Logic BIST: Theory, Problems and Solutions

By
Ahmad Al-Yamani II.
August 5, 2002

Outline

- Built-In Self Test: How?
- Advantages
- Disadvantages and Problems
- Solutions around Problems
- Future Work
Outline

- Solutions around disadvantages
  - Fault Coverage
    - Top-off patterns
    - Reseeding
    - Weighted patterns
    - Test point insertion
    - Mapping Logic
    - Built-In Reseeding
  - Contention
    - Static decoding
    - Priority decoder
    - Illegal State Detection

Built-In Self-Test

- On-chip circuitry
  - Pattern generation
    - Linear Feedback Shift Register (LFSR)
  - Response compression
    - Multiple-Input Shift Register (MISR)
Advantages

- Low-cost ☺
- Circuit speed ☺
- Applicability while in system ☺
- Collateral coverage ☺

Disadvantages

- Test length ☹
- Area overhead ☹
- Fault coverage ☹
- Contention ☹
- Aliasing ☹
Test Per Clock

Combinational Logic

Scan Chain

LFSR

Test Per Scan

Combinational Logic

Scan Chain

LFSR

Sig. Reg.
Why Test Per Scan?

- Less storage with reseeding.
- No additional fanouts for scan chain.
- No routing problems for feedbacks.

How to Calculate the Seed?

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<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>
<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>1</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>
How to Calculate the Seed? (cont.)

Test Pattern = 1 X X 0 1 X X 1 0

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

| S0   | 0 | 1 | 0 | 1 |   |   |   |   |   |   |
| S1   |   | 1 | 1 | 0 | 0 | 1 |   |   |   |   |
| S2   |   | 1 | 1 | 0 | 0 | 0 | 1 |   |   |   |
| S3   |   | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |   |
| S4   |   | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| S5   |   | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| S6   |   |   | 0 | 1 | 0 |   |   |   |   |   |
| S7   |   |   | 0 | 1 | 0 | 0 |   |   |   |   |
| S8   |   |   |   | 0 | 1 | 0 | 0 |   |   |   |
| S9   |   |   |   |   | 0 | 1 | 0 | 0 |   |   |

System of Equations

- S0 = S4 ⊕ S1 = L1 ⊕ L3
- S1 = S5 ⊕ S2 = L0 ⊕ L1 ⊕ L2
- S2 = S6 ⊕ S3 = L1 ⊕ L2 ⊕ L3
- S3 = S7 ⊕ S4 = L0 ⊕ L1 ⊕ L2 ⊕ L3
- S4 = S8 ⊕ S5 = L0 ⊕ L2 ⊕ L3
- S5 = S9 ⊕ S6 = L0 ⊕ L3
- S6 = L0
- S7 = L1
- S8 = L2
- S9 = L3

[0 1 0 1] [1 1 1 0] [0 1 1 1] [1 1 1 1] [1 0 1 1] [1 0 0 1] [1 0 0 0] [0 1 0 0] [0 0 1 0] [0 0 0 1]
How to Calculate the Seed? (cont.)

- Test Pattern = 1 X X 0 1 X X 1 0
  - S9 = 0 = L3
  - S8 = 1 = L2
  - S5 = 1 = L0 ⊕ L3 ⇒ L0 = 1
  - S4 = 0 = L0 ⊕ L2 ⊕ L3 OK
  - S0 = 1 = L1 ⊕ L3 ⇒ L1 = 1
- ⇒ Seed = 1110

Stumps Architecture
Why Doesn’t That Work?

The Solution

- $h$ channels, each $r$ stages deep
- Modify the LFSR s.t.
  - The input to each channel is spaced by $r$ shifts
LFSR States

- LFSR Polynomial \( f(x) = c_n x^n + c_{n-1} x^{n-1} + \cdots + c_2 x^2 + c_1 x + c_0 \)
- LFSR State at time \( k \) \( v_k = [v_{1k} \ v_{2k} \ \cdots \ v_{nk}] \)
- LFSR State at \( k+1 \) \( v_{k+1} = [F_k \ v_{1k} \ v_{2k} \ \cdots \ v_{n-1k}] \)
- Where \( F_k = \sum_{i=1}^{n} c_i \cdot v_{ik} \mod 2 \)

Transition Matrix (S)

\( v_{k+1} = v_k S \)

\[
S = \begin{bmatrix}
    c_1 & 1 & 0 & \cdots & 0 & 0 \\
    c_2 & 0 & 1 & \cdots & 0 & 0 \\
    c_3 & 0 & 0 & \cdots & 0 & 0 \\
    \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
    c_{n-2} & 0 & 0 & \cdots & 1 & 0 \\
    c_{n-1} & 0 & 0 & \cdots & 0 & 1 \\
    c_n & 0 & 0 & \cdots & 0 & 0 \\
\end{bmatrix}
\]

\( v_{k+1} = v_1 S^k \)
Selection Vector

- Selection vector $b_j = [0 \cdots 1 \cdots 0]$
  - Where the 1 is in the $j$th column
- The $k$th bit in the $j$th register is given by
  $$u_k^{(j)} = v_k b_j^T = v_1 S^{k-1} b_j^T$$
- To get the shifted sequence
  $$u_k^{(j)} = v_1 S^{k+q-1} b_j^T$$
- Which is also
  $$u_k^{(j)} = \sum_{i=1}^{n} d_i u_k^{(i)} \mod 2$$

Shifting the LFSR Output

$$v_1 S^{k+q-1} b_j^T = v_k \sum_{i=1}^{n} d_i b_i^T \mod 2$$

$$v_1 S^{k-1} S^q b_j^T = v_1 S^{k-1} \sum_{i=1}^{n} d_i b_i^T \mod 2$$

$$S^q b_j^T = \sum_{i=1}^{n} d_i b_i^T$$

- Let $p_q^{(j)} = \sum_{i=1}^{n} d_i b_i$
  $$S^q b_j^T = p_q^{(j)}$$
Shifting the LFSR Output (cont.)

\[ S^{2r}b_j^T = S^r(S^rb_j^T) = S^r p_r^{T(j)} \]

- Same process repeated for every channel
- Back to the example,

\[ f(x) = x^4 + x^3 + 1 \]
\[ S = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \]

The Phase Shifting Calculation

- Assume 6 channels, of 4 stages each
- Let the 1st channel take the 1st stage output:
  \[ b_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \]
- We need to shift every channel by 4:
  \[ p_4^T = S^4b_1^T \Rightarrow p_4 = \begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix} \]
  \[ p_8^T = S^4p_4^T \Rightarrow p_8 = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \]
  \[ p_{12} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \]
  \[ p_{16} = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix} \]
  \[ p_{20} = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \]
The Phase Shifter

Generating the equations
With the phase shifter

How to Choose LFSR Size?

- LFSR size < care bits $\Rightarrow$ Variables < Equations
- LFSR size ($n$) > care bits ($s$)

$$P(s,n) = \prod_{i=0}^{s-1} \frac{(2^n - 1) - (2^i - 1)}{(2^n - 1) - 1}$$

- For $s < n$

$$P(s,n) \approx 1 - e^{-\frac{1}{2^{(s-n)}}}$$

If $n = s + 20 \Rightarrow P(s,n) \approx 10^{-6}$
Top off Patterns

- Run the LFSR in autonomous mode
  - Catch most of the faults
- Load patterns for remaining faults from ATE
- If reseeding is used
  \[
  s_{\text{max}} = \frac{\text{ChainLength}}{k} \quad \text{and} \quad n = s_{\text{max}} + 20
  \]
  \[
  \text{Saving} = (\text{ChainLength} - n) \times \text{NumberOfPatterns}
  \]
- For ChainLength=10000, k=10, Patterns=500
  - Saving = 89.8%

Weighted Random Patterns

- To fully test an n-input AND gate, we need:
  
  \[
  \begin{array}{cccc}
  0 & 1 & \cdots & 1 & 1 \\
  1 & 0 & \cdots & 1 & 1 \\
  \vdots & & & & \\
  1 & 1 & \cdots & 1 & 0 \\
  1 & 1 & \cdots & 1 & 1 \\
  \end{array}
  \]
- Hoping for this to happen randomly!
- For the random pattern resistant fault
  - Add HW to bias the probability of inputs
Test Point Insertion

- Observation Point
- Control-1 Point
- Control-0 Point

Mapping Logic

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

Diagram:
- Original Test Patterns → Pattern Generator → Mapping Logic → Transformed Test Patterns → Circuit Under Test (CUT)
Built-In Reseeding

<table>
<thead>
<tr>
<th>Mapping Logic</th>
<th>Built-In Reseeding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alter LFSR output</td>
<td>Alter LFSR contents</td>
</tr>
<tr>
<td>H/W for every pattern</td>
<td>1 seed → many patterns</td>
</tr>
<tr>
<td>H/W to detect pattern</td>
<td>H/W to detect pattern</td>
</tr>
<tr>
<td>H/W to map it</td>
<td>NO H/W to map it</td>
</tr>
</tbody>
</table>

Built-In Reseeding Circuit

(a) 

(b) 

Reseeding Logic
### Example

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<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>
<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>1</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>

EOS = $c_0 = 1000$
Seed = $1011 = c_5$
Select Lines = $(c_0) \text{ XOR } (c_4)$

EOS = $c_9 = 0110$
Seed = $0100 = c_{12}$
Select Lines = $(c_9) \text{ XOR } (c_{11})$

### Reseeding Circuit

![Reseeding Circuit Diagram]

- $c_0 = 1000$
- $c_5 = 1011$
- $(c_0) \text{ XOR } (c_4) = 1111$
- $c_9 = 0110$
- $c_{12} = 0100$
- $(c_9) \text{ XOR } (c_{11}) = 1111$
Mapping Logic (Same Example)

System Level View
Illegal States

- Normal operation ✓
- Scan-based or Pseudo-random test ?

<table>
<thead>
<tr>
<th>E1</th>
<th>E2</th>
<th>I1</th>
<th>I2</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>e'</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>a</td>
<td>a</td>
<td>✓</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>a</td>
<td>a'</td>
<td>✂</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>?</td>
</tr>
</tbody>
</table>

Static Decoding

- How about capture cycle? 😞
- Fault coverage 😞
Priority Encoder

- Insert a priority encoder at the 1-hot signals
- Delay
  - Several levels of logic
- Specific to 1-hot problem

Illegal State Detection (ISD) Function

- ISD in terms of PIs and FFs
  - Back trace one-hots to PIs and FFs
    - Add the non-one-hot detection circuitry
  - $E_1 = F_1(\text{PIs, FFs})$
  - $E_2 = F_2(\text{PIs, FFs})$
  - $E_3 = F_3(\text{PIs, FFs})$
  - $\text{ISD} = (F_1F_2F_3 + F_1F_2F_3 + F_1F_2F_3)$
FSM Representation

Two-cycle ISD
Fixing Logic

ISD + Fixing Logic

- Small delay
- During scan-in, scan-out and capture
- No fault coverage sacrifice
- Small ISD $\Rightarrow$ low area overhead

But …
- Delay
- Design change
Skipping Illegal Patterns

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

![Diagram showing SE, ISD, and New SE connections with a bit counter]

Future Work

- Reseeding Observations
  - The order of the seeds has a substantial impact
- Further explore LFSRs properties
  - To calculate the optimal order for picking the seeds
- Built-In Reseeding
  - Direct impact on reseeding circuit area
- Reseeding from ATE
  - Direct impact on seed storage