VLSI Test Technology and Reliability (ET4076)

Lecture 9 (2)

Built-In-Self Test

(Chapter 15)

Said Hamdioui

Computer Engineering Lab
Delft University of Technology
2009-2010

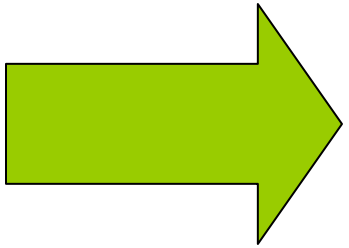
Learning aims

- Describe the concept and the architecture of BIST
- Compare BIST different implementations
- List the pros and cons of BIST
- Develop a BIST at higher level

Contents

- Motivation
- General BIST Architecture
- BIST Hierarchy
- BIST Building blocks
- Aliasing
- BIST Cost
- Logic BIST implementation
  - Test-Per-Clock BIST
  - Scan Based Test
- Memory BIST
- Summary

BIST Motivation.....(1)



BIST Motivation.....(2)

- To solve variety of **test problems**
  - Solve Test-Access problem (Lack of direct pin access)
    - Increasing chip logic-to-pin ratio – harder observability
  - Ability to carry out "At Speed Test"
    - Increasing difficulty in performing at speed test using ATE
  - Useful for field test and diagnosis
    - Less expensive than a local ATE
  - Improve diagnosis (system, board, component)
    - Increasingly dense devices and fast clocks
  - Reduce test cost
    - Increasing test generation and application times
  - Shortage of test/DFT engineers
  - Hard testability insertion
    - Designers unfamiliar with gate-level logic, since they design at behavioral level

BIST Motivation……(3)

- Other advantages
  - Reduce chip pin count = reduced # ATE driver/sensor channels
  - Reduce program runtime (depends on # of tests and clock frequency)
  - Lower system test effort
  - Improved system maintenance and repair
  - Improved component repair
  - Lower test development cost
    - BIST can be automatically added with CAD tools
  - Can test many units in parallel

BIST Motivation……(4)

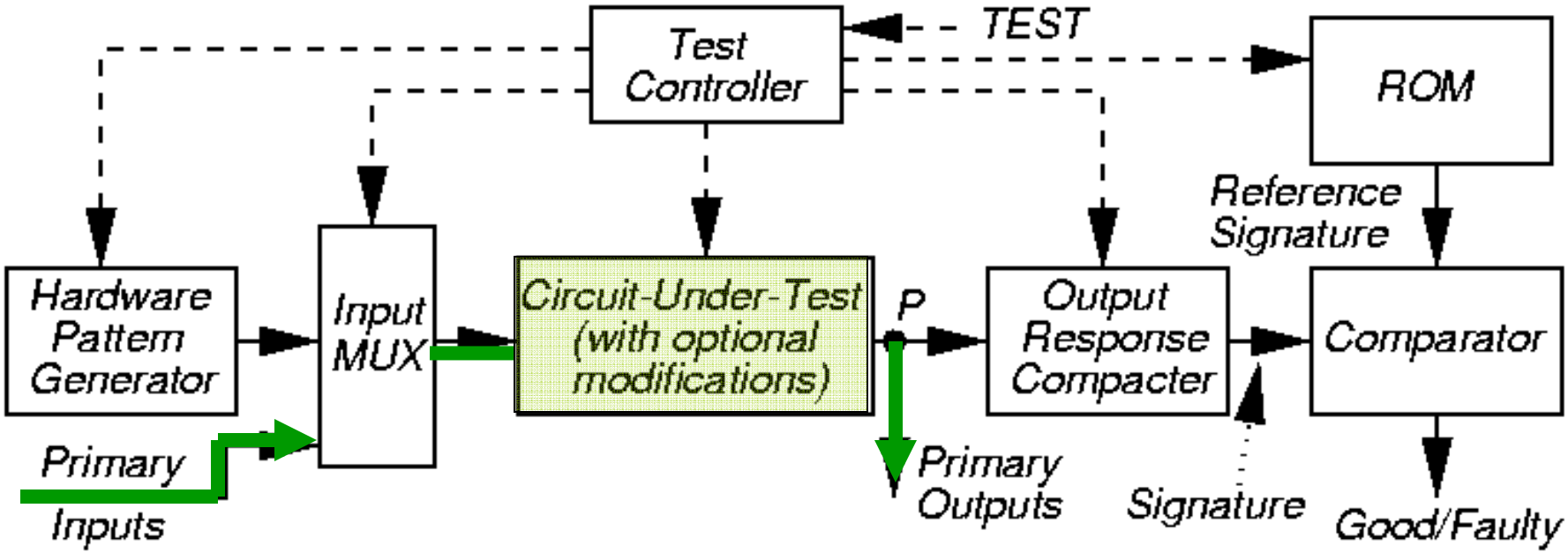
- **In the past**
  - BIST was seen as **"one off"** investment
  - Once it has been used and device passed the tests, there was no further use of BIST

  => Hard to economically justify BIST
- **Today**
  - Boundary scan changed the view
  - Access of BIST through boundary scan
  - BIST can be re-run at all stages of the product cycle
    - Especially system level for diagnostic purposes

  => Various form of BIST are being requested
    - By designers in system companies
    - By EDA companies to respond with appropriate test-synthesis tools

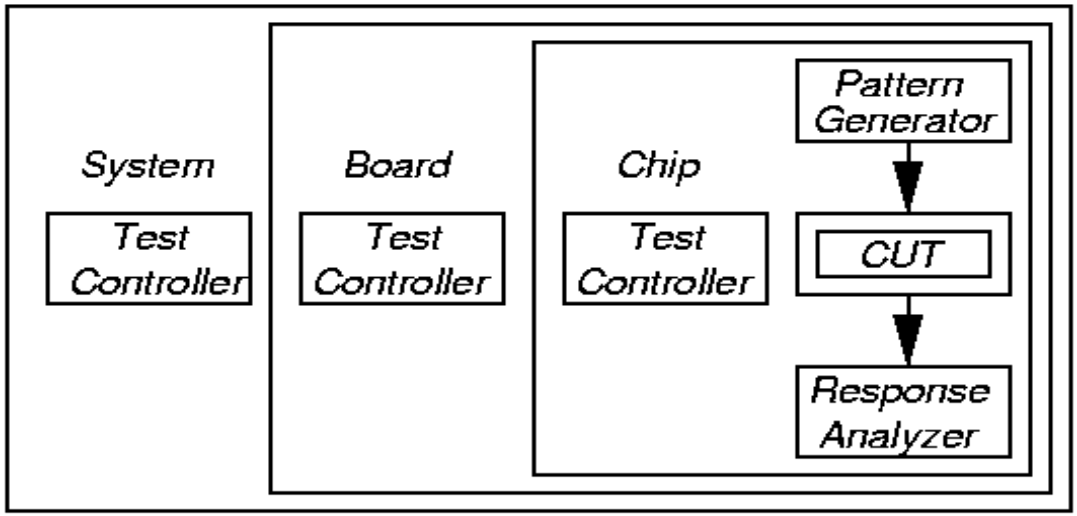
General BIST Architecture

- BIST cannot test wires (and transistors):
  - From PI pins to Input MUX
  - From POs to output pins
  - Can be tested using ATE or JTAG
- In normal mode: PIs → input MUX →Circuit →POs



BIST Hierarchy

- BIST Hierarchy at all three levels of packaging
  - System has multiple PCBs, each with multiple chips
- System Test controller can activate self-test simultaneously on all boards
- Board Test Controller can activate self-test on all chips
  - Chip Test Controller executes self-test and transmits results to Board Test Controller
  - Board Test Controller accumulates results from all chips and transmits them to System Test Controller
  - System Test Controller use the results to e.g., isolate the faulty chips and boards
- BIST Diagnosis effective only if very high fault coverage considered



BIST Building blocks….Pattern generator

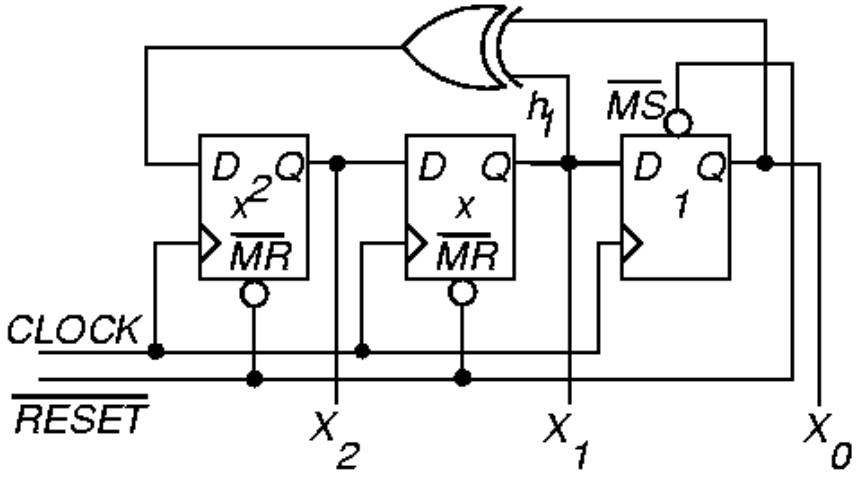
- **Many approaches**
  - ROM
    - Very expensive in chip area
  - LFSR: Linear Feedback Shift Register
    - Generate Pseudo-random tests (1M or more)
    - Use very little hardware
    - Currently the preferred BIST pattern generation method
  - Binary counters
    - Generate exhaustive test sequence
    - Too much test time if # of PIs is huge
    - Use more hardware than LFSR
  - LFRS combined with few patterns in ROM
    - To augment the fault coverage
  - Modified counters
  - …

BIST Building blocks….Pattern generator

- LFSR: Linear Feedback Shift Register
  - A shift register with feedback from the last stage and others
  - It has no other input beside the clock
  - Outputs of FFs form the test pattern
  - Number of unique test patterns is equal to the # of the states of the circuit
    - Determined by the # and locations of feedbacks
  - Two types:
    - **Standard LFSR** (External XOR LFSR)
    - **Modular LFSR** (Internal XOR LFSR)
  - Example: Standard LFSR
  - Characteristic polynomial
    $f(x) = 1 + x + x^3$

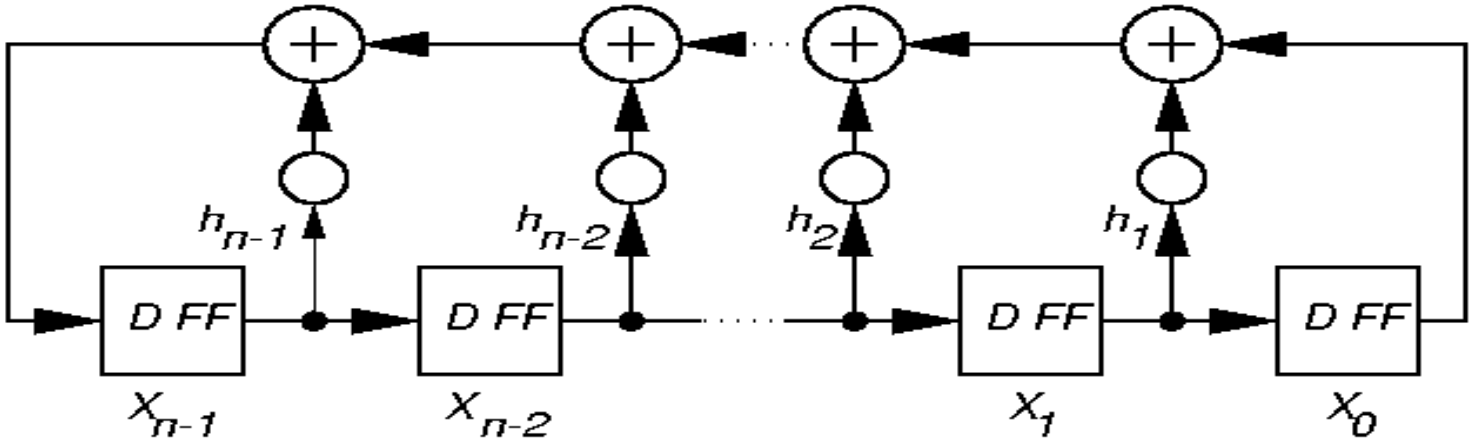
Initialized to 001 (7 patterns)

X0	1	0	0	1	0	1	1	1	0	..
X1	0	0	1	0	1	1	1	0	0	..
X2	0	1	0	1	1	1	0	0	1	..



BIST Building blocks….Pattern generator

- Generic Standard 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

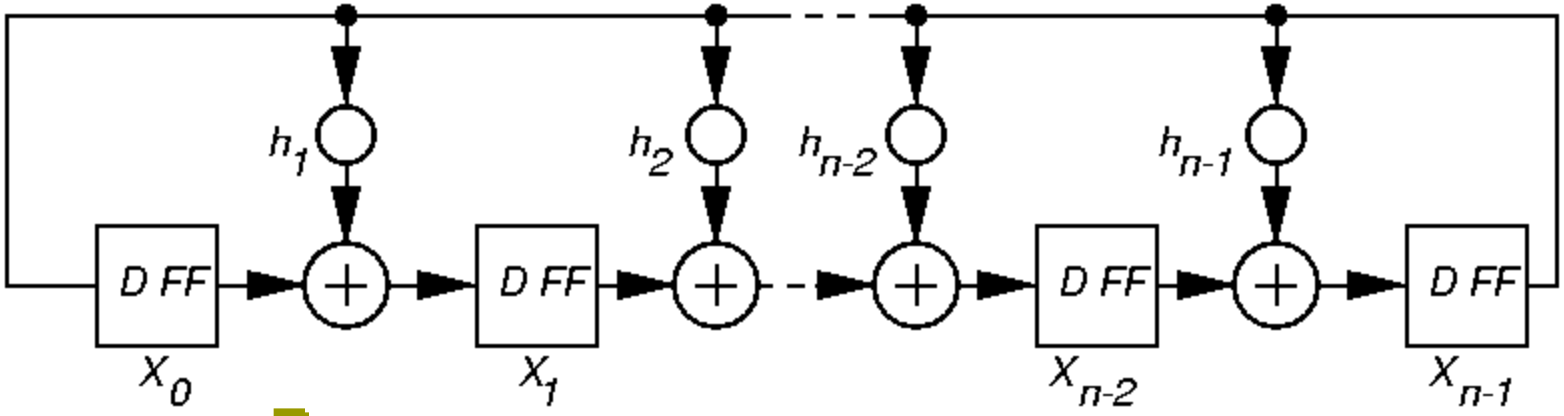


$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ \vdots \\ X_{n-3}(t+1) \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & h_1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

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

BIST Building blocks….Pattern generator

- Generic Modular LFSR
  - Described by *companion matrix* $T_m = T_s^T$; $X(t+1) = T_m \times X(t)$
  - Equivalent to standard External XOR LFSR
    - With a different state assignment; Faster – usually does not matter
    - Same amount of hardware



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

BIST Building blocks….Pattern generator

LFSR Theory (see book)

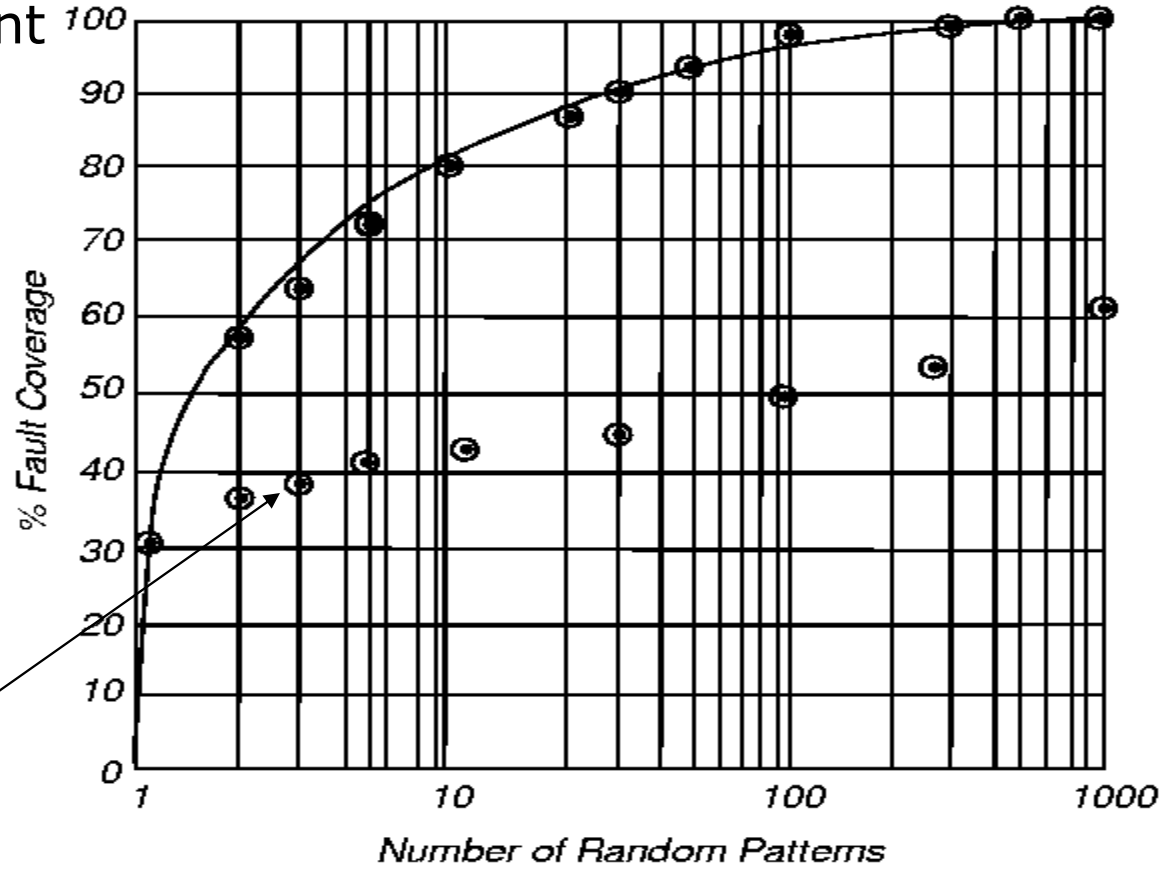
- Cannot initialize to all 0's (it hangs)
- If X is initial state, progresses through states X, $T_s X$, $T_s^2 X$, $T_s^3 X$, …
- *Matrix period*:
  - Smallest k such that $T_s^k = I$
  - k is LFSR cycle length
- Described by **characteristic polynomial**:

$$f(x) = |T_s - I\,X| \; (=|T_m - I\,X| \text{ if modular LFSR})$$
$$= 1 + h_1 x + h_2 x^2 + \ldots + h_{n-1} x^{n-1} + x^n$$

- Maximal-length LFSR if period $K=2^n-1$

BIST Building blocks….Pattern generator

FC versus pattern count

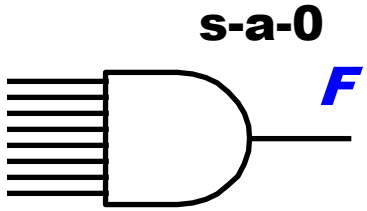


(a) Top curve -- random pattern testing with acceptable fault coverage.
(b) Bottom curve -- unacceptable random pattern testing.

- Bottom: Random-Pattern Resistant circuit (like PLAs)
  - Requires **weighted pseudo-random pattern generation**
  - Or insertion of testability hardware

BIST Building blocks….Pattern generator

- Weighted Pseudo-Random Pattern Generation



- If p (1) at all PIs is 0.5, p_F (1) = 0.5^8 = 1/256

  $p_F(0)=1-(1/256)=255/256$

- 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

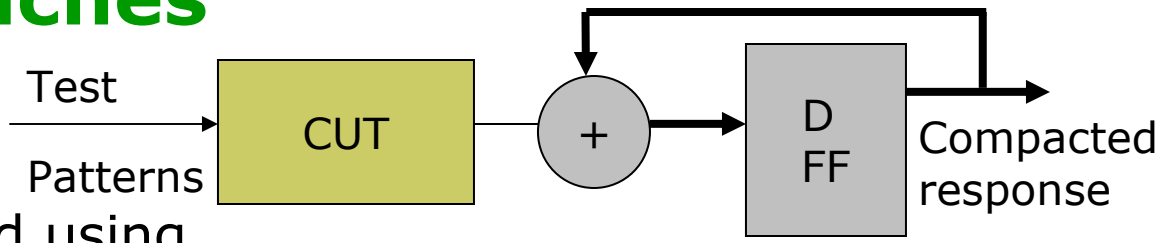
BIST Building blocks….Response Compaction

- 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

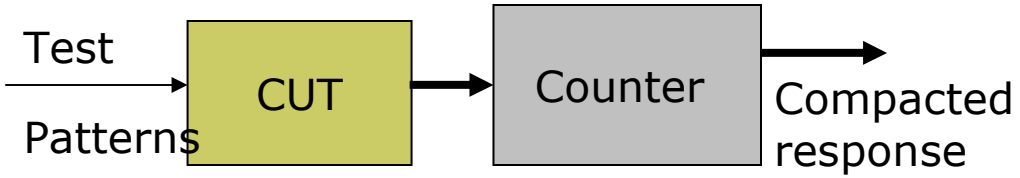
BIST Building blocks….Response Compaction

Different approaches

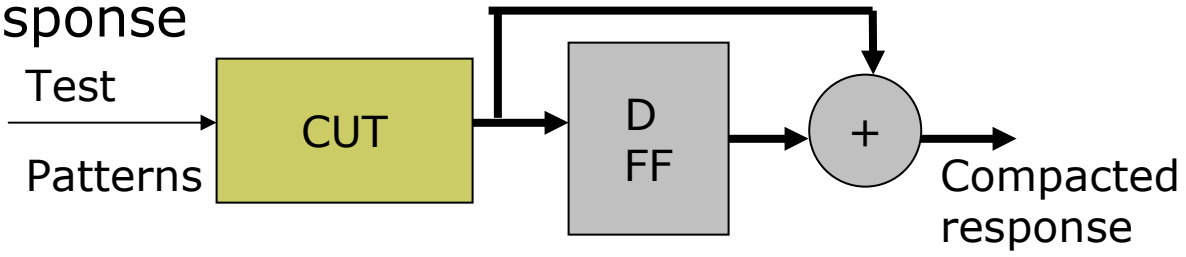
- **Parity Testing**
  - Parity bit calculated using modulo 2 summation
  - i.e., (Σ r_i)modulo 2



- **One Counting**
  - Number of 1s in the response stream is counted



- **Transition Counting**
  - Number of transitions 0→1 and 1→0 in the response counted
  - i.e., Σ (r_i XOR r_{i+1})



BIST Building blocks….Response Compaction

Signature analysis using LFSR

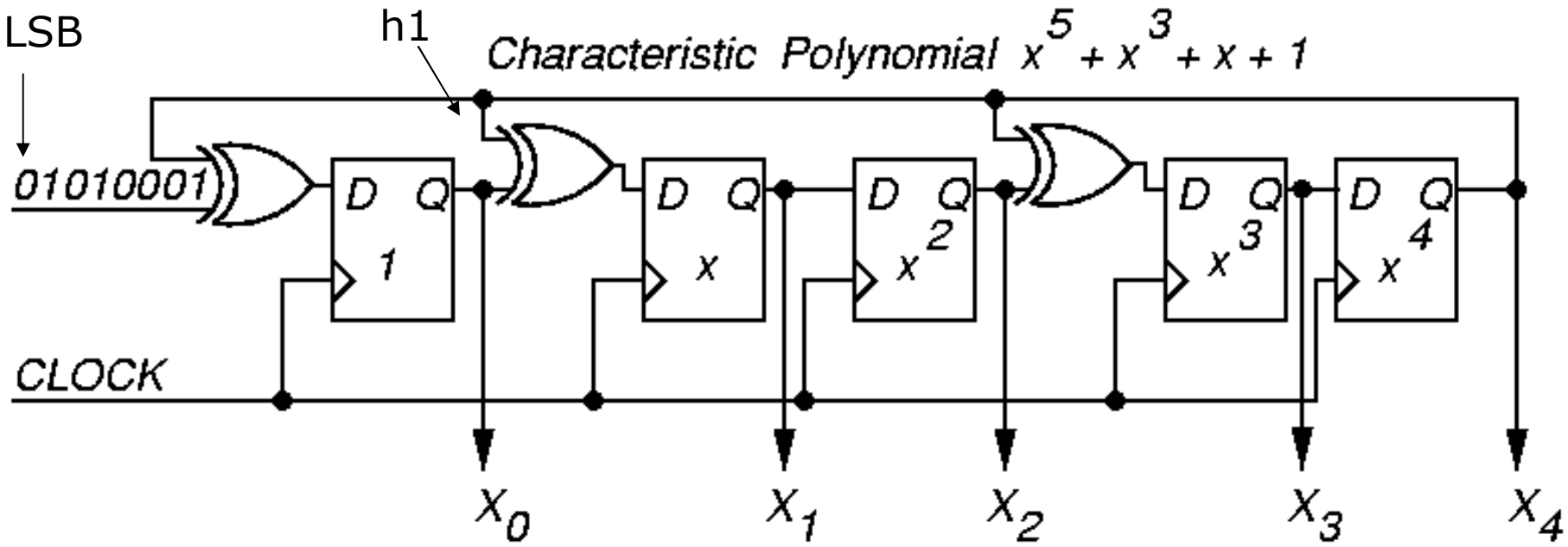
- Single-Input Signature Register/ Analyzer (SISR)
- Also known as **Cyclic Redundant Check Code CRCC**
  - Treat data bits from circuit POs to be compacted as a decreasing order coefficient polynomial
  - SISR divides the **PO polynomial** by its **characteristic polynomial**
    - Leaves remainder 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

BIST Building blocks….Response Compaction

Single-Input Signature Register (SISR)

Example:

- LFSR seed value is **0000**
- PO polynomial: 10001010 →x^7+x^3+x
- Characteristic polynomial: x^5+x^3+x+1
- Remainder of division: x^3+x^2+1 (=**01101**)



BIST Building blocks….Response Compaction

- **Problem Single-Input Signature Register (SISR)**
  - Too much hardware if one of these is put on each *primary output* (PO)
  - E.g., if 200 outputs, 200*5 FFs, and 3*200 XOR gates will be required for the previous example

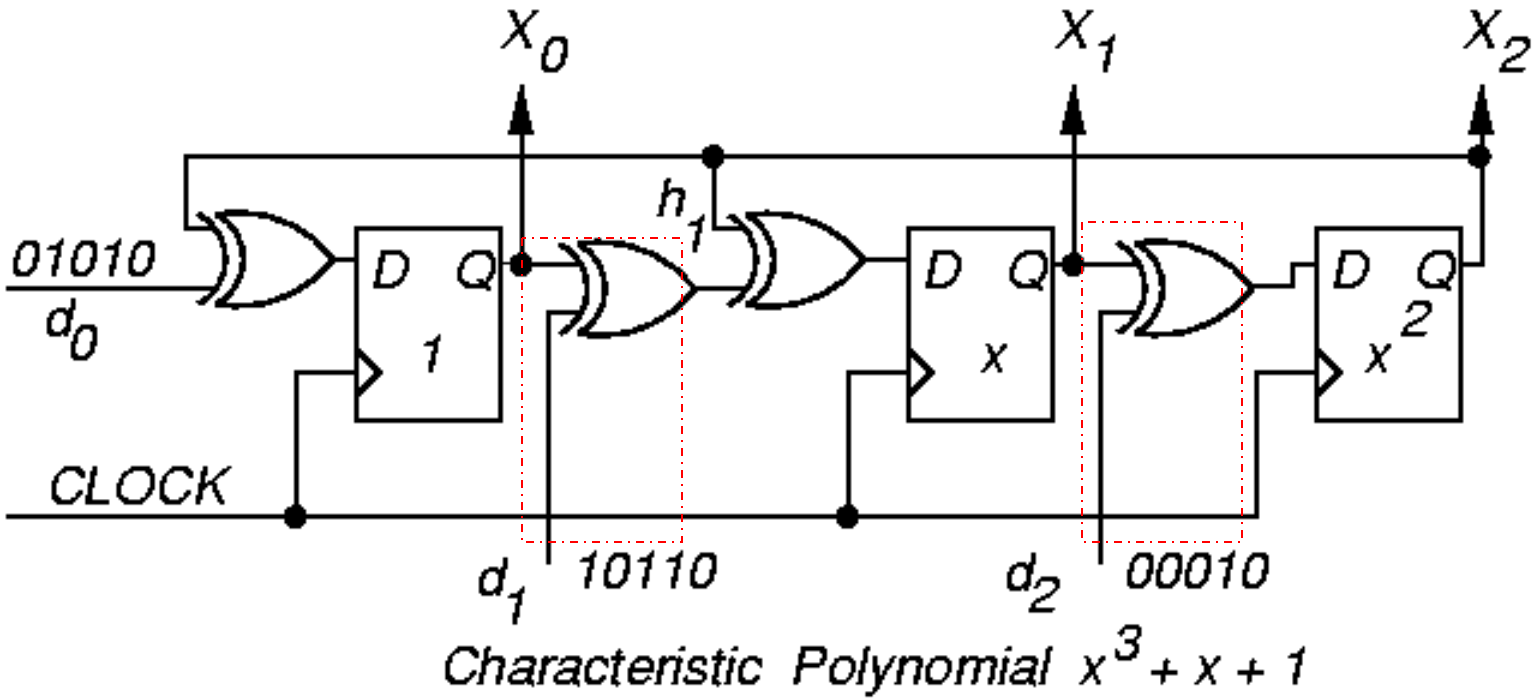
- Solution:
  - **MISR**: **Multiple-Input Signature Register**
  - compacts all outputs into one LFSR
    - Works because LFSR is linear – obeys *superposition principle*
    - Superimpose all responses in one LFSR
    - Final remainder is XOR sum of remainders of polynomial divisions of each PO by the characteristic polynomial

BIST Building blocks….Response Compaction

Multiple-Input Signature Register

Example:

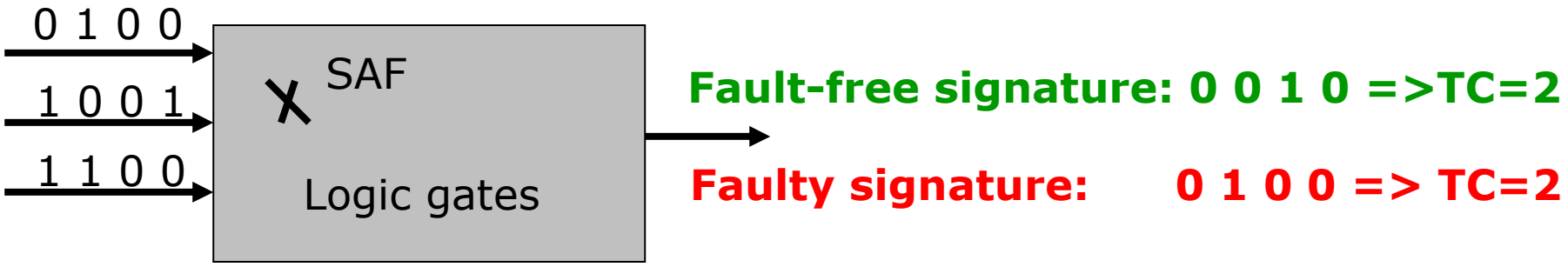
- PO polynomials: PO1=d0=01010 →x^3+x
  - PO2=d1=01101 → x^3+x^2+1 PO3=d2=01000 → x^3
- Characteristic polynomial: x^3+x+1
- Resulting signature is the XOR of the three different signatures due to the polynomial division of each of the three POs



Aliasing

- When using compaction, the resulted compacted signature may be identical to the fault-free signature => **aliasing**
  - **It lowers the fault coverage**

- Example
  - Transition count (TC)

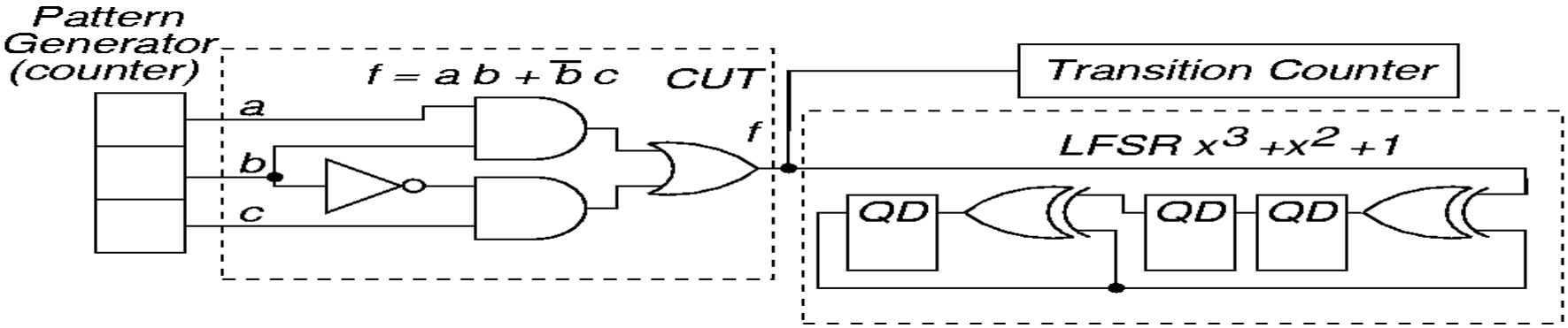


Fault will be **not detected** as TC=2 in both cases

Aliasing

- Parity checking
  - Aliasing frequently occurs
- One counting
  - Permutes placement of 0's and 1's
- Count transition
  - Alias less than one counting
  - It checks for correct number of 0's and 1's AND also *partially* the **correct ordering** of 0's and 1's
- MISR
  - Aliasing depends on k: polynomial degree (e.g., k=3 for x^3+x+1 **)**
  - The larger k, the smaller the aliasing
  - Aliasing probability is about $\mathbf{1/2^k}$
  - MISR has more aliasing than LFSR on single PO

Aliasing …….Experiment Hardware



- 3 bit exhaustive binary counter for pattern generator

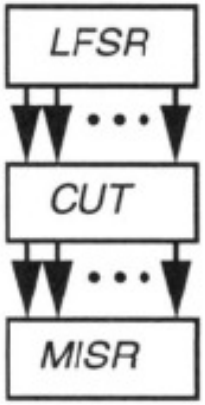
Pattern *abc*	Responses			
	Fault-free	a sa1	f sa1	b sa1
000	0	0	1	0
001	1	1	1	0
010	0	1	1	0
011	0	1	1	0
100	0	0	1	1
101	1	1	1	1
110	1	1	1	1
111	1	1	1	1
Signatures				
Transition count	**3**	**3**	**0**	**1**
LFRS	**001**	**101**	**001**	**010**

BIST Costs

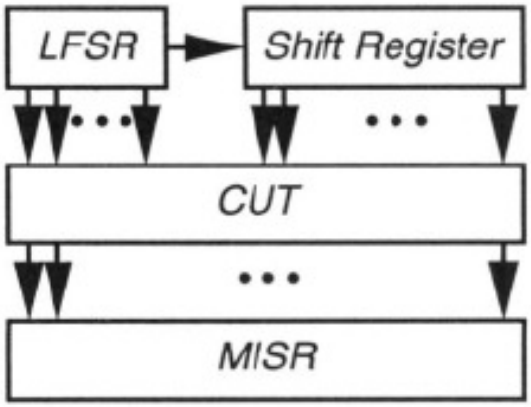
- Chip area overhead and pin overhead:
  - Test controller
  - Hardware pattern generator
  - Hardware response compacter
  - Testing of BIST hardware
  - At least 1 pin needed to activate BIST operation
- Performance overhead
  - Extra path delays due to BIST
- *Yield loss*
  - Due to increased chip 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

Logic BIST implementation…..Classification

- **Test-Per-Clock BIST**
  - Also called Concurrent Built-In Logic Block Observer **CBILBO**
  - FFs at the inputs of the kernel/circuit are configured as parallel pattern generation (e.g., LFSR)
  - FFs at the output of the kernel/circuit are configured as signature analyzer (e.g. SISR, MISR)
  - (Faults are tested every clock period)



(a) Test-per-clock system.



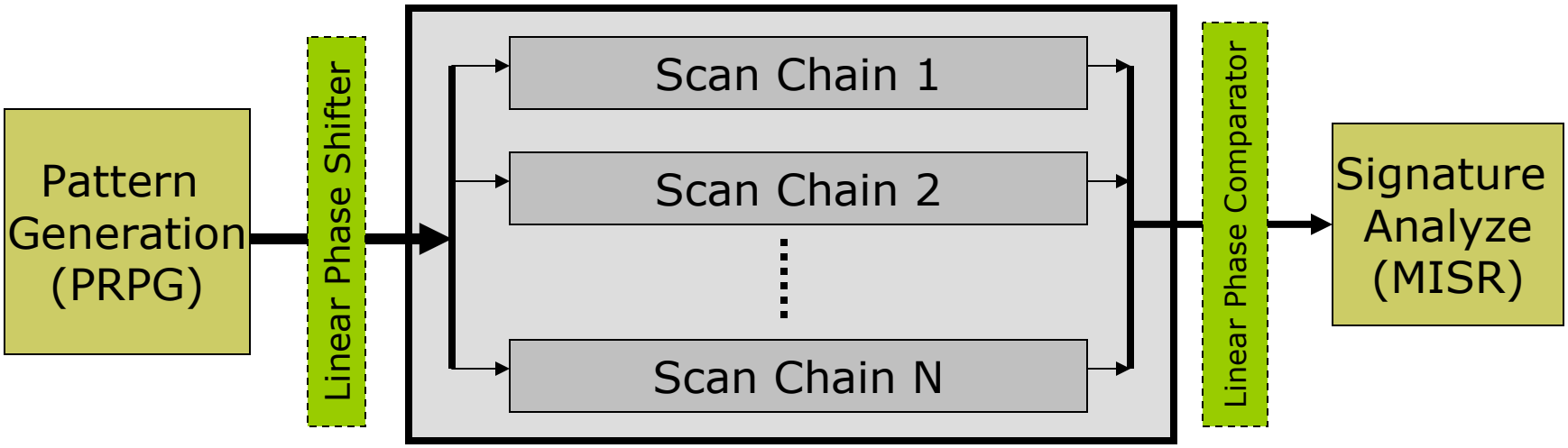
(b) Large input count test-per-clock system.

- **Test-Per-Scan BIST**

Logic BIST implementation…..Classification

Test-Per-Scan BIST

- Also called **STUMPS** (**S**elf-**T**est **U**sing **M**ISR and **P**arallel **S**R**S**G
- The FFs of the circuit are configured as one or more scan chains
- Pattern generator and Signature analyzer are added to the circuit
- Testing of a faults is done in many clock cycles (scan-in test vector, conduct the test, scan-out the response)
- Take significantly more time than Test-Per-Clock BIST
- To reduce the length of PRPG and MISR and improve the randomness of PRPG, optional Linear Phase Shifter and Comparator are used



Logic BIST implementation…..Comparison

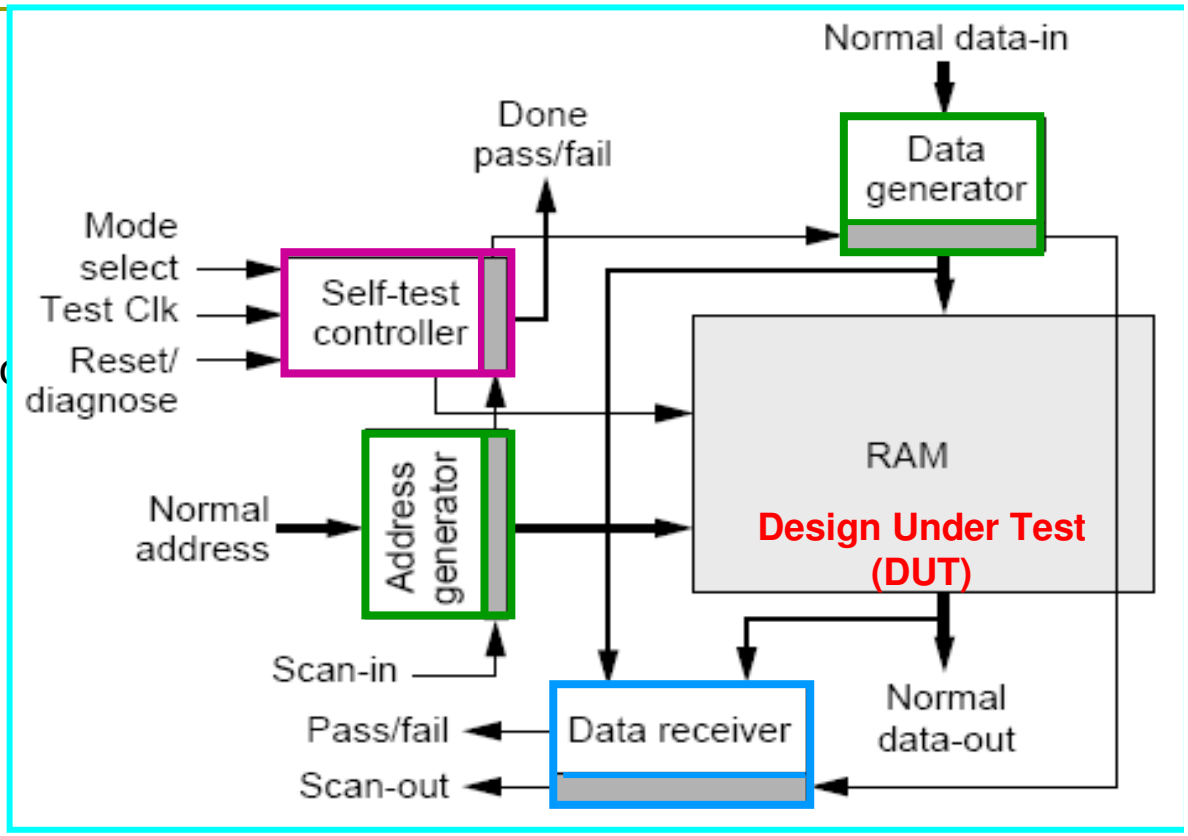
- **Hardware overhead**
  - Test-Per-Clock BIST requires the replacement of the circuit FFs with special configurable registers (normal, scan, pattern generator, signature analyzer)
  - Scan based BIST requires the replacement of circuit FFs with scan FFs; it also requires additional registers for pattern generation and signature analyzer.
  - Area overhead for Scan Based BIST is lower only if the circuit is large.

- **Test application**
  - Test-Per-Clock BIST supports at speed-testing
    - Covers faults due transients in power/ground lines due to switching
  - Scan based BIST requires SCL+1 for a single test
    - SCL: length of the longest scan path
  - Scan based BIST can be implemented at the chip level
    - Even if the modules do not have BIST circuitry (but include scan design)
  - Scan based BIST most used in industry

Memory BIST

General architecture

- Input Test generator
  - Address & Data
- Output analyzer
  - Compare & pass/fail
  - Response Data Evaluat
- Overall controller
  - Communication



Cost

- Some design effort
- Learning curve
- Chip area

Advantages

+ Short test time as compared with scan
+ The only practical & cost-effective solution for large embedded memories (>16KB?)
+ At speed testing
+ Diagnosis capabilities
+ No expensive ATE required

Summary (1)

- BIST architecture
  - Patten generator
  - Response compaction
  - Comparator
  - Test controller
- Preferred BIST methods
  - LFSR pattern generator
  - MISR response compacter
- Two ways to implement BIST
  - Test-per-clock (CBILBO)
  - Test-per-scan (STUMPS)

Summary(2)

- 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
- BIST cost
  - Overhead
    - Test controller, extra circuit delay, Input MUX, pattern generator, response compacter, DFT to initialize circuit & test the test hardware, at least one additional pin
  - Performance overhead and yield loss