Outline

- BIST ... Why? What? & How?
- BIST randomness and RPR faults
- Contention in BIST
- Previous Work
- Solutions
  - Full decoding
  - Condition checking
  - Pattern mapping
  - State skipping
- Conclusions

Built-In Self-Test ... Why?

- External Testers
  - High coverage
  - No additional area overhead
  - No performance degradation
  - High cost
  - Large test application time
  - Large test patterns generation time

Built-In Self-Test ... What?

- Move tester inside the chip
- ROM-based
  - High coverage
  - Large area overhead
  - Large test pattern generation time
- LFSR-based

Built-In Self-Test ... What?

- LFSR
  - Test Generation
    - Random patterns
    - Fast generation
    - Small HW overhead
    - Exhaustive testing ↔ Primitive polynomial LFSR
  - Output Compression
    - Minimal extra HW
  - Signature Analysis
    - Go/No go

Built-In Self-Test ... How?

- Circuit Under Test (CUT)
  - LFSR
  - Scan Chain
  - Signature Reg.
  - Serial BIST (“Test per scan”)
- Circuit Under Test (CUT)
  - LFSR
  - MSR
  - Parallel BIST (“Test per clock”)
**Built-In Self-Test Randomness**

- LFSR ⇒ Random patterns
  - Easy faults
    - Detectable by many patterns
  - Hard faults (Random Pattern Resistant)
    - Detectable by few patterns
    - Solutions
      - Generate more LFSR patterns
      - Apply deterministic patterns

**RPR Faults Solution**

- CUT Modification
  - Test point insertion
- Pseudo-deterministic
  - Weighted patterns
  - Pattern mapping
  - Seed selection
    - Analytically
    - By Simulation
  - Re-seeding ⇒ seeds storage

**Pattern Mapping [1]**

- Given an LFSR and a test length
- Find a cube with no fault dropping patterns
- Map to an image cube to increase coverage
- Add the additional H/W to the LFSR

**Contention in BIST**

- LFSR ⇒ Random patterns
- How about CUTs with restrictions?
  - e.g. with tri-state busses

**Contention ... So What?**

- More than one tri-state enabled
  - Undetermined value on the bus
    - Unknown Output and Signature
- No tri-state is enabled
  - Floating bus
    - Same problem

**Previous Work on Contention [3]**

- Illinois Scan Architecture
Contention Solutions

- CUT Modification
  - Full Decoding
  - One Hot Condition Checking
- LFSR
  - Pattern Mapping
  - State Skipping

Full Decoding [2]

- Fully decode the tri-state buffers input

Choose 2 inputs based on coverage

- Performance degradation
- Intellectual Property problems

One Hot Condition Checking

- Add an external circuit to check for one hot condition

Z might be large
Need to express enables in terms of ffs
No performance degradation

Pattern Mapping

1. Given LFSR and test length
2. Simulate the coverage
3. If any patterns cause contention
   a. Choose source cubes
   b. Choose image cubes
   c. Perform the mapping
4. Loop back to 2

Pattern Mapping (Source Cubes)

\[ CP = \sum (\text{Contention Patterns}) \]

- Apply heuristic minimization to \( CP \)
  - e.g. Espresso (expand, irredundant, reduce)
- Choose all terms in \( CP \) as source cubes

Pattern Mapping (Image Cubes)

- Patterns computed for undetected faults
  - Choose an image cube from those patterns
  - Maximize coverage
- No patterns computed
  - Choose a cube from \( CP \setminus F' \)

\[ F' = \sum (\text{Fault Dropping Patterns}) \]
State Skipping

- Identify contention patterns in the test sequence
- Add logic to the LFSR to skip those patterns

Example (LFSR Sequence)

<table>
<thead>
<tr>
<th>State</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>State</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>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>15</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Example

- Assume test length = 10
  - Since we get 100 FC at pattern 9
  \[ CP = \sum (S_1, S_3, S_5, S_8) = \sum (12, 15, 11, 13) \]
  \[ CP = ABC' + ACD \]
  \[ F = \sum (S_0, S_2, S_4, S_6, S_7, S_9) = \sum (8, 14, 7, 5, 10, 6) \]
  \[ F = A'BD + AB'D' + BCD' \]

Example (Pattern Mapping HW)
Example (State Skipping)

- Skip 110* to 111*
- Skip 1*11 to 0*11

<table>
<thead>
<tr>
<th>State</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>State</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>
<td>0</td>
<td>5</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>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Example (State Skipping)

- New contention patterns appear
- \( CP = AD' + BC'D' + A'B'CD' \)
- Skip 1** to 0**
- Skip *100 to *110

<table>
<thead>
<tr>
<th>State</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>State</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>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Coverage compromised

Conclusion

- For contention free BIST

<table>
<thead>
<tr>
<th></th>
<th>Computational Effort</th>
<th>Coverage Improvement</th>
<th>Performance Degradation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Decoding</td>
<td>😊</td>
<td>😊</td>
<td>😞</td>
</tr>
<tr>
<td>Condition Checking</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
</tr>
<tr>
<td>Pattern Mapping</td>
<td>😞</td>
<td>😊</td>
<td>😊</td>
</tr>
<tr>
<td>State Skipping</td>
<td>😞</td>
<td>😞</td>
<td>😊</td>
</tr>
</tbody>
</table>

References