EEDG/CE 6303: Testing and Testable Design

Mehrdad Nourani

Dept. of ECE
Univ. of Texas at Dallas

Session 10

Built-In Self-Test (BIST)

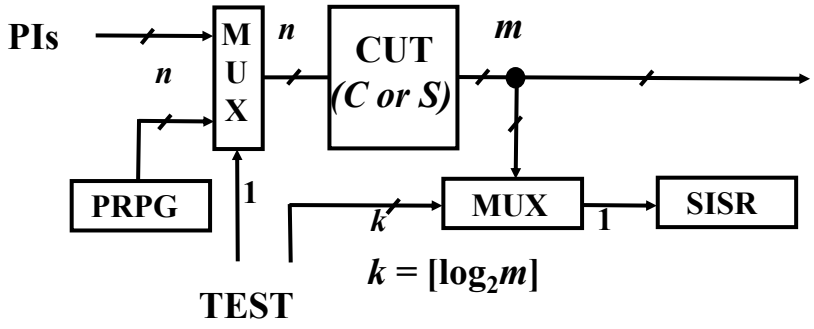
Logic BIST Architectures

Logic BIST Architectures

- Without Scan Chains
  - Centralized and Separate board-level BIST (CSBL)
  - Built-In Evaluation and Self-Test (BEST)
- With Scan Chains (test-per-scan)
  - LSSD on-chip self-test (LOCST)
  - Self-testing using MISR and parallel SRSG (STUMPS)
- Using Register Reconfiguration (test-per-clock)
  - Built-In Logic Block Observer (BILBO)
  - Modified BILBO (MBILBO)
  - Concurrent BILBO (CBILBO)
  - Circular self-test path (CSTP)
- Using Concurrent Checking Circuit
  - Concurrent self verification (CSV)

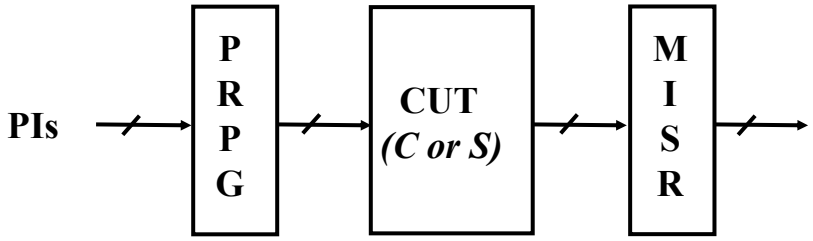
Centralized and Separate Board-Level BIST (CSBL)

- Two LFSRs and two multiplexers are added to the circuit.
  - The first LFSR acts as a PRPG, the second serves as a SISR.
  - The first multiplexer selects the inputs, another routes the PO to the SISR.



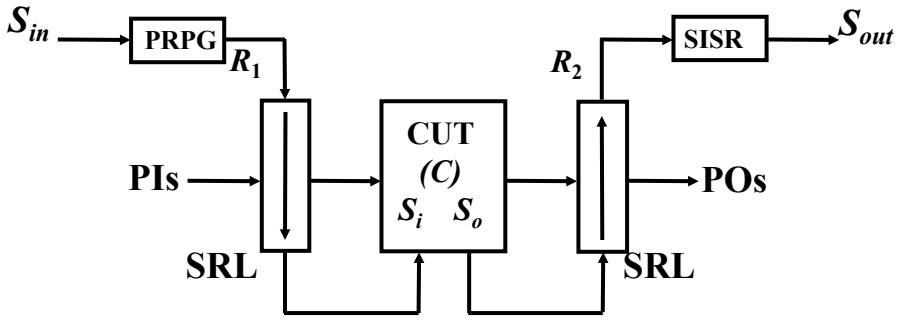
Built-In Evaluation and Self-Test (BEST)

- Use a PRPG and a MISR.
  - —Pseudo-random patterns are applied in parallel from the PRPG to the chip primary inputs (PIs)
  - —An MISR is used to compact the chip output responses



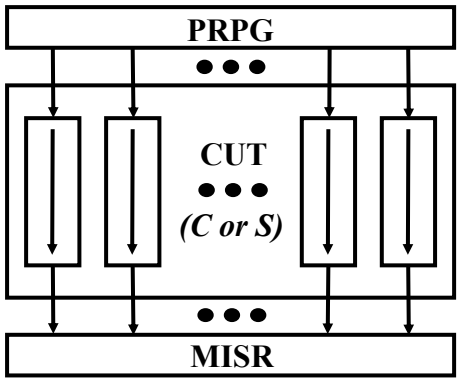
LSSD On-chip Self-Test (LOCST)

- In addition to the internal scan chain, an external scan chain comprising all primary inputs and primary outputs is required.
- The external scan-chain input is connected to the scan-out point of the internal scan chain.

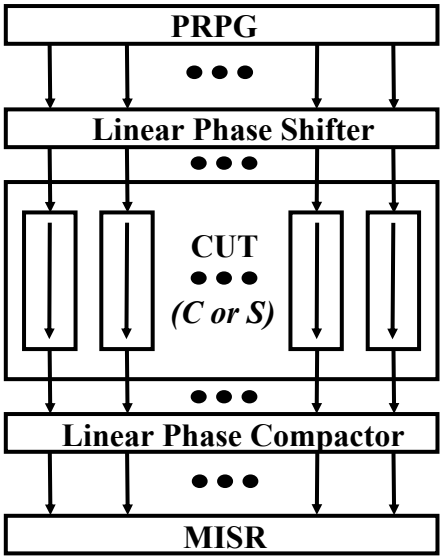


Self-Testing Using MISR & Parallel SRSG (STUMPS)

- The only BIST architecture widely used in industry.
- Contains a PRPG (SRSG) and a MISR.
- The scan chains are loaded in parallel from the PRPG.
- The system clocks are then pulsed and the test responses are scanned out to the MISR for compaction.
- New test patterns are scanned in at the same time when the test responses are being scanned out.



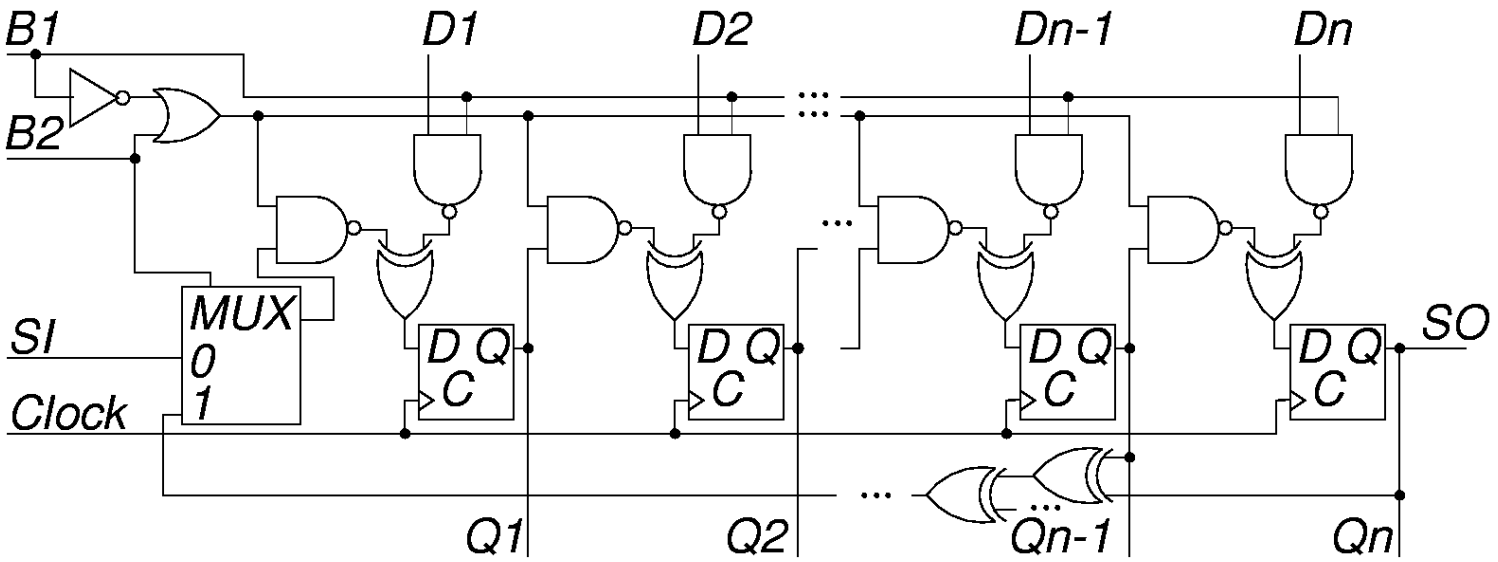
STUMPS



A STUMPS-based Architecture

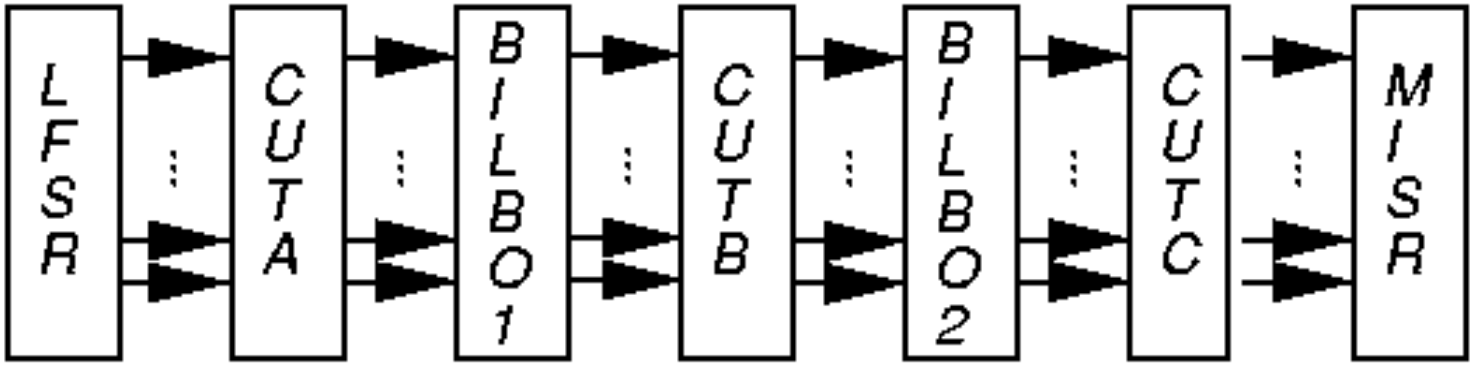
Built-In Logic Block Observer (BILBO)

- Complex systems with multiple chips demand elaborate logic BIST architectures
- *BILBO* and *test / clock* system
  - Shorter test length, more BIST hardware
  - Combined functionality of D flip-flop, *pattern generator*, *response compacter*, & *scan chain*
  - Reset all FFs to 0 by scanning in zeros



Example BILBO Usage

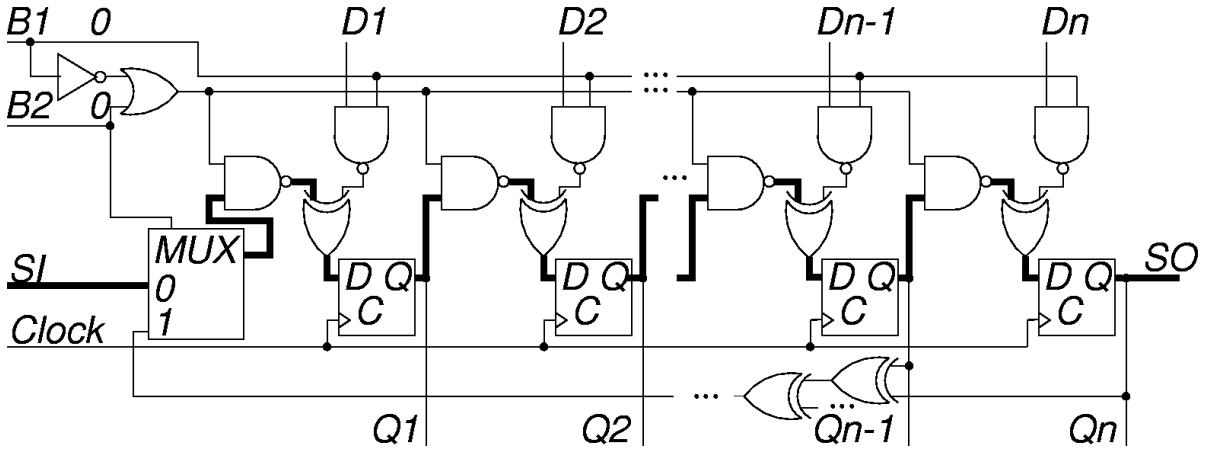
- *SI – Scan In*
- *SO – Scan Out*
- *Characteristic polynomial*: $1 + x + \ldots + x^n$
- CUTs A and C: BILBO1 is MISR, BILBO2 is LFSR
- CUT B: BILBO1 is LFSR, BILBO2 is MISR



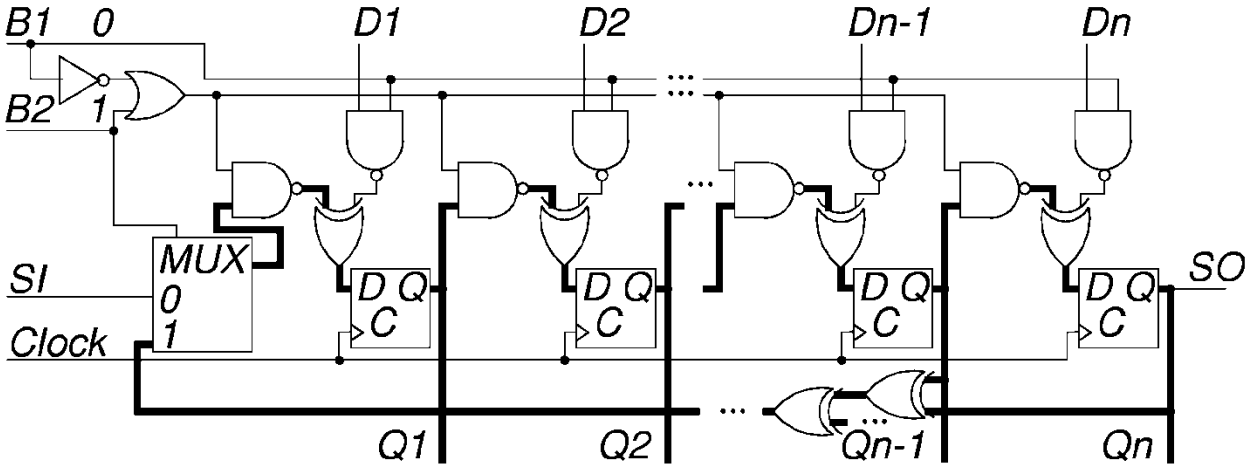
(a) Example test configuration.

BILBO Modes

- **1. Serial Scan Mode -** $B1\ B2$ = "00"

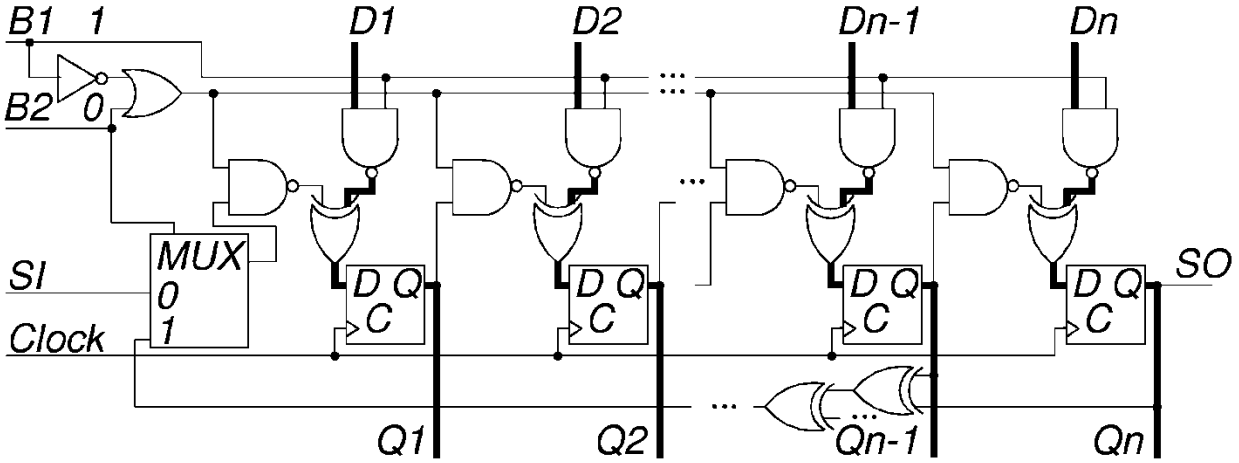


- **2. LFSR Pattern Generator Mode -** $B1\ B2$ = "01"

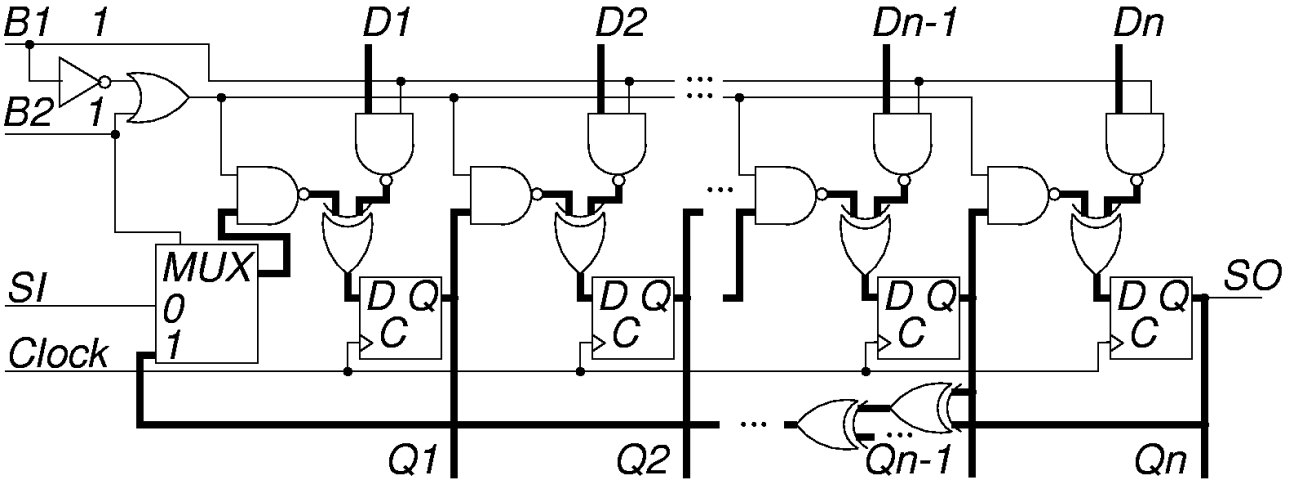


BILBO Modes (cont.)

- **3. D-FF Normal Mode -** $B1\ B2$ = "10"



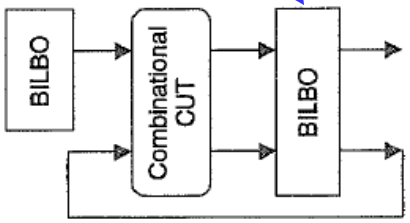
- **4. MISR Mode -** $B1\ B2$ = "11"

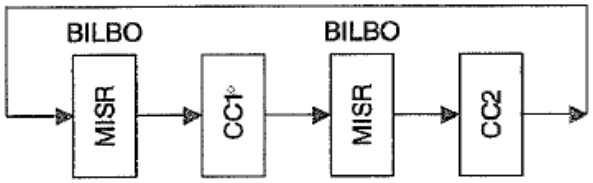


Modified BILBO (MBILBO)

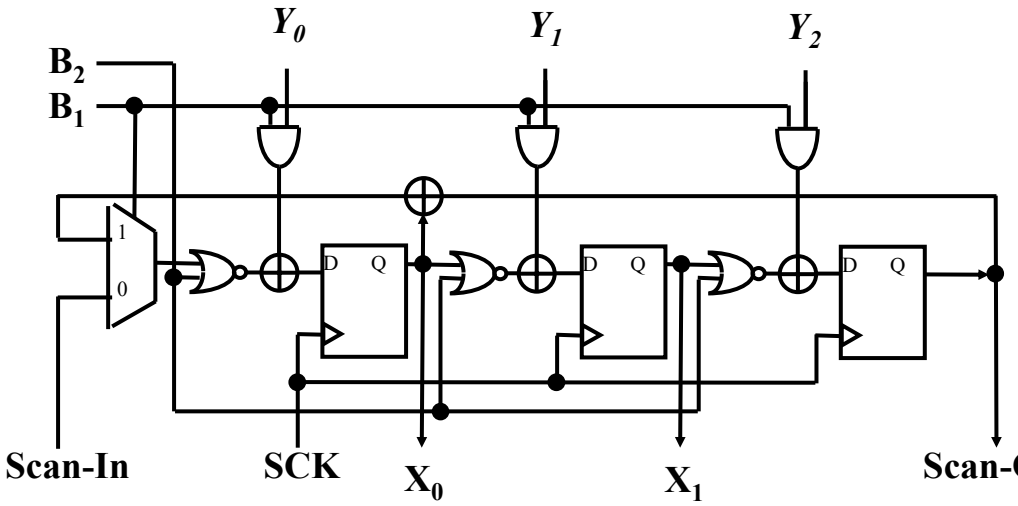
- Example of a 3-stage BILBO using a slightly different structure
  - A NOR gate is added to each input path of D-FF
  - Useful for testing pipelined blocks

Cannot test FSM exhaustively as BILBO must be always in MISR mode.



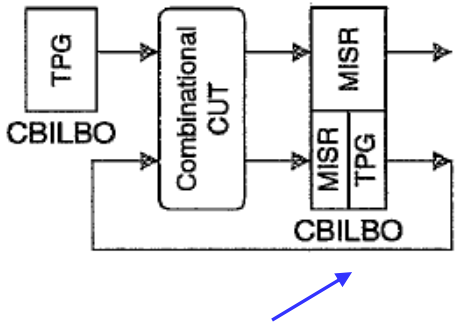


B_1	B_2	Operation mode
1	1	Normal
0	0	Scan
1	0	Mixed Test Generation and Signature Analysis
0	1	Reset

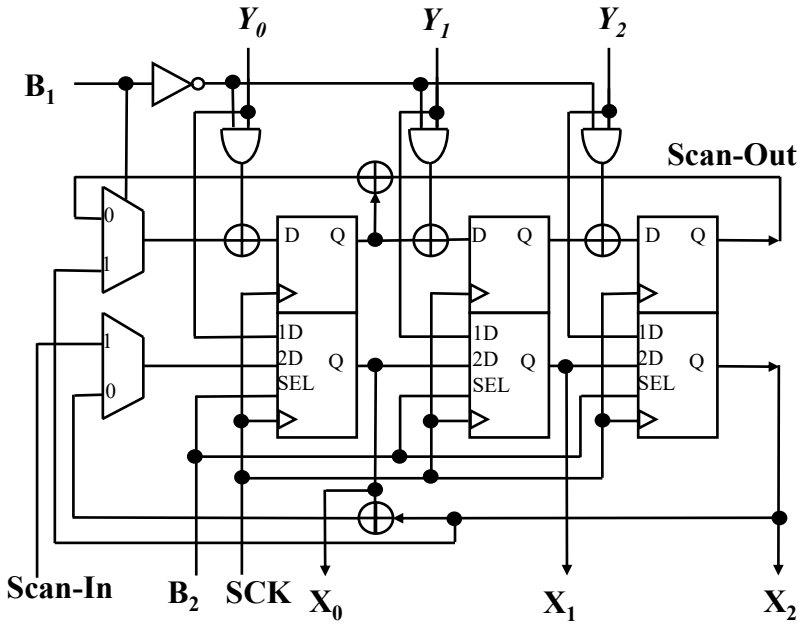


Concurrent Built-In Logic Block Observer (CBILBO)

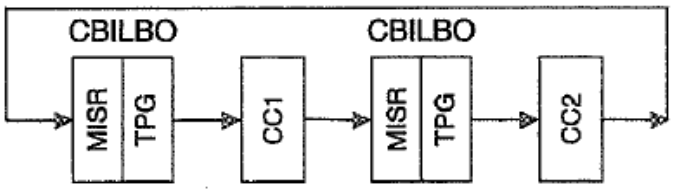
- Three modes of operation
  - B_1B_2=01: the upper D-FFs act as a MISR, the lower two-port D-FFs form a TPG (exhaustive or pseudo-exhaustive TPG possible)



B_1	B_2	Operation mode
-	0	Normal
1	1	Scan
0	1	Test Generation and Signature Analysis



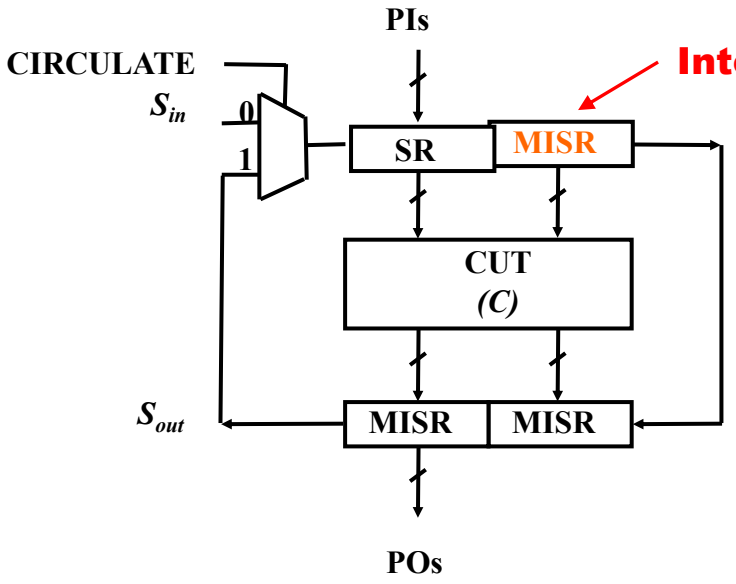
Exhaustive (or pseudo-exhaustive) pattern generation is possible which means 100% s-a-f coverage.



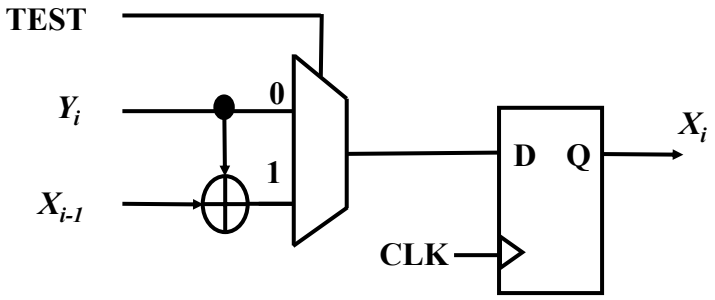
[Wang 1986]

Circular Self-Test Path (CSTP)

- All PIs and POs are reconfigured as external scan cells.
  - They are connected to the internal scan cells to form a circular path. During self-test, all primary inputs (PIs) are connected as a shift register (SR), whereas all internal scan cells and primary outputs (POs) are reconfigured as a MISR.
  - If the entire circular path has n scan cells then it corresponds to a MISR with polynomial $\varphi(x) = 1+x^n$
  - Since $\varphi(x) = 1+x^n$ is non-linear, it may lead to low fault coverage.



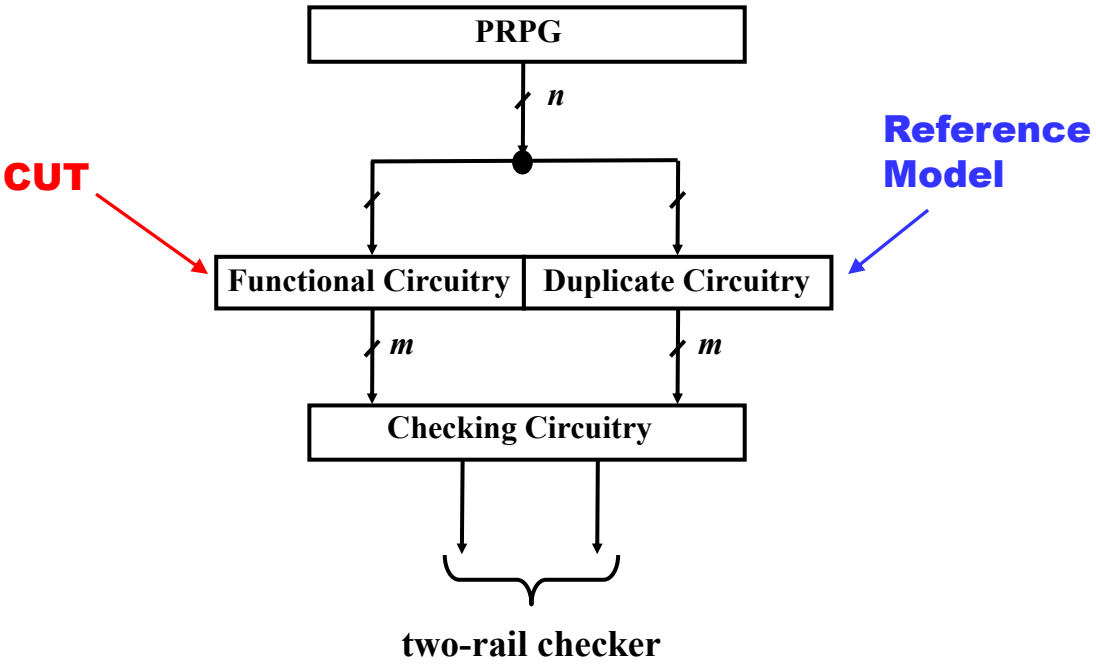
(a) The CSTP architecture



(b) Self-Test cell (MISR cells)

Concurrent Self-Verification (CSV)

- A PRPG is applied to the CUT (functional circuit) and the duplicate circuit and the result is checked.
  - To reduce design and common-mode fault, the duplicate circuit is realized in complementary form.
  - As we compare two modules, this technique avoids aliasing problem and loss of effective fault coverage.



Comparing Logic BIST Architectures

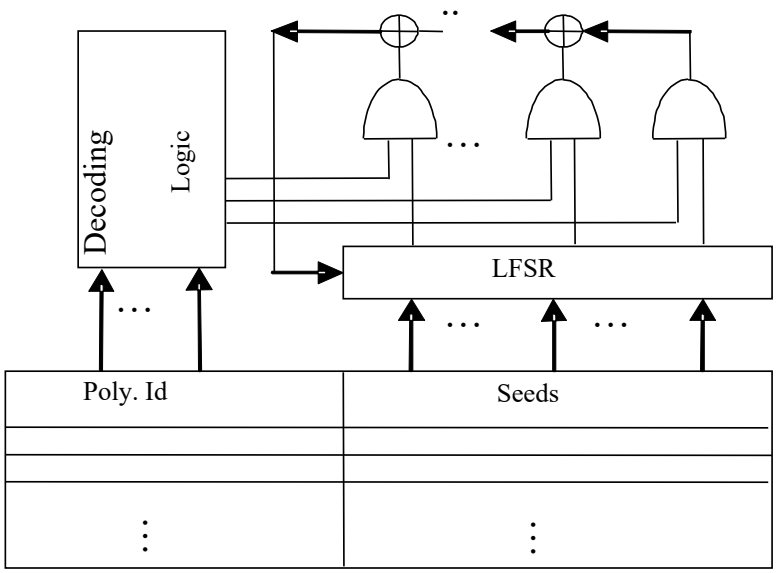
Architecture	*Level*	*TPG*	*ORA*	*Circuit*	*BIST*
CSBL	B or C	PRPG	SISR	C or S	Test-Per-Clock
BEST	B or C	PRPG	MISR	C or S	Test-Per-Clock
LOCST	C	PRPG	SISR	C	Test-Per-Scan
STUMPS	B or C	PRPG	MISR	C	Test-Per-Scan
BILBO	C	PRPG	MISR	C	Test-Per-Clock
CBILBO	C	EPG/PEPG	MISR	C	Test-Per-Clock
CSTP	C	PRPG	MISR	C or S	Test-Per-Clock
CSV	C	PRPG	Checker	C or S	Test-Per-Clock

B: board-level testing
C: combinational circuit
S: sequential circuit

Other BIST Structures/Improvements

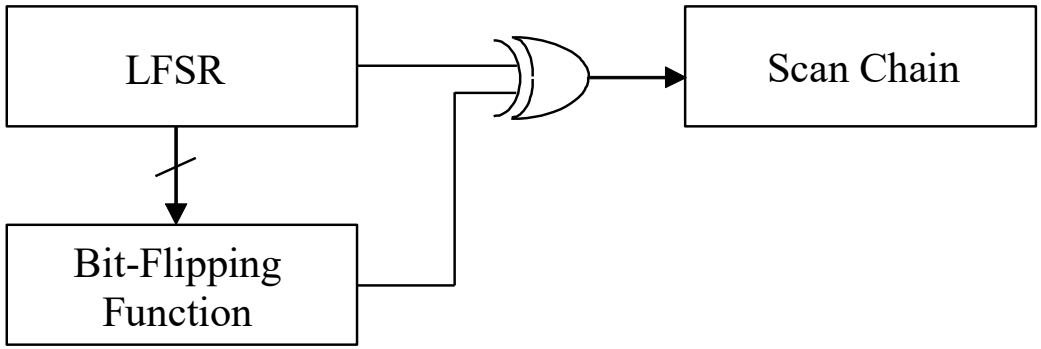
LFSR Reseeding

- The LFSR seeds (initial values) and polynomial are stored to generate both pseudo-random or deterministic patterns.
  - —Multiple-polynomial LFSR (MP-LFSR)
  - —Using variable-length seeds



Embedding Deterministic Patterns

- A bit-flipping function detects useless patterns and maps them to deterministic patterns through an XOR gate that feeds the scan chain.
- The decimation problem causes the sequence of patterns in scan chain to be short (much shorter than the sequence of LFSR).



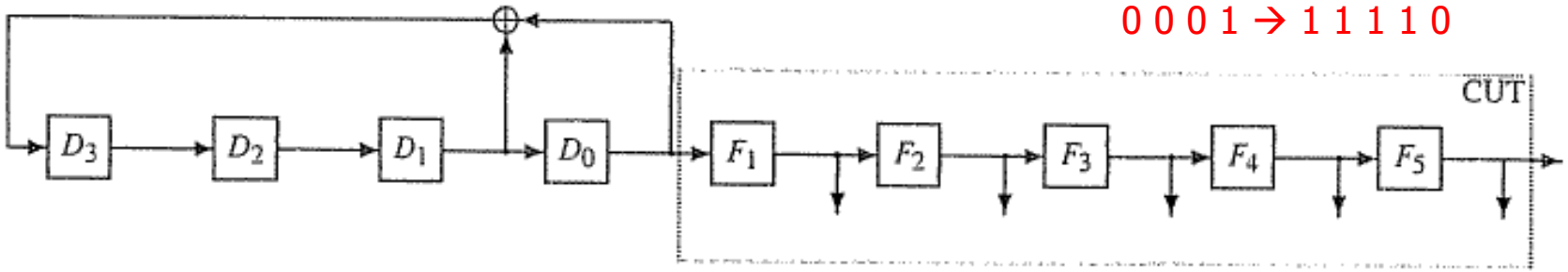
Clocking PRPG in Scan-Based BIST

- A good n-bit PRPG generates 2^n-1 patterns.
- If the length of chain is SCL, every SCL clocks we have one new vector in the chain.
- PRPG goes back to its initial state after (2^n-1)/SCL clocks.
- The reduction of number of vectors is called **decimation**.
- The decimation problem can be eliminated if 2^n-1 and SCL are co-prime.

Clocking PRPG in Scan-Based BIST (cont.)

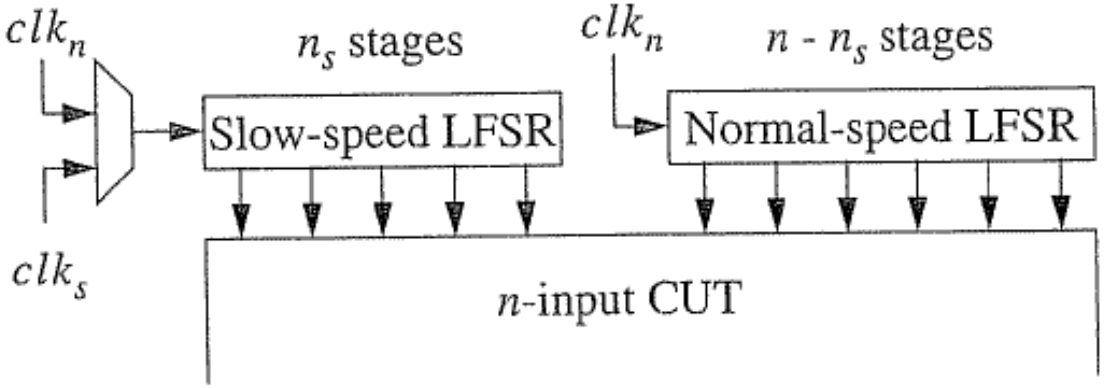
- In this example, n=4, SCL=5, there will be only 2^n-1/SCL= 15/5=3 distinct vectors.
- You can verify that if SCL=7, then 15 distinct 7-bit vectors are applied to the scan chain before the vectors repeat.

$D_3D_2D_1D_0$ →	$F_1F_2F_3F_4F_5$
1 0 0 0 →	0 0 0 0 0
0 1 0 0 →	0 0 0 0 0
0 0 1 0 →	0 0 0 0 0
1 0 0 1 →	1 0 0 0 0
1 1 0 0 →	0 1 0 0 0
0 1 1 0 →	0 0 1 0 0
1 0 1 1 →	1 0 0 1 0
0 1 0 1 →	1 1 0 0 1
1 0 1 0 →	0 1 1 0 0
1 1 0 1 →	1 0 1 1 0
1 1 1 0 →	0 1 0 1 1
1 1 1 1 →	1 0 1 0 1
0 1 1 1 →	1 1 0 1 0
0 0 1 1 →	1 1 1 0 1
0 0 0 1 →	1 1 1 1 0



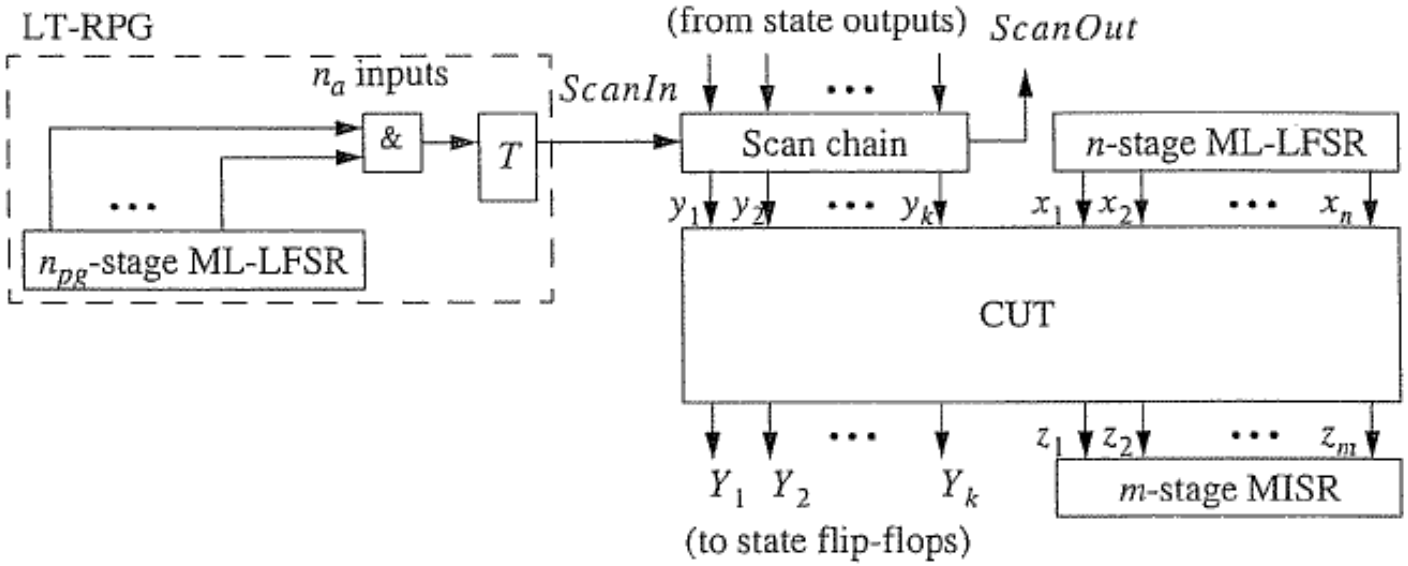
Reducing Switching Activity in BIST

- As the vectors are random, they have low correlation and thus the number of transitions between consecutive vectors can be high.
  - Excessive heat dissipation and damage during test
  - Excessive power/ground noise results in failing a fault-free circuit
- Dual-speed LFSR (DS-LFSR) can be used. For an n-input CUT:
  - n_s stages slow-speed LFSR
  - n-n_s stages normal-speed LFSR
  - For having N_v vectors ($N_v<2^n$) , we need clock-ratio=$clk_n/clk_s=N_v/2^{ns}$)
  - Example: For n=4 and N_v=16, we have 3 choices: (i) clock-ratio=8,n_s=1 , (ii) clock-ratio=4,n_s=2 and (iii) clock-ratio=2,n_s=3



Low-Transition PRPG

- A weighted pseudorandom signal (e.g. output of the AND gate in LT-RPG) drives the toggle input of a T-flip-flop.
  - —If the output of AND gate is 0 for m consecutive cycles, then identical values are applied at ScanIn for m clocks. So, the switching among $y_1 \ldots y_k$ is reduced.



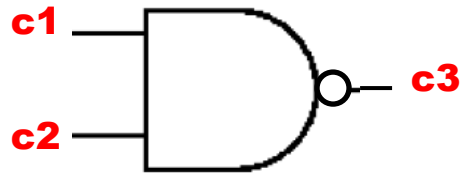
Detectability &
Estimation of Test Length

Importance of Testability Metrics

- The relationship between test length and test quality depends on the nature of the CUT.
  - If all the faults in the CUT have a large number of tests, then a short sequence can provide high fault coverage.
  - In contrast, if many faults in the CUT have very few tests, then a long sequence is required to obtain high fault coverage.
- Controllability and Observability metrics (similar to SCOAP) can be defined to find problematic (weak) points.
- In BIST analysis, we prefer the metrics to be in [0-1] as we deal with probability analysis.

Controllability Metric

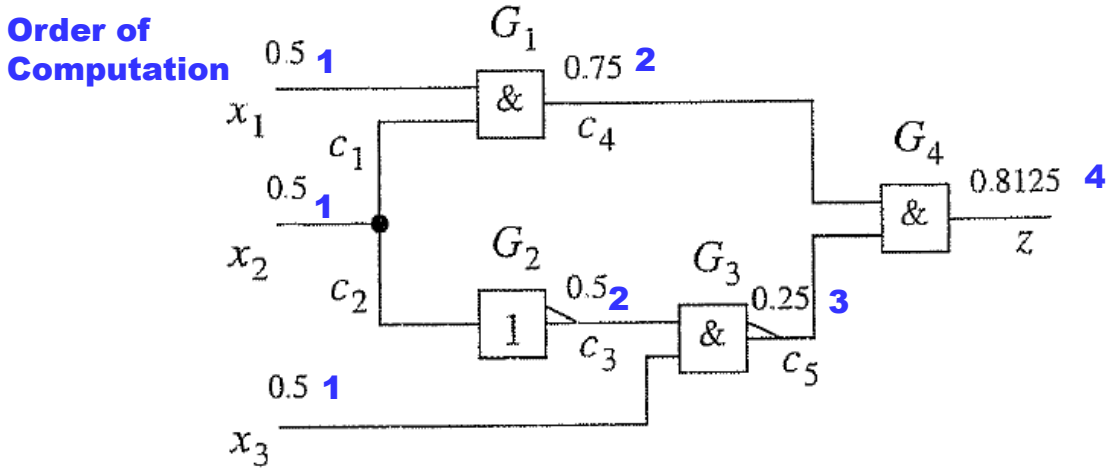
- The overall framework for computation of controllability values can be used to compute the probabilistic controllability values.
- For each primary input x_i, $CC_p^0(x_i)$ and $CC_p^1(x_i)$ are initialized to 0.5.
  - PRPG generates logic 0 and logic 1 at each input with equal probabilities.
- The controllability transfer function provides the rule for computing the controllability values at a gate's output given their values at its inputs.
- For a NAND gate with inputs c_1 and c_2 and output c_3, assuming the logic values in two inputs are independent:
  - $CC_p^0(c_3) = CC_p^1(c_1)\, CC_p^1(c_2)$
  - $CC_p^1(c_3) = 1 - CC_p^1(c_1)\, CC_p^1(c_2)$



Controllability Metric (cont.)

- The breadth-first (forward traversal – level based) approach is used to compute the probabilistic controllability values for each line in the circuit
- Example:
  - The circuit has a fanout at x_2 that reconverges at G_4.
  - The above approach is used to compute the probabilistic controllability values for circuit lines
    - 0-controllability values, CCp0 values are shown in the figure below.
    - For any line c_i, $CC_p^1(c_i) = 1 - CC_p^0(c_i)$

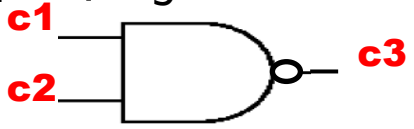
- AND (z=x.y)
  - $CC_p^1(z) = CC_p^1(x)\, CC_p^1(y)$
  - $CC_p^0(z) = 1 - CC_p^1(z)$
- NAND ($z=\overline{x.y}$)
  - $CC_p^0(z) = CC_p^1(x)\, CC_p^1(y)$
  - $CC_p^1(z) = 1 - CC_p^0(z)$



Observability Metric

- We define the probabilistic observability of a line c_i in an n-input combinational circuit as the probability that a randomly selected vector will sensitize one or more paths from line c_i, to one or more of the circuit outputs.
- The observability of each primary output z is $O_p(z_i) = 1$. The probabilistic observability is computed for the other circuit lines via a breadth-first, backward traversal of the circuit lines starting at the primary output.
- For a 2-input NAND gate, assuming the controllability values are known for each lines and the probabilistic observability is known only for the gate output, c_3.

  $O_p(c_1) = CC_p^1(c_2)O_p(c_3)$



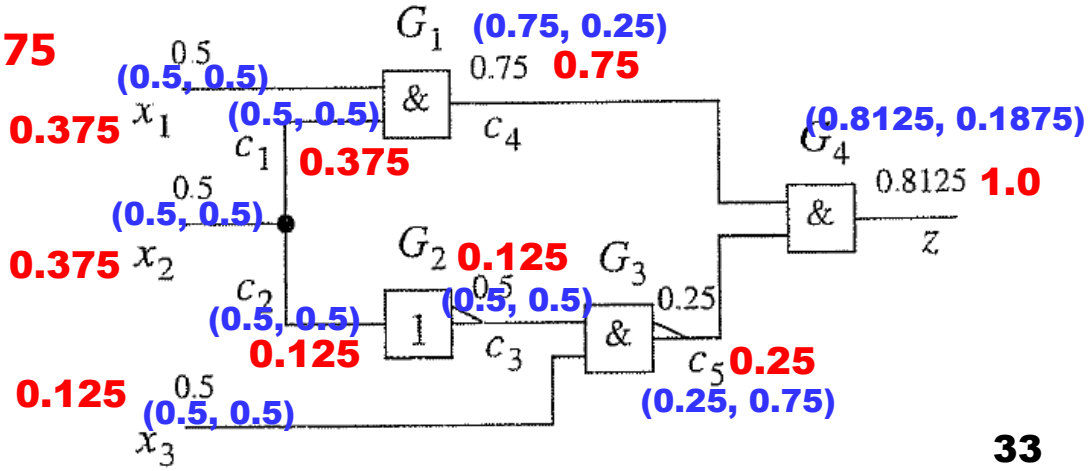
- At a fanout system with stem c_1 and branches $c_2, c_3, \ldots, c_n$:

  $O_p(c_1) = \max[O_p(c_2), O_p(c_3), \ldots, O_p(c_n)]$

Observability Metric (cont.)

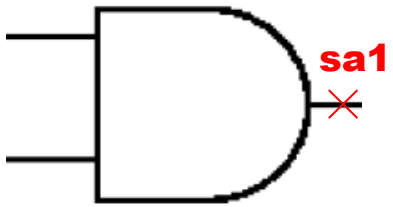
- The backward analysis approach is used
- Example:
  - The circuit has an output at z.
  - $O_p(z)$=**1.0**
  - $O_p(c_4) = CC_p^1(c_5)O_p(z)$=0.75*1.0=**0.75**
  - $O_p(c_5) = CC_p^1(c_4)O_p(z)$=0.25*1.0=**0.25**
  - $O_p(c_1) = CC_p^1(x_1)O_p(c_4)$=0.5*0.75=**0.375**
  - $O_p(x_1) = CC_p^1(c_1)O_p(c_4)$=0.5*0.75=**0.375**
  - $O_p(c_3) = CC_p^1(x_3)O_p(c_5)$=0.5*0.25=**0.125**
  - $O_p(x_3) = CC_p^1(c_3)O_p(c_5)$=0.5*0.25=**0.125**
  - $O_p(c_2) = O_p(c_3)$=**0.125**
  - $O_p(x_2)$ = MAX{$O_p(c_1)$,$O_p(c_2)$}=**0.375**

Legend: (CC^0, CC^1) O



Detectability Metrics

- Once $CC^0()$, $CC^1()$ and $O()$ values are known for all points of a circuit, the probabilistic detectability of a fault f at line c_i can be defined:
  - $D_p(f)=CC_p^1(c_i)O_p(c_i)$, if f is an SA0 fault at c_i
  - $D_p(f)=CC_p^0(c_i)O_p(c_i)$, if f is an SA1 fault at c_i
- The detectability of a fault f in an n-input CUT, κ_f, is the number of vectors that can detect the fault.
- Example: Let f be the stuck-at 1 (SA1) fault at the output of a circuit comprised of a single two-input AND gate. Since f has three tests,{00, 01, 10}, the detectability of f, $\kappa_f=3$.



Escape Probability

- If PRPG generates all possible 2^n vectors then, $\kappa_f = 2^n \cdot D_p(f)$.
  - If we apply N vectors, N-κ_f vectors do not detect the fault f.
- Escape probability of a fault is the probability that a fault f with detectability κ_f will escape detection.
- Detectability Profile: The frequency distribution of the detectabilities of all faults within the circuit. $H=\{h_{\kappa min}, h_{\kappa min+1}, \ldots, h_{\kappa max}\}$
  - Example: If in a 2-input AND gate, all 6 SAFs are targeted, then: $H=\{h_1=5, h_3=1\}$.

There are 5 faults that have 1 vector (for each) to detect.

There is 1 fault that has 3 vectors to detect it.

Escape Probability – Random Testing

- Escape probability of a fault is the probability that a fault f with detectability κ_f will escape detection. Assume $N=2^n$:
  - after application of one vector: $Q_\kappa^r(1)=(N-\kappa)/N$
  - after application of L vectors: $Q_\kappa^r(L)=[(N-\kappa)/N]^L$
- If we use well-known inequality $\ln(a)\leq a-1$, we will get: $[(N-\kappa)/N]^L\leq e^{-\kappa L/N} \rightarrow Q_\kappa^r(L)\leq e^{-\kappa L/N}$
- In practice $\kappa/N\leq 0.1$ and the bound in above inequality is tight ($\leq$ can be changed to approximately equal). So, we have: $Q_\kappa^r(L)\approx e^{-\kappa L/N}$

Test Length – Random Testing

- Assume in a circuit:
  - N_v random vectors (out of total $N=2^n$) are applied
  - The hardest-to-detect-faults have detectability of $h_{\kappa min}$
  - The total number of target faults: $n_f=\Sigma_{\kappa=\kappa min}^{\kappa max} h_\kappa$
  - We want to ensure that each target fault is detected with a probability exceeding 1-ε, where the level of confidence (escape probability) is ε.
- Using probabilistic analysis, we can show:
  - The expected fault coverage:

    $$E(C^r(N_v)) \approx 1-(1/n_f \Sigma_{\kappa=\kappa min}^{\kappa max} h_\kappa e^{-\kappa Nv/N})$$

  - The minimum test length required to achieve level of confidence of ε (i.e. should guarantee $Q_{\kappa min}^r(N_v) \le \varepsilon$). This results in: $N_v \ge -\ln(\varepsilon)N/\kappa_{min}$

($Q_{\kappa min}^r(N_v) \le \varepsilon \rightarrow e^{-\kappa min Nv/N} \le \varepsilon \rightarrow -\kappa_{min}N_v/N \le \ln(\varepsilon) \rightarrow N_v \ge -\ln(\varepsilon)N/\kappa_{min}$)

Escape Probability – Pseudo-Random Testing

- Similar probabilistic analysis shows that:
  - —after application of one vector: $Q_\kappa^{pr}(1)=(N-\kappa)/N$
  - —after application of L vectors: $Q_\kappa^{pr}(L)=\Pi_{i=1}^{L}[(N-\kappa-i+1)/(N-i+1)]$

- Typically, $L<<N$ and $\kappa<<N$, and we will get: $Q_\kappa^{pr}(L)\leq e^{-\kappa L/N}$ (similar to case of random patterns)

Test Length – Pseudo-Random Testing

- Similar to the random case, probabilistic analysis show that:
  - The expected fault coverage:
    $E(C^{pr}(N_v)) \leq 1-(1/n_f \Sigma_{\kappa=\kappa min}^{\kappa max} h_\kappa e^{-\kappa Nv/N})$
  - The minimum test length required to achieve level of confidence of ε (i.e. should guarantee $Q_{\kappa min}^{pr}(N_v) \leq \varepsilon$). This results in N_v that satisfies this relation:
    $[(N-\kappa_{min}-N_v+1)/(N-N_v+1)]^{Nv} \leq \varepsilon$

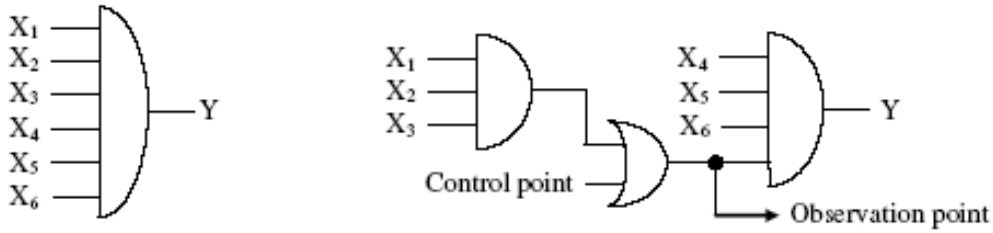
Fault Coverage Enhancement

Test Points

- When the expected fault coverage for a given CUT is less than that desired under the given constraint on test length, one is faced with two alternatives
  1. The CUT can be made more testable via insertion of test points to improve the detectabilities of hard-to-detect faults.
  2. The PG can be customized to generate vectors that are more suitable for the given CUT.
- Two categories of test point improvements
  - Control points
  - Observation points

Test Points (cont.)

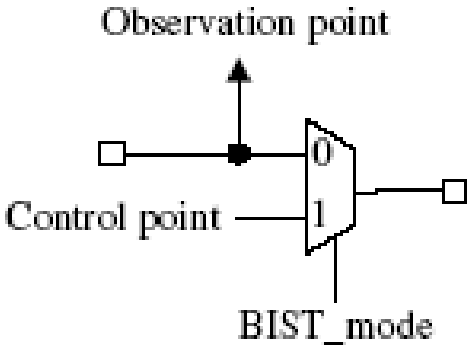
- By enabling control of the value at its site, a control point also, alters controllabilities of the lines in its fanout.
  - A control point also alters the observabilities of lines.
- An observation point enhances the observability of its site by enabling observation of the value at the line.
  - An observation point also enhances observabilities of lines in the fanin of its site.
- Example: Improve detecting Y SA0 in this 6-input AND.



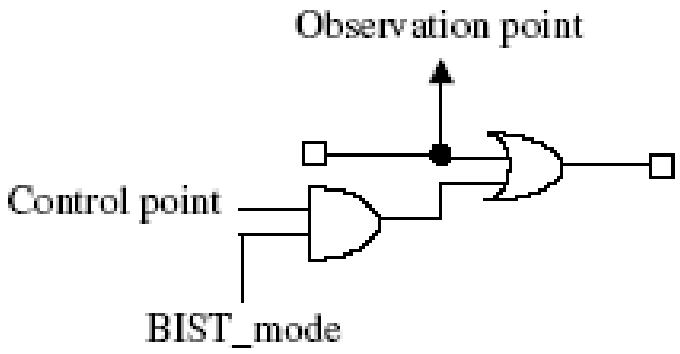
(a) An output RP-resistant stuck-at-0 fault (b) Example inserted test points

Examples of Test Point Insertion

- Typical test point insertion uses MUX/gates to inject values from PIs and connects wires to POs.



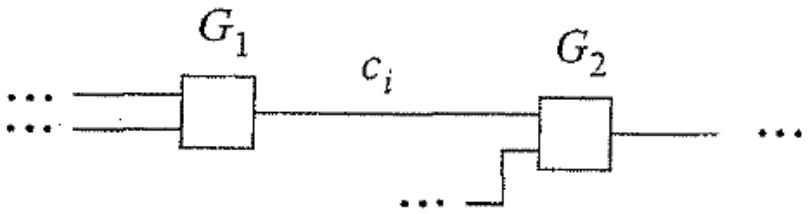
(a) Test point with a multiplexer



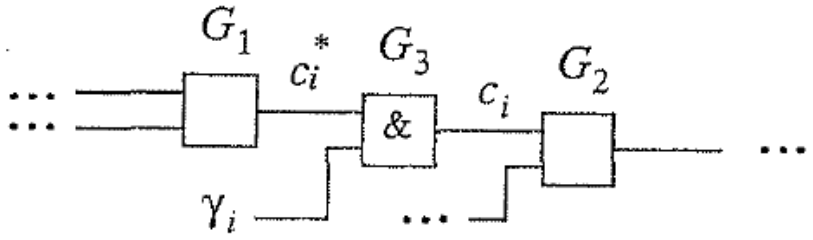
(b) Test point with AND-OR gates

Examples of Test Point Insertion (cont.)

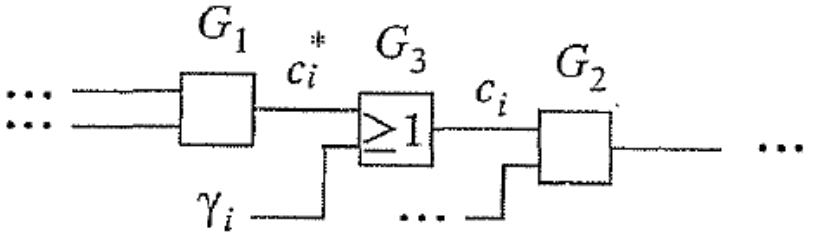
- Various types of test points:



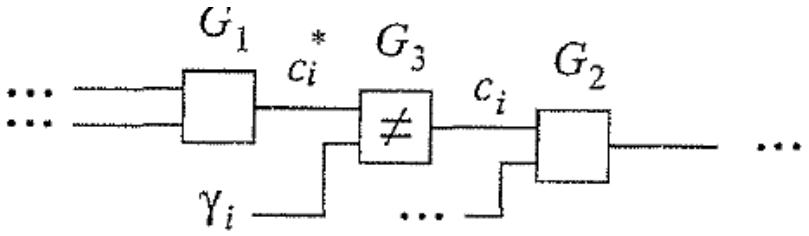
(a) **Original Circuit**



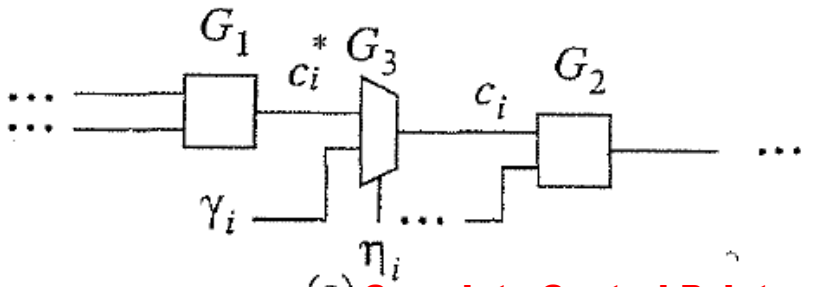
(b) **0-Control Point**



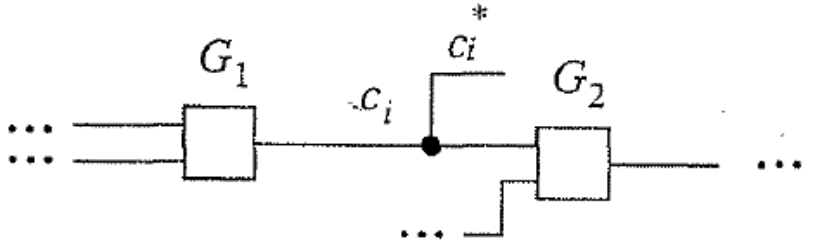
(c) **1-Control Point**



(d) **Inversion-Control Point**



(e) **Complete-Control Point**



(f) **Observation Point**