Hardware Implementation of the Double-Tree Scan Architecture

Nathan Schemm and Sina Balkır
University of Nebraska-Lincoln
Department of Electrical Engineering
209N SEC, Lincoln, Nebraska 68588-0511, USA
Email: nathan.schemm@huskers.unl.edu, sbalkir@unl.edu

Sharad Seth
University of Nebraska-Lincoln
Computer Science and Engineering
256 Avery, Lincoln, Nebraska 68588, USA
Email: seth@cse.unl.edu

Abstract—In a scan-based test architecture, the scan power and test data volume can be reduced by utilizing a double tree scan (DTS) architecture. This paper presents a novel hardware implementation of the DTS architecture and compares the hardware overhead with the conventional scan architecture. The implementation proposed utilizes a clock structure which greatly decreases the number of clocked flip-flops and thereby reduces power consumption. A test chip is designed and fabricated in a 0.5 \( \mu \)m CMOS technology to verify the power saving properties of the architecture.

I. INTRODUCTION

The power and energy consumption during the test of integrated circuits can become very large. Due to the high activity factor of the digital logic during the scan operation, the power consumption of a circuit under test can be many times larger than the same circuit under normal functionality. The larger power consumption causes many problems, ranging from IR-drop and inductive power supply losses to excessive total chip power dissipation. The power dissipation can limit the test application speed and therefore increase test time. A number of different approaches have been proposed to decrease the power consumption or test time.

Breaking a large scan chain into \( N \) smaller ones has the effect of dividing energy consumption by \( N \). However, this does not scale well with increasing design complexity. The Illinois scan architecture [1], [2] shares the inputs of the many scan chains between a common scan input. This dramatically reduces test data volume as well as energy consumption, but enforces a strict correlation between bits in the separate scan chains. Test data compaction uses the large number of don’t care bits to decrease the number of tests required and therefore decrease the testing energy [3], [4]. Test compression compresses the test vectors to save tester memory and possibly increase application speed of the test [5]. The ATPG tool can also be customized to generate low-power test patterns [6]. However, test compaction and compression do not necessarily decrease testing power. The double-tree scan (DTS) architecture can be used to decrease test power consumption.

The double-tree scan architecture was first presented in [7], where the basic architecture is presented, as well as the control architecture for distributed and centralized control. In [8], the DTS approach is extended with concepts from the Illinois Scan architecture. In this form, multiple DTS trees in parallel use the same input data stream. However, the final locations where each input bit is shifted to can be changed by changing the initial states of the path control logic. In this way the strict correlation between bits, present in the conventional Illinois Scan architecture, can be broken. This makes it possible to perform some tests which are not possible with the conventional Illinois scan architecture. The fewer number of input bits allows for a reduction in test time. This architecture also allows for multiple tests to be performed using a single test pattern by changing the initial conditions of the path selection logic, thereby simultaneously reducing the test volume, time, and power.

The concept of a double-tree scan architecture is shown in Figure 1. Each node in the figure is a flip-flop. The structure consists of two binary trees with their leaves merged. Path control logic systematically selects the different paths to scan the data in and out.

The double-tree scan architecture decreases power consumption by shortening the maximum length of the scan chain. The power consumption in a scan chain is proportional to the number of clock cycles required and the length of the scan chain. Because the number of clock cycles required also scales by \( N \), the power consumption of a linear scan chain grows by \( N^2 \). By shortening the chain length, the power consumption can be reduced. For DTS, the length through the tree grows as \( 2 \times \log(N) \), and the number of clock cycles grows by \( N \), which results in a power savings of \( N/\log(N) \).

Fig. 1. DTS Tree.
The double-tree scan requires extra hardware overhead at each flip-flop to implement path selection. A path-sequence must also be implemented to cycle through all the paths in a known manner in order to read and write the data in the flip-flops. The path-sequence can either be distributed or centralized. In the distributed case, extra branch flip-flops are present at either the top or bottom nodes in the tree. These flip-flops select the path through the tree in a distributed manner. In the centralized case, a single control circuit controls global control signals which select the correct path. The centralized control circuit is more efficient in hardware, as it requires substantially fewer flip-flops.

In this paper, a hardware realization of the DTS architecture is presented. The presented hardware uses a clock gating scheme which is able to fully exploit the power-saving properties of the DTS architecture. The DTS logic is inserted into a number of standard testbench circuits and the hardware overhead is measured. A test chip is also designed to measure the total power savings garnered from the DTS architecture.

II. IMPLEMENTATION

One possible implementation of the DTS architecture is by clocking all the flip-flops on every cycle, and enabling the flip-flops along the desired path. The enable logic consists of a multiplexer on the data input of each flip-flop. This has the disadvantage of requiring the entire clock tree to be clocked when only a few flip-flops are actually enabled.

A naïve clock gating structure is to put a clock multiplexer at each node and route the clock and data from the top of the tree to the bottom. It works by routing the clock either to the left or right child node at each node in the top of the tree, and combining the clocks from the two parents in the lower part of the tree. This has the advantage of substantial power savings, because only the active flip-flops are clocked. However, this is not practical in an actual implementation, due to the clock skew problems it generates. Without buffers at each node to delay the clock signal, the clock skew is such that a race is set up on every node. For example, in figure 1, if the clock and the data both enter from the top of the tree, then after the root node is clocked, there is a race on the next node to clock and latch the old data from the root before the new data is driven on the output of the root node. This problem can be fixed with delays at each node, but these require a large area, high power consumption, and would catastrophically affect the functional clock tree.

A better solution, and the one presented in this paper, is to ripple the clock from the bottom of the tree with the data input still at the top of the tree. In this configuration, the clock skew caused by the clock rippling through the clock gating structures at each node is in a direction where it cannot cause race conditions. When the clock arrives at each node, the parent node has not yet been clocked, so its output is stable and no race conditions are present. In this way the functional clock tree can be bypassed, and the test clock tree does not have any strict clock tree synthesis requirements. The clock gating circuits can be replaced by OR gates if a requirement is made that the system clock is held low during the test mode. The hardware for this method is slightly simpler than for the other methods because the enable multiplexer in the other methods is replaced with a simpler OR gate.

The proposed scheme routes the scan-shift clock from the bottom of the tree to the top, while the data is propagated from the top of the tree to the bottom. Once when all the data is shifted in, the single capture cycle is clocked on the main system clock, so all the flip-flops get clocked, while the scan-shift clock is not clocked. Then the scan-out of the captured data and scan-in of the new data commences using the scan-shift clock.

For transition delay testing, the timing uncertainty caused by skew in the scan-shift clock as it propagates through the tree may cause problems when using launch-off-shift type transition testing. However, it will not cause problems for launch-off-capture type transition testing.

Two different scan tree synthesis programs were created to insert the DTS logic; both differing by only a small amount. In one version, the DTS tree with a centralized control structure is implemented. In the other, a distributed control structure is used. The two schemes differ mainly by the fact that in the distributed case the bottom flip-flops have an extra branching control flip-flop present. The schematic of the DTS flip-flops cells for the top, middle, and bottom flip-flops for the centralized control are shown in figure 2. For the distributed control, the top and middle schematics are the same, and the bottom one is modified to include an extra flip-flop. This distributed schematic is shown in figure 3.

From these schematics, it is possible to see how the bottom nodes route the clocks to either their right or left parent node, and the top nodes combine the clocks from both child nodes. The Lx Ctrl line in the bottom node schematic is the path-select control line. All flip-flops at the same level share a common Lx Ctrl line. The control lines go to individual bits of a counter which acts as the path sequencer for the design.

Cadence Encounter® RTL Compiler was used to perform the logic synthesis and technology mapping. A custom program was developed which reads the technology-independent netlist from RTL compiler, inserts the DTS logic, and writes out another technology-independent netlist. RTL compiler can then re-optimize the generic netlist before mapping it to a particular technology. The clock nets in the upper part of the DTS tree must be marked so they are not optimized, otherwise the optimizer would combine the boolean equivalent clock nets at points “a” and “b” in figure 1. However, those nets need to be separate as they have different clock skew times.

To handle the case where the number of functional flip-flops is not precisely equal to the number of flip-flops in a full DTS tree, the implementation uses multiple complete DTS trees, and daisy-chains them together. Eventually, a number of padding flip-flops are added to make the total number of flip-flops fill the last DTS tree. The program allows up to the maximum of either 50 or 2% of the total number of flip-flops to be added as padding flip-flops. Though it is possible to construct partial DTS trees, the path sequencer for partial DTS
The program as developed does not take into consideration any layout information when selecting which circuit flip-flop to place in which location in the trees. Using this information could greatly decrease the routing congestion in the layout. Instead it selects the tree flip-flops sequentially as they appear in the netlist.

For the measurement of the hardware overhead of the DTS architecture, several ITC '99 testbenches were synthesized in a 0.18 \( \mu \text{m} \) CMOS technology. The selected process has 5 routing metal layers, and one top thick metal useful for routing power and ground. The routing resources are such that over 90\% of the area can be utilized for standard cell area, with no additional routing channels needed. The library used is quite extensive with a large number of functions available over a range of drive strengths. It also supports scan versions of each of the flip-flops in the library for minimal scan testing overhead.

For the test chip, a 0.5 \( \mu \text{m} \) CMOS process was used. The test chip is designed to measure the power consumption of both a standard scan architecture and a DTS architecture on the same circuit. Because of the limited area available, two separate circuits were not used. Instead, a single design was configured to be able to be scanned both via a normal scan architecture and via a DTS scan architecture. The hardware for both scan architectures was inserted in the same circuit. The normal scan architecture simply consists of a single 188 bit scan chain. The DTS chain is a single 190 bit 6-level tree.

The digital design fabricated in the test chip is a sinc\(^3\) decimation filter for use as the first filtering stage after a second order delta-sigma modulator. It consists of 188 functional flip-flops and is about 8,000 gates. It has a single clock and is fully synchronous.

### III. RESULTS AND DISCUSSION

A number of test circuits were used to demonstrate the hardware overhead. One circuit is a 16-bit RISC CPU, and the others are ITC '99 benchmarks which were synthesized from the RTL-level VHDL. Each circuit was synthesized in four ways. One with no scan whatsoever, another using the standard scan flow available in the digital design flow, one DTS with centralized control, and finally DTS with decentralized control. The results are shown in table I.

#### TABLE I
SYNTHESIS RESULTS

<table>
<thead>
<tr>
<th>Ckt.</th>
<th># FF</th>
<th>No Scan ($\mu$m(^2))</th>
<th>Std. Scan ($\mu$m(^2))</th>
<th>DTS Cent. ($\mu$m(^2))</th>
<th>%↑</th>
<th>DTS Decent. ($\mu$m(^2))</th>
<th>%↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>322</td>
<td>48955</td>
<td>54252</td>
<td>64006</td>
<td>18</td>
<td>70633</td>
<td>30</td>
</tr>
<tr>
<td>b05</td>
<td>34</td>
<td>6020</td>
<td>6554</td>
<td>8662</td>
<td>32</td>
<td>9370</td>
<td>43</td>
</tr>
<tr>
<td>b14</td>
<td>245</td>
<td>87938</td>
<td>94879</td>
<td>111668</td>
<td>17.7</td>
<td>116602</td>
<td>22.9</td>
</tr>
<tr>
<td>b15</td>
<td>449</td>
<td>81175</td>
<td>88781</td>
<td>102870</td>
<td>15.9</td>
<td>116198</td>
<td>30.9</td>
</tr>
<tr>
<td>b17</td>
<td>1415</td>
<td>263435</td>
<td>280417</td>
<td>316753</td>
<td>12.9</td>
<td>358696</td>
<td>27.9</td>
</tr>
<tr>
<td>b19</td>
<td>6554</td>
<td>1409825</td>
<td>1542676</td>
<td>1820112</td>
<td>18</td>
<td>2101071</td>
<td>36.2</td>
</tr>
<tr>
<td>b20</td>
<td>490</td>
<td>189132</td>
<td>200060</td>
<td>225312</td>
<td>12.6</td>
<td>237909</td>
<td>18.9</td>
</tr>
<tr>
<td>b22</td>
<td>703</td>
<td>281630</td>
<td>297348</td>
<td>334161</td>
<td>12.3</td>
<td>348095</td>
<td>17.1</td>
</tr>
<tr>
<td><strong>Tot:</strong></td>
<td><strong>10212</strong></td>
<td><strong>2368110</strong></td>
<td><strong>2564967</strong></td>
<td><strong>2983532</strong></td>
<td><strong>16.3</strong></td>
<td><strong>3358574</strong></td>
<td><strong>30.9</strong></td>
</tr>
</tbody>
</table>

The percentage increase columns are calculated relative to the standard scan area. There are some factors which
artificially increase the area figures in table I. One effect that increases the DTS area is the fact that the optimizer cannot optimize the clock tree in the upper part of the DTS tree. If it had been allowed to optimize that, the logic would be somewhat smaller. In the standard scan and no scan cases, some flip-flop optimization was disabled in order to do a fair comparison between those and the DTS cases. In the standard cases, the synthesizer would be able to optimize out many of the flip-flops when mapping the logic to gates. It would then delete these flip-flops for substantial area savings. However, in the DTS cases, those same flip-flops could not be optimized out because they were part of the DTS trees. To make it a fair comparison, these optimizations were disallowed for all cases. The circuits were simulated in a digital simulator to make sure the DTS trees function as expected.

The results show an average centralized DTS overhead of 16%. The distributed control overhead is another 15% on top of that. This is generally acceptable in most cases, considering the very substantial power savings accrued. Not only is the scan length greatly reduced, but the clock tree is also not active. The DTS tree insertion has negligible effect on the functional timing of the circuits, as no extra logic is inserted in the critical path of the circuit.

The ITC '99 benchmarks are fully synchronous circuits with a single clock. However, the CPU has some asynchronous logic in it which conventionally makes scan testing difficult. However, for the proposed DTS tree structure, the asynchronous flip-flops, as well as flip-flops from other clock domains, can be incorporated into the DTS structure without problems, as long as the clock input can be set to a zero value during testing. When the functional clock input is set to zero, the clock can propagate up the DTS tree crossing clock domain boundaries without any problems. The requirement that the functional clock be set to zero can be eliminated by replacing some of the OR gates in figure 2 with multiplexers.

The DTS tree also complicates the routing of the design. This is particularly true for the centralized control case where global control signals need to be routed. Up to a certain point, the increase in routing complexity does not affect the design size. This is true as long as the design area does not have to be artificially expanded to allow more routing resources. For modern deep-sub-micron processes with many layers of metal, routing channels are rarely needed under any circumstances, so there is rarely extra area required for routing.

The routing overhead can be decreased and the general circuit performance can be increased by optimizing the assignment of the physical flip-flops to the tree locations. In the present implementation this is not done, as this is much more complicated and requires a complex analysis of the design connectivity with feedback from the placement engine. This optimization would allow for simpler routing as it would decrease the average length of the wires connecting nodes together. This is done by assigning nodes which are physically adjacent to each other in the layout, to nodes in the DTS tree which are electrically connected to each other.

A test chip was designed and fabricated in a 0.5 μm process to measure the power consumption savings. The measured results will be added before the final conference submission. For simulation results, the circuit was simulated in a digital simulator clocking in a random test pattern in both the conventional scan path and the DTS scan path. The total number of logic level toggles in the circuit as well as amount of decrease is reported in table II. As can be seen, there is a very substantial power savings associated with the DTS implementation compared to the standard scan implementation. For this case, the standard scan power consumption could be decreased by breaking the single scan chain to multiple smaller chains, however this does not scale well with circuit complexity. On the other hand, the DTS tree scales well with circuit size.

### Table II

<table>
<thead>
<tr>
<th>Test Pattern</th>
<th>Std. Scan</th>
<th>DTS Scan</th>
<th>Factor Decrease</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>257535</td>
<td>29025</td>
<td>8.6</td>
</tr>
<tr>
<td>2</td>
<td>258228</td>
<td>30913</td>
<td>8.35</td>
</tr>
<tr>
<td>3</td>
<td>252403</td>
<td>31769</td>
<td>7.94</td>
</tr>
<tr>
<td>4</td>
<td>263732</td>
<td>32009</td>
<td>8.24</td>
</tr>
</tbody>
</table>

### IV. Conclusion

In this paper the DTS scan tree is implemented in a number of test circuits and the hardware overhead is measured. A proposed clock gating scheme is used to further reduce power consumption by not requiring the functional clock tree to be active during test. Furthermore, the proposed DTS implementation does not require fully synchronous designs and allows the DTS trees to cross clock domains. The hardware overhead is found to be 16% for a centralized DTS control architecture. A test chip was created to verify the power saving attributes of the DTS architecture, and for the test chip, simulations show a power saving of 8 times.

### REFERENCES