Deterministic Built-In Self Test for Digital Circuits

Ahmad A. Al-Yamani
Department of Electrical Engineering
Stanford University
Ph.D. Orals Presentation
April 21, 2003
Outline

- Background
- Motivation
- Built-In Reseeding
- Seed Ordering
- Testing digital circuits with constraints
- Summary
VLSI Testing

- **Generate test patterns**
  - Applied through primary inputs and flip-flops

- **Simulate good circuit response**
  - On primary outputs and flip-flops

- **Apply patterns**

- **Compare**
  - simulated response
  - measured response
Design for Testability

Combinational Logic

I₁, I₂, ..., Iₙ → MUX → D → Q

SDI → MUX
SE → MUX
CLK

D → Q

... → MUX

D → Q

O₁, O₂, ..., Oₘ → SDO
Defects vs. Faults

- **Failure**
  - Deviation from designed characteristics

- **Defect**
  - Failure mode
  - Very hard to generate tests for 😞

- **Fault**
  - Models effect of failure on logical signals
  - Easy to generate tests for 😊
  - Imperfect 😞
Fault Models

- Single-stuck fault
  - Signal line stuck at 0 or 1 (SSF)
- Transition fault
  - A node doesn’t switch within allowed time
- Bridging fault
- Stuck Open

![Fault Model Diagram]
Test Pattern Generation

Circuit HDL Netlist:
Module core(in, out)
Begin
...
End

Requirements:
• Fault model
• Fault coverage
• ...

ATPG TOOL

Test Patterns:
• 100110 ...
• 010110 ...

Fault Lists:
• Detected
• Undetected
Built-In Self Test

- **On-chip circuitry**
  - Pattern generation
    - Linear feedback shift register (LFSR)
  - Output response analysis
- **Pseudorandom patterns**
- **Circuit response simulated**
- **Additional circuitry to start and stop testing**
Built-In Self Test

Combination Logic

Scan Chain(s)

LBIST Controller

PIs

POs
Why Built-In Self Test?

- **Low cost 😊**
  - Low tester memory requirement
    - Generation and compression on-chip

- **Circuit speed 😊**
  - Limited speed between tester and chip

- **Applicability in field 😊**
  - Tester inside chip

- **Unmodeled defect coverage 😊**
Why NOT Built-In Self Test?

- **Contention 😞**
  - Pseudorandom patterns
  - Constraints on signals
    - Circuit damaged
    - Test results corrupted

- **Low fault coverage 😞**
  - Doesn’t target specific faults

- **Long test length 😞**
  - To achieve reasonable fault coverage

- **Area overhead 😞**
Outline

- Background
- Motivation
- Built-In Reseeding
- Seed Ordering
- Testing digital circuits with constraints
- Summary
Motivation (1)

- BIST becoming more popular
  - Commercially automated
  - Devices getting smaller
  - External testing getting more difficult
  - Silicon area less critical
Motivation (2)

- Pseudorandom BIST problems
  - Contention
  - Low fault coverage
  - Long test length
Motivation (3)

Pseudorandom BIST [10]

- Test thoroughness 😊
  - Better than theoretical estimation
- Collateral fault coverage 😊
Contributions (1)

- Built-in reseeding [1,2,3,8]
  - On-chip deterministic test
- Seed Ordering [4]
  - Minimizing overhead through seed reordering
- Seed Encoding [5]
  - Minimizing test storage and test time
  - Not covered today
Contributions (2)

- Testing circuits with constraints [6,7,9]
  - Built-in self test for circuits with constraints
- Defect based test [10,11]
  - BIST: theory, simulation and tester data
  - Defects vs. faults
  - Not covered today
Outline

- Background
- Motivation
- Built-In Reseeding
- Seed Ordering
- Testing digital circuits with constraints
- Summary
Random Pattern Resistant (r.p.r.) Faults

- Faults with low detectability
  - Few patterns detect them
  - Missed by pseudorandom patterns

- Solutions
  - Modifying the circuit
  - Weighted pseudorandom patterns
  - Mixed mode testing
Mixed Mode Testing

- **Pseudorandom phase**
  - Pseudorandom patterns
  - Easy faults detected

- **Deterministic phase**
  - Deterministic patterns for r.p.r. faults
  - From tester or on-chip ROM
Objective

- BIST
- Maximum fault coverage
  - Stuck-at and transition faults
- No test point insertion
- No design modification
- No performance overhead
Previous Work

- PRPG reseeding [Koenemann 91]
  - Seed = PRPG size << pattern size 😊
- External tester or on-chip ROM 😞
- No external testing, No ROM 😊
  - [Kim 95, Crouch 95, Savir 90]
  - But pseudorandom seeds ⇒ No 100% FC 😞
- Mapping logic [Touba 95]
  - No ATE, No ROM & deterministic patterns
Seed Size

- Test patterns
  - Unspecified bits (don’t cares)
- PRPG size = $S_{max} +20$
- $S_{max} = \text{Max } \# \text{ of specified bits in a pattern}$
- $S_{max} \sim [2\% - 10\%]$ of total bits
Linear Feedback Shift Register
Main Contributions

- **Built-In Reseeding**
  - On-chip hardware based reseeding

- **Ordering the seed**
  - Minimizing number of seeds to be loaded

- **Seed Encoding**
  - Reducing seed size and test time

- **Illegal State Detection**
  - Enabling BIST for circuits with constraints
Built-In Reseeding Logic

Reseeding Logic
## Reseeding Circuit Example

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I1</th>
<th>I2</th>
<th>I3</th>
<th>I4</th>
<th>O1</th>
<th>O2</th>
<th>O3</th>
<th>O4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
</tbody>
</table>

Note: 'd' represents a don't care state.
Reseeding Circuit Hardware
System Level View

Reeseeding Circuit

LFSR

Control Signal Generator

Pattern Counter

SE & TM

Combinational Logic

Scan Chain(s)

LBIST Controller
Reseeding Algorithm

- Pseudorandom patterns initially
- ATPG patterns for undetected faults
  - Convert them to seeds (system of equations)
- When a seed is loaded
  - Run in pseudorandom mode
- Load a new seed
  - Iff pseudorandom patterns are NOT effective
- CIT = Coverage Improvement Threshold
  - Measures effectiveness of PR patterns
### Benchmark Circuits Characteristics

<table>
<thead>
<tr>
<th>Circuit Name</th>
<th>PIs</th>
<th>POs</th>
<th>Scan Chain Size</th>
<th>LFSR Size</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1423</td>
<td>17</td>
<td>5</td>
<td>74</td>
<td>50</td>
<td>4,531</td>
</tr>
<tr>
<td>s1488</td>
<td>8</td>
<td>19</td>
<td>6</td>
<td>6</td>
<td>3,555</td>
</tr>
<tr>
<td>s1494</td>
<td>8</td>
<td>19</td>
<td>6</td>
<td>6</td>
<td>3,563</td>
</tr>
<tr>
<td>s5378</td>
<td>35</td>
<td>49</td>
<td>179</td>
<td>61</td>
<td>14,376</td>
</tr>
<tr>
<td>s9234</td>
<td>36</td>
<td>39</td>
<td>211</td>
<td>80</td>
<td>25,840</td>
</tr>
<tr>
<td>s13207</td>
<td>62</td>
<td>152</td>
<td>638</td>
<td>45</td>
<td>44,255</td>
</tr>
<tr>
<td>s15850</td>
<td>77</td>
<td>150</td>
<td>534</td>
<td>150</td>
<td>48,494</td>
</tr>
<tr>
<td>s35932</td>
<td>35</td>
<td>320</td>
<td>1728</td>
<td>60</td>
<td>106,198</td>
</tr>
<tr>
<td>s38417</td>
<td>28</td>
<td>106</td>
<td>1636</td>
<td>200</td>
<td>120,180</td>
</tr>
<tr>
<td>s38584</td>
<td>38</td>
<td>304</td>
<td>1426</td>
<td>200</td>
<td>115,855</td>
</tr>
</tbody>
</table>
## Comparison with Previous Work

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Number of seeds to be encoded</th>
<th>One seed per pattern</th>
<th>Built-in reseeding</th>
<th>Reduction%</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1423</td>
<td>7</td>
<td>4</td>
<td>7</td>
<td>43</td>
</tr>
<tr>
<td>s1488</td>
<td>4</td>
<td>1</td>
<td>7</td>
<td>75</td>
</tr>
<tr>
<td>s1494</td>
<td>4</td>
<td>1</td>
<td>7</td>
<td>75</td>
</tr>
<tr>
<td>s5378</td>
<td>60</td>
<td>20</td>
<td>6</td>
<td>67</td>
</tr>
<tr>
<td>s9234</td>
<td>93</td>
<td>62</td>
<td>3</td>
<td>33</td>
</tr>
<tr>
<td>s13207</td>
<td>121</td>
<td>26</td>
<td>7</td>
<td>79</td>
</tr>
<tr>
<td>s15850</td>
<td>41</td>
<td>38</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>s35932</td>
<td>18</td>
<td>1</td>
<td>9</td>
<td>94</td>
</tr>
<tr>
<td>s38417</td>
<td>78</td>
<td>62</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>s38584</td>
<td>107</td>
<td>36</td>
<td>6</td>
<td>66</td>
</tr>
</tbody>
</table>
## Test Length Comparison

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Built-In Reseeding</th>
<th>Pseudorandom BIST</th>
<th>Reduction %</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1423</td>
<td>8K</td>
<td>500K</td>
<td>98%</td>
</tr>
<tr>
<td>s1488</td>
<td>6.5K</td>
<td>10K</td>
<td>35%</td>
</tr>
<tr>
<td>s1494</td>
<td>6.5K</td>
<td>10K</td>
<td>35%</td>
</tr>
<tr>
<td>s5378</td>
<td>1K</td>
<td>100K</td>
<td>99%</td>
</tr>
<tr>
<td>s9234</td>
<td>4.5K</td>
<td>500K</td>
<td>99%</td>
</tr>
<tr>
<td>s13207</td>
<td>1.6K</td>
<td>500K</td>
<td>99%</td>
</tr>
<tr>
<td>s15850</td>
<td>2.4K</td>
<td>500K</td>
<td>99%</td>
</tr>
<tr>
<td>s35932</td>
<td>6K</td>
<td>500K</td>
<td>98%</td>
</tr>
<tr>
<td>s38417</td>
<td>6K</td>
<td>500K</td>
<td>98%</td>
</tr>
<tr>
<td>s38584</td>
<td>60K</td>
<td>500K</td>
<td>88%</td>
</tr>
</tbody>
</table>
## Area Overhead (SSF)

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Area</th>
<th>Reseeding Area</th>
<th>Area Overhead%</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1423</td>
<td>4,531</td>
<td>113.2</td>
<td>2.5</td>
</tr>
<tr>
<td>s1488</td>
<td>3,555</td>
<td>12</td>
<td>0.3</td>
</tr>
<tr>
<td>s1494</td>
<td>3,563</td>
<td>12</td>
<td>0.3</td>
</tr>
<tr>
<td>s5378</td>
<td>14,376</td>
<td>462</td>
<td>3.2</td>
</tr>
<tr>
<td>s9234</td>
<td>25,840</td>
<td>3566</td>
<td>13.8</td>
</tr>
<tr>
<td>s13207</td>
<td>44,255</td>
<td>122</td>
<td>0.3</td>
</tr>
<tr>
<td>s15850</td>
<td>48,494</td>
<td>1988</td>
<td>4.1</td>
</tr>
<tr>
<td>s35932</td>
<td>106,198</td>
<td>52</td>
<td>0.05</td>
</tr>
<tr>
<td>s38417</td>
<td>120,180</td>
<td>9418</td>
<td>7.8</td>
</tr>
<tr>
<td>s38584</td>
<td>115,855</td>
<td>1760</td>
<td>1.5</td>
</tr>
<tr>
<td>Circuit</td>
<td>Area</td>
<td>Reseeding Area</td>
<td>Area Overhead%</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>s1423</td>
<td>4,531</td>
<td>205.2</td>
<td>4.5</td>
</tr>
<tr>
<td>s1488</td>
<td>3,555</td>
<td>35.2</td>
<td>1.0</td>
</tr>
<tr>
<td>s1494</td>
<td>3,563</td>
<td>45.2</td>
<td>1.3</td>
</tr>
<tr>
<td>s5378</td>
<td>14,376</td>
<td>855.2</td>
<td>5.9</td>
</tr>
<tr>
<td>s9234</td>
<td>25,840</td>
<td>3937.6</td>
<td>15.2</td>
</tr>
<tr>
<td>s13207</td>
<td>44,255</td>
<td>225.2</td>
<td>0.5</td>
</tr>
<tr>
<td>s15850</td>
<td>48,494</td>
<td>2513.6</td>
<td>5.2</td>
</tr>
<tr>
<td>s35932</td>
<td>106,198</td>
<td>174</td>
<td>0.2</td>
</tr>
<tr>
<td>s38417</td>
<td>120,180</td>
<td>3022.8</td>
<td>2.5</td>
</tr>
<tr>
<td>s38584</td>
<td>115,855</td>
<td>2668.4</td>
<td>2.3</td>
</tr>
</tbody>
</table>
Conclusions (Built-In Reseeding)

- Maximum fault coverage fully with BIST
- Better trade off between area and TL
- Directly applicable for SSF and transition
  - Suitable for different fault-models
- Low area overhead
Main Contributions

- **Built-In Reseeding**
  - On-chip hardware based reseeding

- **Ordering the seed**
  - Minimizing number of seeds to be loaded

- **Seed Encoding**
  - Reducing seed size and test time

- **Illegal State Detection**
  - Enabling BIST for circuits with constraints
Reseeding

❖ Seed

➤ *Initial State* for PRPG
➤ *m* clock cycles to fill chains
➤ After *m* cycles, *Final State*

![Diagram](Diagram.png)
Problem Definition

- The order at which seeds are applied
  - Impacts hardware overhead
  - Impacts seed storage
- Optimizing the order
Related Work

- **Discrete logarithms [Lempel 95]**
  - Location of the pattern in the LFSR sequence
  - High computational complexity 😞

- **Simulation [Fagot 99]**
  - Simulate different sequences of LFSR
  - Pick one that will generate most patterns
  - Unnecessary
Seed Ordering

- Seed = Initial state of PRPG $s(0)$
- After filling the scan chain, final state $s(m+1)$
- If $s(m+1)$ can generate another pattern
  - No need to load a seed for that pattern
  - If not, try for few more cycles - user specified
- While solving for the seeds
  - Increase the chance for a match
Seed Ordering Flow Chart

1. Apply PR patterns until no coverage improvement
2. Generate test patterns \( \{P\} \) for undetected faults
3. Pick a pattern, calculate its seed, and add it to \( \{S\} \)
4. Find a seed \( s_i(0) \) s.t. \( s_i(m+1) \) generates a pattern \( k \)

- Is such a seed found?
  - NO: \( S = S + \{s_k(0)\} \)
  - YES: \( P = P - \{k\} \)

- Is \( P \) empty?
  - NO: Return to step 3
  - YES: END
Seed Calculation

\[ s_i(m + 1) = s_i(0) \times H^{(m+1)} \]

- H = LFSR transition matrix
- Only one matrix multiplication
  \( H^{(m+1)} \) precomputed only once.
- Simulation techniques
  \( \text{Simulate every step of LFSR} \)
## Benchmark Circuits Characteristics

<table>
<thead>
<tr>
<th>Circuit Name</th>
<th>PIs</th>
<th>POs</th>
<th>Scan Chain Size</th>
<th>LFSR Size</th>
<th>Cell Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1238</td>
<td>14</td>
<td>14</td>
<td>18</td>
<td>15</td>
<td>2,740</td>
</tr>
<tr>
<td>s1423</td>
<td>17</td>
<td>5</td>
<td>74</td>
<td>50</td>
<td>4,531</td>
</tr>
<tr>
<td>s1488</td>
<td>8</td>
<td>19</td>
<td>6</td>
<td>6</td>
<td>3,555</td>
</tr>
<tr>
<td>s1494</td>
<td>8</td>
<td>19</td>
<td>6</td>
<td>6</td>
<td>3,563</td>
</tr>
<tr>
<td>s5378</td>
<td>35</td>
<td>49</td>
<td>179</td>
<td>61</td>
<td>14,377</td>
</tr>
<tr>
<td>s9234</td>
<td>36</td>
<td>39</td>
<td>211</td>
<td>80</td>
<td>25,840</td>
</tr>
<tr>
<td>s13207</td>
<td>62</td>
<td>152</td>
<td>638</td>
<td>45</td>
<td>44,255</td>
</tr>
<tr>
<td>s15850</td>
<td>77</td>
<td>150</td>
<td>534</td>
<td>150</td>
<td>48,494</td>
</tr>
<tr>
<td>s38584</td>
<td>38</td>
<td>304</td>
<td>1426</td>
<td>200</td>
<td>115,855</td>
</tr>
</tbody>
</table>
## Comparison with Seed Per Pattern

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Cell Area</th>
<th>Seed/Pattern</th>
<th>Our Tech.</th>
<th>Red. %</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1238</td>
<td>2,740</td>
<td>30</td>
<td>7</td>
<td>77</td>
</tr>
<tr>
<td>s1423</td>
<td>4,531</td>
<td>7</td>
<td>3</td>
<td>57</td>
</tr>
<tr>
<td>s1488</td>
<td>3,555</td>
<td>4</td>
<td>1</td>
<td>75</td>
</tr>
<tr>
<td>s1494</td>
<td>3,563</td>
<td>4</td>
<td>1</td>
<td>75</td>
</tr>
<tr>
<td>s5378</td>
<td>14,377</td>
<td>35</td>
<td>7</td>
<td>80</td>
</tr>
<tr>
<td>s9234</td>
<td>25,840</td>
<td>89</td>
<td>57</td>
<td>36</td>
</tr>
<tr>
<td>s13207</td>
<td>44,255</td>
<td>74</td>
<td>12</td>
<td>84</td>
</tr>
<tr>
<td>s15850</td>
<td>48,494</td>
<td>38</td>
<td>35</td>
<td>8</td>
</tr>
<tr>
<td>s38584</td>
<td>115,855</td>
<td>55</td>
<td>16</td>
<td>71</td>
</tr>
</tbody>
</table>
## Comparison with Arbitrary Ordering

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Cell Area</th>
<th>Number of Seeds</th>
<th>Order1</th>
<th>Order2</th>
<th>Order3</th>
<th>Our Technique</th>
<th>Reduction %</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1238</td>
<td>2,740</td>
<td></td>
<td>28</td>
<td>27</td>
<td>28</td>
<td>7</td>
<td>75</td>
</tr>
<tr>
<td>s1423</td>
<td>4,531</td>
<td></td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>50</td>
</tr>
<tr>
<td>s1488</td>
<td>3,555</td>
<td></td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>73</td>
</tr>
<tr>
<td>s1494</td>
<td>3,563</td>
<td></td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>73</td>
</tr>
<tr>
<td>s5378</td>
<td>14,377</td>
<td></td>
<td>32</td>
<td>33</td>
<td>32</td>
<td>7</td>
<td>78</td>
</tr>
<tr>
<td>s9234</td>
<td>25,840</td>
<td></td>
<td>87</td>
<td>85</td>
<td>87</td>
<td>57</td>
<td>34</td>
</tr>
<tr>
<td>s13207</td>
<td>44,255</td>
<td></td>
<td>66</td>
<td>65</td>
<td>64</td>
<td>12</td>
<td>82</td>
</tr>
<tr>
<td>s15850</td>
<td>48,494</td>
<td></td>
<td>37</td>
<td>38</td>
<td>36</td>
<td>35</td>
<td>5</td>
</tr>
<tr>
<td>s38584</td>
<td>115,855</td>
<td></td>
<td>47</td>
<td>50</td>
<td>48</td>
<td>16</td>
<td>67</td>
</tr>
</tbody>
</table>
Built-In Reseeding Circuit

Reseeding Logic
## Hardware Overhead Comparison

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Cell Area</th>
<th>Area Overhead %</th>
<th>Our Technique</th>
<th>Reduction %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Order1 Order2 Order3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1238</td>
<td>2,740</td>
<td>39.6 37.0 42.7</td>
<td>9.9</td>
<td>75</td>
</tr>
<tr>
<td>s1423</td>
<td>4,531</td>
<td>5.8   4.8 4.6</td>
<td>2.1</td>
<td>59</td>
</tr>
<tr>
<td>s1488</td>
<td>3,555</td>
<td>2.1   2.0 1.6</td>
<td>0.3</td>
<td>84</td>
</tr>
<tr>
<td>s1494</td>
<td>3,563</td>
<td>2.1   2.0 1.6</td>
<td>0.3</td>
<td>84</td>
</tr>
<tr>
<td>s5378</td>
<td>14,377</td>
<td>13.0  16.6 15.6</td>
<td>3.1</td>
<td>80</td>
</tr>
<tr>
<td>s9234</td>
<td>25,840</td>
<td>14.4  13.8 14.4</td>
<td>13.2</td>
<td>7</td>
</tr>
<tr>
<td>s13207</td>
<td>44,255</td>
<td>2.0   1.8 1.9</td>
<td>0.3</td>
<td>84</td>
</tr>
<tr>
<td>s15850</td>
<td>48,494</td>
<td>4.1   3.7 4.2</td>
<td>3.5</td>
<td>13</td>
</tr>
<tr>
<td>s38584</td>
<td>115,855</td>
<td>1.5   1.4 1.5</td>
<td>1.2</td>
<td>20</td>
</tr>
</tbody>
</table>
Application to Real Silicon

- Test chip experiment
  - Compare
    - Actual production defects
    - Fault models
  - How
    - Collect
      - Defective chips
      - Test sets
      » Various test metrics & conditions
ELF35 Chip

- LSI Logic G10P technology
- $L_{\text{eff}} = 0.35 \ \mu m$
- Nominal $V_{DD} = 3.3 \ \text{V}$

6 core designs

- 4 Combinational core designs
  - 1 translator, 3 datapath
- 2 Sequential (2901’s)
  - full scan
## ELF35 Cores

<table>
<thead>
<tr>
<th>Name</th>
<th>Inputs</th>
<th>Outputs</th>
<th>Gate count</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSI2901</td>
<td>61</td>
<td>64</td>
<td>12,338</td>
<td>Arithmetic processor LSI Logic</td>
</tr>
<tr>
<td>TOPS2901</td>
<td>57</td>
<td>48</td>
<td>18,090</td>
<td>Arithmetic processor generated by TOPS tool</td>
</tr>
<tr>
<td>MULT-ADDER</td>
<td>65</td>
<td>33</td>
<td>4,499</td>
<td>Multiplier followed by an adder</td>
</tr>
<tr>
<td>M12</td>
<td>24</td>
<td>12</td>
<td>1,309</td>
<td>12 × 12 multiplier</td>
</tr>
<tr>
<td>SQR</td>
<td>12</td>
<td>6</td>
<td>538</td>
<td>6 × 6 multiplier followed by a squarer</td>
</tr>
<tr>
<td>PB</td>
<td>12</td>
<td>12</td>
<td>17,468</td>
<td>Pseudo-random-to-binary translator</td>
</tr>
</tbody>
</table>
## Number of ELF35 Defective Cores

<table>
<thead>
<tr>
<th></th>
<th>LSI</th>
<th>TOPS</th>
<th>MA</th>
<th>M12</th>
<th>SQR</th>
<th>PB</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>FATS</td>
<td>63</td>
<td>18</td>
<td>16</td>
<td>17</td>
<td>7</td>
<td>102</td>
<td>223</td>
</tr>
<tr>
<td>FOSTS</td>
<td>26</td>
<td>11</td>
<td>11</td>
<td>17</td>
<td>7</td>
<td>29</td>
<td>101</td>
</tr>
<tr>
<td>Total</td>
<td>89</td>
<td>29</td>
<td>27</td>
<td>34</td>
<td>14</td>
<td>131</td>
<td>324</td>
</tr>
</tbody>
</table>
Test Escapes

- Undetected Defective Cores
## Test Escapes

<table>
<thead>
<tr>
<th>Core</th>
<th>100 % SSF Set</th>
<th>Reseeding</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSI2901</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>TOPS2901</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MA</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>M12</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SQR</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PB</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>5/101</strong></td>
<td><strong>0/101</strong></td>
</tr>
</tbody>
</table>
Main Contributions

- **Built-In Reseeding**
  - On-chip hardware based reseeding

- **Ordering the seed**
  - Minimizing number of seeds to be loaded

- **Seed Encoding**
  - Reducing seed size and test time

- **Illegal State Detection**
  - Enabling BIST for circuits with constraints
Seed Encoding

- Encode a seed
  - Number of cycles needed by generator
- Save on seed size
- Save on test time
Seed Encoding Seed Size

Seed size example

- Core with 10,000 flip-flops / 10 scan chains
  - Chain depth = 1000 bits
- $S_{max} = 500$ (5%)
- PRPG size = 520 flip-flops = seed size
- Bit counter size = $\lceil \log(1000) \rceil = 10$ flops
  - Size of encoded seed is 10 bits vs. 520 bits
  - 98% (52x) reduction in seed size
Seed Encoding Test Time

- Test Time
  - Same example, 520 cycles to load PRPG
  - 11 cycles to load bit counter
  - Search for a seed match in up to 508 cycles
    - Saving in test time
    - Saving in test storage
Conclusions (Seed Ordering)

- A seed ordering technique
- Substantial reduction in:
  - Area overhead
  - Storage on tester
- Why does it work?
  - LFSR linearity
  - Avoids unnecessary complexity
  - Exploits don’t cares in test patterns
Main Contributions

- **Built-In Reseeding**
  - On-chip hardware based reseeding

- **Ordering the seed**
  - Minimizing number of seeds to be loaded

- **Seed Encoding**
  - Reducing seed size and test time

- **Illegal State Detection**
  - Enabling BIST for circuits with constraints
Purpose

- Pseudorandom testing
  - Digital Circuits with Constraints on Signals
    - e.g. One-hot signals
One-Hot Signals

- One-Hot Condition
  - Exactly one signal active at a time

- Multiple-Hot Problem
  - Multiple lines driving output
    - CONTENTION

- Zero-Hot Problem
  - No line driving output
    - Floating output
One-Hot Signals Sources

- **Busses**
  - Controlled by tri-state drivers

- **N-to-1 Selectors**
  - Implemented with N transmission gates

- **Scan-based pseudorandom test**
  - One-hot condition may not be satisfied
### Example

<table>
<thead>
<tr>
<th>$E_1$</th>
<th>$E_2$</th>
<th>$I_1$</th>
<th>$I_2$</th>
<th>$O$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>✓</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>🔥</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
</tbody>
</table>
Problem Definition

- **Scan-based Pseudorandom Testing**
  - Ensure one-hot signals
    - Scan and Capture cycles
    - No fault coverage loss
Scan Operation

- Standard fix for one-hot signals
Previous Work

- **Priority encoder** [Fleming92, Mitra97]
  - Insert a priority encoder at the 1-hot signals
  - Delay 😞
    - Several levels of logic
  - Specific to 1-hot problem 😞
Previous Work

- Fixed value throughout test
  - [Levitt95, Hetherington99, Raina00]
    - Fault coverage 😞
Illegal State Detection (ISD-Trace)

- **ISD in terms of PIs and FFs**
  - Back trace one-hots to PIs and FFs
  - \( E_1 = F_1(\text{PIs, FFs}) \)
  - \( E_2 = F_2(\text{PIs, FFs}) \)
  - \( E_3 = F_3(\text{PIs, FFs}) \)
  - \( \text{ISD} = (F_1 F_2 \neg F_3 + F_1 \neg F_2 F_3 + F_1 \neg F_2 \neg F_3) \)
ISD - BIST

- BIST
- Test set known
- ISD-BIST
  - Logical Sum of patterns causing illegal values
    - Logic simulation

\[ ISD = \sum (\text{IllegalPatterns}) \]
ISD with BIST

- **PRPG**
- **ISD**
- **Fixing Logic**
- **SE & TM**
- **Control Signals Generator**
- **Combinational Logic**
- **Scan Chain(s)**
- **LBIST Controller**

Input pins (PIs) and output pins (POs) are connected through the diagram.
Fixing Logic

ISD Circuit

TM

SE

PIs

Bistables

Bus

E1

E2

E3

E4

Fixing logic
ISD + Fixing Logic

- Small delay😊
- During scan-in, scan-out and capture😊
- No fault coverage sacrifice😊
- Small ISD ⇒ low area overhead😊

But …

- Delay😊
- Design change😊
Skip Illegal Patterns

- Instead of fixing logic
  - Skip bad patterns
- Change SE
- Use bit counter

```
SE -> New SE
ISD
End of scan shift
```
ITC99 Benchmark

- Only benchmark with tri-state drivers

<table>
<thead>
<tr>
<th>Pls</th>
<th>POs</th>
<th>InOuts</th>
<th>Bistables</th>
<th>Busses</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>88</td>
<td>176</td>
<td>508</td>
<td>6</td>
<td>29817</td>
</tr>
</tbody>
</table>

- Only benchmark with tri-state drivers
## Results

<table>
<thead>
<tr>
<th>Busses</th>
<th>Bus width</th>
<th>Tri-state drivers</th>
<th>ISD circuit area</th>
<th>%Area overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus 1</td>
<td>17</td>
<td>2</td>
<td>10</td>
<td>0.03</td>
</tr>
<tr>
<td>Bus 2</td>
<td>17</td>
<td>2</td>
<td>10</td>
<td>0.03</td>
</tr>
<tr>
<td>Bus 3</td>
<td>6</td>
<td>3</td>
<td>40</td>
<td>0.1</td>
</tr>
<tr>
<td>Bus 4</td>
<td>6</td>
<td>5</td>
<td>82</td>
<td>0.3</td>
</tr>
<tr>
<td>Bus 5</td>
<td>6</td>
<td>7</td>
<td>93</td>
<td>0.3</td>
</tr>
<tr>
<td>Bus 6</td>
<td>6</td>
<td>3</td>
<td>31</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Summary and Conclusions (ISD)

- ISD technique applicability
  - IC production, board-level and system-level
- No fault coverage sacrifice
- Non-intrusive
- Very low area overhead
- Very low delay overhead
- Any constraints, not only one-hot
Outline

- Background
- Motivation
- Built-In Reseeding
- Seed Ordering
- Testing digital circuits with constraints
- Summary
Summary

- **Built-In Reseeding**
  - Maximum fault coverage
  - Fully BIST
  - Better trade off between area and TL
  - Fault-model independent
  - Low area overhead
Summary

- **Seed Ordering**
  - Substantial reduction in:
    - Area overhead
    - Storage on tester

- **Seed Encoding**
  - Reducing seed size and test time
  - Slight architecture change
Summary

- **Illegal state detection**
  - BIST with no illegal states
  - No fault coverage sacrifice
  - Non-intrusive
  - Very low overhead
  - Any constraints, not only one-hot


Acknowledgements

 Advisor: Prof. McCluskey
 Committee: Prof. Gill, Prof. DeMicheli, Prof. Widrow
 Administrative help: Siegrid, Valerie, Eileen, Diane
 Funding: KFUPM, LSI Logic
 Tester and Support:
   Advantest: Mike Purtell, Arnaud Stehle
 RATS: Anjana, Chaowen, Catherine, Eddie, Edward, Erik C, Erik V, Ernesto, Francois, Hong, James, Mehdi, Nahmsuk, Philip, Ray, Robert, Sam, Santiago, Sungroh, Vinod and others
 CRC visitors & affiliates: Prof. Abraham, Prof. Koren, Prof. Mitra, Prof. Saxena
Acknowledgements

- **My family**
  - My wife, Alaa
  - My son, Abdul
  - My parents, brother and sisters
  - My In-laws

- **My friends at Stanford**

- **My teachers and professors**

- **My friends at KFUPM**
THANK YOU !
References (1)


References (2)

References (3)

- [Raina 00], R., *et al.*, “DFT Advances in Motorola's Next-Generation 74xx PowerPC™ Microprocessor,” *Proc. ITC00*.