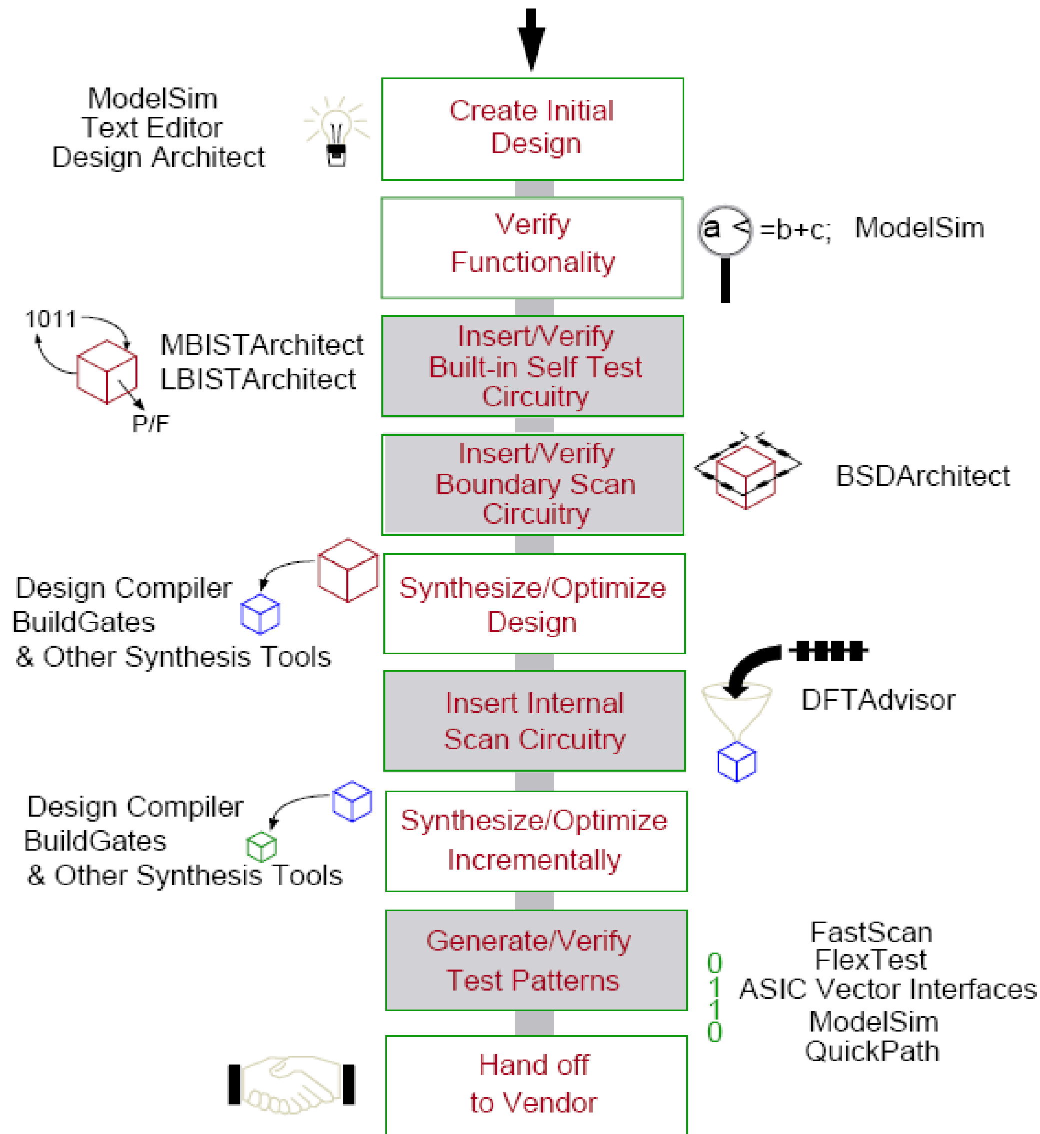
Built-In Self Test

Smith Text: Chapter 14.7

Top-down test design flow

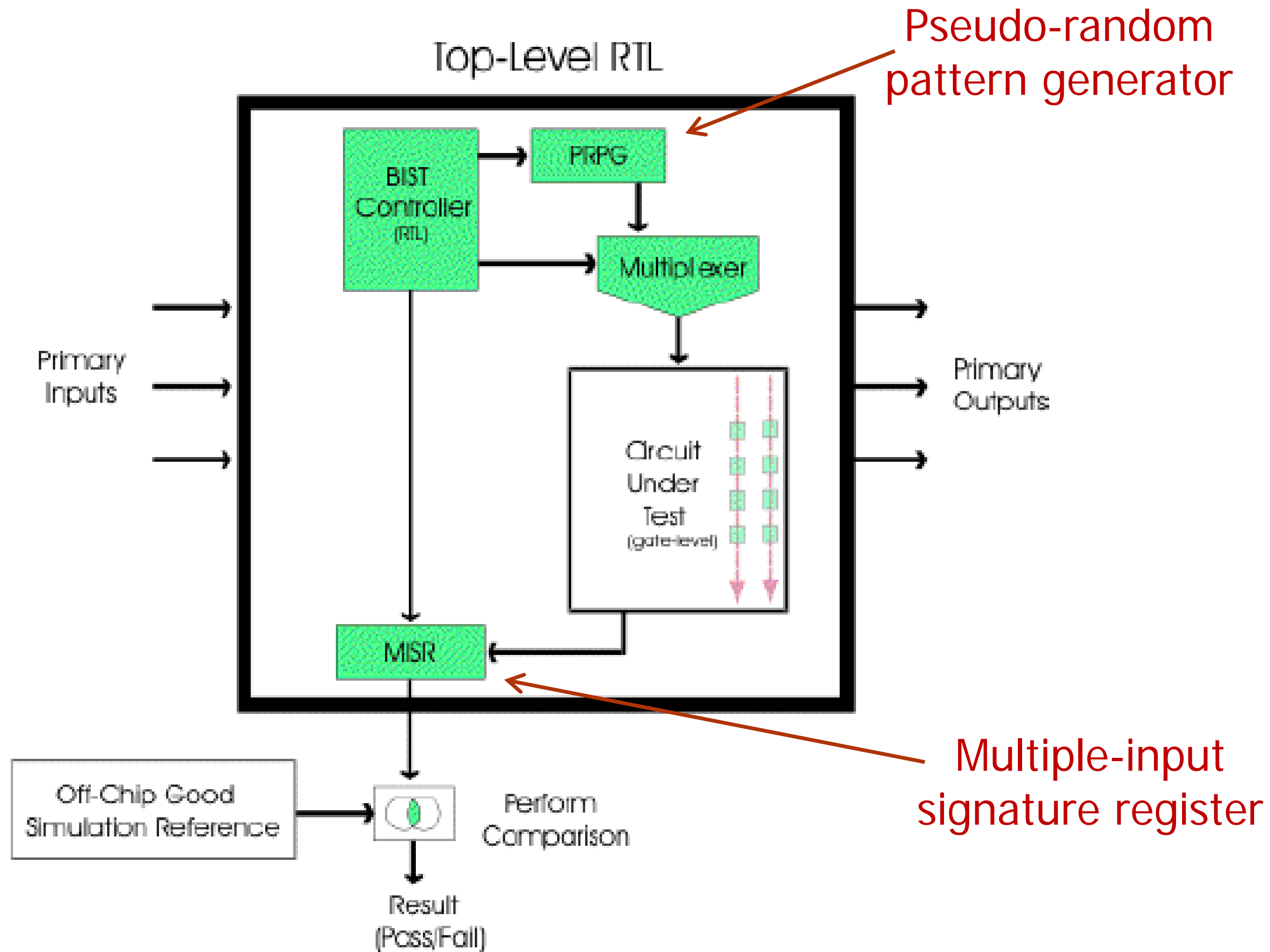


Source: FlexTest Manual

Built-In Self-Test (BIST)

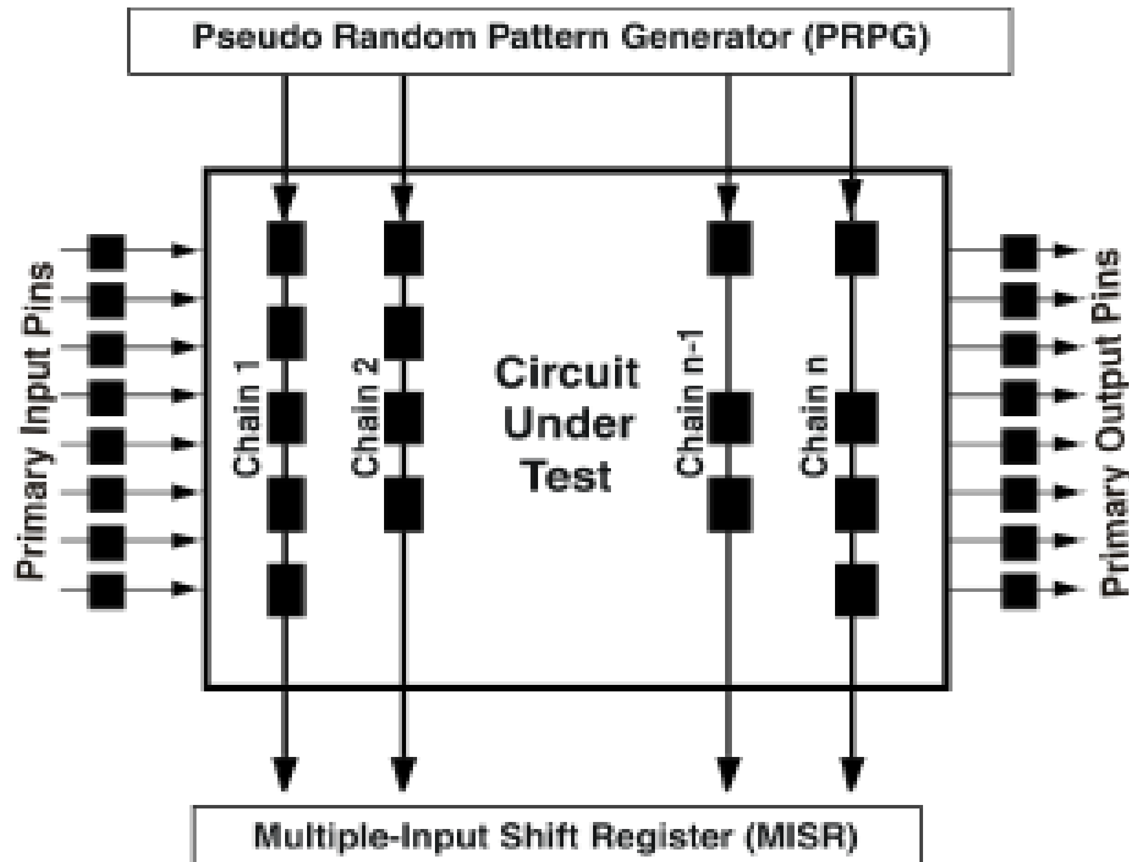
- Structured-test techniques for logic circuits to improve access to internal signals from primary inputs/outputs
- BIST procedure:
  - generate a test pattern
  - apply the pattern to "circuit under test" (CUT)
  - check the response
  - repeat for each test pattern
- Most BIST approaches use pseudo-random test vectors

Logic BIST architecture



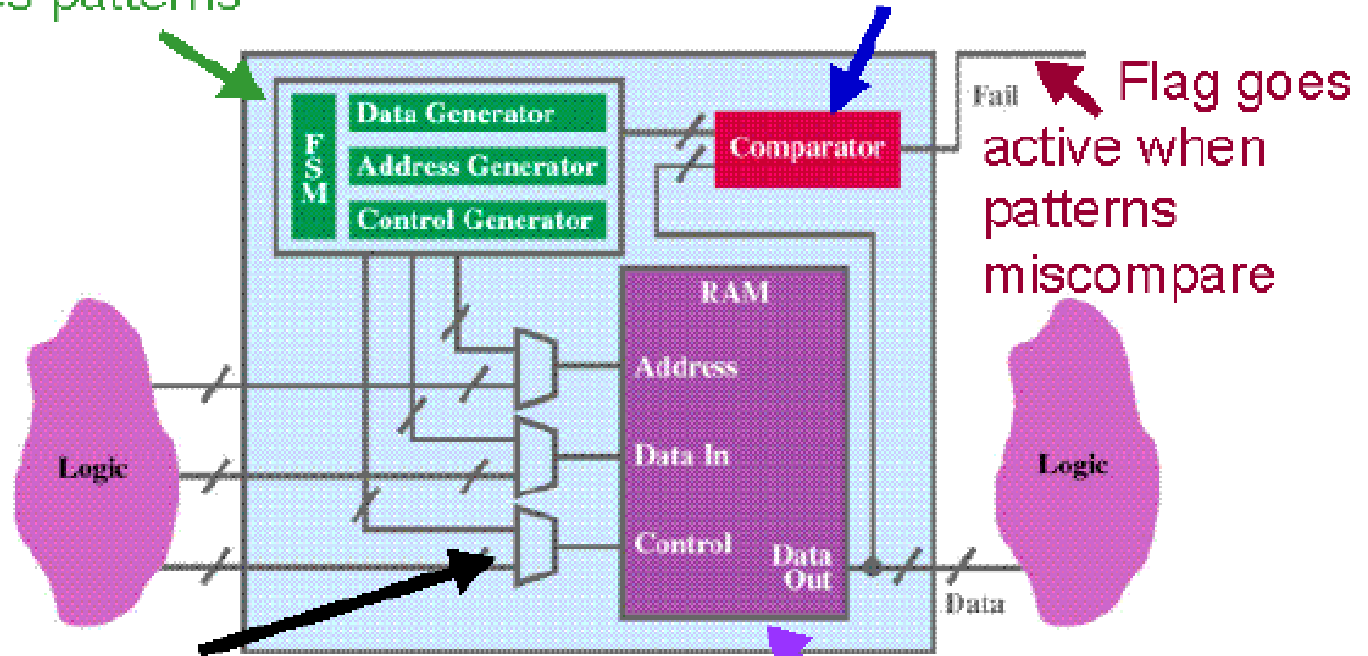
Source: Mentor Graphics "LBISTArchitect Process Guide"

Logic BIST general architecture



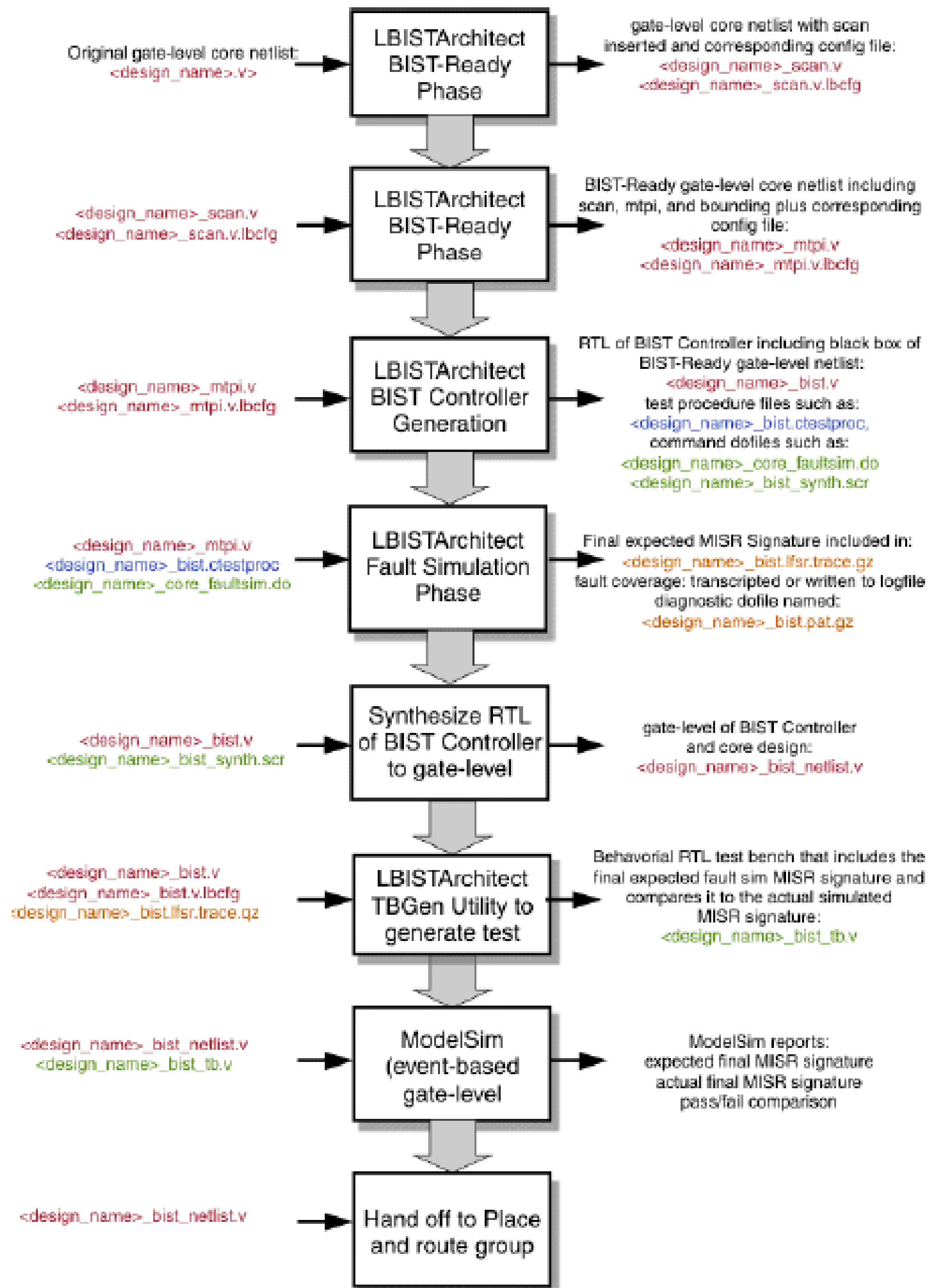
Source: Mentor Graphics "LBISTArchitect Process Guide"

Memory BIST architecture



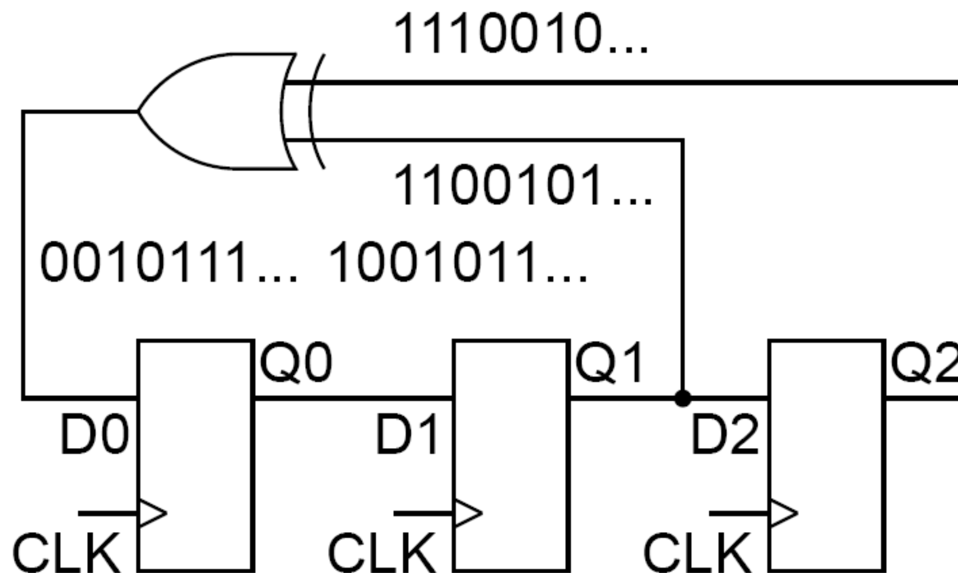
Source: Mentor Graphics "MBISTArchitect Process Guide"

LBIST Process Flow



Linear Feedback Shift Register (LFSR)

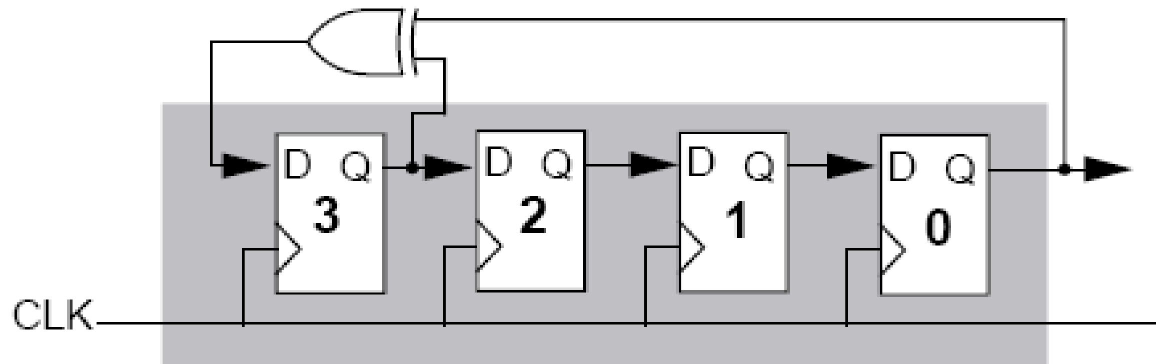
- Produce pseudorandom binary sequences (PRBS)
- Implement with shift register and XOR gates
- Selection of feedback points allows n-bit register to produce a PRBS of length 2^n-1



LFSR produces pattern: 7,3,1,4,2,5,6 (PRBS length 7)

Smith text figure 14.23

4-stage LFSR with one tap point

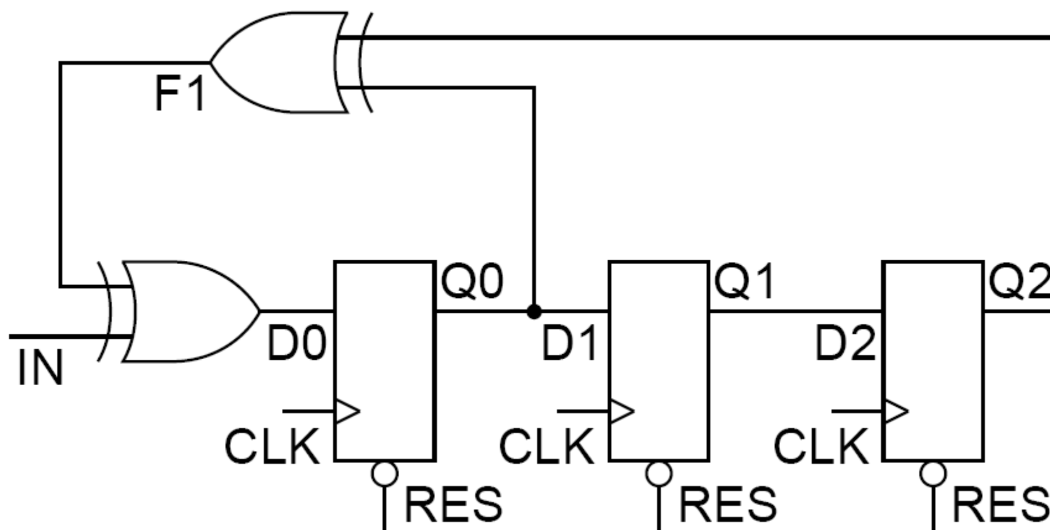


State	Pattern	State	Pattern
0	1000	8	1101
1	1100	9	0110
2	1110	10	0011
3	1111	11	1001
4	0111	12	0100
5	1011	13	0010
6	0101	14	0001
7	1010	15	1000

Source: Mentor Graphics "LBISTArchitect Process Guide"

Serial Input Signature Register (SISR)

- Use an LFSR to compact serial input data into an n-bit "signature"
- For sufficiently large n, two different sequences producing the same signature is unlikely
- Good circuit has a unique signature



Text figure 14.24

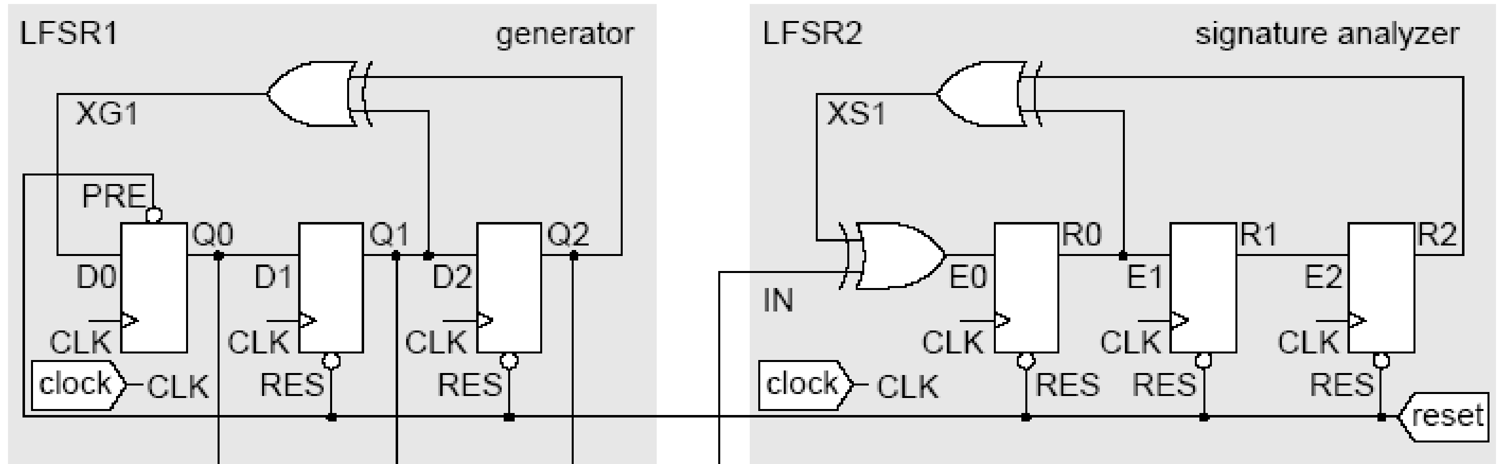
Initialize LFSR to '000' via RES.

Signature formed via shift & add

BIST Example (Fig. 14.25)

Pattern generator

Signature analyzer



Generated test patterns

good 1011100 0101110 0010111

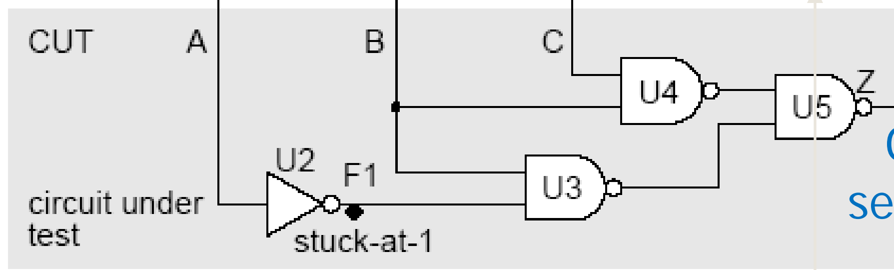
bad 1011100 0101110 0010111

01001100

01011100

00111110 00011111 00001111

00111000 00011100 00001110



Output sequences

Circuit under test

signatures:

good = hex 3 = 011

R0 = 0, R1 = 1, R2 = 1

bad (F1) = hex 0 = 000

R0 = 0, R1 = 0, R2 = 0

Signatures

Aliasing

- Good and bad circuits might produce the same signature ("aliasing") – masking errors
- <u>Previous example:</u>
  - 7-bit sequence applied to signature analyzer
    $2^7 = 128$ possible patterns
  - 3-bit signature register: $2^3 = 8$ possible signatures
  - $128/8 = 16$ streams can produce the good signature: 1 corresponds to good circuit, 15 to faulty circuits
    (assume all bit streams equally likely)
  - $128-1 = 127$ streams correspond to bad circuits
  - $15/127 = 11.8\%$ of bad bit streams produce the good signature, and therefore will be undetected
    (Probability of missing a bad circuit = 11.8%)

Aliasing – Error Probability

- Given test sequence length L & signature register length R
- Probability of aliasing is:

$$p = \frac{2^{L-R} - 1}{2^L - 1}$$

- For L >> R:

$$p \approx 2^{-R}$$

- Use long sequences to minimize aliasing

LFSR Theory (chap 14.7.5)

- Operation based on polynomials and Galois-field theory used in coding
- Each LFSR has a "characteristic polynomial"
  - Called a "primitive polynomial" if it generates a maximum-length PRBS
  - General form: $P(x) = c_0 \oplus c_1 x^1 \oplus \ldots \oplus c_n x^n$

    c_k always 0 or 1, $\oplus$ = xor
  - Reciprocal of P(x) is also primitive:

    $P^*(x) = x^n P(x^{-1})$
- LFSR can be constructed from P(x) or P*(x)

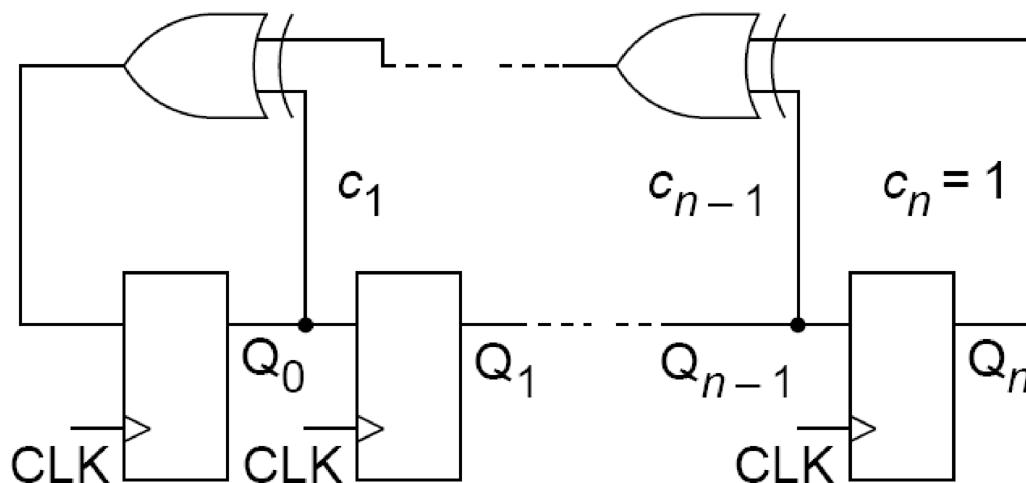
Primitive polynomial examples

- $P(x) = 1 \oplus x^1 \oplus x^3$
  - Order: $n = 3$
  - Coefficients: $c_0=1$, $c_1=1$, $c_2=0$, $c_3=1$
  - LFSR feedback taps: $s = 0, 1, 3$

    (non-zero coefficients)

- $P^*(x) = 1 \oplus x^2 \oplus x^3$

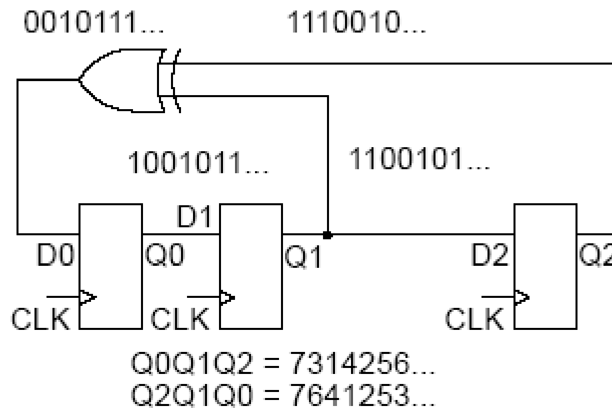
“Type 1” LFSR schematic



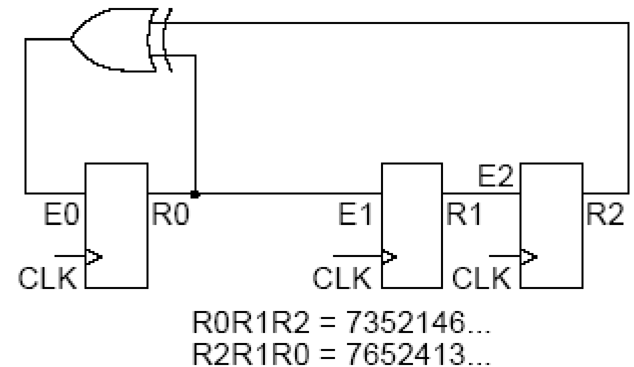
If ck=1 add feedback connection & xor gate in position k

Four LFSR structures for every primitive polynomial

Type 1
- -external XOR
- -easy to build from existing registers
- -Q outputs delayed by 1 clock (test seq's are correlated)

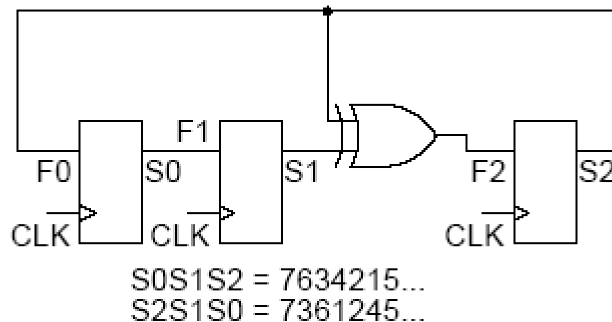


Type 1, P*(x)

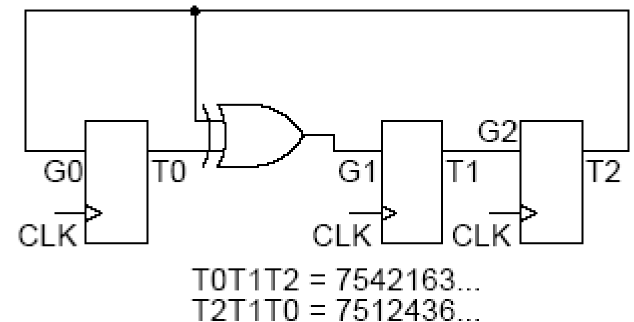


Type 1, P(x)

Type 2
- -internal XOR
- -fewer series XORs (faster)
- -outputs not correlated
- -usually used for BIST



Type 2, P(x)



Type 2, P*(x)

$$P(x) = 1 \oplus x \oplus x^3 \qquad P^*(x) = 1 \oplus x^2 \oplus x^3$$

Common LFSR Configurations

Table 2-1. Common LFSR Configuration

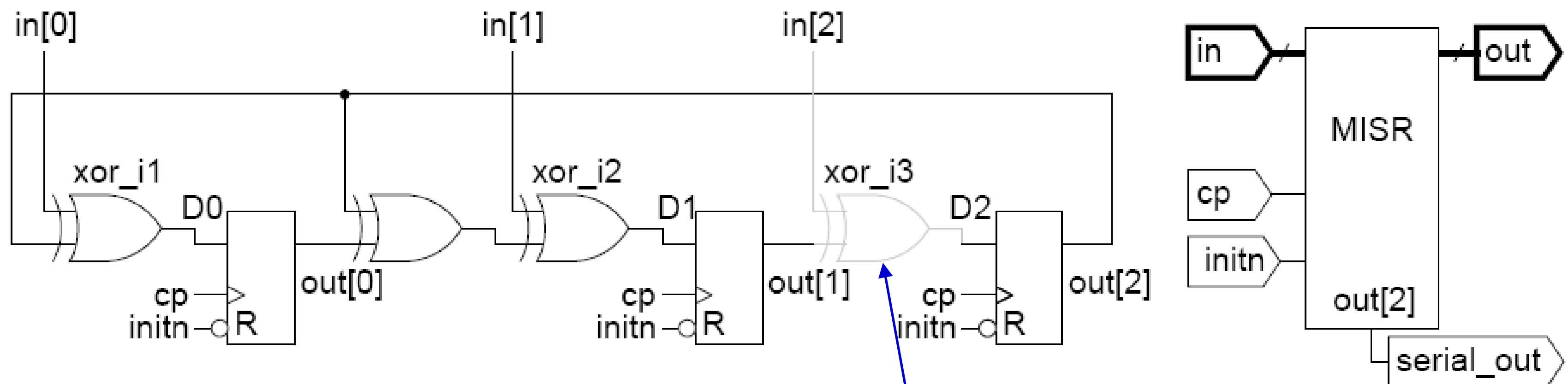
LFSR Length	Primitive Polynomial	Tap Points ("in"/Type2)	Tap Points ("out"/Type1)
8-bits	$x^8+x^4+x^3+x^2+1$	6, 5, 4	4, 3, 2
16-bits	$x^{16}+x^5+x^4+x^3+1$	13, 12, 11	5, 4, 3
24-bits	$x^{24}+x^7+x^2+x+1$	23, 22, 17	7, 2, 1
32-bits	$x^{32}+x^{22}+x^2+x+1$	31, 30, 10	22, 2, 1

Source: Mentor Graphics "LBISTArchitect Process Guide"

Also see Figure 14.27 and Table 14.11 in the Smith Text

Multiple-Input Signature Register (MISR)

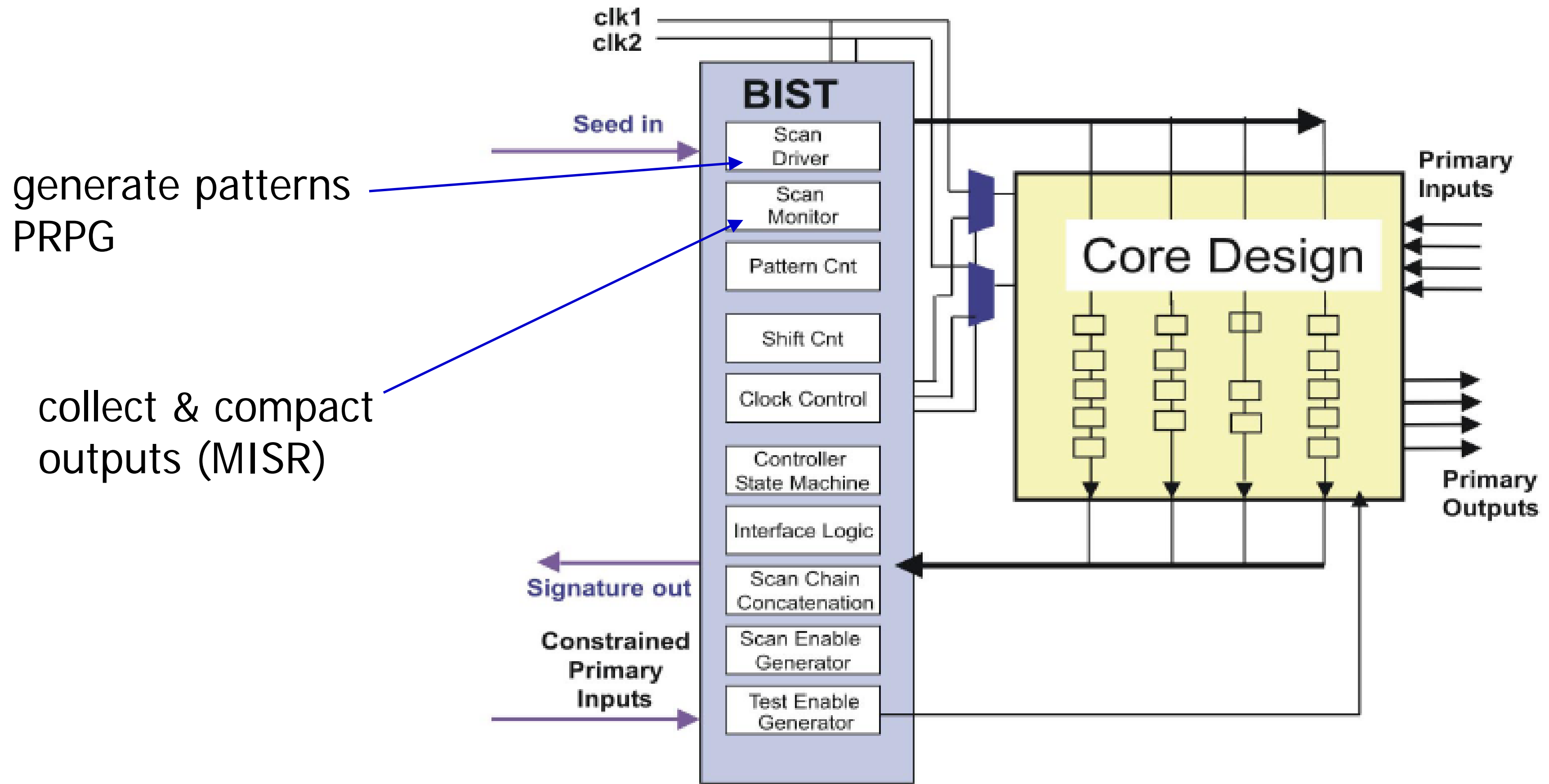
- Reduce test logic by using multiple bit streams to create a signature
- BILBO (built-in logic block observer) – uses MISR as ***both*** PRBS generator and signature register



Mentor Graphics Tools

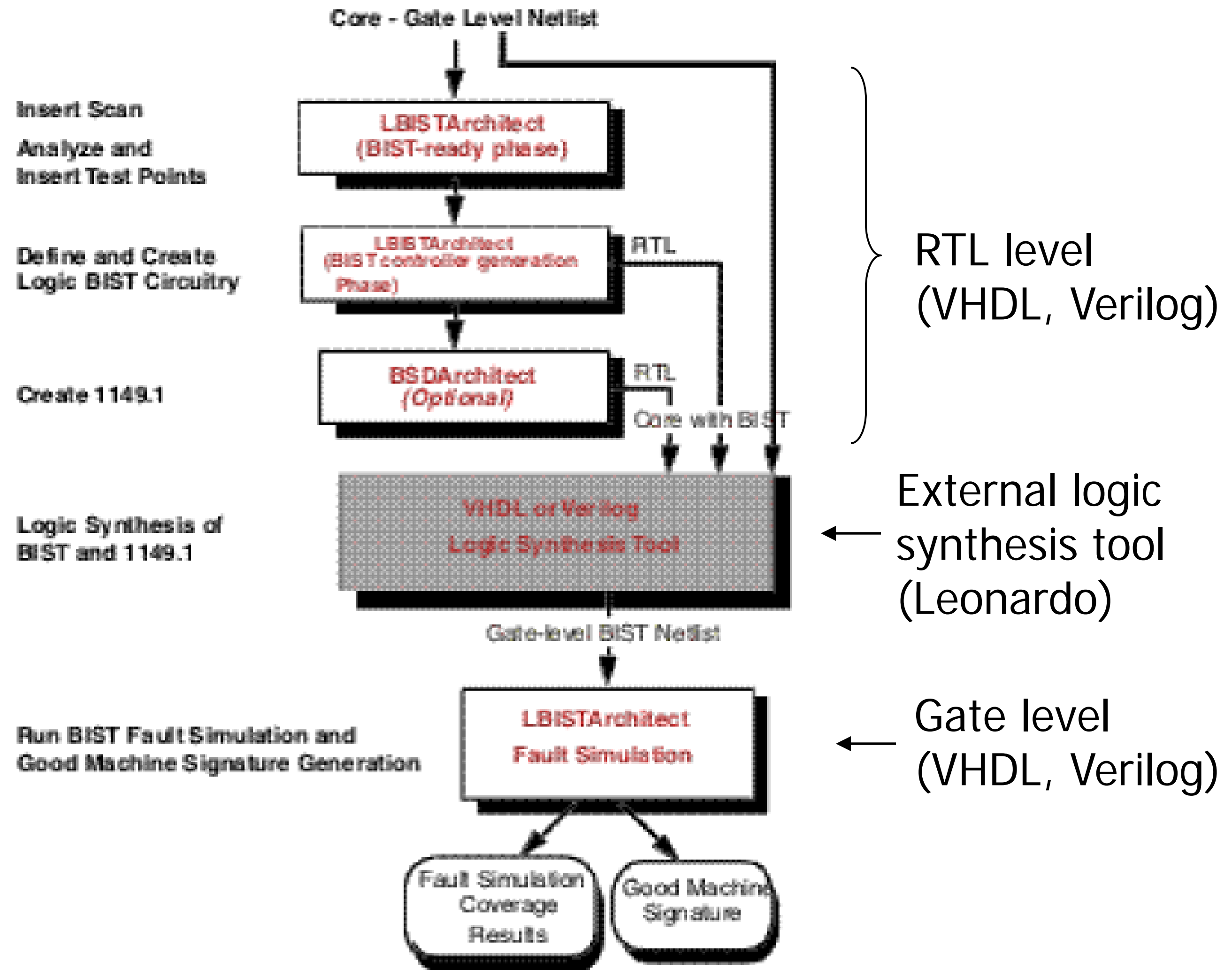
- LBISTArchitect
  - logic BIST design & insertion
  - Reference: "LBISTArchitect Process Guide"
- MBISTArchitect
  - memory BIST design & insertion

Architecture produced by LBISTarchitect

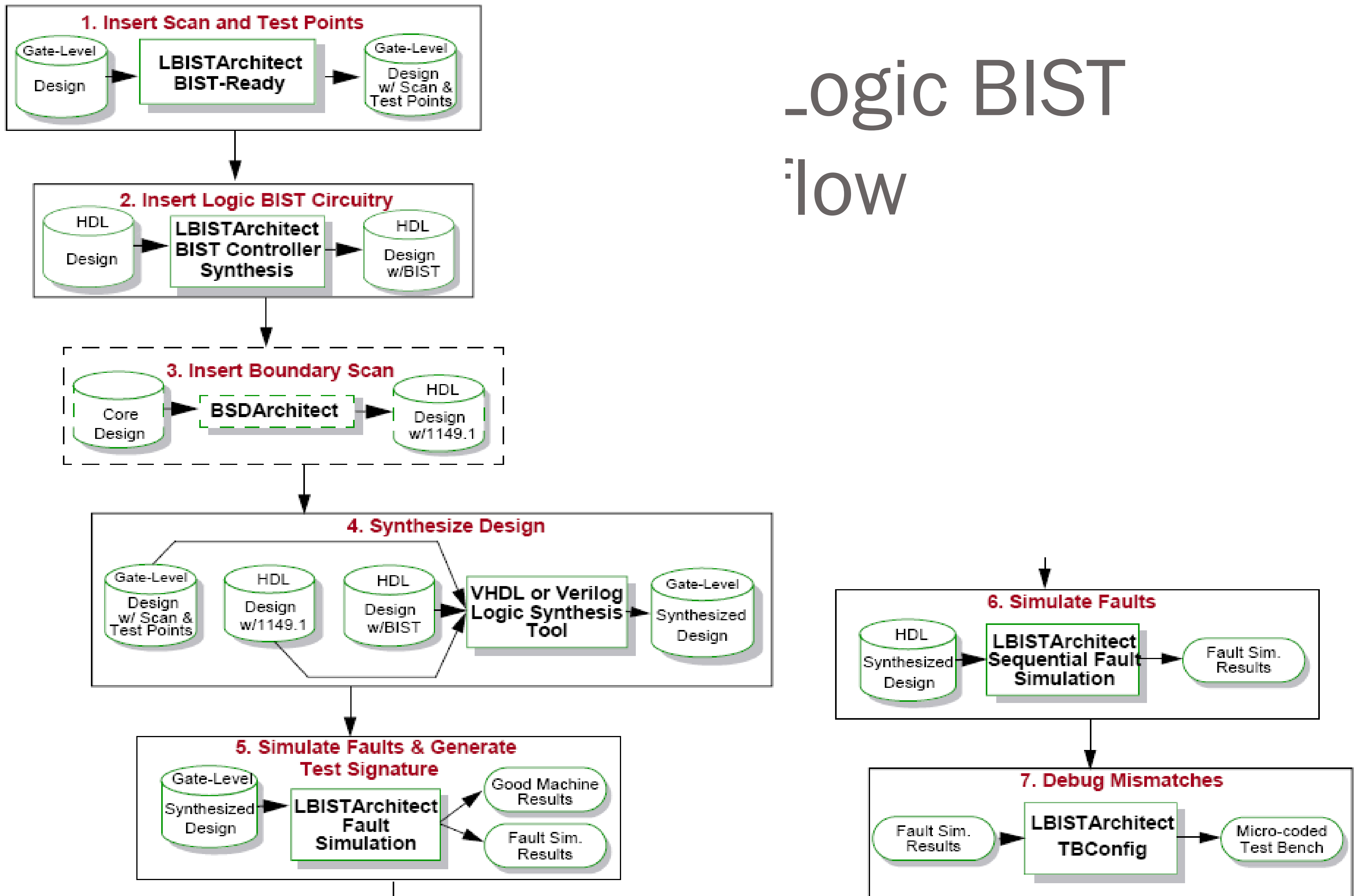


Source: Mentor Graphics “LBISTArchitect Process Guide”

Logic BIST design flow

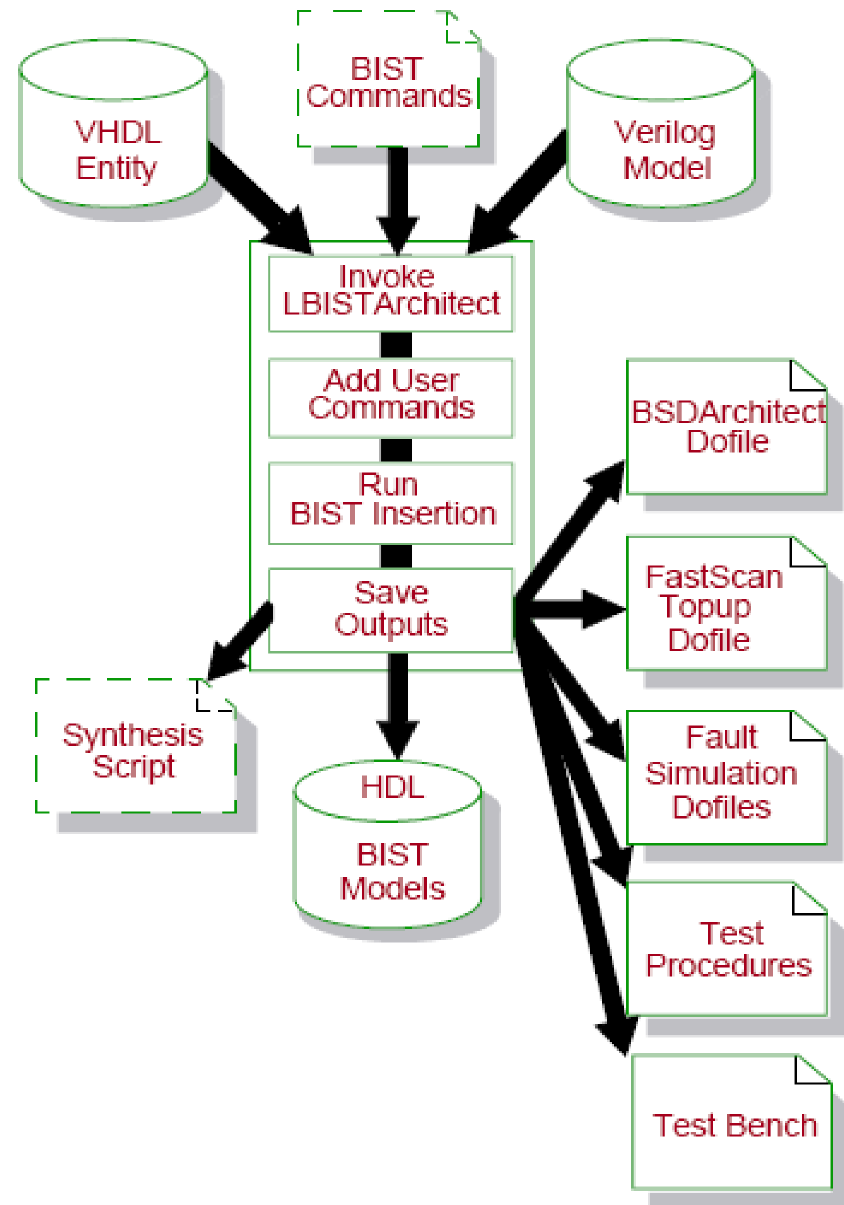


Source: Mentor Graphics “LBISTArchitect Process Guide”



Source: Mentor Graphics “LBISTArchitect Process Guide”

Logic BIST insertion flow



Logic BIST design phases

- BIST-Ready:
  - check design for testability
  - insert scan circuits & test points
- BIST Controller Generation:
  - produce synthesizable RTL model (VHDL, Verilog)
  - includes scan driver/PRPG, scan monitor/MISR
- Boundary Scan Insertion (optional)
  - BSDarchitect can tie 1149.1 to logic BIST
  - inserts boundary scan ckts & TAP controller

LOGIC BIST design phases (2)

- Fault simulation & signature generation
  - determine fault coverage of BIST patterns
  - generate signature of "good circuit"
- Sequential fault simulation (optional)
  - determine fault coverage of BIST hardware
- BIST verification (optional)
  - generate test bench for full simulation
- Manufacturing diagnostics (optional)
  - generate info to assist in fault diagnosis

BIST-ready phase: test point insertion

- Add control test points to gain access to inputs of difficult-to-test gates
- Add observe test points to gain access to outputs of difficult-to-test gates
- MTPI: Multiphase Test Point Insertion
  - break test into phases (ex. 256 patterns each)
  - activate only test points used in a phase
  - add points to improve detection of faults not detected in previous test phases

MTPI Example

Table 2-2. Four-Phase Test Point Control Activity

Phase	Patterns	Activity
0	1—256	No active TP controls
1	257—512	1st TP control active
2	513—768	2nd TP control active
3	769—1024	3rd TP control active