

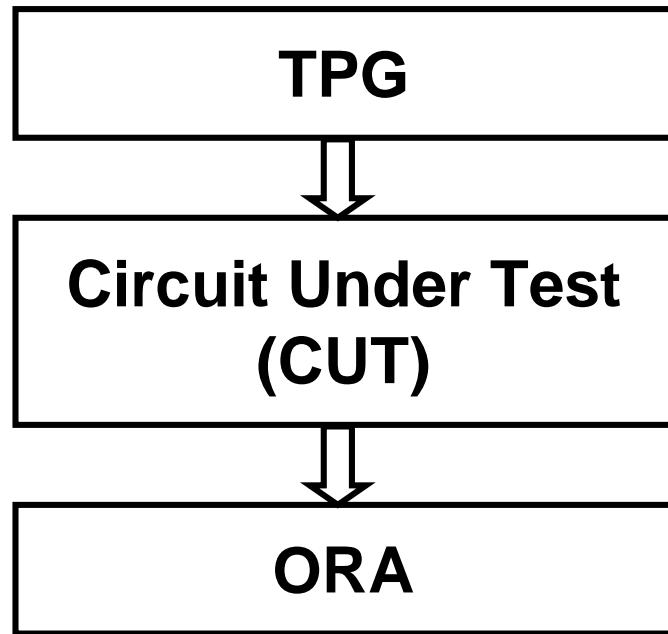
# **Built-in Self-Test (BIST)**

- **Introduction**
- **Test Pattern Generation**
- **Test Response Analysis**
- **BIST Architectures**
- **Scan-Based BIST**

# **Built-in Self-Test (BIST)**

- **Capability of a circuit to test itself**
- **On-line:**
  - **Concurrent : simultaneous with normal operation**
  - **Nonconcurrent : idle during normal operation**
- **Off-line:**
  - **Functional : diagnostic S/W or F/W**
  - **Structural : LFSR-based**
- **We deal primarily with structural off-line testing here.**

# Basic Architecture of BIST



- TPG: Test pattern generator
- ORA Output response analyzer

# Glossary of BIST

- **TPG** - Test pattern generator
- **PRPG** - pseudorandom pattern generator  
(Pseudorandom number generator)
- **SRSG** – Shift register sequence generator (a single output PRPG)
- **ORA** – Output response analyzer
- **SISR** – Single-input signature register
- **MISR** – Multiple-input signature register
- **BILBO** – Built-in logic block observer

# Built-in Self Testing

- **Test pattern generation**
  - **Exhaustive**
  - **Pseudoexhaustive**
  - **Pseudorandom**
- **Test response compression**
  - **One's count**
  - **Transition count**
  - **Parity checking**
  - **Syndrome checking**
  - **Signature analysis**

# Test Pattern Generation for BIST

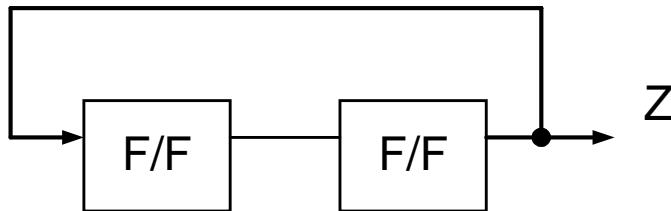
- **Exhaustive testing**
- **Pseudorandom testing**
  - Weighted and Adaptive TG
- **Pseudoexhaustive testing**
  - Syndrome driver counter
  - Constant-weight counter
  - Combined LFSR and shift register
  - Combined LFSR and XOR
  - Cyclic LFSR

# Exhaustive Testing

- **Apply all  $2^n$  input vectors, where  $n = \#$  inputs to CUT**
- **Impractical for large  $n$**
- **Detect all detectable faults that does not cause sequential behavior**
- **In general not applicable to sequential circuits**
- **Can use a counter or a LFSR for pattern generator**

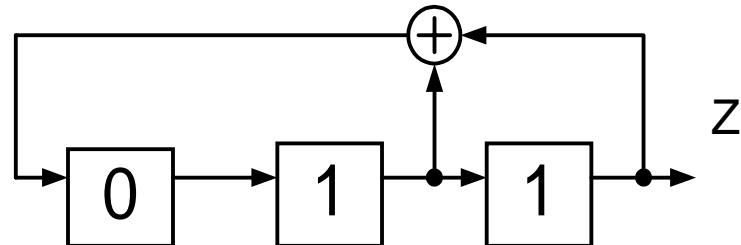
# Linear Feedback Shift Register (LFSR)

- Example



S1	1	0	
S2	0	1	
S1	1	0	
	⋮		

$Z = 0101\dots$   
2 states

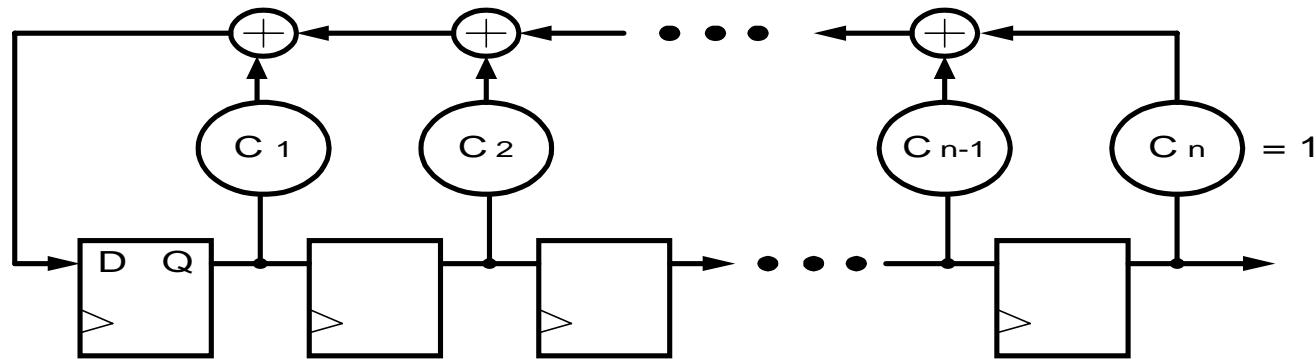


S1	0	0	1
S2	1	0	0
S3	0	1	0
S4	1	0	1
S5	1	1	0
S6	1	1	1
S7 = S0	0	1	1
	⋮		

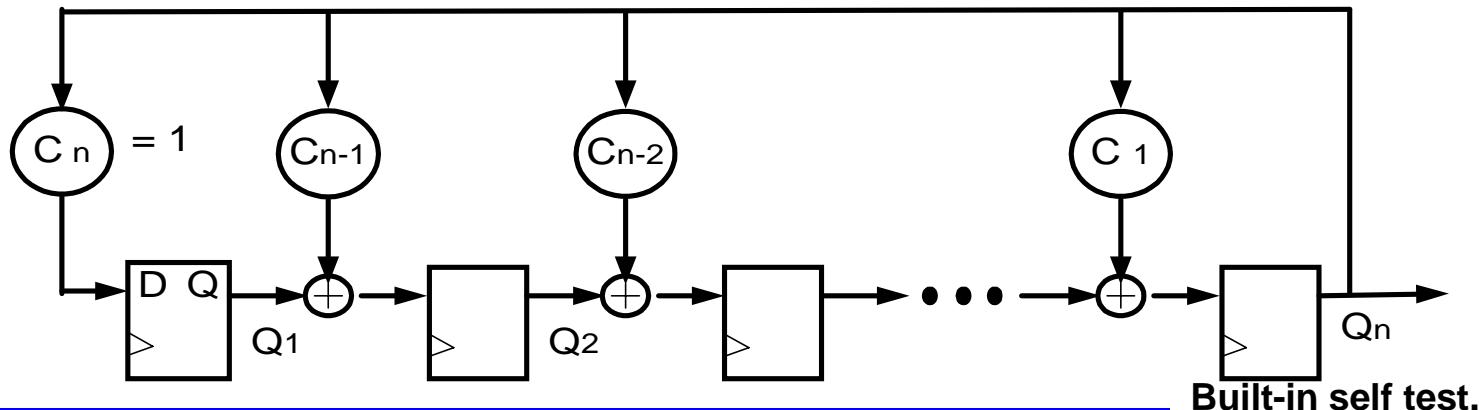
$Z = 1100101\dots$   
7 states

# Two Types of LFSRs

- Type 1: External type



- Type 2: Internal type



# Mathematical Operations over GF(2)

- **Multiplication ( $\bullet$ )**

$\bullet$	0	1
0	0	0
1	0	1

- **Addition ( $\oplus$  or simply  $+$ )**

$\oplus$	0	1
0	0	1
1	1	0

**Example:**

$$C_1 = 0, \quad C_2 = 1, \quad C_3 = 1$$

$$a_{-1} = 0, \quad a_{-2} = 1, \quad a_{-3} = 1,$$

**if  
then**

$$a_0 = a_{-1} \bullet C_1 + a_{-2} \bullet C_2 + a_{-3} \bullet C_3$$

$$a_0 = 0 + 1 + 1 = 0$$

# Analysis of LFSR using Polynomial Representation

- A sequence of binary numbers can be represented using a generation function (polynomial)
- The behavior of an LFSR is determined by its initial “seed” and its feedback coefficients, both can be represented by polynomials.

# Characteristic Polynomials

- $a_0, a_1, \dots, a_m, \dots$  : **sequence of binary numbers.**
- **Generating function :**  $G(x) = a_0 + a_1x + a_1x^2 + \dots + a_mx^m + \dots = \sum_{m=0}^{\infty} a_mx^m$
- Let  $\{a_m\} = a_0, a_1, \dots, a_m, \dots$   
**be output sequence of an LFSR of type1**

$$\Rightarrow a_m = \sum_{i=1}^n C_i * a_{m-i}$$

- Let initial state be  $a_{-1}, a_{-2}, \dots, a_{-n}$

$$\begin{aligned}
 \rightarrow G(x) &= \sum_{m=0}^{\infty} a_m * x^m = \sum_{m=0}^{\infty} \sum_{i=1}^n C_i * a_{m-i} * x^m \\
 &= \sum_{i=1}^n C_i * x^i \sum_{m=0}^{\infty} C_i * a_{m-i} * x^{m-i} \\
 &= \sum_{i=1}^n C_i * x^i [a_{-i} * x^{-i} + \dots + a_{-1} * x^{-1} + \sum_{m=0}^{\infty} a_m * x^m] \\
 &\rightarrow G(x) = \frac{\sum_{i=1}^n C_i * x^i (a_{-i} * x^{-i} + \dots + a_{-1} * x^{-1})}{1 + \sum_{i=1}^n C_i * x^i}
 \end{aligned}$$

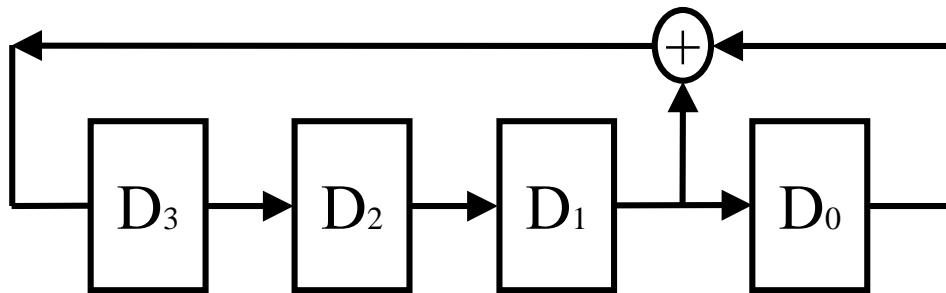
$\therefore G(x)$  depends on initial state and feedback coefficients

# Denominator

$$P(x) = 1 + c_1 * x + c_2 * x^2 + \dots + c_n * x^n$$

is called the characteristic polynomial of the LFSR

Example:



$$P(x) = 1 + x^3 + x^4$$

# LFSR Theory

- **Definition:** If period  $p$  of sequence generated by an LFSR is  $2^n - 1$ , then it is a maximum length sequence
- **Definition:** The characteristic polynomial associated with a maximum length sequence is a primitive polynomial
- **Theorem:** # of primitive polynomials for an  $n$ -stage LFSR is given by

$$\lambda_2(n) = \phi(2^n - 1) / n$$

where

$$\phi(n) = n * \prod_{P \mid n} \left(1 - \frac{1}{P}\right)$$

# Primitive Polynomial

- # primitive polynomials of degree n

- Some primitive polynomials

N	$\lambda_2(n)$
1	1
2	1
4	2
8	16
16	2048
32	67108864

---

1:	0			
2:	1	0		
3:	1	0		
4:	1	0		
5:	2	0		
6:	1	0		
7:	1	0		
8:	6	5	1	0
9:	4	0		
10:	3	0		
11:	2	0		
12:	7	4	3	0

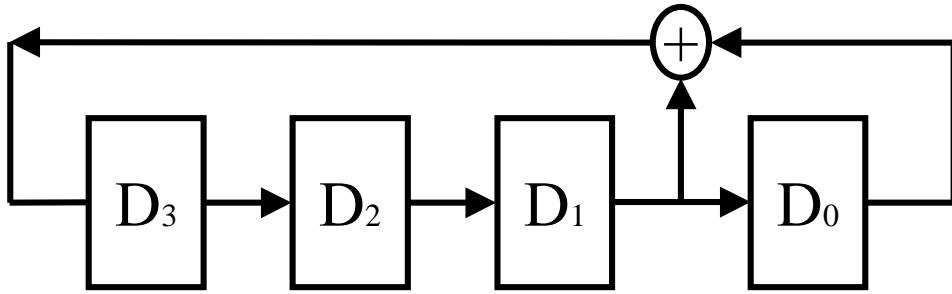
13:	4	3	1	0
14:	12	11	1	0
15:	1	0		
16:	5	3	2	0
17:	3	0		
18:	7	0		
19:	6	5	1	0
20:	3	0		
21:	2	0		
22:	1	0		
23:	5	0		
24:	4	3	1	0

25:	3	0		
26:	8	7	1	0
27:	8	7	1	0
28:	3	0		
29:	2	0		
30:	16	15	1	0
31:	3	0		
32:	28	27	1	0
33:	13	0		
34:	15	14	1	0
35:	2	0		
36:	11	0		

# Primitive Polynomial (Cont.)

- **Characteristic of maximum-length sequence:**
  - Pseudorandom though deterministic and periodic
  - $\# 1's = \# 0's + 1$
- ⇒ **Can be used as a (pseudo)-random or exhaustive number generator.**

# LFSR Example



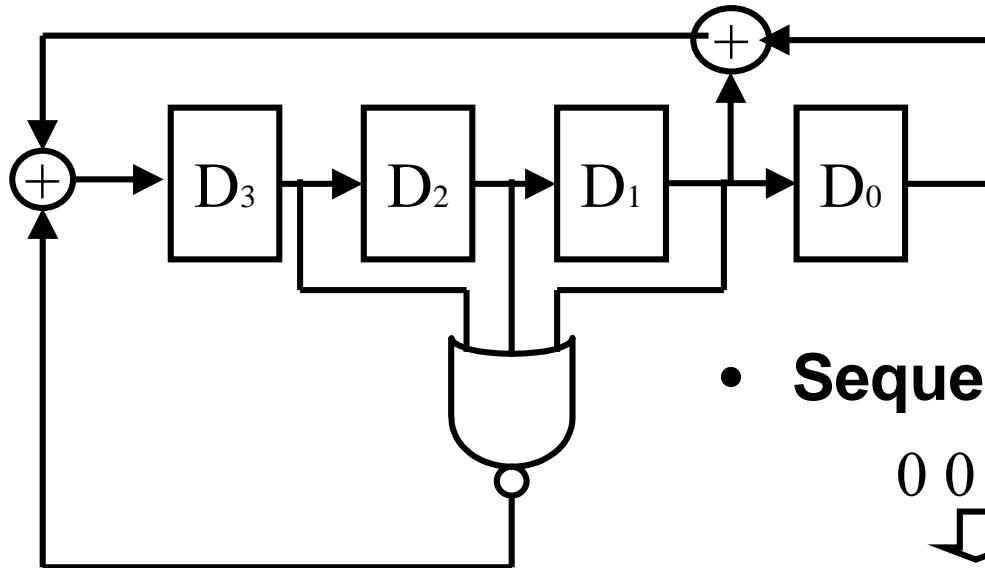
$D_0$	$D_1$	$D_2$	$D_3$
1	0	0	0
0	0	0	1
0	0	1	1
0	1	1	1
1	1	1	1
1	1	1	0
1	1	0	1
1	0	1	0
0	1	0	1
1	0	1	1
0	1	1	0
1	1	0	0
1	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0
			(repeat)

$$P(x) = 1 + x^3 + x^4$$

- $2^4 - 1 = 15$  “near” complete patterns are generated

# LFSR Example (Cont.)

- To generate  $2^n$  patterns using LFSR



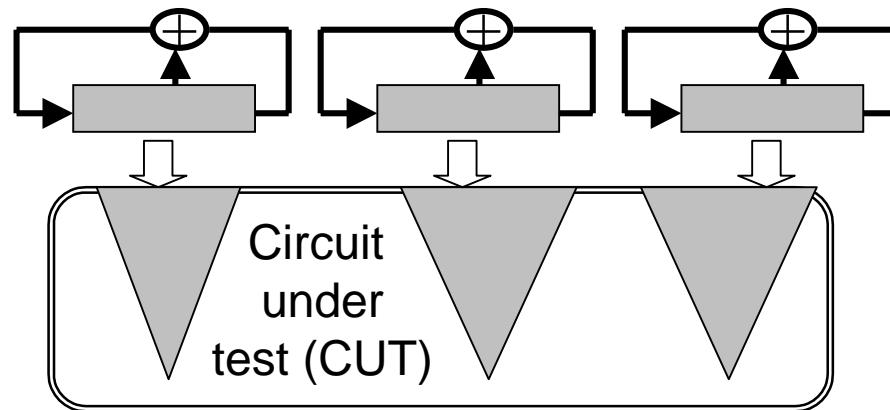
- Sequence becomes:

0	0	0	1	↓	0	0	0	1
0	0	0	0	↓	⋮	0	0	0
1	0	0	0					

# What to Do if $2^n$ is too Large?

- Using “pseudorandom”  
e.g. generate  $2^{32}$  pattern only

- Partitioning



- Using pseudo-exhaustive

# Constant Weight Patterns (for pseudoexhaustive testing)

- $T$ , set of binary  $n$ -tuples, exhaustively covers all  $k$ -subspaces if for all subsets of  $k$  bits positions, each of the  $2^k$  binary patterns appears at least once in  $T$ , where
- Example:

$$T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad \begin{array}{l} k \leq n \\ n = 3 \\ |T| = 4 \\ k = 2 \end{array}$$

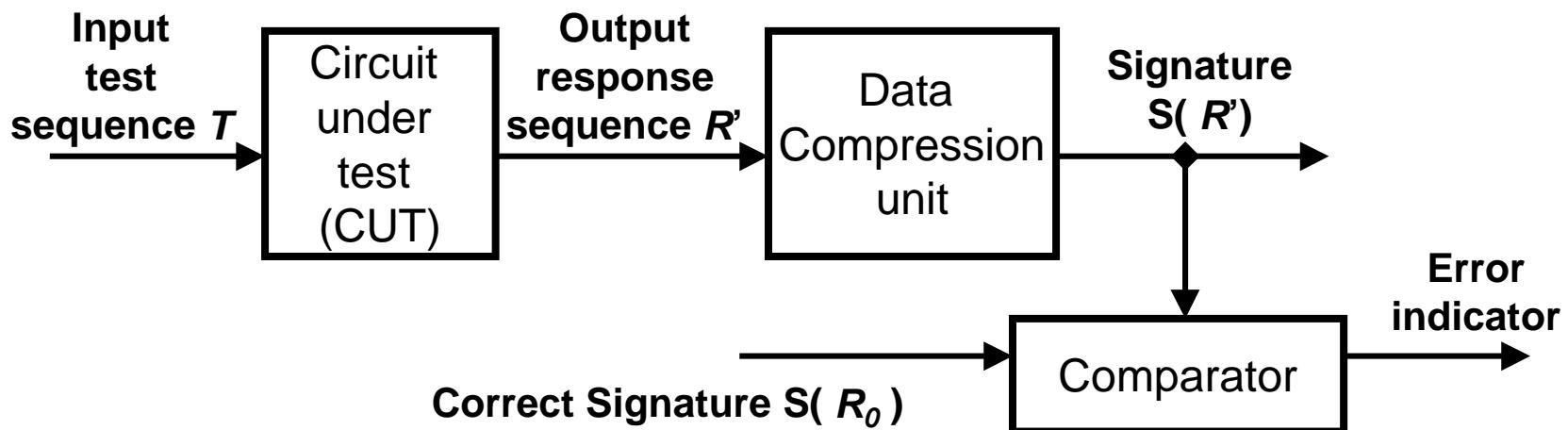
⇒  $T$  can be a pseudoexhaustive test for a  $(n,w)$ -CUT if  $k \geq n$

# Constant Weight Patterns (Cont.)

- A binary  $n$ -tuple has weight  $k$  if it contains exactly  $k$  1's
- There are  $C_k^n$  binary  $n$ -tuples having weight  $k$ .
- Theorem: Given  $n$  and  $k$ ,  $T$  exhaustively covers all binary  $k$ -subspaces if it contains all binary  $n$ -tuples of weight(s)  $w$  such that  $w \equiv c \pmod{n-k+1}$  for some integer  $c$ , where 
$$\begin{cases} 0 \leq c \leq n - k, \\ 0 \leq w \leq n \end{cases}$$

# Compression Techniques

- Bit-by-bit comparison is inefficient
- Compress information into “signature”
  - ➡ Response compacting or compressing



# **Compression Techniques to be Discussed**

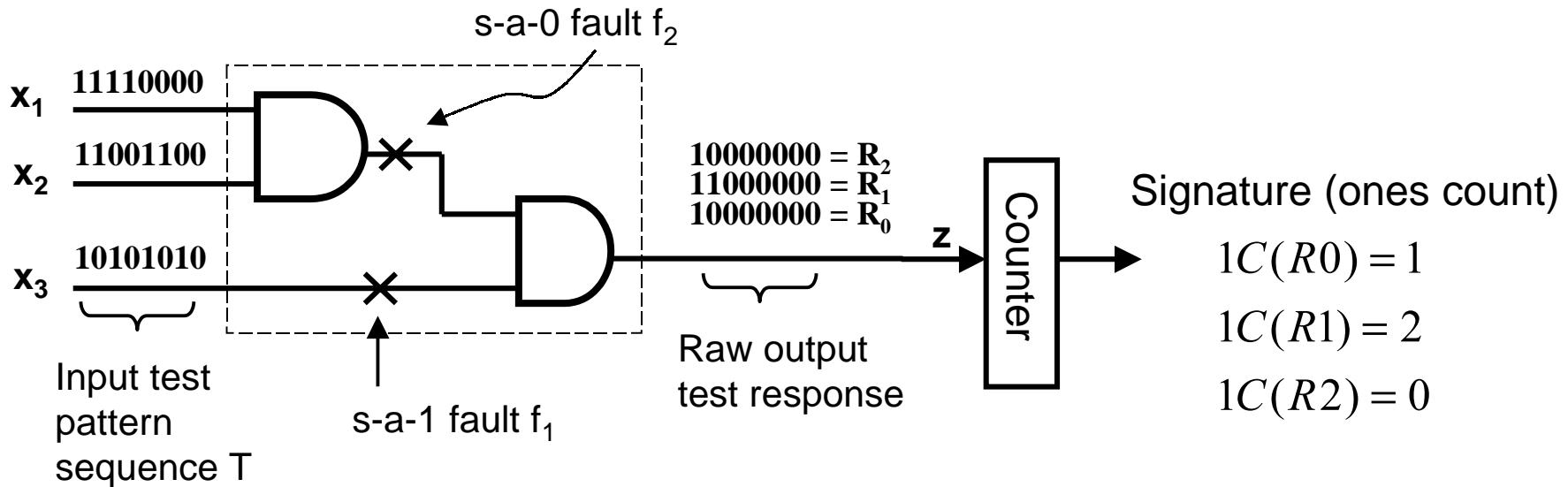
- **Ones Count**
- **Transition Count**
- **Parity Checking**
- **Syndrome Checking**
- **Signature Analysis**

# Practical Compression Techniques

- Easy to implement, part of BIST logic
- Small performance degradation
- High degree of compaction
- No or small aliasing errors
- Problems:
  1. Aliasing error:  
**signature (faulty ckt) = signature (fault-free ckt)**
  2. Reference (good) signature calculation:
    - ⇒ Simulation
    - ⇒ Golden unit
    - ⇒ Fault tolerant

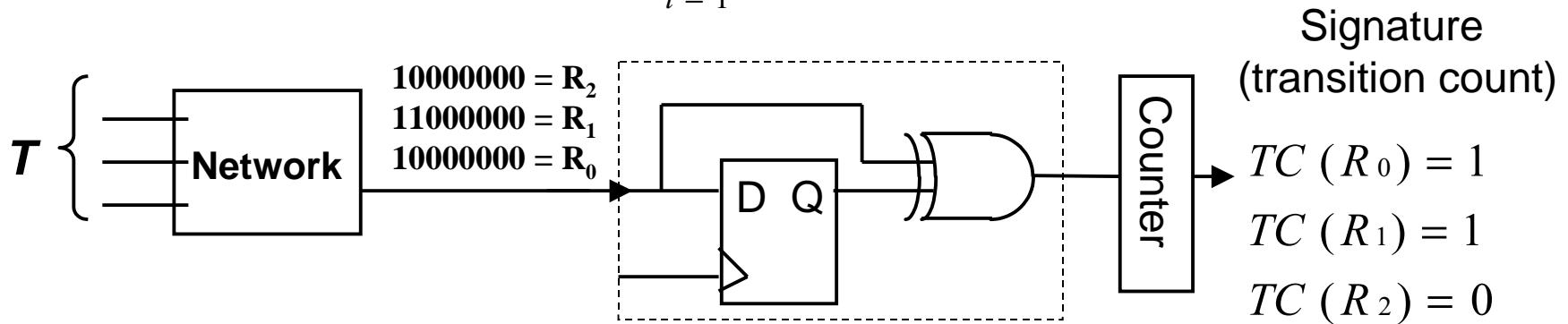
# Ones-count Compression

- **C: single-output circuit**
- **R: output response**  $R = r_1r_2\dots r_m$
- **1C(R) = # ones in**  $R = \sum_i r_i$



# Transition-count Compression

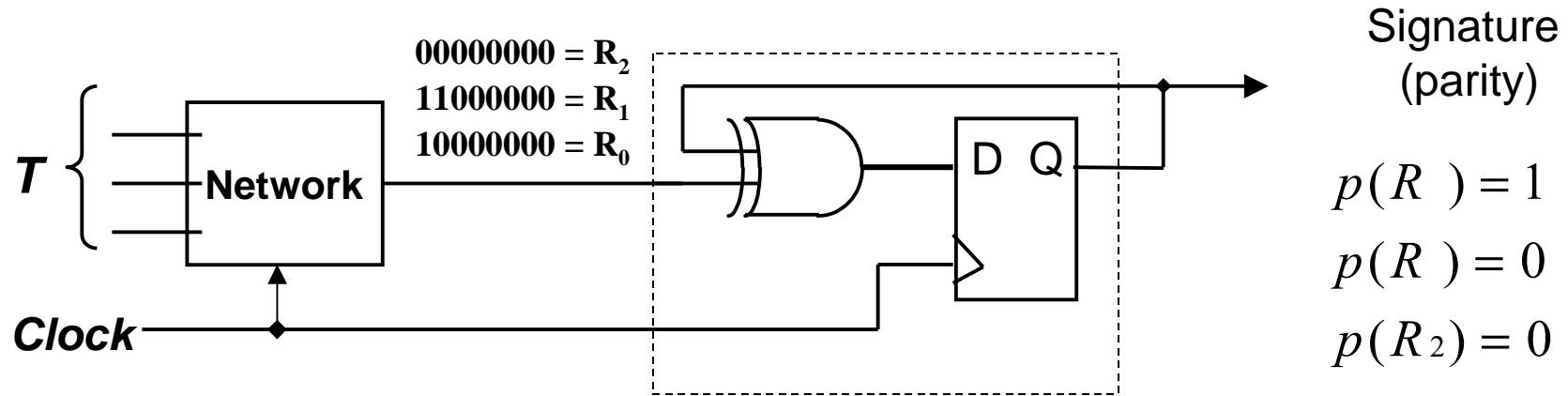
$$TC(R) = \sum_{i=1}^m r_i \oplus r_{i+1}$$



- $TC(R_0, R_1, R_2) = \frac{2C_r^{m-1} - 1}{2^m - 1}$

- **Does not guarantee the detection of single-bit errors**
- **Prob. (single-bit error masked)  $= \frac{m-2}{2^m}$**
- **Prob. (masking error)  $\rightarrow (\pi m)^{-\frac{1}{2}}$**

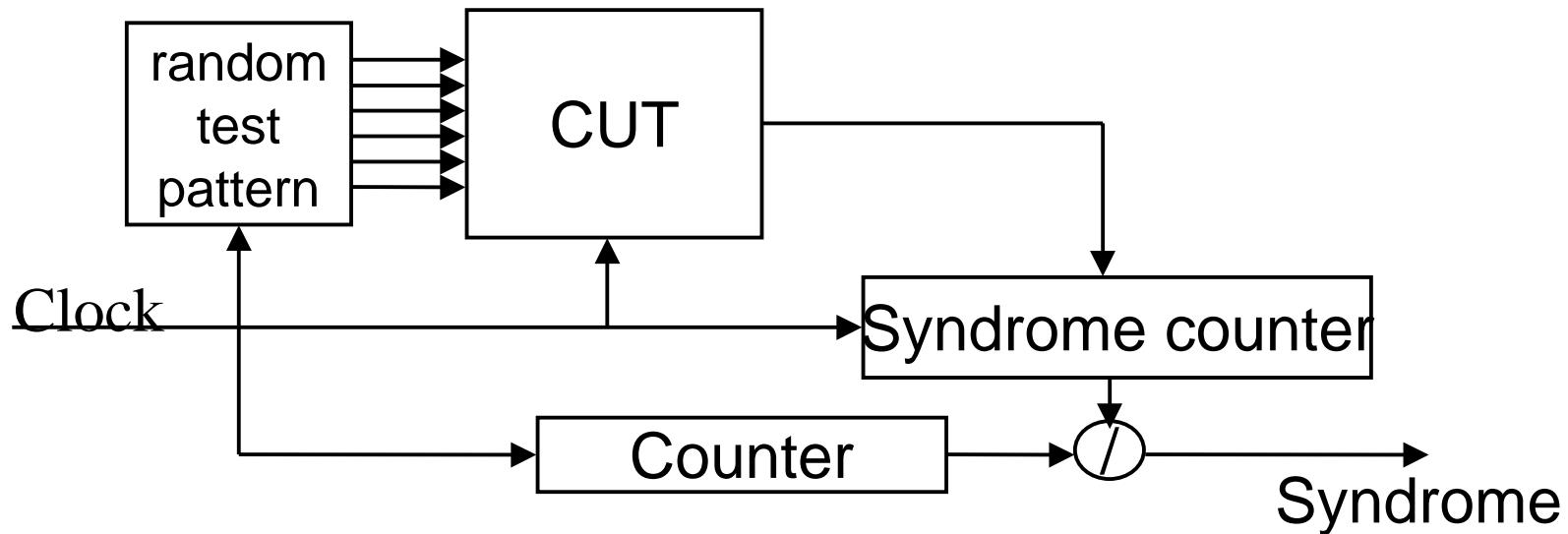
# Parity-check Compression



- Detect all single bit errors
- All errors consisting of odd number of bit errors are detected
- All errors consisting of even number of bit errors are masked
- Prob. (error masked)  $\approx \frac{1}{2}$

# Syndrome Checking

- Apply random patterns.
- Count the probability of 1.
- The property is similar to that of ones count.



# Signature Analysis

- Based on linear feedback shift register (LFSR)
- A linear ckt. composed of
  - ⇒ unit delay or D FFs
  - ⇒ Modulo-2 scalar multipliers or adders

+/-	0	1
0	0	1
1	1	0

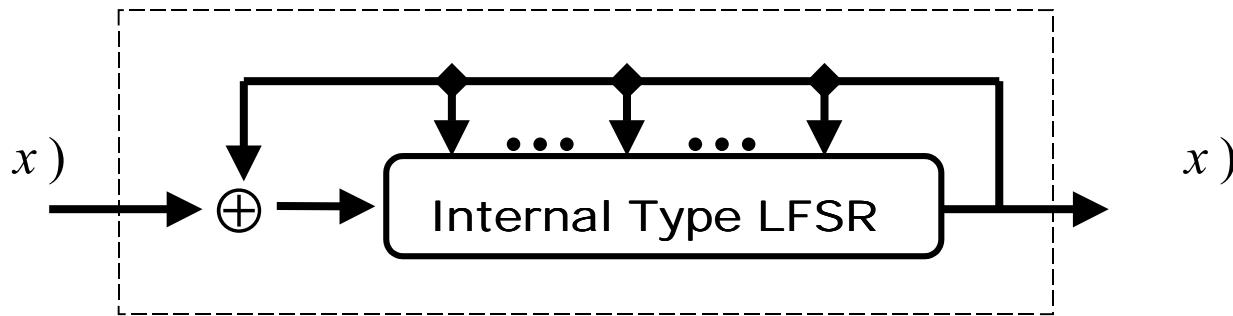
*	0	1
0	0	0
1	0	1

- Linear ckt.:
  - Response of linear combination of inputs
  - = Linear combination of responses of individual inputs

$$I_1 + I_2) = (I_1) + (I_2)$$

# Use LFSR as Signature Analyzer

- Single-input LFSR



- Initial state  $x = 0$
  - Final state  $x)$
- $$\Rightarrow \frac{x)}{x)} = x) + \frac{x)}{x)} \quad \text{or} \quad x) = x) - x) + x)$$

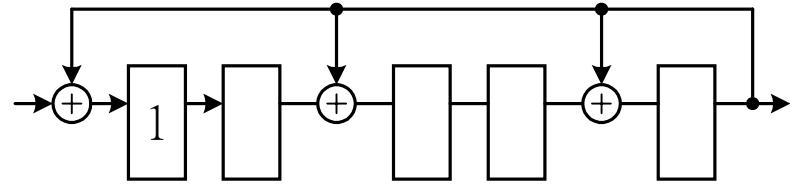
$x)$  : The remainder, or the signature

# Signature Analyzer (SA)

**Input sequence:**

1 1 1 1 0 1 0 1 (8 bits)

$$G(x) = x^7 + x^6 + x^5 + x^4 + x^2 + 1$$



$$P(x) = 1 + x^2 + x^4 + x^5$$

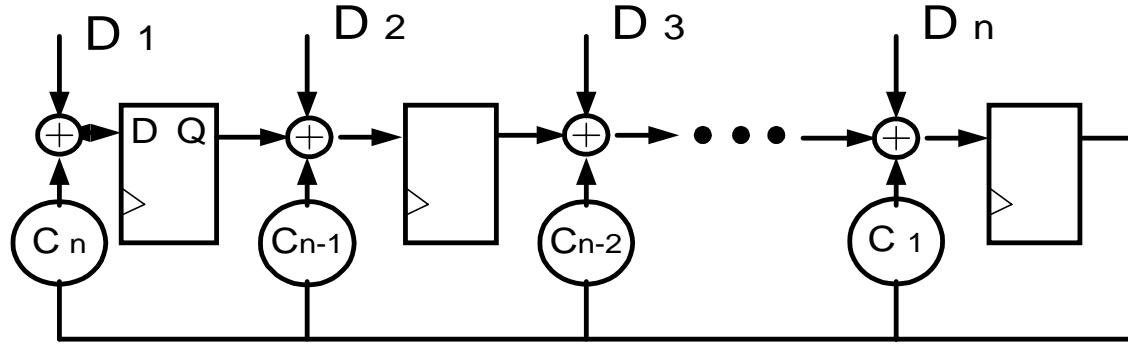
Time	Input stream	Register contents	Output stream
------	--------------	-------------------	---------------

		1 2 3 4 5	
0	1 0 1 0 1 1 1	0 0 0 0 0	Initial state
1	1 0 1 0 1 1 1	1 0 0 0 0	
:	:	:	
5	1 0 1	0 1 1 1 1	
6	1 0	0 0 0 1 0	1
7	1	0 0 0 0 1	0 1
8	Remainder	0 0 1 0 1	1 0 1
			Quotient
	Remainder	$R(x) = x^6 + x^4$	$Q(x) = 1 + x^2$

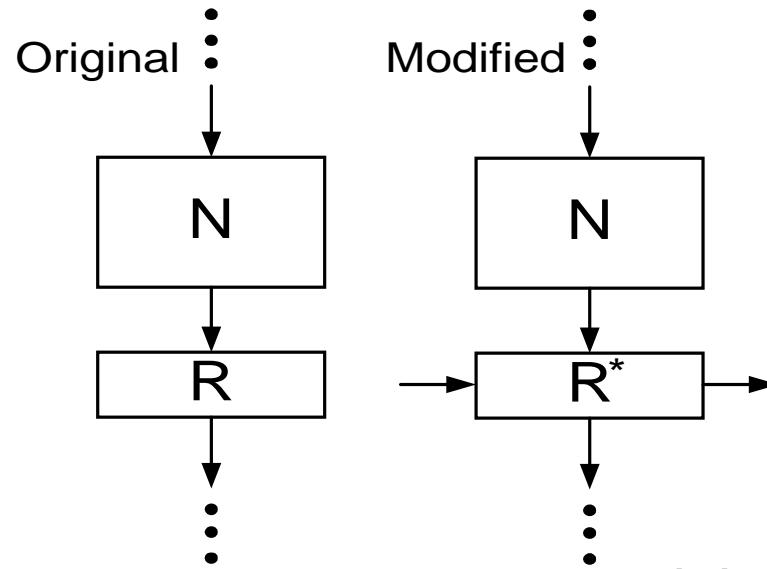
# Signature Analyzer (SA) (Cont.)

$$\begin{array}{r} x \quad x + x + x^2 + 1 \\ \times \quad x \quad x^2 + 1 \\ \hline \\ = \quad \quad \quad x + x + x^5 + 1 \end{array}$$
$$x) \quad x) + \quad x) = x + x + x + x + x^2 + 1 = \quad x)$$

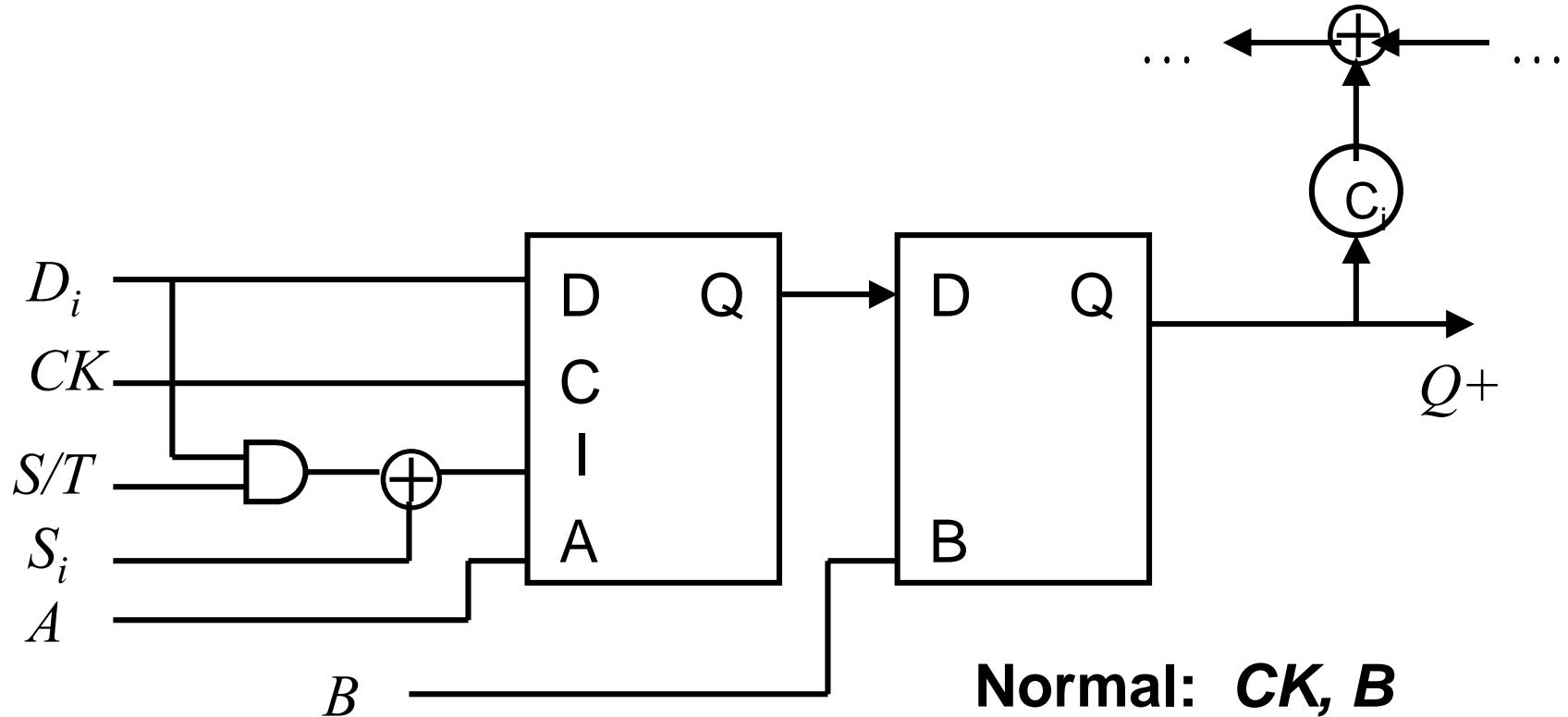
# Multiple-input Signature Register (MISR)



- **Implementation:**



# Storage Cell of a SA



**Normal:**  $CK, B$   
**Shift:**  $S/T=0, A, B$   
**MISR:**  $S/T=1, A, B$

# Performance of Signature Analyzer

- For a test bit stream of length  $m$  :  
# possible response =  $2^m$ , of which only one is correct
- The number of bit stream producing a specific signature is

$$\frac{2}{2^n} = 2^{-n}$$

# Performance of Signature Analyzer (Cont.)

- Among these stream , only one is correct.

$$SA(\quad | \quad, n) = \frac{2^{m-n} - 1}{2^m - 1} \cong 2^{-n}$$

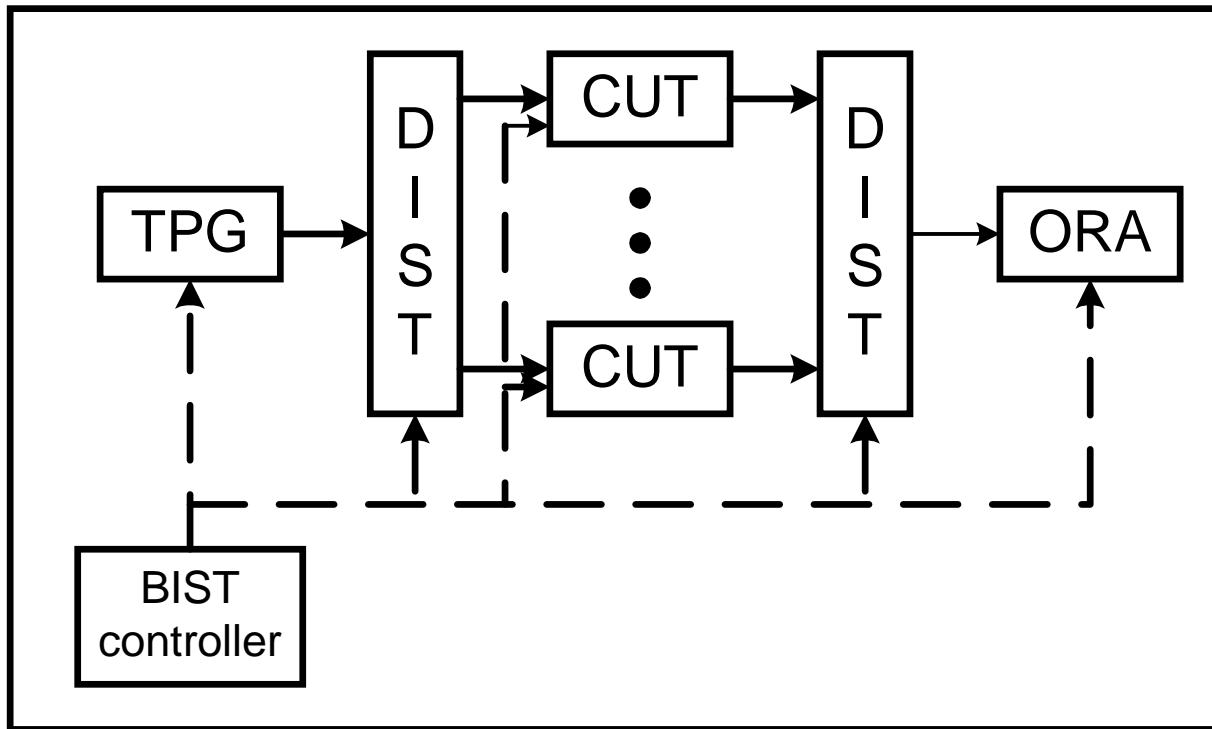
- If  $n=16$ , then  $1 - 2^{-16}$  ) 1 % = 99.99% of erroneous response are detected.  
*(Note that this is not % of faults !)*

# Generic Off-line BIST Architecture

- Categories of architectures
  - Centralized or Distributed
  - Embedded or Separate BIST elements
- Key elements in BIST architecture
  - Circuit under test (CUTs)
  - Test pattern generators (TPGs)
  - Output-response analyzers (ORAs)
  - Distribution system for data transmission between TPGs, CUTs and ORAs
  - BIST controllers

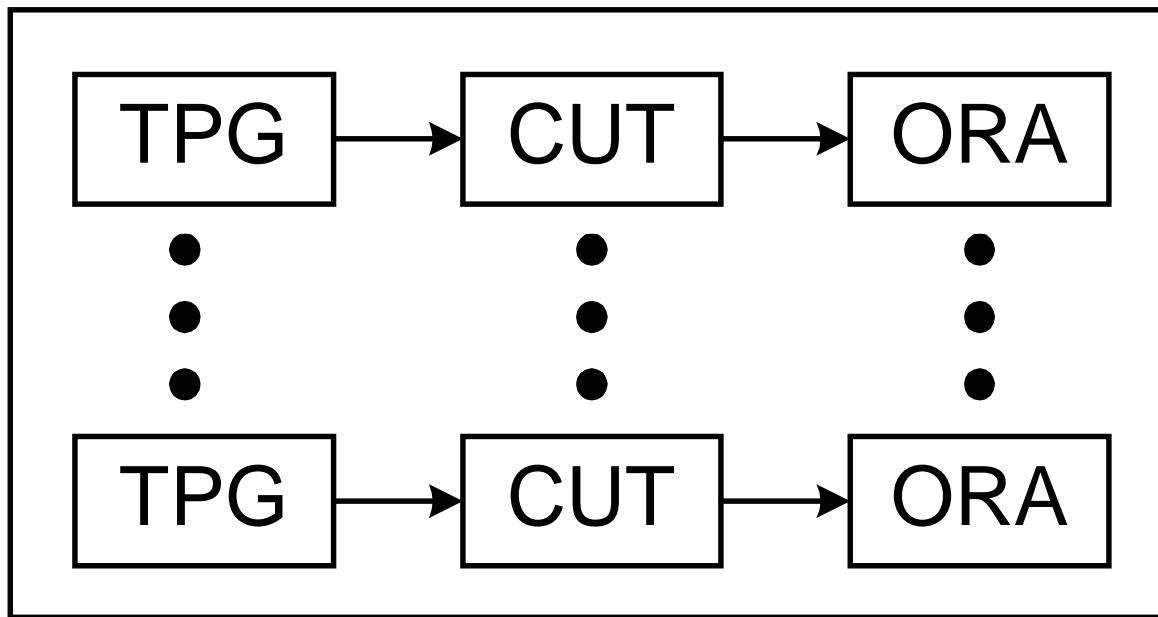
# Centralized/ Separate BIST

Chip, board, or system



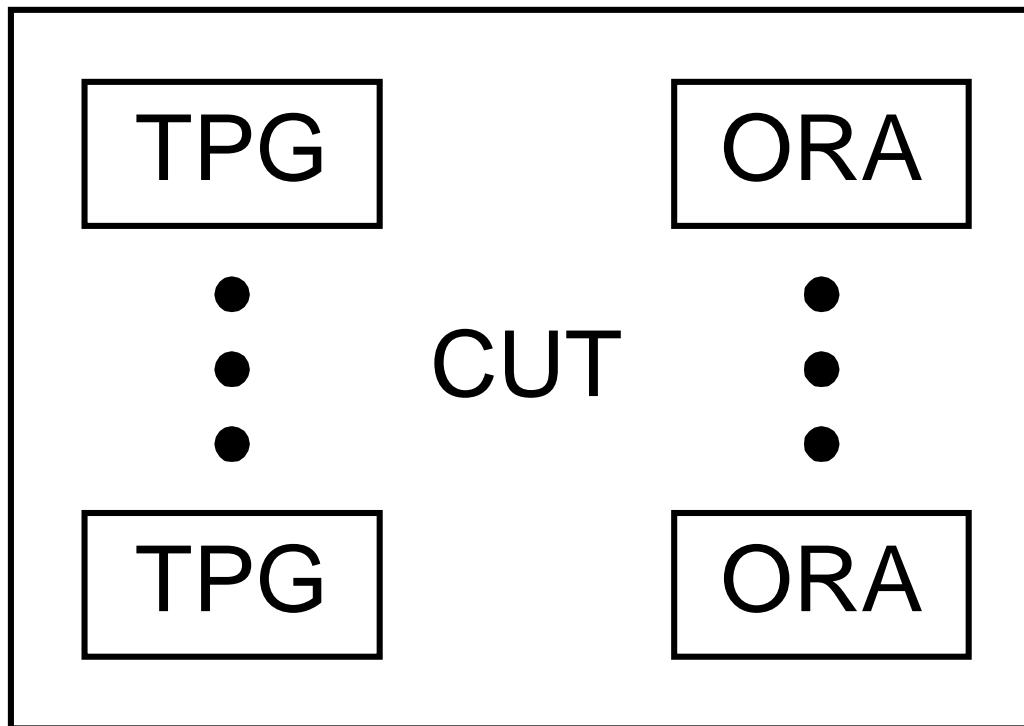
# Distributed / Separate BIST

Chip, board, or system



# Distributed / Embedded BIST

Chip, board, or system



# Factors Affecting the Choice of BIST

- Degree of test parallelism
- Fault coverage
- Level of packaging
- Test time
- Complexity of replaceable unit
- Factory and field test-and-repair strategy
- Performance degradation
- Area overhead

# Specific BIST Architectures

- Ref. Book by Abramovici, Breuer and Friedman
- Centralized and Separate Board-Level BIST (CSBL)
- Built-in Evaluation and Self-Test (BEST)
- Random-Test Socket (RTS)
- LSSD On-Chip Self-Test (LOCST)
- Self-Testing Using MISR and Parallel SRSG (STUMPS)

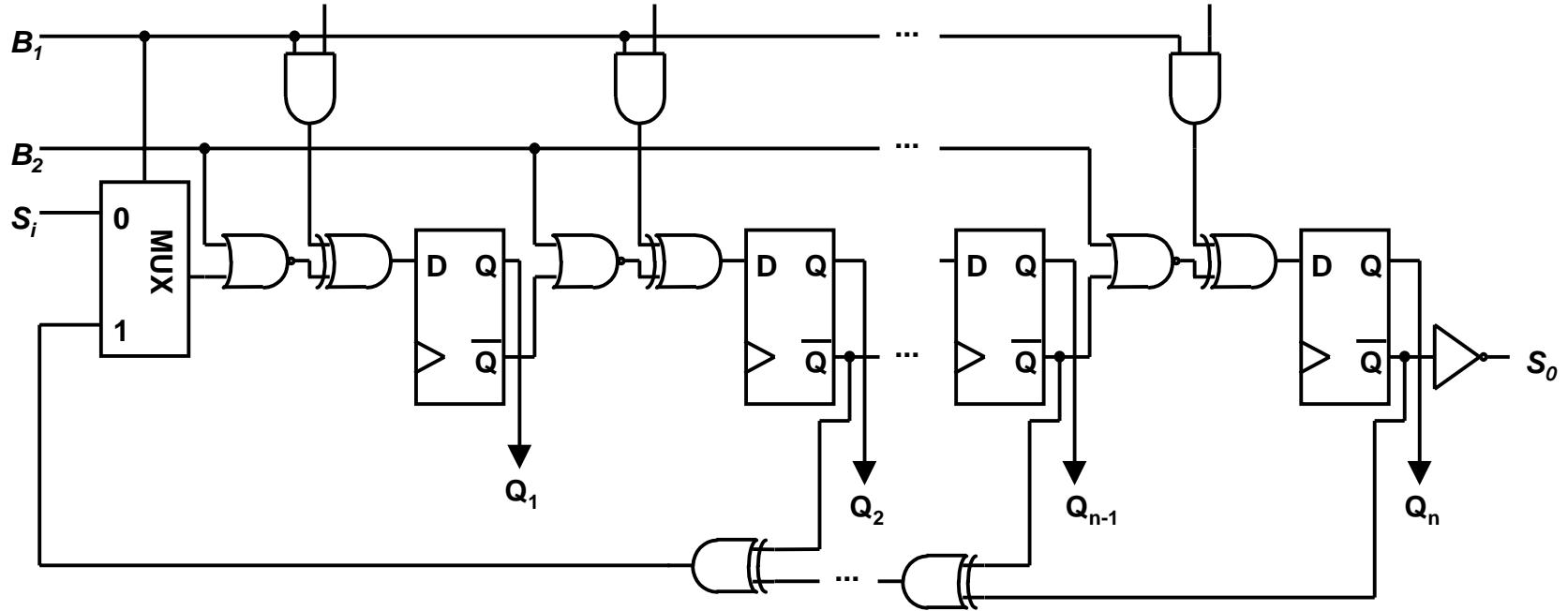
# Specific BIST Architectures (Cont.)

- **Concurrent BIST (CBIST)**
- **Centralized and Embedded BIST with Boundary Scan (CEBS)**
- **Random Test Data (RTD)**
- **Simultaneous Self-Test (SST)**
- **Cyclic Analysis Testing System (CATS)**
- **Circuit Self-Test Path (CSTP)**
- **Built-In Logic-Block Observation (BILBO)**

# **Built-In Logic Block Observation (BILBO)<sup>[1]</sup>**

- **Distributed**
- **Embedded**
- **Combinational “Kernels”**
- **Chip level**
- **“Clouding” of circuit**
- **Registers based description of circuit**
- **BILBO registers**
  
- [1]. B. Konemann, *et al.*, “Built-In Logic\_Block Observation Technique,” Digest of papers 1979 Test Conf., pp.37-41, Oct., 1979

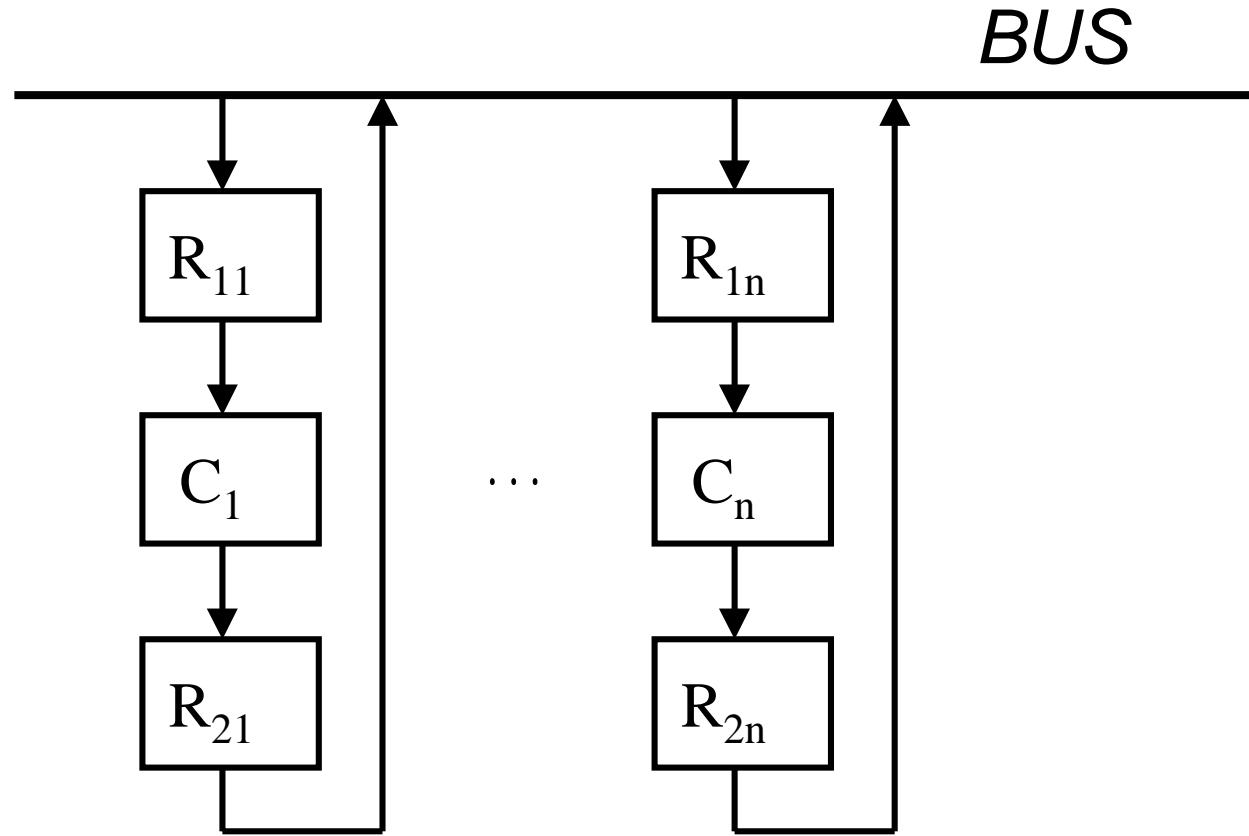
# BILBO Registers



$B_1$	$B_2$	BILBO
0	0	shift register
0	1	reset
1	0	MISR (input * constant * LFSR)
1	1	parallel load (normal operation)

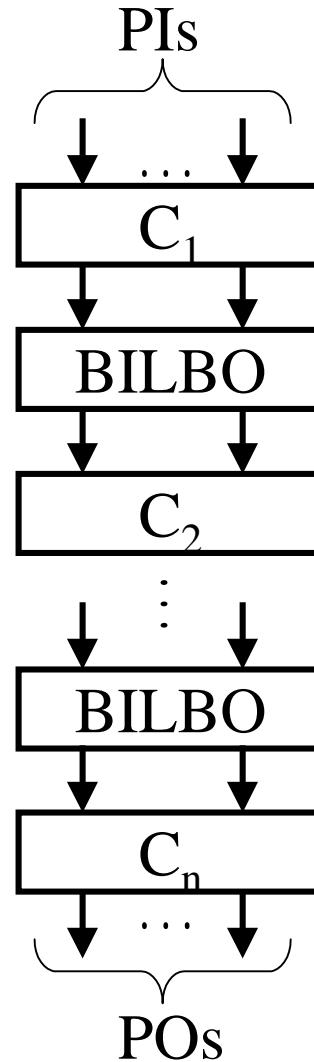
# Applications of BILBO

- Bus-oriented structure

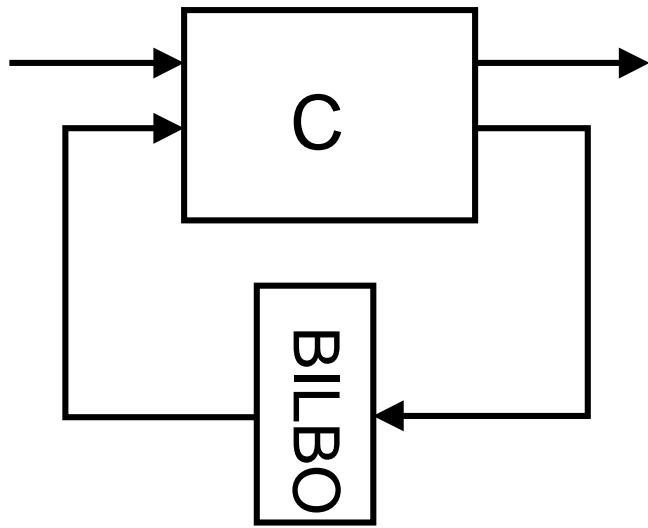


# Applications of BILBO (Cont.)

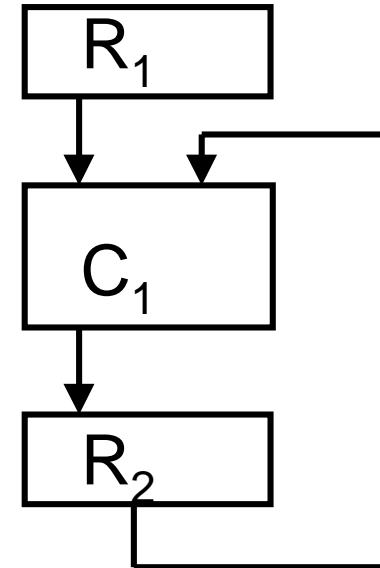
- Pipeline-oriented structure



# Problems with BILBO



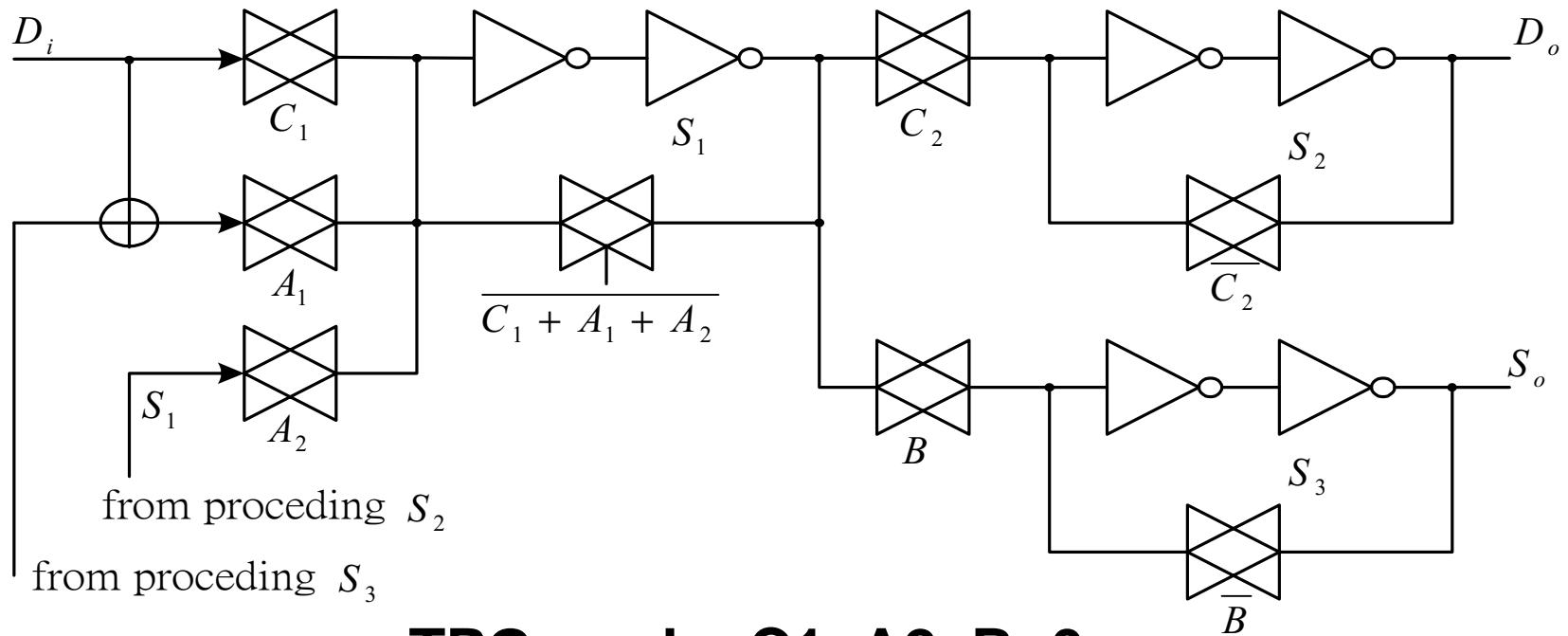
MISR or TPG ?



R2 ?

⇒ Using CBILBO

# Transistor Level Implementation of CBILBO

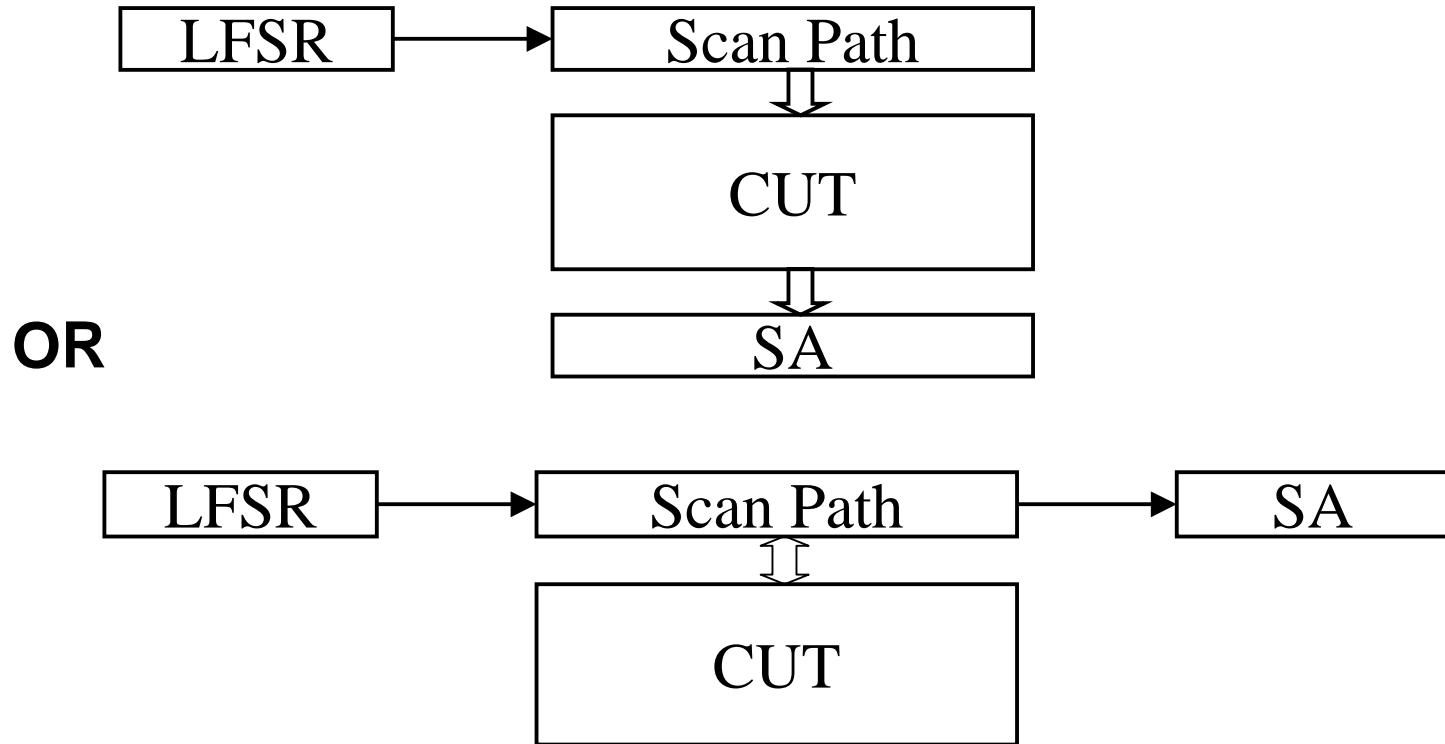


**TPG mode:  $C_1, A_2, B=0$**

**MISR mode:  $C_1, A_1, C_2=0$**

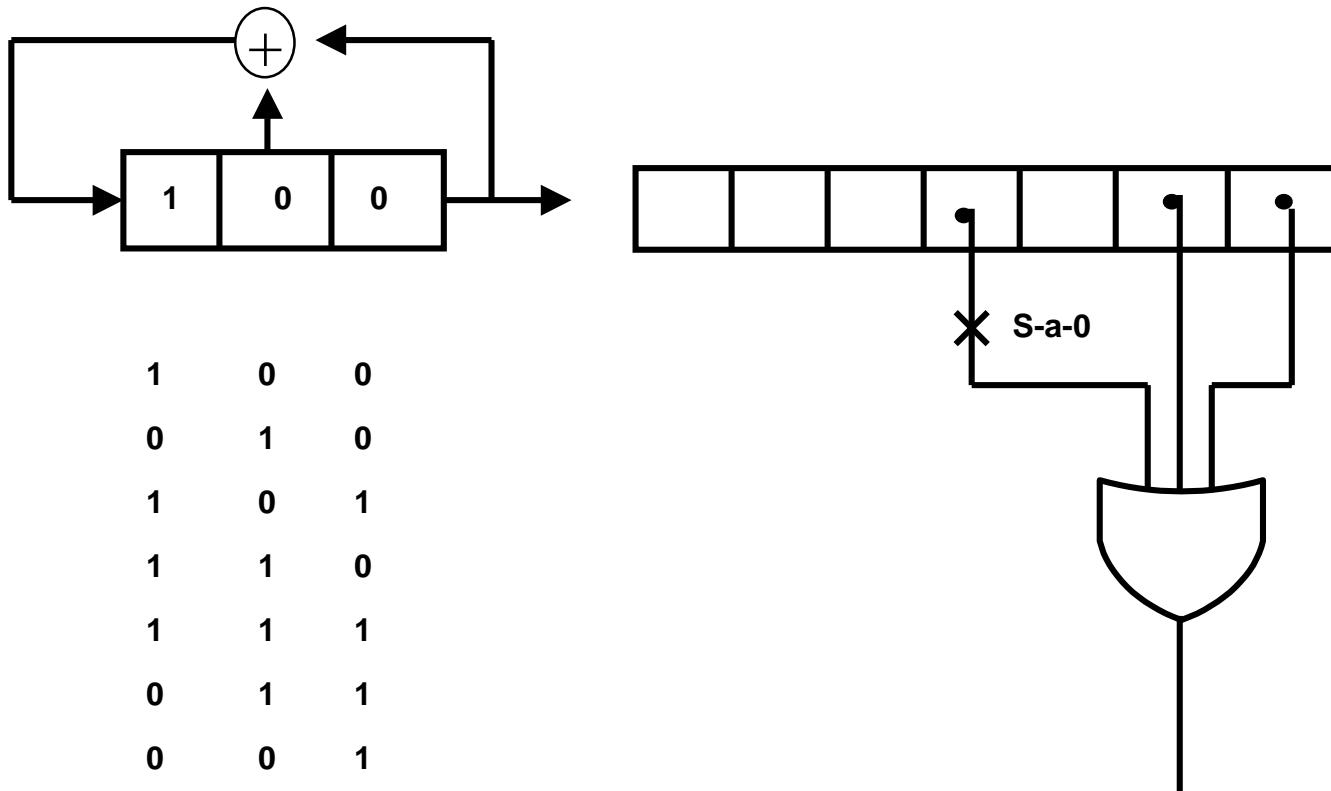
**(c) A simultaneous TPG/MISR S-Cell**

# Combination of LFSR and Scan Path



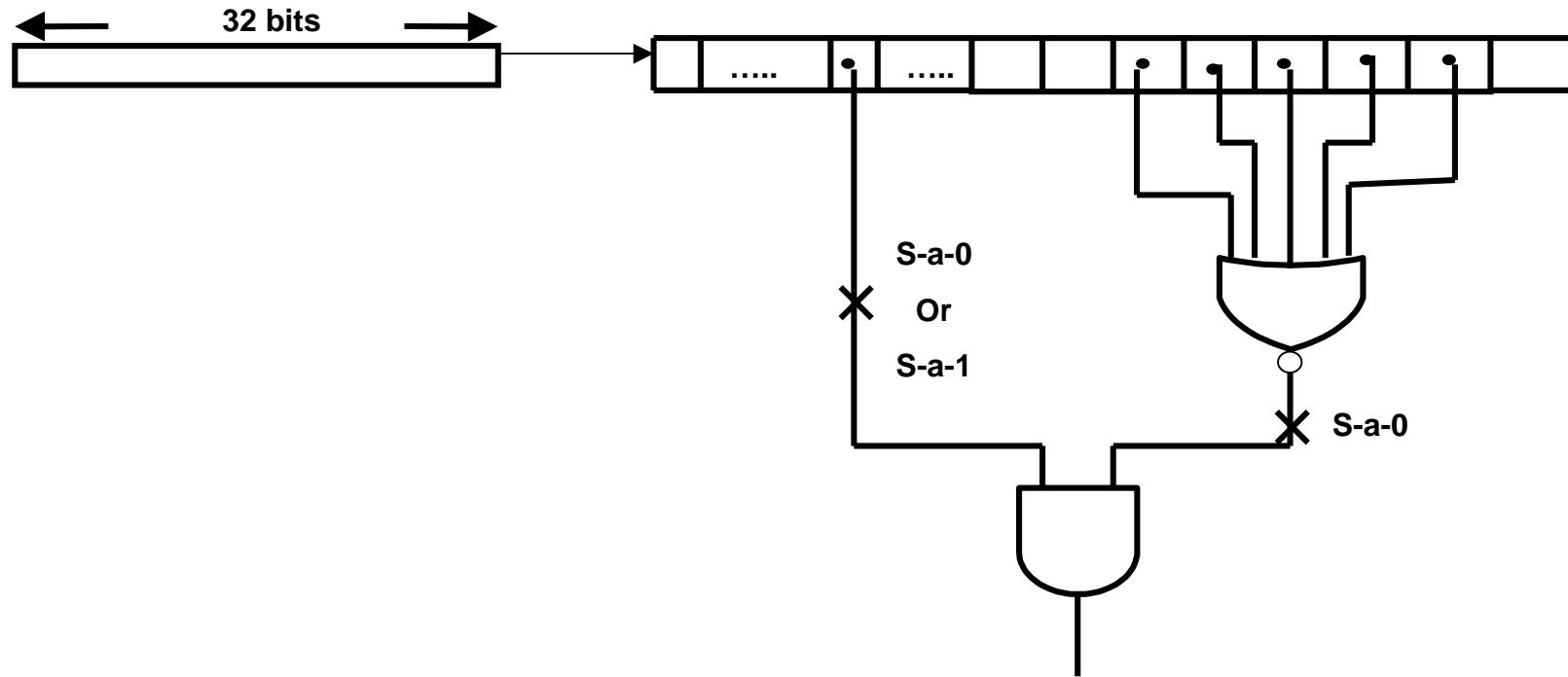
- **Problem:** Some hard-to-detect faults may never be exercised

# Example 1



- The fault can never be detected

# Example 2



- The faults are difficult to detect

# Solutions:

- **Exhausting testing**
  - **Weighted random testing**
  - **Mixed mode vector pattern generation**
    - **Pseudorandom vectors first**
    - **Deterministic tests followed**
- **Do not consider the fact that the test vectors are given in a form of testcubes with many unspecified inputs.**
4. **Reseeding**
    - **Change the seeds as needed**
  5. **Reprogram the characteristic polynomial**
  6. **Combination of two or more of the above methods**