Circular BIST - Organization

- Architecture
- Operation
- BIST Controller
- Selective Replacement
- Register Adjacency
- Limit Cycling
  - Design Guidelines
  - Hardware Solutions
- Benefits and Limitations
BILBO Concerns

- Why not run all BILBOs in MISR mode?
  - Let signatures be the next test patterns & forget about
    - register self-adjacency
    - test session scheduling
    - combining BILBOs to create large TPGs

- Do we need to read every MISR at end of BIST sequence?
  - Let signatures propagate through chain to output register

- Does every flip-flop need to be part of a BILBO?
  - Avoid critical timing paths

- Do we need polynomials for every register?
  - Different polynomials for different size registers
Circular BIST Design

Step 1: Extract flip-flops from design

Step 2: Replace flip-flops with BIST flip-flops

Step 3: Connect BIST flip-flops to form circular shift register
Circular BIST FFs

(a) CSTP flip-flop and modes of operation

(b) SST flip-flop and modes of operation

(c) Circular BIST flip-flop and modes of operation
Circular BIST

- BIST logic added to subset of flip-flops
  - Lower area overhead
  - Less performance penalty
    - Avoid critical timing paths
- CBIST control leads B0 & B1 facilitate
  - System mode - normal operation
  - Reset mode - initialization
  - Scan mode - scan testing
  - MISR mode – BIST
Circular BIST (cont)

- Compacts CUT output patterns (both POs and FF inputs)
  - simultaneously provides CUT input patterns (bypass PIs)
    - a *non-linear* feedback shift register since feedback via CUT
- Test-per-clock BIST
  - only one test session
- CUT may be sequential
  - Problem: initialize non-BIST FFs for reproducible results
    - CBIST FFs reset & CUT clocked \( k \) times to initialize non-BIST FFs
  - Advantage: reduced area overhead by selectively replacing FFs
- Embedded RAMs have separate BIST (run before CBIST)
  - RAM BIST also initializes RAM contents before CBIST

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Gates</th>
<th>FFs</th>
<th>RAM</th>
<th>CBIST FFs</th>
<th>Logic Overhead</th>
<th>Area Overhead</th>
<th>Fault Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ckt 1</td>
<td>15,394</td>
<td>790</td>
<td>8K</td>
<td>656</td>
<td>17.6%</td>
<td>13.1%</td>
<td>95.6%</td>
</tr>
<tr>
<td>Ckt 2</td>
<td>13,937</td>
<td>784</td>
<td>32K</td>
<td>575</td>
<td>18.8%</td>
<td>7.4%</td>
<td>94.2%</td>
</tr>
<tr>
<td>Ckt 3</td>
<td>16,820</td>
<td>920</td>
<td>0</td>
<td>739</td>
<td>18.9%</td>
<td>18.9%</td>
<td>91.6%</td>
</tr>
</tbody>
</table>
Complete CBIST Design

- Provides
  - Sequential logic BIST approach for system-level use
  - That can easily be automated
- For system-level use we need
  - Input isolation
  - Test controller
- Not incorporated in:
  - Circular Self-Test Path (CSTP)
  - Simultaneous Self-Test (SST)
Circular BIST

- Selective replacement of FFs with CBIST FFs reduces area overhead and performance penalty
  - Only about 60% to 70% of FFs need to be replaced
- Fault coverage was 91% to 98% for my applications
  - Used partial scan mode for improving fault coverage
  - Krasniewski & Pilarski claimed CSTP would get 100%
    - They ignored limit cycling
- Problems observed in my applications and by others
  - Register adjacency
  - Limit cycling
Circular BIST Scan Mode

Used to augment fault coverage for manufacturing test but not used in my system applications
Problems with Circular BIST

- Register adjacency
  - $Q_{i-1} \oplus Q_{i-1} = 0$
  - Solution: reorder chain
- Limit cycling
  - Solutions:
    - Apply partial scan vectors
    - Find head state
    - Re-seed CBIST chain using partial scan mode
    - Add FFs to circuits prone to limit cycling
      - Increases number of possible states
When to Worry about Limit Cycling?

- Circuits prone to limit cycling: \( N_{FF}/C_{in} < 2 \)
  - where \( N_{FF} = \) number of FFs
  - and \( C_{in} = \) number of inputs to largest logic cone
Circular BIST Summary

**Benefits**
- Easy to implement and easily automated
- Test-per-clock architecture
  - Only needs one test session
- Selective replacement of flip-flops
  - Low area overhead
  - Can avoid critical timing paths

**Limitations**
- Does not guarantee high fault coverage
- Register adjacency
- Limit cycling
## Table 9.1 Area overhead of Circular BIST

<table>
<thead>
<tr>
<th>BIST Approach</th>
<th>Flip-Flops</th>
<th>Exclusive-ORs</th>
<th>Multiplexers</th>
<th>Gates</th>
<th>Performance Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSTP</td>
<td>0</td>
<td>$N_{FF}$</td>
<td>$N_{FF} + N_{PI}$</td>
<td>$N_{FF}$</td>
<td>1 MUX + 1 gate</td>
</tr>
<tr>
<td>SST</td>
<td>0</td>
<td>$N_{FF}$</td>
<td>$N_{FF} + N_{PI}$</td>
<td>$N_{FF}$</td>
<td>1 MUX</td>
</tr>
<tr>
<td>Circular BIST</td>
<td>0</td>
<td>$N_{FF}$</td>
<td>$N_{PI}$</td>
<td>$2N_{FF}$</td>
<td>1 XOR + 1 gate</td>
</tr>
</tbody>
</table>

where: $N_{FF} =$ number of flip-flops in CUT replaced by BIST flip-flops  

$N_{PI} =$ number of primary input to CUT requiring input isolation