Built-In Self-Test (BIST)

Virendra Singh
Associate Professor
Computer Architecture and Dependable Systems Lab
Department of Electrical Engineering
Indian Institute of Technology Bombay
http://www.ee.iitb.ac.in/~viren/
E-mail: viren@ee.iitb.ac.in

EE-709: Testing & Verification of VLSI Circuits

Lecture 33 (08 April 2013)
BIST Architecture

Note: BIST cannot test wires and transistors:
- From PI pins to Input MUX
- From POs to output pins
Pattern Generation

- Store in ROM – too expensive
- Exhaustive
- Pseudo-exhaustive
- Pseudo-random (LFSR) – Preferred method
- Binary counters – use more hardware than LFSR
- Modified counters
- Test pattern augmentation
  - LFSR combined with a few patterns in ROM
  - Hardware diffracter – generates pattern cluster in neighborhood of pattern stored in ROM
Pseudo-Random Pattern Generation

- Standard Linear Feedback Shift Register (LFSR)
  - Produces patterns algorithmically – repeatable
  - Has most of desirable random # properties

- Need not cover all $2^n$ input combinations
- Long sequences needed for good fault coverage
Matrix Equation for Standard LFSR

\[
\begin{bmatrix}
X_0(t+1) \\
X_1(t+1) \\
\vdots \\
X_{n-3}(t+1) \\
X_{n-2}(t+1)
\end{bmatrix}
= 
\begin{bmatrix}
0 & 1 & 0 & \ldots & 0 & 0 \\
0 & 0 & 1 & \ldots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \ldots & 1 & 0 \\
0 & 0 & 0 & \ldots & 0 & 1 \\
1 & h_1 & h_2 & \ldots & h_{n-2} & h_{n-1}
\end{bmatrix}
\begin{bmatrix}
X_0(t) \\
X_1(t) \\
\vdots \\
X_{n-3}(t) \\
X_{n-2}(t) \\
X_{n-1}(t)
\end{bmatrix}
\]

\(X(t+1) = T_s X(t)\)  \(T_s\) is companion matrix
Response Compaction

- Severe amounts of data in CUT response to LFSR patterns – example:
  - Generate 5 million random patterns
  - CUT has 200 outputs
  - Leads to: \(5 \text{ million} \times 200 = 1 \text{ billion bits}\) response
- Uneconomical to store and check all of these responses on chip
- Responses must be compacted
Definitions

- **Aliasing** – Due to information loss, signatures of good and some bad machines match

- **Compaction** – Drastically reduce # bits in original circuit response – lose information

- **Compression** – Reduce # bits in original circuit response – no information loss – fully invertible (can get back original response)

- **Signature analysis** – Compact good machine response into good machine signature. Actual signature generated during testing, and compared with good machine signature

- **Transition Count Response Compaction** – Count # transitions from 0 → 1 and 1 → 0 as a signature
1’s Count Signature

(a) Logic simulation of good machine and fault a stuck-at-1.

(b) Transition counts of good and failing machines.
Transition Counting

Transition count:

\[ C(R) = \sum_{i} (r_i \oplus r_{i-1}) \]  for all \( m \) primary outputs

To maximize fault coverage:

- Make \( C(R0) \) – good machine transition count – as large or as small as possible
Transition Counting

(a) Logic simulation of good machine and fault a stuck-at-1.

(b) Transition counts of good and failing machines.
LFSR for Response Compaction

- Use *cyclic redundancy check code* (CRCC) generator (LFSR) for response compacter
- Treat data bits from circuit POs to be compacted as a decreasing order coefficient polynomial
- CRCC divides the PO polynomial by its characteristic polynomial
  - Leaves remainder of division in LFSR
  - Must initialize LFSR to *seed value* (usually 0) before testing
- After testing – compare signature in LFSR to known good machine signature
- Critical: *Must compute good machine signature*
Example Modular LFSR Response Compacter

- LFSR seed value is “00000”
## Polynomial Division

<table>
<thead>
<tr>
<th>Inputs</th>
<th>$x^0$</th>
<th>$x^1$</th>
<th>$x^2$</th>
<th>$x^3$</th>
<th>$x^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial State</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>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Logic Simulation:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Logic simulation: $Remainder = 1 + x^2 + x^3$

| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |

$0 \cdot x^0 + 1 \cdot x^1 + 0 \cdot x^2 + 1 \cdot x^3 + 0 \cdot x^4 + 0 \cdot x^5 + 0 \cdot x^6 + 1 \cdot x^7$
Symbolic Polynomial Division

\[ x^5 + x^3 + x + 1 \]

\[ x^7 \]
\[ x^7 + x^5 \]
\[ x^7 + x^5 + x^3 \]
\[ x^7 + x^5 + x^3 + x^2 \]
\[ x^7 + x^5 + x^3 + x^2 + x \]
\[ x^7 + x^5 + x^3 + x^2 + x + 1 \]

Remainder matches that from logic simulation of the response compacter!
Multiple-Input Signature Register (MISR)

- Problem with ordinary LFSR response compacter:
  - Too much hardware if one of these is put on each primary output (PO)
- Solution: MISR – compacts all outputs into one LFSR
  - Works because LFSR is linear – obeys superposition principle
  - Superimpose all responses in one LFSR – final remainder is XOR sum of remainders of polynomial divisions of each PO by the characteristic polynomial
MISR Matrix Equation

- $d_i(t)$ – output response on $PO_i$ at time $t$

$$
\begin{bmatrix}
X_0(t+1) \\
X_1(t+1) \\
\vdots \\
X_{n-3}(t+1) \\
X_{n-2}(t+1)
\end{bmatrix}
= 
\begin{bmatrix}
0 & 1 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & 1 & 0 \\
0 & 0 & \cdots & 0 & 1 \\
1 & h_1 & \cdots & h_{n-2} & h_{n-1}
\end{bmatrix}
\begin{bmatrix}
X_0(t) \\
X_1(t) \\
\vdots \\
X_{n-3}(t) \\
X_{n-2}(t)
\end{bmatrix}
+ 
\begin{bmatrix}
d_0(t) \\
d_1(t) \\
\vdots \\
d_{n-3}(t) \\
d_{n-2}(t)
\end{bmatrix}
$$
Modular MISR Example

Characteristic Polynomial \( x^3 + x + 1 \)

\[
\begin{bmatrix}
X_0(t + 1) \\
X_1(t + 1) \\
X_2(t + 1)
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & 1 \\
1 & 0 & 1 \\
0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
X_0(t) \\
X_1(t) \\
X_2(t)
\end{bmatrix} +
\begin{bmatrix}
d_0(t) \\
d_1(t) \\
d_2(t)
\end{bmatrix}
\]
Multiple Signature Checking

• Use 2 different testing epochs:
  - 1\textsuperscript{st} with MISR with 1 polynomial
  - 2\textsuperscript{nd} with MISR with different polynomial

• Reduces probability of aliasing –
  - Very unlikely that both polynomials will alias for the same fault

• Low hardware cost:
  - A few XOR gates for the 2\textsuperscript{nd} MISR polynomial
  - A 2-1 MUX to select between two feedback polynomials
Aliasing Probability

- **Aliasing** – when bad machine signature equals good machine signature
- Consider error vector \( e(n) \) at POs
  - Set to a 1 when good and faulty machines differ at the PO at time \( t \)
- \( P_{al} \equiv \) aliasing probability
- \( p \equiv \) probability of 1 in \( e(n) \)
- Aliasing limits:
  - \( 0 < p \leq \frac{1}{2} \), \( \bar{p}^k \leq P_{al} \leq (1 - p)^k \)
  - \( \frac{1}{2} \leq p \leq 1 \), \( (1 - \bar{p})^k \leq P_{al} \leq p^k \)
Aliasing Theorems

- Theorem: Assuming that each PO $d_{ij}$ has probability $p_j$ of being in error, where the $p_j$ probabilities are independent, and that all outputs $d_{ij}$ are independent, in a $k$-bit MISR, $P_{al} = 1/(2^k)$, regardless of the initial condition.
Transition Counting vs. LFSR

- LFSR aliases for $f_{sa1}$, transition counter for $a_{sa1}$

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>$abc$</td>
<td>$Good$</td>
</tr>
<tr>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
</tr>
</tbody>
</table>

**Signatures**

<table>
<thead>
<tr>
<th>Transition Count</th>
<th>LFSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>3</td>
</tr>
<tr>
<td>$a_{sa1}$</td>
<td>3</td>
</tr>
<tr>
<td>$f_{sa1}$</td>
<td>0</td>
</tr>
<tr>
<td>$b_{sa1}$</td>
<td>1</td>
</tr>
</tbody>
</table>

08 Apr 2013 EE-709@IITB 22 CADSL
Logic BIST

• Complex systems with multiple chips demand elaborate logic BIST architectures
  ➢ BILBO and test / clock system
    ➢ Shorter test length, more BIST hardware
  ➢ STUMPS & test / scan systems
    ➢ Longer test length, less BIST hardware
• Benefits: cheaper system test, Cost: more hdwe.
• Must modify fully synthesized circuit for BIST to boost fault coverage
  ▪ Initialization, loop-back, test point hardware
Test / Clock System Example

• New fault set tested every clock period
• Shortest possible pattern length
  ▪ 10 million BIST vectors, 200 MHz test / clock
  ▪ Test Time = $10,000,000 / 200 \times 10^6 = 0.05$ s
  ▪ Shorter fault simulation time than test / scan
BILBO – Works as PG and RC

- **Built-in Logic Block Observer (BILBO)** -- 4 modes:
  1. Flip-flop
  2. LFSR pattern generator
  3. LFSR response compacter
  4. Scan chain for flip-flops
Complex BIST Architecture

- Testing epoch I:
  - LFSR1 generates tests for CUT1 and CUT2
  - BILBO2 (LFSR3) compacts CUT1 (CUT2)
- Testing epoch II:
  - BILBO2 generates test patterns for CUT3
  - LFSR3 compacts CUT3 response
Bus-Based BIST Architecture

- **Self-test control** broadcasts patterns to each CUT over bus – parallel pattern generation
- Awaits bus transactions showing CUT’s responses to the patterns: serialized compaction
Built-in Logic Block Observer (BILBO)

- Combined functionality of D flip-flop, *pattern generator*, *response compacter*, & *scan chain*
  - Reset all FFs to 0 by scanning in zeros
Example BILBO Usage

- **SI** – Scan In
- **SO** – Scan Out
- **Characteristic polynomial:** $1 + x + \ldots + x^n$
- CUTs A and C: BILBO1 is MISR, BILBO2 is LFSR
- CUT B: BILBO1 is LFSR, BILBO2 is MISR

(a) Example test configuration.
BILBO Serial Scan Mode

- \( B1 \, B2 = \text{“00”} \)
- Dark lines show enabled data paths
BILBO LFSR Pattern Generator Mode

- \( B_1 B_2 = \text{"01"} \)
BILBO in D FF (Normal) Mode

- \( B1 \ B2 = "10" \)
BILBO in MISR Mode

- \( B_1 \, B_2 = "11" \)
Test / Scan System

- New fault tested during 1 clock vector with a complete scan chain shift
- Significantly more time required per test than test / clock
  - **Advantage**: Judicious combination of scan chains and MISR reduces MISR bit width
  - **Disadvantage**: Much longer test pattern set length, causes fault simulation problems

- Input patterns – time shifted & repeated
  - Become *correlated* – reduces fault detection effectiveness

- Use XOR network to phase shift & *decorrelate*
STUMPS Example

- SR1 … SRn – 25 full-scan chains, each 200 bits
- 500 chip outputs, need 25 bit MISR (not 5000 bits)
STUMPS

- **Test procedure:**
  1. Scan in patterns from LFSR into all scan chains (200 clocks)
  2. Switch to normal functional mode and clock 1 x with system clock
  3. Scan out chains into MISR (200 clocks) where test results are compacted
    - Overlap Steps 1 & 3

- **Requirements:**
  - Every system input is driven by a scan chain
  - Every system output is caught in a scan chain or drives another chip being sampled
Summary

- LFSR pattern generator and MISR response compacter – preferred BIST methods
- BIST has overheads: test controller, extra circuit delay, Input MUX, pattern generator, response compacter, DFT to initialize circuit & test the test hardware
- BIST benefits:
  - At-speed testing for delay & stuck-at faults
  - Drastic ATE cost reduction
  - Field test capability
  - Faster diagnosis during system test
  - Less effort to design testing process
  - Shorter test application times
Thank You
Problem

- Consider a circuit with no self loops and an S-graph that is a complete graph. In a complete graph, a directed edge from a vertex $v_i$ to vertex $v_j$ exists for all $i$ and $j$. Show that to convert into acyclic graph for test generation you need to scan all but one flip-flop.