Lecture 25

Built-In Self-Testing
Pattern Generation and Response Compaction

- Motivation and economics
- Definitions
- Built-in self-testing (BIST) process
- BIST pattern generation (PG)
- BIST response compaction (RC)
- Aliasing probability
- Example
- Summary
BIST Motivation

- Useful for field test and diagnosis (less expensive than a local automatic test equipment)

- Software tests for field test and diagnosis:
  - Low hardware fault coverage
  - Low diagnostic resolution
  - Slow to operate

- Hardware BIST benefits:
  - Lower system test effort
  - Improved system maintenance and repair
  - Improved component repair
  - Better diagnosis
Costly Test Problems Alleviated by BIST

- Increasing chip logic-to-pin ratio – harder observability
- Increasingly dense devices and faster clocks
- Increasing test generation and application times
- Increasing size of test vectors stored in ATE
- Expensive ATE needed for 1 GHz clocking chips
- Hard testability insertion – designers unfamiliar with gate-level logic, since they design at behavioral level
- In-circuit testing no longer technically feasible
- Shortage of test engineers
- Circuit testing cannot be easily partitioned
Typical Quality Requirements

- 98% single stuck-at fault coverage
- 100% interconnect fault coverage
- Reject ratio – 1 in 100,000
### Benefits and Costs of BIST with DFT

<table>
<thead>
<tr>
<th>Level</th>
<th>Design and test</th>
<th>Fabrication</th>
<th>Manuf. Test</th>
<th>Maintenance test</th>
<th>Diagnosis and repair</th>
<th>Service interruption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chips</td>
<td>+ / -</td>
<td>+</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boards</td>
<td>+ / -</td>
<td>+</td>
<td>-</td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>System</td>
<td>+ / -</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

+  Cost increase  
-  Cost saving  
+/−  Cost increase may balance cost reduction
Economics – BIST Costs

- Chip area overhead for:
  - Test controller
  - Hardware pattern generator
  - Hardware response compacter
  - Testing of BIST hardware
- Pin overhead -- At least 1 pin needed to activate BIST operation
- Performance overhead – extra path delays due to BIST
- Yield loss – due to increased chip area or more chips in system because of BIST
- Reliability reduction – due to increased area
- Increased BIST hardware complexity – happens when BIST hardware is made testable
BIST Benefits

- Faults tested:
  - Single combinational / sequential stuck-at faults
  - Delay faults
  - Single stuck-at faults in BIST hardware

- BIST benefits:
  - Reduced testing and maintenance cost
  - Lower test generation cost
  - Reduced storage / maintenance of test patterns
  - Simpler and less expensive ATE
  - Can test many units in parallel
  - Shorter test application times
  - Can test at functional system speed
Definitions

- BILBO – Built-in logic block observer, extra hardware added to flip-flops so they can be reconfigured as an LFSR pattern generator or response compacter, a scan chain, or as flip-flops
- Concurrent testing – Testing process that detects faults during normal system operation
- CUT – Circuit-under-test
- Exhaustive testing – Apply all possible $2^n$ patterns to a circuit with $n$ inputs
- Irreducible polynomial – Boolean polynomial that cannot be factored
- LFSR – Linear feedback shift register, hardware that generates pseudo-random pattern sequence
More Definitions

- **Primitive polynomial** – Boolean polynomial $p(x)$ that can be used to compute increasing powers $n$ of $x^n$ modulo $p(x)$ to obtain all possible non-zero polynomials of degree less than $p(x)$

- **Pseudo-exhaustive testing** – Break circuit into small, overlapping blocks and test each exhaustively

- **Pseudo-random testing** – Algorithmic pattern generator that produces a subset of all possible tests with most of the properties of randomly-generated patterns

- **Signature** – Any statistical circuit property distinguishing between bad and good circuits

- **TPG** – Hardware test pattern generator
- Test controller – Hardware that activates self-test simultaneously on all PCBs
- Each board controller activates parallel chip BIST
- Diagnosis effective only if very high fault coverage
BIST Architecture

- **Note:** BIST cannot test wires and transistors:
  - From PI pins to Input MUX
  - From POs to output pins
BILBO – Works as Both a PG and a RC

- Built-in Logic Block Observer (BILBO) -- 4 modes:
  - Flip-flop
  - LFSR pattern generator
  - LFSR response compacter
  - Scan chain for flip-flops
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
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
Exhaustive Pattern Generation

- Shows that every state and transition works
- For $n$-input circuits, requires all $2^n$ vectors
- Impractical for $n > 20$
Pseudo-Exhaustive Method

- Partition large circuit into fanin cones
  - Backtrace from each PO to PIs influencing it
  - Test fanin cones in parallel
- Reduced # of tests from $2^8 = 256$ to $2^5 \times 2 = 64$
  - Incomplete fault coverage
Pseudo-Exhaustive Pattern Generation

Five-Bit Binary Counter 1

Five-Bit Binary Counter 2

2-Bit 2-1 MUX

0 for Counter 1
1 for Counter 2

X1
X2
X3
X4
X5
X6
X7
X8

h

f

2
3
4
5
6
7

1
Random Pattern Testing

Bottom: Random Pattern Resistant circuit

(a) Top curve -- random pattern testing with acceptable fault coverage.
(b) Bottom curve -- unacceptable random pattern testing.
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

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

\[X (t + 1) = T_s \cdot X (t)\]  \( (T_s \text{ is companion matrix}) \)
LFSR Implements a Galois Field

- Galois field (mathematical system):
  - Multiplication by $x$ same as right shift of LFSR
  - Addition operator is XOR ($\oplus$)
- $T_s$ companion matrix:
  - 1st column 0, except nth element which is always 1 ($X_0$ always feeds $X_{n-1}$)
  - Rest of row $n$ – feedback coefficients $h_i$
  - Rest is identity matrix $I$ – means a right shift
- Near-exhaustive (maximal length) LFSR
  - Cycles through $2^n - 1$ states (excluding all-0)
  - 1 pattern of $n$ 1’s, one of $n-1$ consecutive 0’s
- Autocorrelation – any shifted sequence same as original in $2^{n-1} - 1$ bits, differs in $2^{n-1}$ bits
- If $h_i = 0$, that XOR gate is deleted
LFSR Theory

- Cannot initialize to all 0’s – hangs
- If $X$ is initial state, progresses through states $X, T_s X, T_s^2 X, T_s^3 X, \ldots$

- Matrix period:
  
  Smallest $k$ such that $T_s^k = I$
  
  - $k \equiv$ LFSR cycle length

- Described by characteristic polynomial:

  $f(x) = |T_s - I X|$

  $= 1 + h_1 x + h_2 x^2 + \ldots + h_{n-1} x^{n-1} + x^n$
Fault detection probability by a random number

\[ p(x) \, dx = \text{fraction of detectable faults with detection probability between } x \text{ and } x + dx \]

- \[ p(x) \, dx \geq 0 \text{ when } 0 \leq x \leq 1 \]
- \[ \int_0^1 p(x) \, dx = 1 \]

Exist \( p(x) \, dx \) faults with detection probability \( x \)

Mean coverage of those faults is \( x \int p(x) \, dx \)

Mean fault coverage \( y_n \) of 1st \( n \) vectors:

\[ I(n) = 1 - \int_0^1 (1 - x)^n \, p(x) \, dx \]

\[ y_n \equiv 1 - I(n) + \frac{n}{\text{total faults}} \quad (15.6) \]
LFSR Fault Coverage & Vector Length Estimation

Random-fault-detection (RFD) variable:
- Vector # at which fault first detected
- \( w_i \equiv \# \text{ faults with RFD variable } i \)

So

\[
p(x) = \frac{1}{n_s} \sum_{i=1}^{N} w_i p_i(x)
\]

- \( n_s \equiv \text{size of sample simulated; } N \equiv \# \text{ test vectors} \)
- \( w_0 \approx n_s - \sum_{i=1}^{N} w_i \)

Method:
- Estimate random first detect variables \( w_i \) from fault simulator using fault sampling
- Estimate \( I(n) \) using book Equation 15.8
- Obtain test length by inverting Equation 15.6 & solving numerically
Example External XOR LFSR

Characteristic polynomial \( f(x) = 1 + x + x^3 \) (read taps from right to left)
External XOR LFSR

- Pattern sequence for example LFSR (earlier):
  - \(X_0\) | 1 0 0 1 0 1 1 1 1 0
  - \(X_1\) | 0 0 1 0 1 1 1 1 0 0 ...
  - \(X_2\) | 0 1 0 1 1 1 0 0 1

- Always have 1 and \(x^n\) terms in polynomial
- Never repeat an LFSR pattern more than 1 time
  - Repeats same error vector, cancels fault effect

\[
\begin{bmatrix}
X_0 (t + 1)
X_1 (t + 1)
X_2 (t + 1)
\end{bmatrix}
= 
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 1 & 0 \\
\end{bmatrix}
\begin{bmatrix}
X_0 (t)
X_1 (t)
X_2 (t)
\end{bmatrix}
\]
Generic Modular LFSR
Modular Internal XOR LFSR

- Described by companion matrix $T_m = T_s^T$
- Internal XOR LFSR – XOR gates in between D flip-flops
- Equivalent to standard External XOR LFSR
  - With a different state assignment
  - Faster – usually does not matter
  - Same amount of hardware
- $X(t + 1) = T_m \times X(t)$
- $f(x) = |T_m - I X|$
  - $= 1 + h_1 x + h_2 x^2 + \ldots + h_{n-1} x^{n-1} + x^n$
- Right shift – equivalent to multiplying by $x$, and then dividing by characteristic polynomial and storing the remainder
Modular LFSR Matrix

\[
\begin{bmatrix}
X_0 (t + 1) \\
X_1 (t + 1) \\
X_2 (t + 1) \\
\vdots \\
X_{n-3} (t + 1) \\
X_{n-2} (t + 1) \\
X_{n-1} (t + 1)
\end{bmatrix}
= 
\begin{bmatrix}
0 & 0 & 0 & \ldots & 0 & 0 & 1 \\
1 & 0 & 0 & \ldots & 0 & 0 & h_1 \\
0 & 1 & 0 & \ldots & 0 & 0 & h_2 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \ldots & 0 & 0 & h_{n-3} \\
0 & 0 & 0 & \ldots & 1 & 0 & h_{n-2} \\
0 & 0 & 0 & \ldots & 0 & 1 & h_{n-1}
\end{bmatrix}
\begin{bmatrix}
X_0 (t) \\
X_1 (t) \\
X_2 (t) \\
\vdots \\
X_{n-3} (t) \\
X_{n-2} (t) \\
X_{n-1} (t)
\end{bmatrix}
\]
Example Modular LFSR

$f(x) = 1 + x^2 + x^7 + x^8$

Read LFSR tap coefficients from left to right
Primitive Polynomials

- Want LFSR to generate all possible $2^n - 1$ patterns (except the all-0 pattern)

- Conditions for this – must have a primitive polynomial:
  - Monic – coefficient of $x^n$ term must be 1
  - Modular LFSR – all D FF’s must right shift through XOR’s from $X_0$ through $X_1$, ..., through $X_{n-1}$, which must feed back directly to $X_0$
  - Standard LFSR – all D FF’s must right shift directly from $X_{n-1}$ through $X_{n-2}$, ..., through $X_0$, which must feed back into $X_{n-1}$ through XORing feedback network
Characteristic polynomial must divide the polynomial $1 + x^k$ for $k = 2^n - 1$, but not for any smaller $k$ value.

- See Appendix B of the book for tables of primitive polynomials.

- If $p(\text{error}) = 0.5$, no difference between behavior of primitive & non-primitive polynomial.

- But $p(\text{error})$ is rarely $= 0.5$. In that case, non-primitive polynomial LFSR takes much longer to stabilize with random properties than primitive polynomial LFSR.
Weighted Pseudo-Random Pattern Generation

If \( p(1) \) at all PIs is 0.5, \( p_F(1) = 0.5^8 = \frac{1}{256} \)

\[
p_F(0) = 1 - \frac{1}{256} = \frac{255}{256}
\]

Will need enormous \# of random patterns to test a stuck-at 0 fault on \( F \) -- LFSR \( p(1) = 0.5 \)

- We must not use an ordinary LFSR to test this

IBM – holds patents on weighted pseudo-random pattern generator in ATE
Weighted Pseudo-Random Pattern Generator

- LFSR $p(1) = 0.5$
- Solution: Add programmable weight selection and complement LFSR bits to get $p(1)$’s other than 0.5
- Need 2-3 weight sets for a typical circuit
- Weighted pattern generator drastically shortens pattern length for pseudo-random patterns
Weighted Pattern Gen.

Weight select \( W_1 \) \( W_2 \)

<table>
<thead>
<tr>
<th>( w_1 )</th>
<th>( w_2 )</th>
<th>Inv.</th>
<th>( p ) (output)</th>
<th>( w_1 )</th>
<th>( w_2 )</th>
<th>Inv.</th>
<th>( p ) (output)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( \frac{1}{2} )</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( \frac{1}{8} )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( \frac{1}{2} )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( \frac{7}{8} )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( \frac{1}{4} )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( \frac{1}{16} )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( \frac{3}{4} )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( \frac{15}{16} )</td>
</tr>
</tbody>
</table>
Cellular Automata (CA)

- Superior to LFSR – even “more” random
  - No shift-induced bit value correlation
  - Can make LFSR more random with linear phase shifter
- Regular connections – each cell only connects to local neighbors

\[
x_{c-1}(t) \times x_c(t) \times x_{c+1}(t)
\]
Gives CA cell connections

<table>
<thead>
<tr>
<th>(x_{c-1}(t))</th>
<th>(x_c(t))</th>
<th>(x_{c+1}(t))</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>110</td>
<td>101</td>
</tr>
<tr>
<td>100</td>
<td>011</td>
<td>010</td>
</tr>
<tr>
<td>001</td>
<td>000</td>
<td></td>
</tr>
</tbody>
</table>

\[2^6 + 2^4 + 2^3 + 2^1 = 90\] Called Rule 90

\[x_c(t+1) = x_{c-1}(t) \oplus x_{c+1}(t)\]
Cellular Automaton

- Five-stage hybrid cellular automaton
- Rule 150: $x_c(t+1) = x_{c-1}(t) \oplus x_c(t) \oplus x_{c+1}(t)$
- Alternate Rule 90 and Rule 150 CA
Test Pattern Augmentation

- Secondary ROM – to get LFSR to 100% SAF coverage
  - Add a small ROM with missing test patterns
  - Add extra circuit mode to Input MUX – shift to ROM patterns after LFSR done
  - Important to compact extra test patterns
- Use diffracter:
  - Generates cluster of patterns in neighborhood of stored ROM pattern
- Transform LFSR patterns into new vector set
- Put LFSR and transformation hardware in full-scan chain
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 million x 200 = 1 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.
Transition Counting

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

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

Transition count:

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

To maximize fault coverage:

- Make \( C(R_0) \) – good machine transition count – as large or as small as possible
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

Characteristic Polynomial: $x^5 + x^3 + x + 1$

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>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Logic Simulation:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</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** = $1 + x^2 + x^3$

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

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

\[ \frac{x^7 + x^3 + x^2 + x}{x^2 + 1} \]

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

\[ x^5 + x^2 + x \]

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

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

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 \( t \)

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

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

Characteristic Polynomial: \( x^3 + x + 1 \)
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:
  - $0 < p \leq \frac{1}{2}$, $p^k \leq P_{al} \leq (1 - p)^k$
  - $\frac{1}{2} \leq p \leq 1$, $(1 - p)^k \leq P_{al} \leq p^k$
Aliasing Probability Graph

Bounds on Aliasing

- $k = 1$
- $k = 3$
- $k = 11$

Bound for $0 \leq p \leq 1/2$
Bound for $1/2 \leq p \leq 1$
MISR has more aliasing than LFSR on single PO

- Error in CUT output $d_j$ at $t_i$, followed by error in output $d_{j+h}$ at $t_{i+h}$, eliminates any signature error if no feedback tap in MISR between bits $Q_j$ and $Q_{j+h}$. 
Aliasing Theorems

Theorem 15.1: Assuming that each circuit PO $d_{ij}$ has probability $p$ of being in error, and that all outputs $d_{ij}$ are independent, in a $k$-bit MISR, $P_{al} = 1/(2^k)$, regardless of initial condition of MISR. Not exactly true – true in practice.

Theorem 15.2: 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.
3 bit exhaustive binary counter for pattern generator
### Transition Counting vs. LFSR

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

<table>
<thead>
<tr>
<th>Pattern</th>
<th>abc</th>
<th>Good</th>
<th>$a_{sa1}$</th>
<th>$f_{sa1}$</th>
<th>$b_{sa1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Signatures

<table>
<thead>
<tr>
<th>Transition Count</th>
<th>LFSR</th>
<th>$a_{sa1}$</th>
<th>$f_{sa1}$</th>
<th>$b_{sa1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
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