. The definition in this section is for a full DTS, which can be defined for an arbitrary size scan chain. In Section 5, we discuss the issues related to controlling the loading and unloading of test data in a full DTS. Both centralized and distributed control schemes are described. The scan cells in the centralized scheme are simpler but they require routing of global signals that are largely avoided in the distributed scheme. Section 6 extends the full DTS to partial DTS that can be obtained by pruning a larger full DTS or composing several smaller ones. In Section 7 we estimate the peak and total switching activity in DTS-based designs and compare it against designs employing linear and multiple scan chains. Experimental results on scan-shift switching activities for ISCAS89 benchmarks appear in Section 8. In this paper we have focused on the DTS as a transparent replacement of a single scan chain for the purpose of reducing the scan-shift power. Extensions of DTS to reducing the peak capture power, test-data volume, and test-application time are briefly discussed in Section 9. Section 10 concludes the paper with a summary and discussion of future work.
4. **Double-Tree Scan (DTS) Structure**

We organize the scan flip-flops in a radically different fashion called *double-tree scan (DTS)* architecture (*Figs. 2a, 2b, 2c*), to achieve drastic power/energy reduction. A complete binary tree of level \( k \) (considering the root at level 0) consists of \( 2^k \) leaf nodes and \( (2^k - 1) \) internal nodes. The proposed scan structure resembles two complete \( k \)-level binary trees whose leaf nodes are merged pairwise. Thus, a full *double-tree* DTS(\( k \)) consists of \( N = (2^k - 1 + 2^k + 2^k - 1) = 3 \cdot 2^k - 2 \) nodes. Each node of the tree represents a scan flip-flop. All edges in the tree are directed from top to bottom. A directed edge \((i, j)\) in the DTS indicates that \(Q(i)\), i.e., the Q-output of the flip-flop \( i \) drives \( D(j)\), the D-input of the flip-flop \( j \). For each node with in-degree 2 (i.e., a merge node) in the bottom half of the DTS, a 2-1 MUX is needed to select the predecessor flip-flop during scan operation.

![Figure 2: Full DTS(\( k \)) for \( k=1,2, \) and 3.](image)
5. Full DTS

We will denote the DTS structure just defined as *full DTS*(*)k*), where the examples shown in Figure 2 correspond to full DTS(*)k* for *k* = 1, 2, 3. The topmost node (source) serves as the scan-in node, and the bottommost node (sink) serves as the scan-out node. The full DTS(*)k* has 2^*k* scan-paths, each of length (2* *k* + 1), from the source to the sink. Table 1 shows the design characteristics of full DTS (*k*), for *k* ranging from 0 to 10. For example, a full DTS (10) consists of 3070 FFs organized as 1024 overlapping scan-paths, each of length 21. Thus, the chain length is dramatically reduced in going from linear scan to DTS. To be sure, if the same number of FFs were organized as 1024 independent multiple scan chains, the chain length be even smaller (just 3), however, a 1024-input MUX block at the scan output will need very large fan-in/fan-out, causing additional power loss and a slowdown of the shift clock. The low-power benefits of the DTS structure accrue primarily because of the way the large fanins and fanouts are distributed throughout the structure.

<table>
<thead>
<tr>
<th><em>k</em></th>
<th># FFs in full DTS(<em>)k</em>)</th>
<th>Number of scan-paths</th>
<th>Length of each scan-path</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>22</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>46</td>
<td>16</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>94</td>
<td>32</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>190</td>
<td>64</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>382</td>
<td>128</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>766</td>
<td>256</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>1534</td>
<td>512</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>3070</td>
<td>1024</td>
<td>21</td>
</tr>
</tbody>
</table>

The full DTS structure can be divided into three roughly equal parts by the functionality of nodes in the tree. The top part consists of (2^*k* − 1) *forking* nodes with in-degree 1 and out-degree 2; the middle part consists of just one level of 2^*k* nodes, with in-degree and out-degree equal to 1; and the bottom part consists of 2^*k* − 1 *merging* nodes with in-degree 2 and out-degree 1. The forking (merging) nodes require the addition of a 1→2 DEMUX (2→1 MUX) to the corresponding scan cells, while the middle part can be built from standard scan cells.

The scan-shift control must select different paths from the source to sink during the loading and unloading of scan FFs in the test mode. We will consider two basic approaches to implementing the scan-shift control: *centralized* and *distributed*. In the *centralized* approach a separate control unit generates the signals to control the shift operations; in the *distributed* approach the control logic is integrated into the scan-cell structure.

5.1. Centralized Control

Here, we assume that the paths in DTS(*)k*) are identified by a *k*-bit vector L_1, L_2, .. L_ *k*, and the path number is simply the binary number corresponding to this vector. Thus, in DTS(2) of Fig. 2b, we have:
Further, we assume that a finite-state Path Sequencer produces the $k$-bit binary number of the path to be activated in each scan-shift cycle and feeds it to the DTS Controller that produces the control signals for individual cells in DTS($k$), as shown schematically in Fig. 3.

The DTS controller may control either the clock lines (clock gating) or the data lines of the scan cells in the selected path. During functional operation, all the flip-flops should receive the clock signals, and hence, the DEMUX design is slightly modified as in Fig. 4.

We will consider first the design of the DTS controller and then the path sequencer.

**A Naïve DTS Controller**

We show a design that employs clock gating. In this design, for every forking node in the top part of the DTS, a 1→2 DEMUX routes the clock to an appropriate successor flip-flop, where the DEMUX units are controlled by $L_1, L_2, \ldots, L_k$, as shown in Fig. 5 for DTS(3) of Fig. 2c. In the figure, $CL(i)$ denotes the clock signal of the FF $i$. During every cycle, only one path is activated by letting the clock signal reach only the FFs along the path; clocks to FFs in other paths are frozen. Note that the path selection needs to happen only in the top part of the tree. The selected path is simply mirrored at nodes in the bottom part by the use of 2→1 MUX units that are controlled identically. Thus, the architecture reduces both the shift and clock activity simultaneously. The number of MUX units needed for merge nodes in the bottom half of the tree is $2^k - 1$. Hence, for a DTS($k$) with $(3 \cdot 2^k - 2)$ FFs, the additional hardware overhead would be a total of $(2 \cdot 2^k - 2)$ DEMUX/MUX units. However, the maximum fanout of a control line is $2^k - 1$, which would be unacceptable when the number of scan flip-flops is large. For example, a controller for DTS(10) with 3070 flip-flops would require a fanout of 1024.
The above design can be easily adapted to avoid clock gating. This is accomplished by renaming the CL signal to E (enable) and all the CL(i) signals to E(i), which are then used with a modified design of the scan cells in the top and bottom parts of the tree, as shown in Fig. 6(a). The MUX units that are needed for the merge nodes in the bottom part of the DTS tree can be integrated into the scan flip-flops independent of clock gating, as shown in Fig. 6(b). Although they appear on the shift path, they only increase latency slightly, but not the throughput (i.e., they do not need to slow down the shift clock, unlike the multiple scan-path architecture of Fig. 1(c)).

Compared to the scheme of Fig. 5, this design requires an additional MUX in each scan cell in the top part of the DTS; however, as this MUX is not in a functional path of the circuit, it does not impact the timing of the circuit. Further, without clock gating, power is saved only in the scan cells and not on the clock lines.
A Hierarchical DTS Controller

A better way to design the path-selection logic is to use a recursive structure. We illustrate the design for clock-gating logic, which can be modified to a non-clock-routing solution as described above for the naïve design.

The recursive structure is used to route the clock and the control signals in a hierarchical fashion. To implement this architecture, a different clock control mechanism is needed. For DTS(2) of Fig. 2b, the controller is shown in Fig. 7. To activate scan-shift along the path 1→2→5→8→10, we set the control lines L1 = 0 and L2 = 1. Thus, the DEMUX tree enables CL(1), CL(2), CL(5), CL(8), and CL(10). The MUX control lines for the flip-flops 8 and 10 are also set accordingly.

In this method, DTS(k+1) can be constructed recursively by using two copies of DTS(k), and combining them by adding (k+1) new (1→2) DEMUX blocks. The design for DTS(3) is shown in Fig. 8. Thus, if D(k) denotes the number of DEMUX units to realize DTS(k), then D(k+1) = 2.D(k) + (k+1). Solving this recurrence yields, D(k) = 2^{k+1} − (k + 2). The maximum fanout of a control line for DTS(k) will be only k, instead of 2^{k+1} as in the centralized design. Thus, DTS(10) with 3070 FFs would need a maximum fanout of 10. As before, the number of MUX units needed for the merge nodes in the bottom part of the tree is 2^k-1. Hence, for a DTS(k) with \((3 \cdot 2^k - 2)\), the additional hardware overhead would be a total of \(3 \cdot 2^k - k - 3\) DEMUX/MUX units.

A hierarchical controller of DTS(k) has several advantages. First, the maximum fanout of a control line is k. Second, it takes care of driving clock signals to all the scan flip-flops, and further, each output of a DEMUX unit drives at most two signals. Hence, it obviates the need of a separate clock tree for buffering. Third, for each shift clock, only a single path of length k is activated through the interior of the clock tree, and hence, the additional power loss in the tree is low.

![Hierarchical clock gating logic for DTS(2)](image)
Path Sequencer

The path sequencer complements the DTS control in defining the control part of the DTS architecture (Fig. 3). It defines the order in which source-to-sink paths are selected for activation in each cycle, during the load/unload sequence. A correct path sequence would ensure the proper loading and unloading of the test data during each load/unload sequence. One way to define proper loading is to insist that the result of every load/unload sequence be identical to the behavior observed for the linear scan, where the order of bits loaded into the scan chain is the same as the order of captured valued unloaded from the scan chain. Strict preservation of the order, however, is not necessary for the DTS architecture; it is sufficient to have two fixed one-to-one mappings: (1) from the index of bits in the input stream to the scan FFs and (2) from the scan FFs to the index of bits in the output stream. For the linear scan, these two mappings are inverse of each other, but an automatic test equipment can be programmed easily to work with any fixed mappings.

We illustrate the design of the path sequence for full DTS(2) with 10 flip-flops (Fig. 9a). Let (p10 p9 p8 p7 p6 p5 p4 p3 p2 p1) denote a 10-bit vector to be shifted in the DTS. Assume that the current contents of the FFs are Q1, Q2, …., Q10, where Qi denotes the content of the i-th FF. There are two ways the scan-paths can be filled: depth-first load (DFL) and breadth-first load (BFL).
Figure 9: Scan-in and scan-out sequence for DTS(2) in DFL mode.

**Depth-first load (DFL)**

In this scheme, a scan-path is loaded serially up to a certain depth in the tree once for all, and then the other paths are processed. The path sequencer to scan-in and scan-out a complete 10-bit vector for DTS(2) is shown in Table 2. When the shift-in process is completed, the contents of the FFs will look like as in Fig. 9(b).

<table>
<thead>
<tr>
<th>shift clock</th>
<th>L_1</th>
<th>L_2</th>
<th>active path</th>
<th>scanned-in bit</th>
<th>scanned-out bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Path-0</td>
<td>p1</td>
<td>Q10</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>Path-0</td>
<td>p2</td>
<td>Q8</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>Path-0</td>
<td>p3</td>
<td>Q4</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>Path-1</td>
<td>p4</td>
<td>Q2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>Path-1</td>
<td>p5</td>
<td>Q1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>Path-2</td>
<td>p6</td>
<td>Q5</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>Path-2</td>
<td>p7</td>
<td>Q9</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>Path-3</td>
<td>p8</td>
<td>Q6</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>Path-3</td>
<td>p9</td>
<td>Q3</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>Path-3</td>
<td>p10</td>
<td>Q7</td>
</tr>
</tbody>
</table>

**Breadth-first load (BFL)**

From Table 2, it may be observed that the path sequencer for the DFL scheme (e.g. specified in Table 2 for DTS(2)) is not a simple counter, requiring extra effort and logic in its implementation. A breadth-first load scheme, on the other hand, can be implemented with a simple binary counter. For a DTS(k),
we run a modulo-$2^k$ counter continuously to load and unload the FFs, where the counter value is used to choose the corresponding path in the tree. For example, to load into DTS(2) with 10 FFs, we run the counter for 10 shift clocks cycling through states 0, 1, 2, 3, 0, 1, 2, 3, 0, 1. The resulting mapping from the input stream $<p_1, p_2, ..., p_{10}>$ to the scan FFs is shown graphically in Figure 10. When the captured values in the scan FFs are unloaded simultaneously with the load, the resulting output stream can be verified to be $<Q_1, Q_2, Q_5, Q_3, Q_7, Q_4, Q_8, Q_6, Q_9, Q_{10}>$, where, $Q_i$ represents the scan FF corresponding to where input bit $p_i$ is stored in Figure 10. Thus, it is seen that the two mappings are fixed but not inverse of each other.

### 5.2. Distributed Control

In the previous section, we considered a centralized control for the DTS architecture shown in Fig. 3. The distribution of centrally generated control signals to the DTS scan cells can pose a routing challenge that is avoided in the distributed control in which the two control blocks of Fig. 3 replaced with logic integrated into individual scan cells in the DTS structure.

A key idea of the scheme is to implement the definition of the currently active path in DTS(k) by means of $2^k - 1$ branch FFs attached to the nodes in the top part of the DTS. The path starts at the source node of the DTS (which is always selected in the scan mode), then continues to the left or right depending on the value of the branch FFs along its route in the top part, following its mirror-image route in the bottom part.

Another key idea of the scheme is to implement a breadth-first load (BFL) of the scan FFs by complementing the branch FFs on the active path during each clock cycle. This is illustrated for DTS(2) in Table 3, where BR[1], BR[2], and BR[3] denote the branch FF values, 0 for left and 1 for right, in the top part of the DTS.

It is not hard to see that the DTS(k) would load correctly independent of the initial state of the branch FFs, which determines only the specific static mappings realized for the input and output streams. In the example shown in Table 3, the states of the three FFs are assumed to be 0, 1, and 1 respectively at the beginning of each load/unload cycle. Note that the input and output mappings are not inverses of
each other: If the input stream is loaded in the increasing order of indices (1, 2, 3, ..., 10) the index stream at the output is: (1, 2, 3, 6, 7, 4, 5, 8, 9, 10).

Table 3: Load/Unload of DTS(2) under distributed control

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Initial Load Sequence (assumes scan data stream indexed with integers 1, 2, 3, …)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td>LR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>RR</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>LL</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>RL</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>LR</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>RR</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>LL</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>-</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>RL</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>LR</td>
<td>8</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>RR</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>8</td>
<td>9</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Subsequent Load/Unload Sequences (assumes reinitialization of the BR-FF states)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td>LR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>RR</td>
<td>1*</td>
<td>10</td>
<td>9</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>LL</td>
<td>2*</td>
<td>10</td>
<td>1*</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>9</td>
<td>6</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>RL</td>
<td>3*</td>
<td>2*</td>
<td>1*</td>
<td>10</td>
<td>8</td>
<td>5</td>
<td>9</td>
<td>4</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>LR</td>
<td>4*</td>
<td>2*</td>
<td>3*</td>
<td>10</td>
<td>8</td>
<td>1*</td>
<td>9</td>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>RR</td>
<td>5*</td>
<td>4*</td>
<td>3*</td>
<td>10</td>
<td>2*</td>
<td>1*</td>
<td>9</td>
<td>8</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>LL</td>
<td>6*</td>
<td>4*</td>
<td>5*</td>
<td>10</td>
<td>2*</td>
<td>1*</td>
<td>3*</td>
<td>8</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>RL</td>
<td>7*</td>
<td>6*</td>
<td>5*</td>
<td>4*</td>
<td>2*</td>
<td>1*</td>
<td>3*</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>LR</td>
<td>8*</td>
<td>6*</td>
<td>7*</td>
<td>4*</td>
<td>2*</td>
<td>5*</td>
<td>3*</td>
<td>10</td>
<td>1*</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>RR</td>
<td>9*</td>
<td>8*</td>
<td>7*</td>
<td>4*</td>
<td>6*</td>
<td>5*</td>
<td>3*</td>
<td>2*</td>
<td>1*</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10*</td>
<td>8*</td>
<td>9*</td>
<td>4*</td>
<td>6*</td>
<td>5*</td>
<td>7*</td>
<td>2*</td>
<td>3*</td>
<td>1*</td>
</tr>
</tbody>
</table>

The compound scan cells needed in the top part for implementing the distributed control are similar to those shown in Figure 6 for the centralized control, except for additional logic to propagate the branch enable signals. The extra logic is shown in Figure 11. The signal E(i) not only enables scan shifting but also toggling of the branch FF in the same cell. It is easily verified that the MUX/DEMUX overhead remains the same as in the naïve scheme (Figure 3), but one toggle FF is added to every cell in the top part of the full DTS(k).
Figure 11: Extra logic in the scan cells, shown in the top part of Figure 4(a), to implement the distributed control for path selection.

For both the centralized and distributed control schemes, every DEMUX in the top part of DTS(k) has a corresponding MUX in the bottom part that is controlled by the same enable signal. This suggests a folded layout of the DTS which brings the two scan cells physically together for ease of routing. The layout is schematically shown in Figure 12(a) for a scan chain implemented as a DTS. The layout of multiple scan chains, for a STUMPS-like architecture [[25]] is shown in Figure 12(b).

Figure 12: Folded layout of DTS(k) for ease of routing (a) with extension to multiple scan chains in a STUMPS-like architecture (b).

An analysis in [29] shows that, on average, the area overhead of scan cells in the distributed vs. centralized control is approximately 32%. Counter-balancing this, are the savings realized in the distributed control from simplified routing and the absence of a separate path sequencer.

6. Partial DTS

For a full DTS(k), the number of FFs is \(3.2^k - 2\). In real life, if the number of flip-flops in the CUT does not satisfy this equality, then we have a partial DTS. A partial DTS of any given size can be obtained by composing it from DTS’s of smaller size or by pruning a full DTS of larger size.

Fig. 13 shows three examples of constructing an 18-node DTS. In part (a), three full DTSs of size 10, 4, and 4, respectively, are concatenated serially, while the examples in parts (b) and (c) are obtained by
Figure 13: Three examples a partial DTS with 18 nodes by: (a) serial concatenation of smaller full DTSs, (b) replacing a 4-node DTS in DTS(3) with a direct connection, and (c) deleting 4 nodes with in and out degree 1, thereby producing a balanced structure.

pruning a DTS of size 22. The depth of the partial DTS in Fig. 13(a) (i.e. the number of nodes in its longest path from source ot sink) is 11 while it is 7 in the structures of Figs. 13(b) and 13(c).

In Fig. 14, we consider s35932.scan, one of the larger ISCAS89 circuits. We assume that the scan-path also includes the PI/PO for test application, and hence, its length should be \#FFs + max \{\#PI, \#PO\} = 1728 + 320 = 2048.

The pruned design of a DTS with 2048 FFs is shown in Fig. 14(a). The maximum scan-path length is 21, and the minimum is 3. The serial composition of smaller full DTSs is shown in Fig. 14(b). It needs serial concatenation of DTS(9), DTS(7), DTS(5), DTS(3), DTS(2), DTS(1), one each, and two single nodes (i.e., DTS(0)). This follows from the fact that 2048 = 1534 + 382 + 94 + 22 + 10 + 4 + 1 + 1 (see Table 1). All paths are of equal length (19 + 15 + 11 + 7 + 5 + 3 + 1 + 1 = 62).

Fig. 13(c) preserves the balance property of the full DTS, namely, every path between the source and the sink has the same length. Furthermore, it preserves the minimum-depth property of a full DTS because no partial DTS of the same size can be found with smaller depth. The first property simplifies the shift control for loading and unloading, while the second property minimizes the shift power per scan load, which grows as the square of the tree depth. Therefore it is worthwhile to develop a pruning algorithm constrained to preserve the two desirable properties. The following rule-based algorithm accomplishes this goal:
Algorithm to Produce a Balanced Partial DTS

Objective: Obtain a minimum-depth balanced DTS for a given size $N$, by pruning DTS($k$), such that, $3 \cdot 2^k - 2 > N > 3 \cdot 2^{k-1} - 2$.

Input: $N$ and DTS($k$)

Output: A balanced partial DTS of minimum depth.

Method: Prune the tree according to the following set of rules, applied repeatedly in the stated order:
Rule 1: If there is a source vertex with out-degree one, or a sink vertex with in-degree one, remove the vertex and the edge connected to it. Note that this rule has the effect of reducing the depth of the tree by 1.

Figure 15: Pruning a DTS(3) according to the algorithm to produce a 16 node balanced DTS with minimum depth. In the first step, nodes W-Z of figure (a) are pruned using Rule 2 to yield figure (b); next node pairs (W,X) and (Y,Z) of (b) are pruned using Rule 3 to yield (c).

Rule 2: If there is a node with both in-degree and out-degree equal to 1, delete the node and the two edges connected to it, as long as the deletion does not disconnect a node adjacent to this node. Note that this rule will transform the adjacent branch and merge nodes to pass nodes.

Rule 3: If there is a branch (merge) node x, with branch (merge) edges \(<x,y>\) and \(<x,z>\) (\(<y,x>\) and \(<z,x>\)), such that the out-degree (in-degree) of both y and z is one, then, merge nodes y and z, i.e. replace the two edges by a single edge \(<x,new>\) (\(<new,x>\)), where new is a new node corresponding to the merging of the nodes y and z. Note that this results in moving the branch (merge) from node x to the merge (branch) node new.

Figure 15 shows an example of applying the algorithm to the full DTS(3) with 22 nodes to obtain a 16-node partial DTS. Note that the final result is not unique because of the choices that may be available to apply a rule at a given step. A proof of correctness of the algorithm is omitted here.

For the s35932.scan benchmark circuit, considered earlier, a balanced design according to the algorithm will only modify the innermost two levels of the tree, consisting of DTS(1) and (2) and will otherwise preserve the structure of the of the original tree; all paths in this design will have the same depth of 21.

6.1 Path Sequencer for Partial DTS

The full DTS has both horizontal and vertical symmetries that simplify the path-sequencing mechanism for loading and unloading of test patterns. Because of these symmetries, simple breadth-first-load (BFL) and depth-first-load (DFL) mechanism work correctly for the full DTS but may not do so for a partial DTS. We call a load operation correct and the corresponding path sequencer valid if the DTS is loaded with all the bits of the test pattern at the end of the operation. Deviations from correct behavior can occur if a test-input bit, reaches the sink node before the last bit of the test pattern is
shifted into the source node. For example, in the 3-node DTS, shown in Fig. 16, any load sequence that starts with choosing the right path would be invalid because the first bit would exit the sink node when the second bit is shifted into the source node. In this section we provide a solution to specifying a valid shift control for any partial (or full) DTS.

The solution is based on a topologically sorted numbering of the given DTS nodes. Let the nodes of the DTS be numbered uniquely with integers 1, 2, ..., and let N(x) be the number of node x. Then, the numbering is said to be topologically sorted if for every directed edge <x,y> of the DTS, N(x) is greater than N(y). Clearly, in any such numbering the sink node will be numbered as 1 and the source node as M, where M is the total number of nodes in the tree. Now, it is not difficult to prove that a shift-control sequence that retires the numbers of the nodes in the sequential order is guaranteed to be valid. Further, such a sequence must preserve the topological sort of the remaining numbers in the tree throughout the process. Based on these two rules, it is easy to obtain the unique valid shift-control sequence corresponding to a topological numbering of the tree.

Consider the 9-node partial DTS shown in Fig. 17, along with the topological numbering of its node. Assume the three paths in the tree are labeled as p1, p2, and p3 in the left-to-right order. Then it is clear that in order to retire 1 and 2 in the correct sequence, the shift sequence must start with either p1 or p2 but not with p3. Now, if p2 is selected as the first path, then after the shift, label 4 would occupy the node currently occupied by 2, which would violate the topological-sort because a smaller label (3) would be above this label. Hence, we conclude that the sequence must start with p1. Proceeding in this way, we can derive the full sequence as follows: (p1,p2,p1,p3,p2,p3,p3,p1,p1).

Now, we can determine the two mappings, corresponding to the input and output streams, defined earlier. Without loss of generality, we can assume the mapping from the indices of the input stream to the indices of the FFs to be the identity permutation and then determine the resulting output stream,(as discussed earlier in relation to Table 3) as: (1,2,4,7,3,8,5,6,9).
7. Power and Energy Savings

It is easy to show that in the DTS architecture, for a test length $T$ and a scan chain with $f$ flip-flops, the total shift as well as the clock activity is $O(T \cdot f \cdot \log f)$ including the additional energy consumption in the control circuit. Thus, the percentile energy savings of DTS for both shifting and clocking over a linear scan chain is $(1 - (\log f)/f)$, which asymptotically approaches 100% when $f$ becomes large.

![Figure 17: A partial DTS, with topologically sorted numbering of nodes, used to illustrate the derivation of a valid shift-control sequence.](image)

For the s35932.scan circuit, we compare in Table 4 the estimated switching activity for loading/unloading of DTS against a single and three linear chains. The computed SA refers to shifting of only one vector. It is a measure of the worst-case scan-shift activity and the number of FFs that are clocked during shifting. The activity per shift clock is determined by the length of the currently active path and is a measure of instantaneous power. The total SA is a measure of energy required to complete the shift. The total SA per scan load in each case is essentially the product of the per-clock SA with the total number of scan FFs, the savings in the DTS designs come from the significant reduction in the per clock SA.

<table>
<thead>
<tr>
<th></th>
<th>Linear Scan</th>
<th>3 Linear Scan</th>
<th>Serial DTS (Fig 11b)</th>
<th>DTS Pruned (Fig 11a)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Peak SA</strong></td>
<td>2048</td>
<td>683</td>
<td>62</td>
<td>21</td>
</tr>
<tr>
<td><strong>Total SA</strong></td>
<td>4,194,304</td>
<td>1,399,467</td>
<td>126,976</td>
<td>42,987</td>
</tr>
</tbody>
</table>

(100%) (33.3%) (3.0%) (1.0%)
8. Experimental Results

We carried out experiments on several ISCAS-89 scan circuits and the results are reported in Table 5. We assume that the primary inputs (PI) are also scanned in, so the effective number of scan flip-flops is increased accordingly. A 25-bit LFSR is used to generate 20,000 pseudorandom test vectors. After doing forward and reverse fault simulation for single stuck-at faults, only those vectors that contribute to fault dropping are retained, the number of which is shown in column 4. The total scan-shift switching activity (SA) for loading the test vectors and shifting out response vectors are then computed over the entire test session, and the results are reported for fully linear scan chain (column 5), pruned DTS architecture (column 6), serial DTS architecture (column 8), and multiple scan-path with 3 linear chains (column 10) as in [[22]]. Energy savings for the last three structures over fully linear scan-path are shown in columns 7, 9, 11 respectively. It may be observed that for a 3-chain multiple scan-path [[22]], the expected savings would be \[ \frac{(Tf^2 - sT(f/s)^2)}{Tf^2} = \left(1 - \frac{1}{3}\right) = 66.6\% \], which is a constant independent of the number of FFs. This is reflected in column 11. On the other hand, energy savings will tend to increase significantly for the DTS architecture as the number of FFs increases. This is depicted in columns 7 and 9. The pruned DTS provides more savings over the serial version. As discussed earlier, the instantaneous power, both for shifting and driving clock signals (not shown in the table), will also reduce in the same fashion.

<table>
<thead>
<tr>
<th>Circuit</th>
<th># FFs</th>
<th># PIs</th>
<th># Test patterns</th>
<th>Fully linear scan chain</th>
<th>Pruned DTS SA savings</th>
<th>Serial DTS SA savings</th>
<th>3 linear chains SA savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>s208.scan</td>
<td>8</td>
<td>11</td>
<td>31</td>
<td>5015</td>
<td>1917</td>
<td>61.77</td>
<td>2688</td>
</tr>
<tr>
<td>s713.scan</td>
<td>19</td>
<td>35</td>
<td>53</td>
<td>77610</td>
<td>13536</td>
<td>82.56</td>
<td>25608</td>
</tr>
<tr>
<td>s838.scan</td>
<td>32</td>
<td>35</td>
<td>67</td>
<td>139428</td>
<td>21954</td>
<td>84.25</td>
<td>33837</td>
</tr>
<tr>
<td>s953.scan</td>
<td>29</td>
<td>16</td>
<td>96</td>
<td>91436</td>
<td>19088</td>
<td>79.12</td>
<td>35170</td>
</tr>
<tr>
<td>s1196.scan</td>
<td>18</td>
<td>14</td>
<td>151</td>
<td>77958</td>
<td>19952</td>
<td>74.41</td>
<td>34442</td>
</tr>
<tr>
<td>s1238.scan</td>
<td>18</td>
<td>14</td>
<td>156</td>
<td>79686</td>
<td>20550</td>
<td>74.21</td>
<td>35256</td>
</tr>
<tr>
<td>s1423.scan</td>
<td>74</td>
<td>17</td>
<td>67</td>
<td>269550</td>
<td>33160</td>
<td>87.70</td>
<td>54090</td>
</tr>
<tr>
<td>s5378.scan</td>
<td>179</td>
<td>35</td>
<td>269</td>
<td>6187659</td>
<td>382791</td>
<td>93.81</td>
<td>662756</td>
</tr>
<tr>
<td>s9234.1.scan</td>
<td>211</td>
<td>36</td>
<td>287</td>
<td>8763780</td>
<td>494283</td>
<td>94.36</td>
<td>1061960</td>
</tr>
<tr>
<td>s13207.1.scan</td>
<td>638</td>
<td>62</td>
<td>428</td>
<td>104410365</td>
<td>2542069</td>
<td>97.56</td>
<td>8509093</td>
</tr>
<tr>
<td>s15850.1.scan</td>
<td>534</td>
<td>77</td>
<td>330</td>
<td>61520276</td>
<td>1693672</td>
<td>97.25</td>
<td>4726576</td>
</tr>
<tr>
<td>s35932.scan</td>
<td>1728</td>
<td>35</td>
<td>72</td>
<td>111434168</td>
<td>1231818</td>
<td>98.89</td>
<td>3209995</td>
</tr>
<tr>
<td>s38417.scan</td>
<td>1636</td>
<td>28</td>
<td>593</td>
<td>819768164</td>
<td>9517503</td>
<td>98.84</td>
<td>24628905</td>
</tr>
<tr>
<td>s38584.1.scan</td>
<td>1426</td>
<td>38</td>
<td>695</td>
<td>742811515</td>
<td>9655405</td>
<td>98.70</td>
<td>37564000</td>
</tr>
</tbody>
</table>
9. DTS Extensions

While our focus is on reducing peak and average shift power (during the load/unload cycles of scan testing), DTS can also reduce peak capture power (during the test cycles of scan testing) by compatible techniques available in the literature. This is because, externally, DTS presents the functionality of a scan chain, the underlying structure assumed in many peak-capture-power reduction methods. These include: power-aware ATPG algorithms that fill the don’t cares in the test patterns to minimize the switching activity during capture [[9], [30], [6], [31],[32]] and various scan-cell partitioning or disabling methods for reducing capture power [[2], [33], [34], [9], [35],[36]].

Along with test-power, reductions of test-data volume and test-application time have also gained urgency for SoC devices [1]. Most research on test-data volume reduction is circuit and test specific, relying on filling the don’t-care (X) bits of test patterns in specific ways to facilitate data compression [[37], [38], [39],[40]]. Such X-filling schemes can be used independently with DTS to minimize both test-data volume and test power. Similarly, some schemes for test-application-time reduction, such as the Illinois Scan [[26], [41], [42]] are orthogonal to and compatible with DTS. In particular, as a replacement for individual scan chains, DTS can enhance the Illinois Scan performance. With a minor change in the design of the compound scan cell, discussed in Section 5, it is possible to change the initial state for loading different DTS chains, thus breaking the strict correlation between bits at the same position of different chairs in Illinois Scan.

10. Conclusion and Future Work

A novel scan-path architecture called double-tree scan is proposed for low-power testing. The architecture is simple and provides a very significant amount of power/energy savings for both scan-shift and clock activity. Evaluation of other variations of DTS needs further study. Minimization of test application time on DTS is another open area for further investigation.
References


