VLSI Design Verification and Testing

Built-In Self-Test (BIST)

Mohammad Tehranipoor

Electrical and Computer Engineering

University of Connecticut

Overview

- **Motivation and economics**
- **Definitions**
- ***Built-in self-testing* (BIST) process**
- **BIST *pattern generation* (PG)**
- **BIST *response compaction* (RC)**
- ***Aliasing* definition and 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 at component level

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 GHz clocking chips
- Hard testability insertion – designers unfamiliar with gate-level logic, since they design at behavioral level
- Shortage of test engineers
- Circuit testing cannot be easily partitioned

Benefits and Costs of BIST with DFT

Level	Design and test	Fabri-cation	Manuf. Test	Maintenance test	Diagnosis and repair	Service interruption
Chips	+ / -	+	-			
Boards	+ / -	+	-		-	
System	+ / -	+	-	-	-	-

+ Cost increase
- Cost saving
+/- Cost increase may balance cost reduction

15 April 2007 5

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**

15 April 2007 6

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**

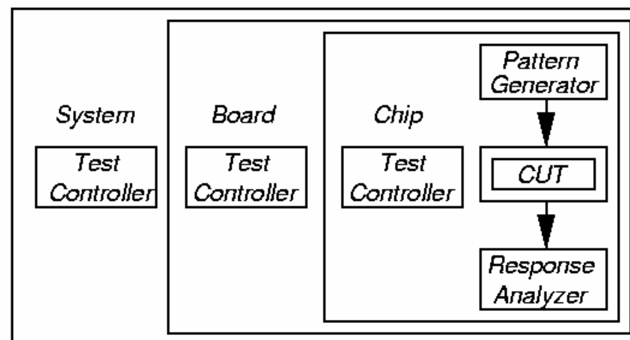
Definitions

- **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**
- **LFSR – *Linear feedback shift register,* hardware that generates pseudo-random pattern sequence**

More Definitions

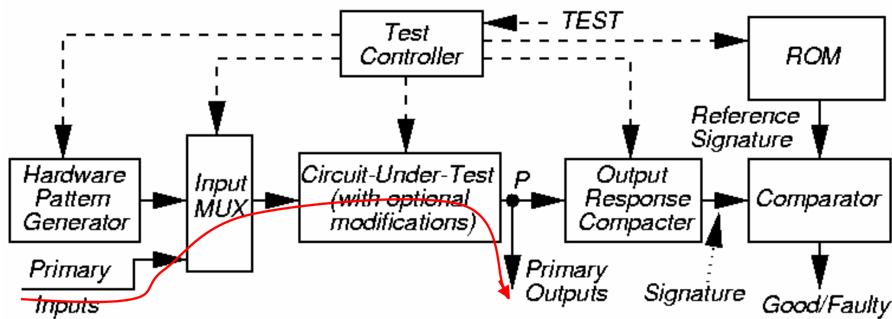
- ***Primitive polynomial*** **– Boolean polynomial *p (x)* that can be used to compute increasing powers *n* of x^n *modulo p (x)* to obtain all possible non-zero polynomials of degree less than *p (x)***
- ***Pseudo-exhaustive testing*** **– Break circuit into small, overlapping blocks and test each 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***

BIST Process



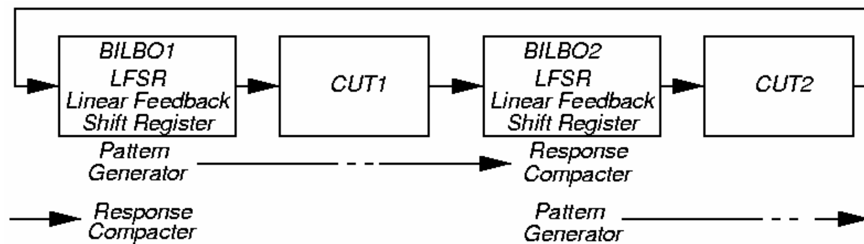
- ***Test controller*** **– Hardware that activates self-test simultaneously on all PCBs**
- **Each board controller activates parallel chip BIST Diagnosis effective only if very high fault coverage**

BIST Architecture



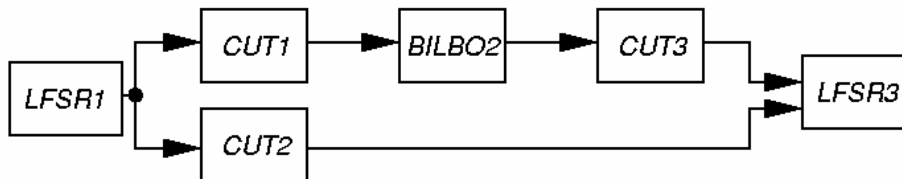
- **Note: BIST cannot test wires and transistors:**
  - **From PI pins to Input MUX**
  - **From POs to output pins**

BILBO – Works as Both a TPG and a RC



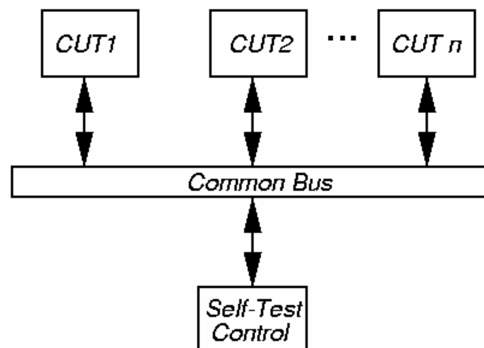
- ***Built-in Logic Block Observer* (BILBO) -- 4 modes:**
  1. **Flip-flop**
  2. **LFSR pattern generator**
  3. **LFSR response compacter**
  4. **Scan chain for flip-flops**

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

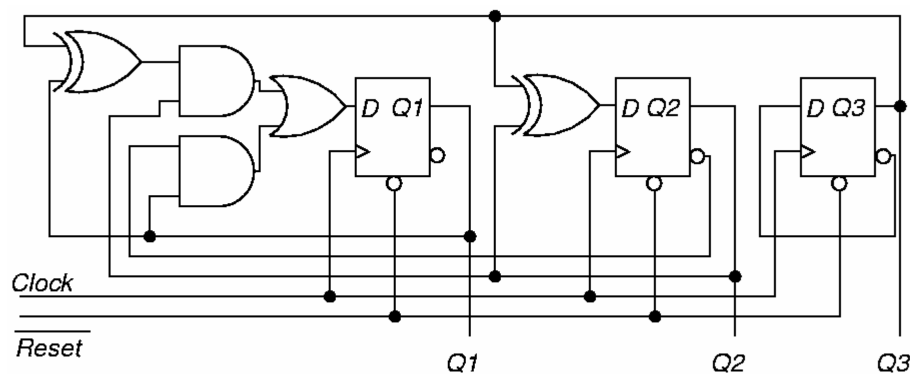


- ***Self-test control*** **broadcasts patterns to each CUT over bus – parallel pattern generation**
- **Awaits bus transactions showing CUT's responses to the patterns: serialized compaction**

Pattern Generation

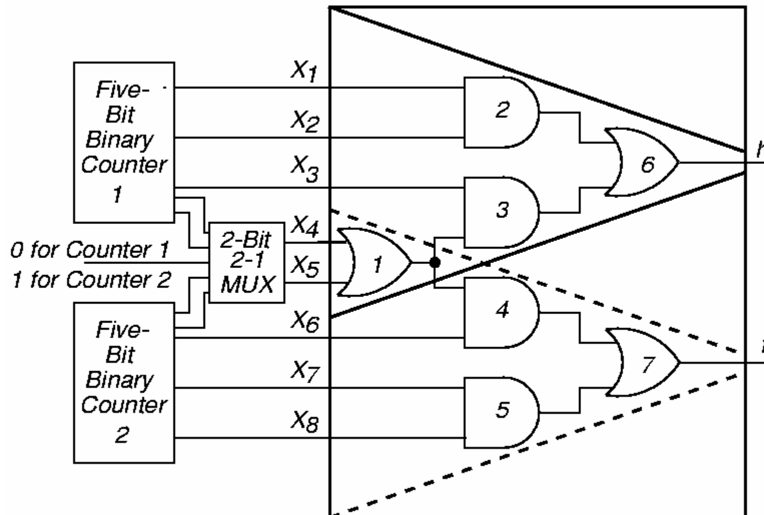
- **Store in ROM – too expensive**
- ***Exhaustive***
- ***Pseudo-exhaustive***
- ***Pseudo-random* (LFSR) – Preferred method**
- **Binary counters – use more hardware than LFSR**
- **Modified counters**
- **Test pattern *augmentation***
  - **LFSR combined with a few patterns in ROM**
  - ***Hardware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

Exhaustive Pattern Generation (A Counter)



- **Shows that every state and transition works**
- **For *n*-input circuits, requires all 2^n vectors**
- **Impractical for large *n* (> 20)**

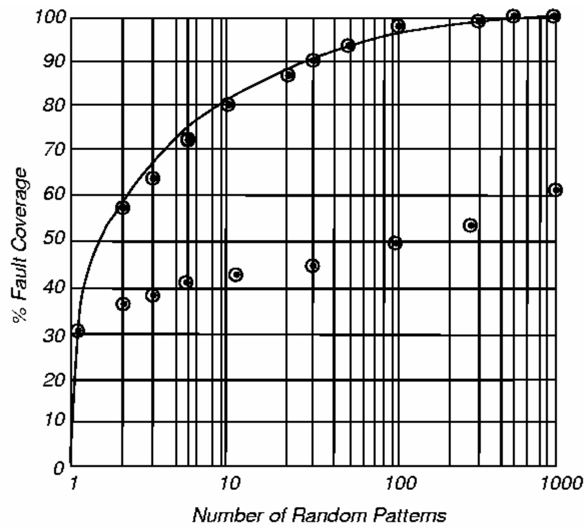
Pseudo-Exhaustive Pattern Generation



15 April 2007 17

Random Pattern Testing

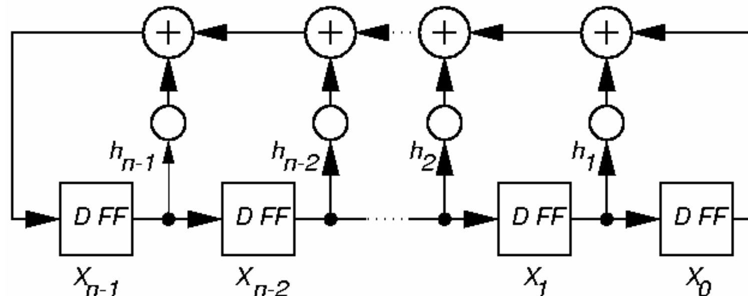
Bottom: Random-Pattern Resistant circuit



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

15 April 2007 18

Pseudo-Random Pattern Generation

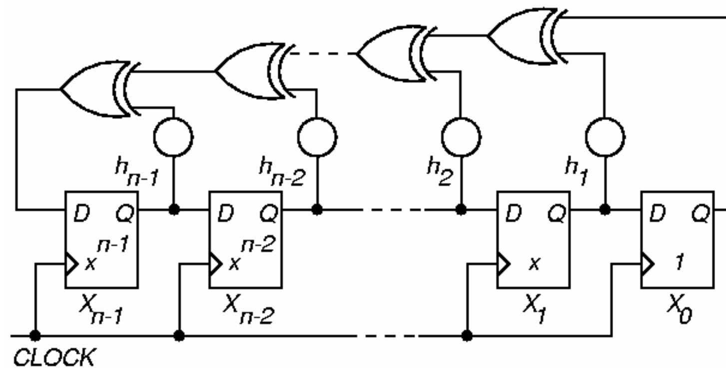


- ***Standard Linear Feedback Shift Register*** **(LFSR)**
  - **Normally known as *External XOR* type LFSR**
  - **Produces patterns algorithmically – repeatable**
  - **Has most of desirable random # properties**
- **Need not cover all 2^n input combinations**
- **Long sequences needed for good fault coverage**

Theory: LFSRs

- ***Galois field*** **(mathematical system):**
  - **Multiplication by x same as right shift of LFSR**
  - **Addition operator is XOR ($\oplus$)**
- **T_s companion matrix for a standard (external XOR type) LFSR:**
  - **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**

Standard *n*-Stage LFSR



- **If $h_i = 0$, that XOR gate is deleted**

Matrix Equation for Standard LFSR

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

$X(t+1) = T_s X(t)$ **(T_s is *companion matrix*)**

LFSR Theory (contd.)

- **Cannot initialize to all 0's – hangs**
- **If X is initial state, progresses through states X, $T_s X$, $T_s^2 X$, $T_s^3 X$, …**
- ***Matrix period*:**

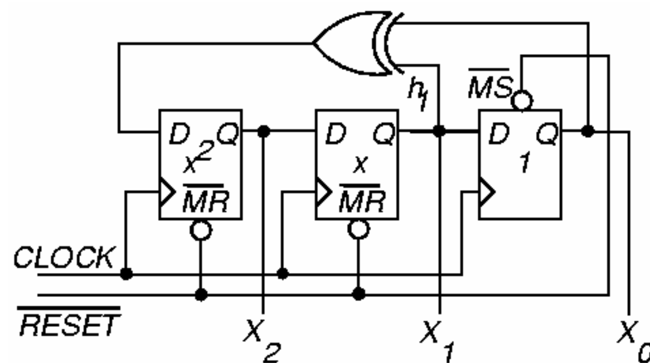
  Smallest k such that $T_s^k = I$

  - **$k \equiv$ LFSR cycle length**
- **Described by characteristic polynomial:**

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

  $= 1 + h_1 x + h_2 x^2 + \ldots + h_{n-1} x^{n-1} + x^n$

Example External XOR LFSR



$F(x) = 1 + x + x^3$

Example: External XOR LFSR (contd.)

- **Matrix equation:**

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \end{bmatrix}$$

- **Companion matrix:**

$$T_s = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

- **Characteristic polynomial:**
  - $f(x) = 1 + x + x^3$

    (read taps from right to left)
- **Always have 1 and x^n terms in polynomial**

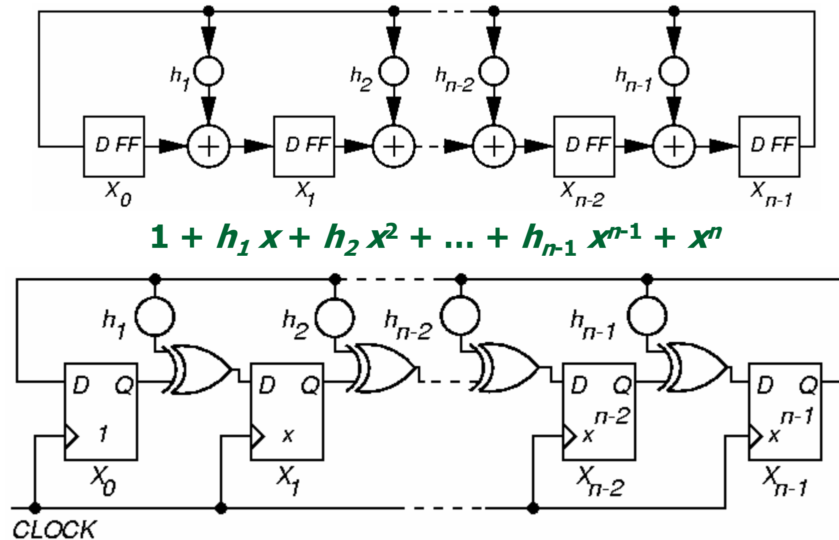
External XOR LFSR

- **Pattern sequence for example LFSR (earlier):**

X_0	1	0	0	1	0	1	1	1	0	
X_1	0	0	1	0	1	1	1	0	0	...
X_2	0	1	0	1	1	1	0	0	1	

- **Never repeat an LFSR pattern more than 1 time –Repeats same error vector, cancels fault effect**

Generic Modular (Internal XOR) LFSR



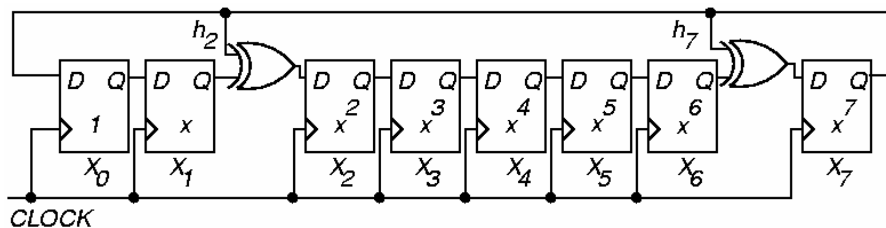
Modular Internal XOR LFSR

- 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

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \\ \vdots \\ X_{n-3}(t+1) \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 & 1 \\ 1 & 0 & 0 & \ldots & 0 & 0 & h_1 \\ 0 & 1 & 0 & \ldots & 0 & 0 & h_2 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & 0 & h_{n-3} \\ 0 & 0 & 0 & \ldots & 1 & 0 & h_{n-2} \\ 0 & 0 & 0 & \ldots & 0 & 1 & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \\ \vdots \\ X_{n-3}(t) \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix}$$

Example Modular LFSR

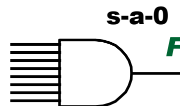


- $f(x) = 1 + x^2 + x^7 + x^8$
- **Read LFSR tap coefficients from left to right**

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

Weighted Pseudo-Random Pattern Generation



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

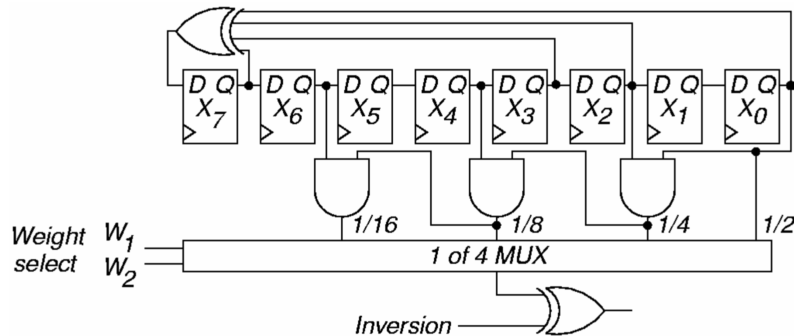
$$p_F(0) = 1 - \frac{1}{256} = \frac{255}{256}$$

- 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.



w_1	w_2	Inv.	p (output)	w_1	w_2	Inv.	p (output)
0	0	0	½	1	0	0	1/8
0	0	1	½	1	0	1	7/8
0	1	0	¼	1	1	0	1/16
0	1	1	3/4	1	1	1	15/16

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

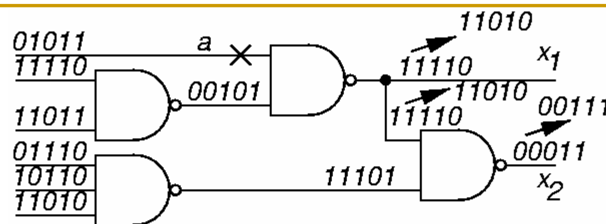
- **Severe amounts of data in CUT response to LFSR patterns – example:**
  - **Generate 5 million random patterns**
  - **CUT has 200 outputs**
  - **Leads to: 5 million x 200 = 1 billion bits response**
- **Uneconomical to store and check all of these responses on chip**
- **Responses must be compacted**

Definitions

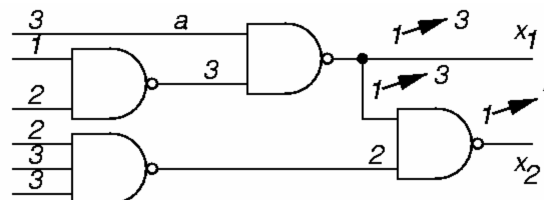
- ***Aliasing*** **– Due to information loss, signatures of good and some bad machines match**
- ***Compaction*** **– Drastically reduce # bits in original circuit response – lose information**
- ***Compression*** **– Reduce # bits in original *circuit response – no information loss – fully invertible* (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**

15 April 2007 37

Transition Counting



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



(b) Transition counts of good and failing machines.

15 April 2007 38

Transition Counting Details

- **Transition count:**

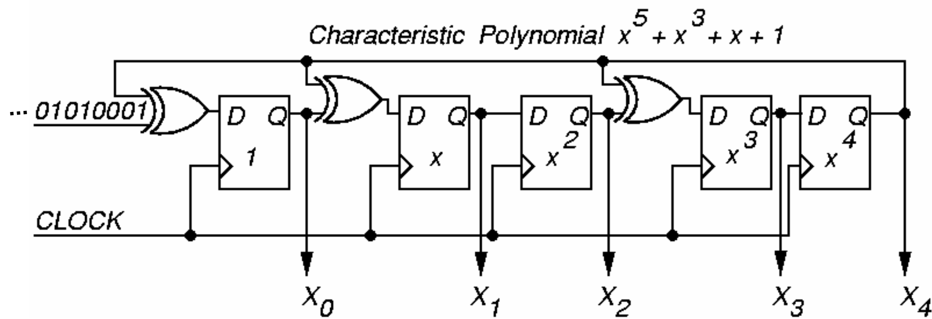
  $$C(R) = \sum_{i=1}^{m} (r_i \oplus r_{i-1}) \text{ for all } m \text{ primary outputs}$$

- **To maximize fault coverage:**
  - **Make $C(R0)$ – 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



- **LFSR seed value is "00000"**

Polynomial Division

	Inputs	x^0	x^1	x^2	x^3	x^4
	Initial State	0	0	0	0	0
	1	1	0	0	0	0
	0	0	1	0	0	0
Logic Simulation:	0	0	0	1	0	0
	0	0	0	0	1	0
	1	1	0	0	0	1
	0	1	0	0	1	0
	1	1	1	0	0	1
	0	1	0	1	1	0

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

$$
\begin{array}{r|l}
 & x^2 + 1 \\
\cline{2-2}
x^5 + x^3 + x + 1 & x^7 \qquad + x^3 \qquad + x \\
 & x^7 + x^5 + x^3 + x^2 \\
\cline{2-2}
 & \quad x^5 \qquad + x^2 + x \\
 & \quad x^5 + x^3 \qquad + x + 1 \\
\cline{2-2}
\text{remainder} \rightarrow & \qquad x^3 + x^2 \qquad + 1
\end{array}
$$

Remainder matches that from logic simulation of the response compacter!

Multiple-Input Signature Register (MISR)

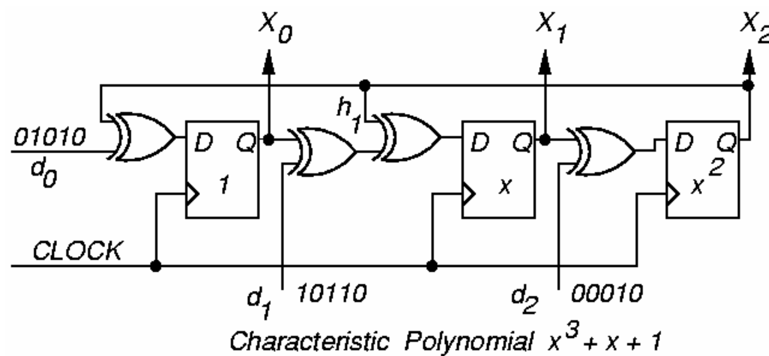
- **Problem with ordinary LFSR response compacter:**
  - **Too much hardware if one of these is put on each *primary output* (PO)**
- **Solution: MISR – compacts all outputs into one LFSR**
  - **Works because LFSR is linear – obeys *superposition principle***
  - **Superimpose all responses in one LFSR – 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

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

Modular MISR Example



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

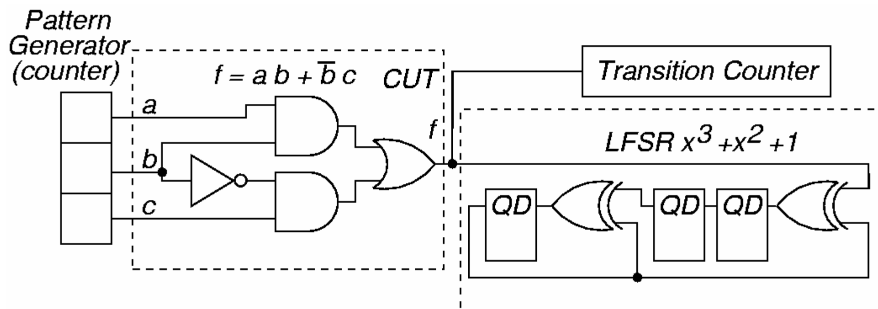
Multiple Signature Checking

- **Use 2 different testing epochs:**
  - **1st with MISR with 1 polynomial**
  - **2nd 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 2nd MISR polynomial**
  - **A 2-1 MUX to select between two feedback polynomials**

Aliasing Probability

- **Aliasing – when bad machine signature equals good machine signature**
- **Aliasing: $1/2^n$**
- **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:**
  - $0 < p \leq 1/2, \quad p^k \leq P_{al} \leq (1 - p)^k$
  - $1/2 \leq p \leq 1, \quad (1 - p)^k \leq P_{al} \leq p^k$

Experiment Hardware



- **3 bit exhaustive binary counter for pattern generator**

Transition Counting vs. LFSR

- **LFSR aliases for *f* sa1, transition counter for *a* sa1**

Pattern	Responses			
abc	***Good***	***a* sa1**	***f* sa1**	***b* sa1**
000	0	0	1	0
001	1	1	1	0
010	0	1	1	0
011	0	1	1	0
100	0	0	1	1
101	1	1	1	1
110	1	1	1	1
111	1	1	1	1
	Signatures			
Transition Count	3	3	0	1
LFSR	001	101	001	010

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**

Appendix

LFSR Fault Coverage Projection

- **Fault detection probability by a random number**
  - $p(x)\,dx$ = fraction of detectable faults with detection probability between x and $x + dx$
  - $p(x)\,dx \geq 0$ 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 $x\,p(x)\,dx$**
- **Mean fault coverage y_n of 1st n vectors:**

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

$$y_n \equiv 1 - I(n) + \frac{n}{\text{total faults}} \qquad (15.6)$$

LFSR Fault Coverage & Vector Length Estimation

- ***Random-fault-detection* (RFD) variable:**
  - Vector # at which fault first detected
  - $w_i \equiv$ # faults with RFD variable i
- **So** $p(x) = \frac{1}{n_s} \sum_{i=1}^{N} w_i\,p_i(x)$
- $n_s \equiv$ **size of sample simulated;** $N \equiv$ **# 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

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**

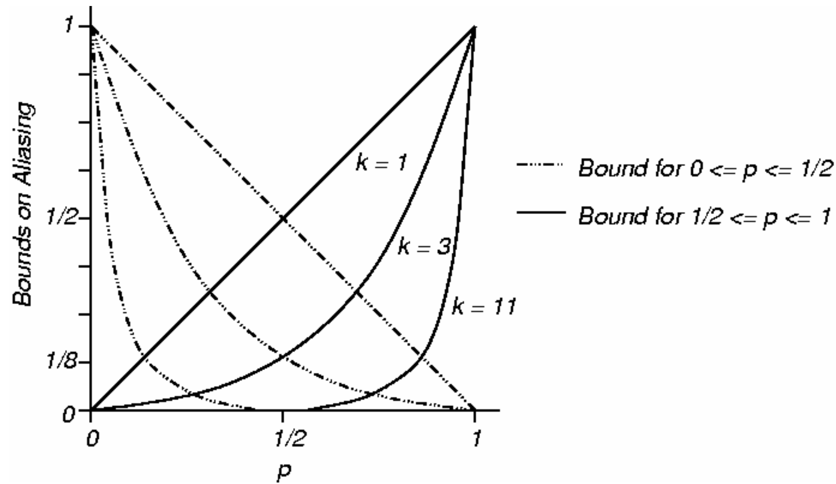
15 April 2007 55

Primitive Polynomials (continued)

  - **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 book for tables of primitive polynomials**
- **Following is related to aliasing:**
  - **If $p(error) = 0.5$, no difference between behavior of primitive & non-primitive polynomial**
  - **But $p(error)$ is rarely $= 0.5$ In that case, non-primitive polynomial LFSR takes much longer to stabilize with random properties than primitive polynomial LFSR**

15 April 2007 56

Aliasing Probability Graph



Additional MISR Aliasing

- **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.**