**Custom BIST Architectures**

- Partition the target circuit into (combinational) CUT blocks
- Insert registers, or modify existing ones, to act as test generators (TGs) or response data compacters (DCs)

**Bus-Based BIST Architectures**

- Self-test control unit broadcasts patterns to each CUT over bus (parallel test pattern generation)
- Receives bus transactions showing CUT’s responses to the patterns (serial data compaction)
**LFSRs**

- Linear feedback shift registers (LFSRs) are widely used both as TGs and DCs.
- They are related to cyclic redundancy check (CRC) codes, which use LFSRs for encoding and decoding of serial data streams.
- Original “Signature Analyzer” [Hewlett-Packard 1977]:

![16-bit LFSR Diagram]

**Simple FSRs**

- Simple $n$-bit shift register repeats after $n$ shifts.
- To increase the number of states generated, we can introduce feedback logic (a few gates).
- If we restrict the feedback gates to XORs, we get an LFSR.

State

<table>
<thead>
<tr>
<th>$s_0$</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

State transitions:

- $s_0$: 1 → 0 → 1
- $s_1$: 1 → 1 → 0
- $s_2$: 1 → 1 → 0
- $s_3$: 1 → 1 → 0

Output: 101
LFSRs

- The symbol $\oplus$ represents an XOR gate or modulo-2 adder
- An $n$-bit LFSR can have as many as $2^n - 1$ different states
- We get different types of LFSRs depending on the XOR logic

<table>
<thead>
<tr>
<th>State</th>
<th>$s_0$</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>$s_4$</th>
<th>$s_5$</th>
<th>$s_6$</th>
<th>$s_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Type 1 LFSR

- Above is a “standard” (type 1 or external) LFSR
- It produces pattern sequences algorithmically
- It has most of the desirable properties of a random number generator
- Long sequences are needed for good fault coverage, but need not include all $2^n$ possible patterns
**Type 1 LFSR**

**Matrix Equation**

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

$$X (t + 1) = T_S X (t) \quad (T_S \text{ is companion matrix})$$

---

**Type 2 LFSR**

- Equivalent to standard (external or type 1) LFSR
- Described by companion matrix $T_m = T_S^T$
Cellular Automaton

- Each cell only connects to neighbors and all connections are regular
- Superior to LFSR – even more random (no shift-induced bit value correlations)

Example

- Five-stage cellular automaton

LFSRs in BIST

D flip-flop

2-input XOR gate (modulo-2 adder)

Basic structure (internal or type 2)

Parallel test generator (test per clock)

Serial test generator (test per scan)

Parallel response compactor

Serial response compactor

MISR compactor

scan path
LFSRs

- Let \( A = a_0a_1a_2... \) be a binary sequence, e.g., the output sequence produced by an LFSR
- Elements of \( A \) can be considered coefficients of a polynomial called a generating function

\[
G(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n + \ldots
\]

- Passing sequence \( A \) through an \( n \)-bit LFSR causes \( G(x) \) to be divided by an order-\( n \) polynomial, the LFSR’s characteristic polynomial \( P(x) \).
- This is just long division implemented serially by a shift-and-subtract method; note that an XOR gate is both a subtracter and an adder mod 2
- At the end, the contents (state) of the LFSR is the remainder from \( G(x)/P(x) \) which is normally arranged to be zero if \( G(x) \) is the fault-free response
- In CRC codes, \( G(x) \) is a codeword obtained by multiplying an input data sequence by \( P(x) \). CRC codes detect single-bit and burst errors.

Example 1: Polynomial Division

- Input sequence = 1 1 0 0 0 11
- Input polynomial \( G(x) = 1.x^6 + 1.x^5 + 0.x^4 + 0.x^3 + 0.x^2 + 1.x^1 + 1.x^0 \)
- Characteristic polynomial \( P(x) = 1.x^4 + 0.x^3 + 1.x^2 + 0.x^1 + 1.x^0 \)

\[
\begin{array}{c|cccccc}
\text{Quotient} & x^2 & + & x & + & 1 \\
\text{Divisor P(x)} & x^6 + x^5 & + & x^4 + x^2 & + & x^1 & + & 1 \\
\text{Dividend G(x)} & x^5 & + & x^4 & + & x^2 & + & x & + & 1 \\
\text{Remainder R(x)} & x^4 & + & x^3 & + & x^2 & + & x & + & 1 \\
\end{array}
\]

- Remember ADD = SUBTRACT and MULTIPLY = AND
### LFSRs

#### Example 1 (contd.)

<table>
<thead>
<tr>
<th>Time</th>
<th>Input stream</th>
<th>Register state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 1 0 0 0 1 1</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 1 1 0 0 0 1</td>
<td>1 0 0 0</td>
</tr>
<tr>
<td>2</td>
<td>0 1 1 0 0 0</td>
<td>1 1 0 0</td>
</tr>
<tr>
<td>3</td>
<td>0 1 1 1 0 0</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>4</td>
<td>0 1 0 1 1 1</td>
<td>0 0 1 1</td>
</tr>
<tr>
<td>5</td>
<td>0 1 0 1 1</td>
<td>1 0 1 1</td>
</tr>
<tr>
<td>6</td>
<td>0 1 0 1</td>
<td>1 0 1 1</td>
</tr>
<tr>
<td>7</td>
<td>0 0 0 1</td>
<td>0 0 0 1</td>
</tr>
</tbody>
</table>

Signature = $x^3$

---

### LFSRs: Example 2

\[
P(x) = 1 + x^2 + x^4 + x^5
\]

\[
G(x) = x^7 + x^6 + x^5 + x^4 + x^2 + 1
\]

\[
\begin{align*}
\text{Input sequence:} & \quad 1 1 1 1 0 1 0 1 \quad (8 \text{ bits}) \\
\text{Initial state} & \quad 0 0 0 0 0
\end{align*}
\]

---

<table>
<thead>
<tr>
<th>Time</th>
<th>Input stream</th>
<th>Register contents</th>
<th>Output stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 0 1 0 1 1 1 1</td>
<td>1 2 3 4 5</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>1 0 1 0 1 1 1</td>
<td>1 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1 0 1</td>
<td>0 1 1 1 1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1 0 0 0 1 0 1 0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1 0 0 0 0 1 0 1</td>
<td>0 1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Remainder</td>
<td>0 0 1 0 1 1 1</td>
<td></td>
</tr>
</tbody>
</table>

\[
R(x) = x^2 + x^4 + 1 + x^2
\]

\[
\text{Quotient} \quad \text{Remainder}
\]

---

ABF p. 443
LFSR Properties

- LFSR’s with maximum-length sequences (period = \(2^n - 1\)) are most useful for TG and DC. Their CPs are called primitive.
- The error detection capabilities of an LFSR depend on \(P(x)\):
  \[ P(x) \text{ must be primitive} \]
  If \(P(x)\) has at least two nonzero coefficients, the LFSR detects all single-bit errors (but not necessarily all SSL faults)
  If \(P(x)\) has degree \(n\) and the coefficient of \(x^0\) is 1, then all \(k\)-bit burst errors are detected for \(k \leq n\)
- Experimental evidence indicates that LFSR-based compression is effective. Current theory for selecting \(P(x)\) is sketchy
- Errors can be represented by an error polynomial \(E(x)\) whose nonzero coefficients mark error bits. Data streams \(R(x)\) and \(R_0(x)\) have the same signature (aliasing) if \(R(x) = R_0(x) + E(x)\) and \(E(x)\) is a multiple of \(P(x)\)
- The probability of aliasing approaches \(2^{-n}\) for long streams

LFSR-Based BIST

- Test-per-clock is fast but has high hardware overhead
- Test-per-scan has low hardware overhead but is slow
BILBO (Built-in Logic Block Observation)

- Based on a BILBO register which has three modes:
  - Normal register
  - Shift register
  - LFSR (test) mode

BILBO (Built-in Logic Block Observation)

TMS 320101 (partial)  BILBO version