Built-in Self-test

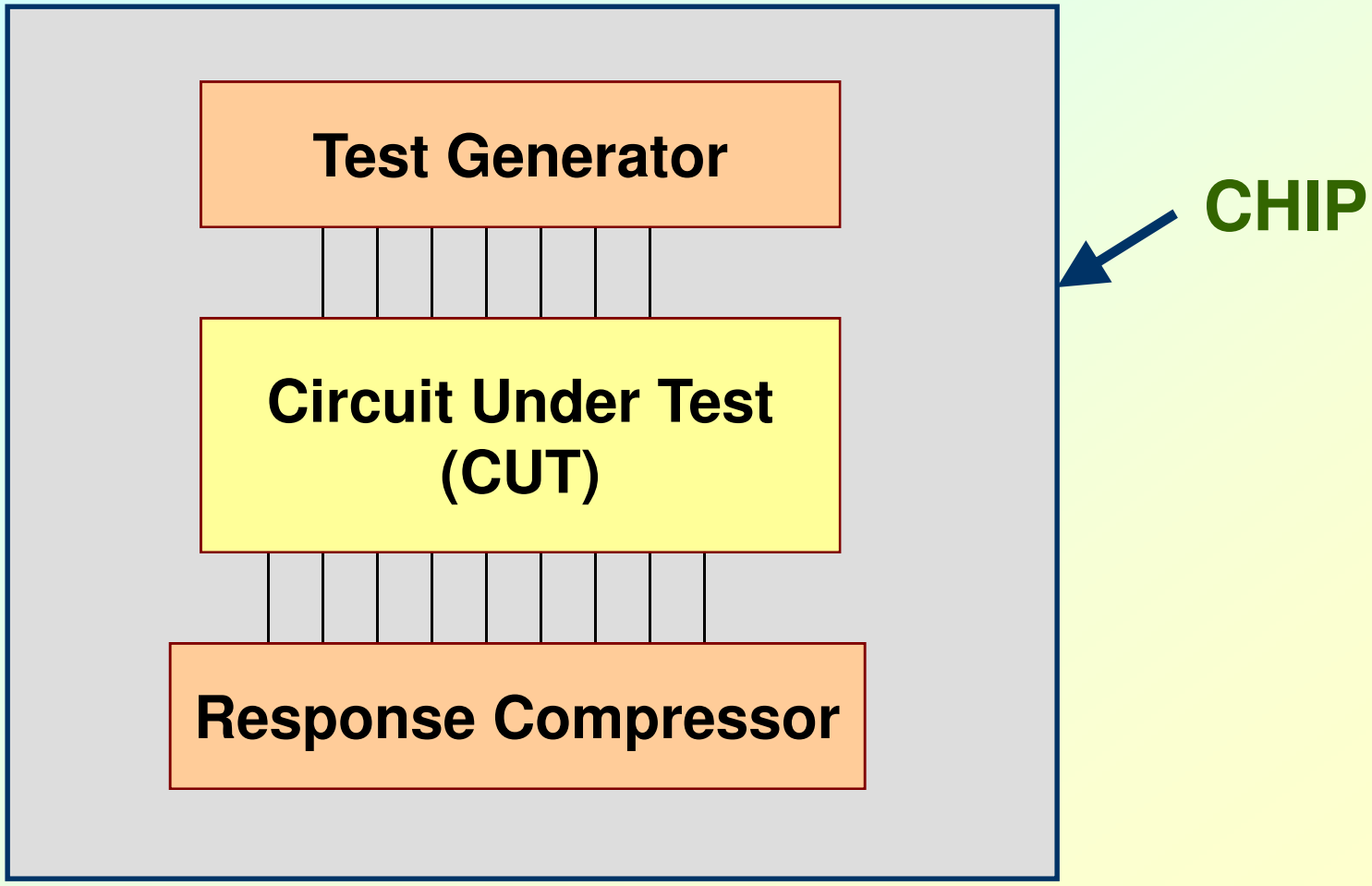
Introduction

- Test generation and response evaluation done on-chip.
- Only a few external pins to control BIST operation.
- Additional hardware overhead.
- Offers a number of benefits.

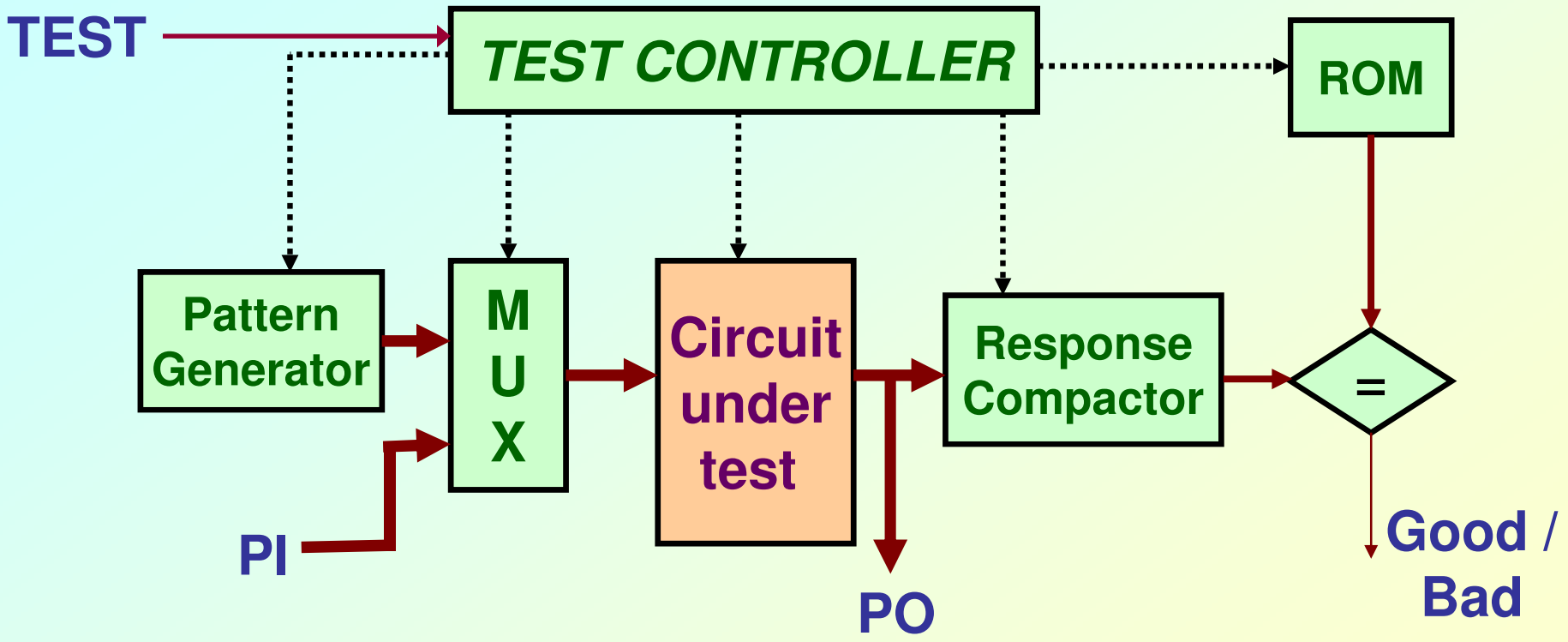
BIST Motivation

- 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

BIST – Basic Idea



BIST Architecture



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

BIST Costs

- Chip area overhead for:
  - Test controller
  - Hardware pattern generator / response compactor
  - 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
- Reliability reduction
  - Due to increased area
- Increased BIST hardware complexity
  - Happens when BIST hardware is made testable

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

BIST Techniques

- **Stored Vector Based**
  - **Microinstruction support**
  - **Stored in ROM**
- **Algorithmic Hardware Test Pattern Generators**
  - **Counter :: exhaustive, pseudo-exhaustive**
  - **Linear Feedback Shift Register**
  - **Cellular Automata**

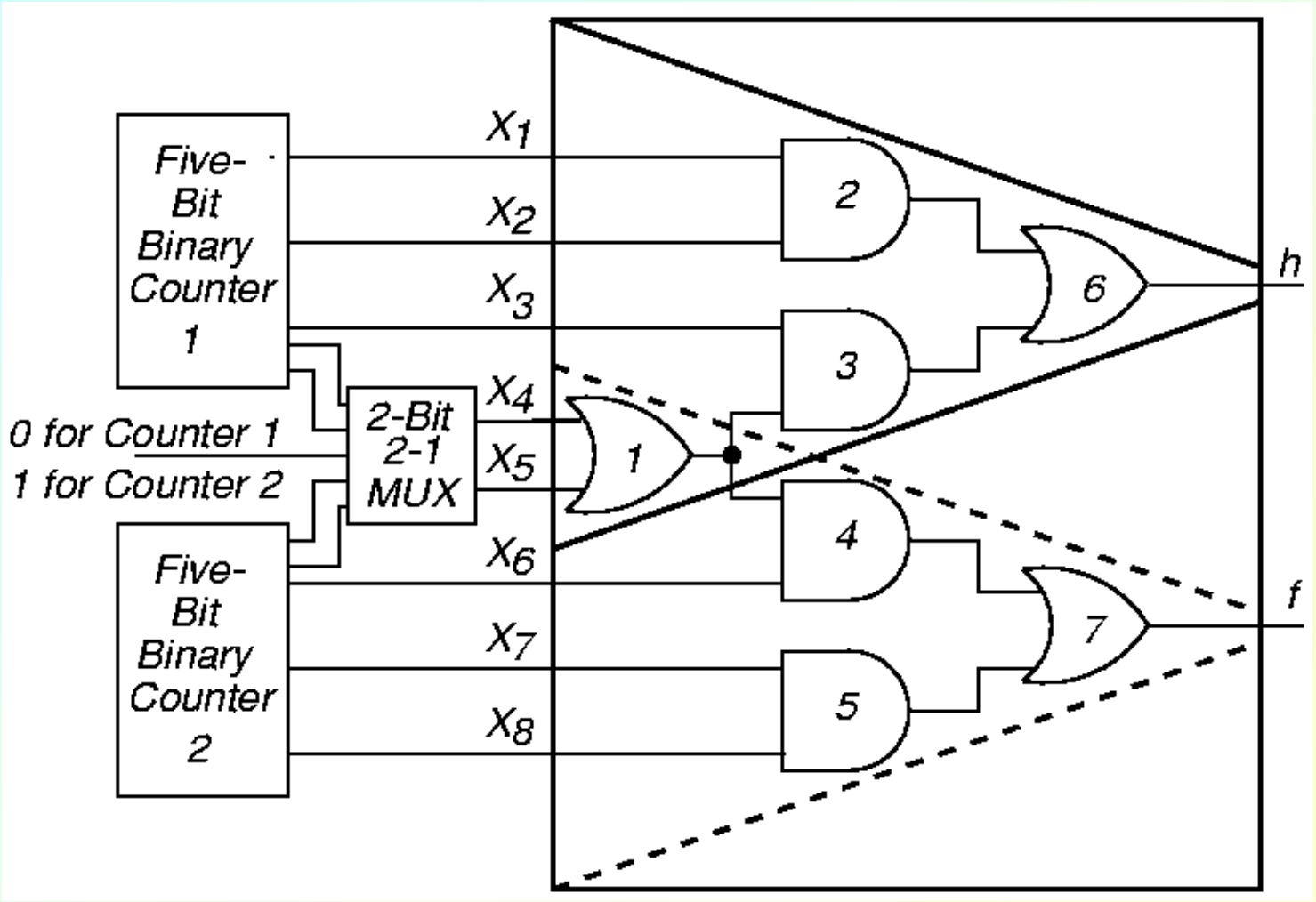
Exhaustive Pattern Generation

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

Pseudo-Exhaustive Method

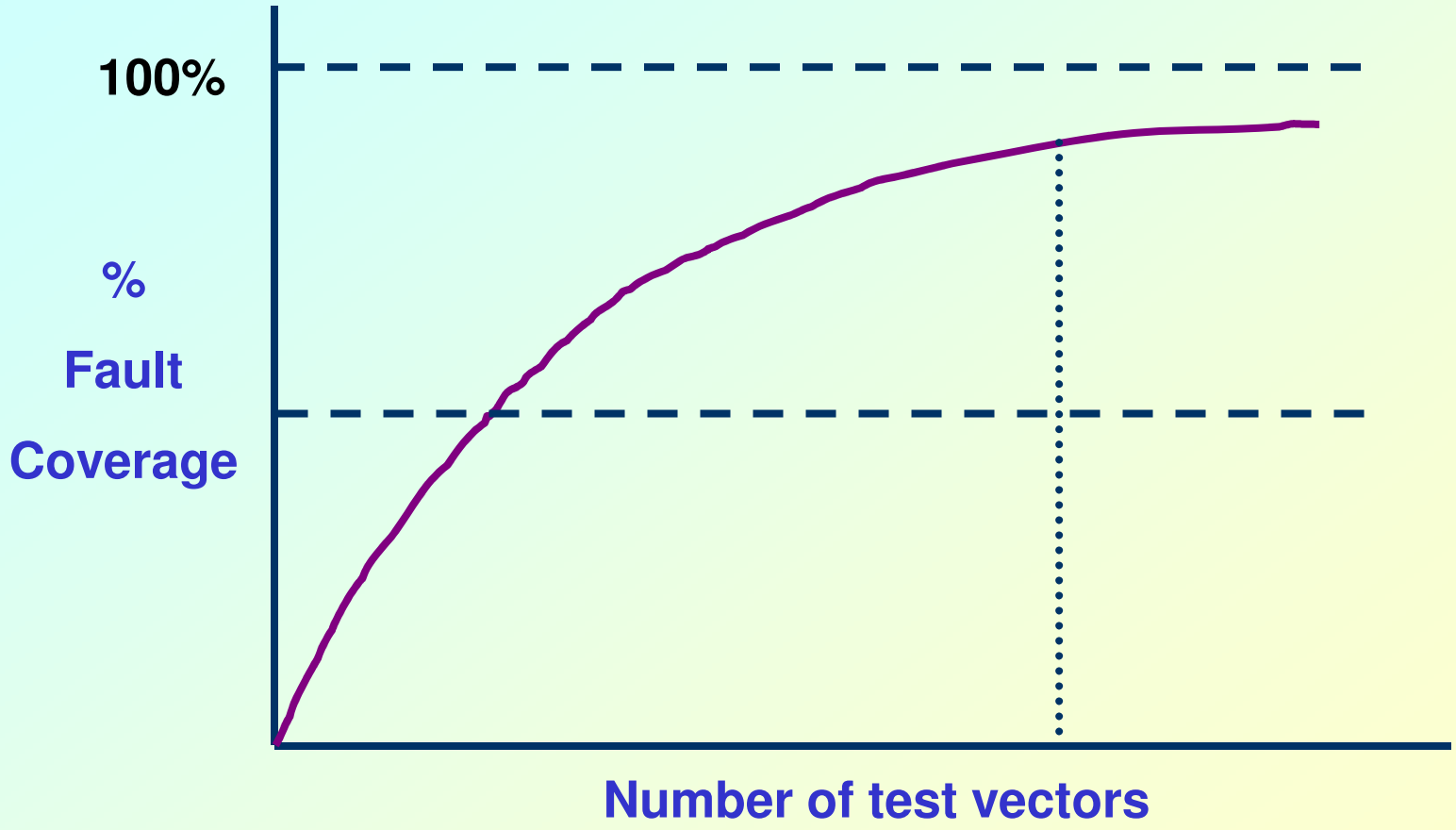
- Partition large circuit into *fanin cones*
  - Backtrace from each PO to PIs influencing it
  - Test fanin cones in parallel
- An illustrative example (next slide):
  - No. of tests reduced from $2^8 = 256$ to $2^5 \times 2 = 64$

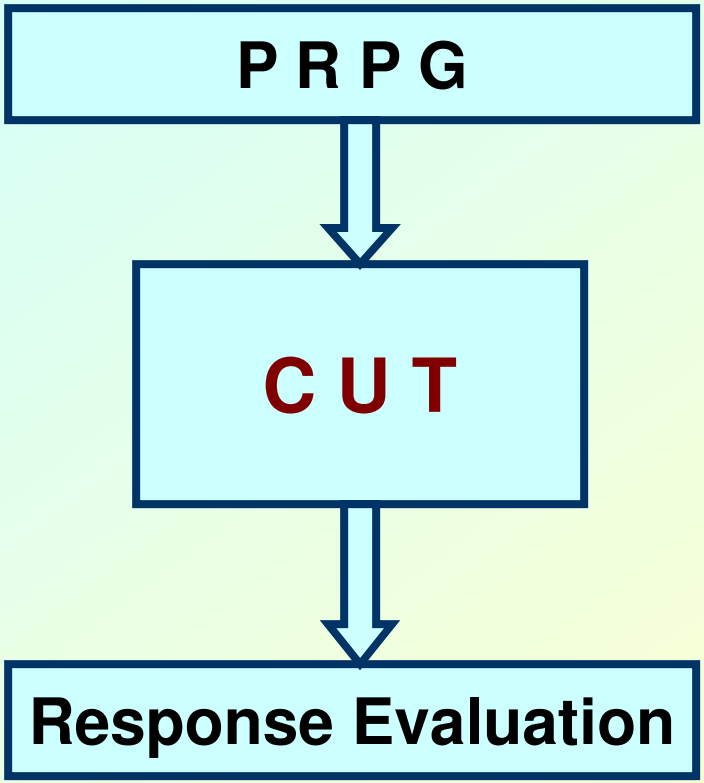
Pseudo-Exhaustive Pattern Generation



Random Pattern Testing

- Generate pseudo-random patterns as test input vectors.
- Evaluate fault coverage through fault simulation.
- Motivation:
  - Test length may be larger.
  - Faster test generation.
- Used to get tests for 60-80% of faults, then switch to ATPG for rest.
- Some circuits may be *random pattern resistant*.


100%
%
Fault
Coverage
Number of test vectors


P R P G
C U T
Response Evaluation

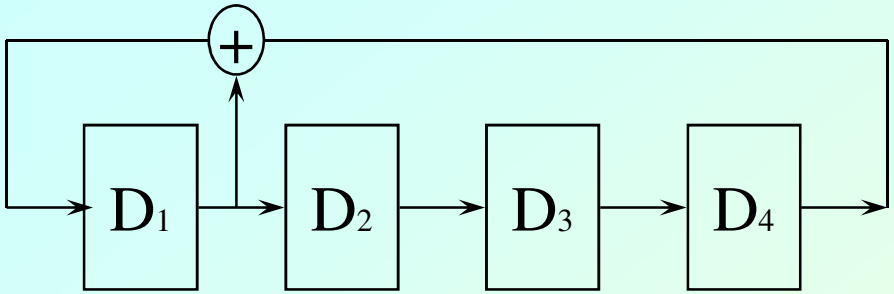
Linear Feedback Shift Register (LFSR)

What is LFSR?

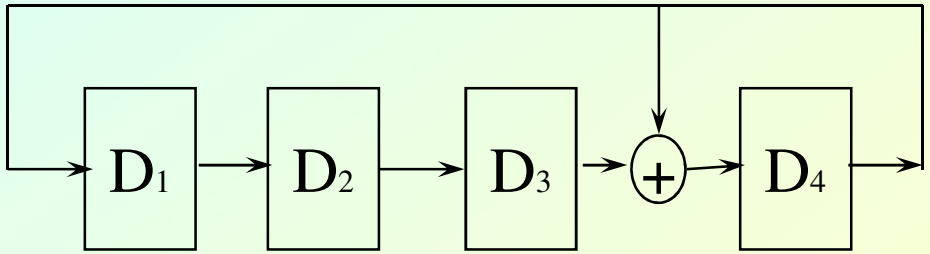
- A simple hardware structure based on shift register.
  - Linear feedback circuit.
- Has a number of useful applications:
  - Pseudo-random number generation
  - Response compression
  - Error checking (Cyclic Redundancy Code)
  - Data compression

Two types of LFSR

Type 1

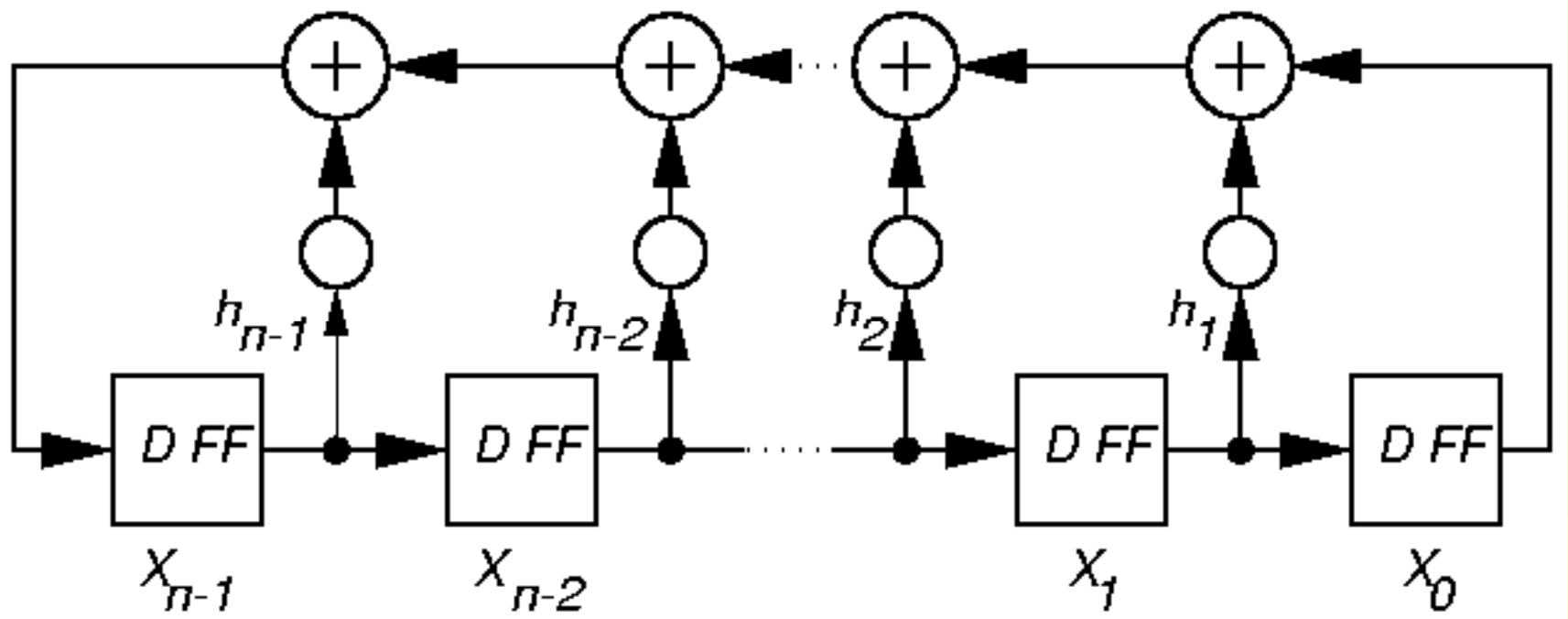


Type 2

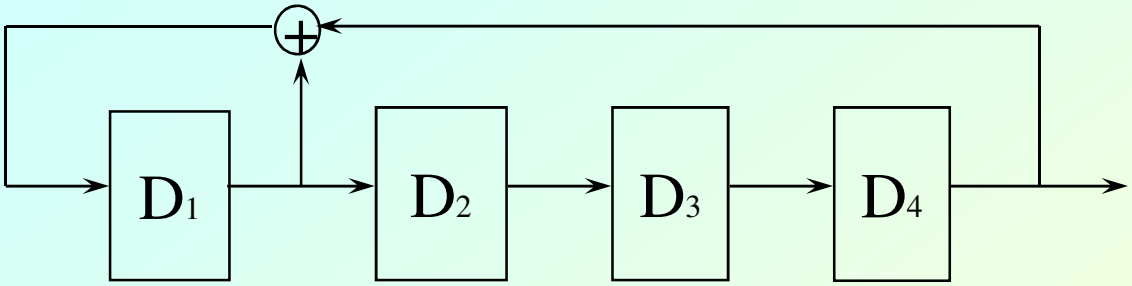


- Unit delay
  - D Flip flop
- Modulo 2 adder
  - XOR gate
- Modulo 2 multiplier
  - Connection

General Type-1 LFSR



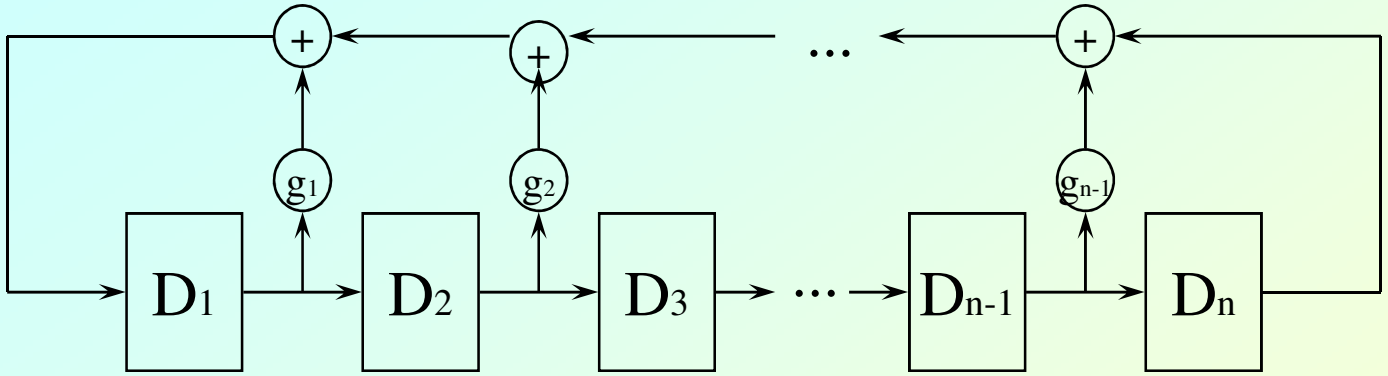
LFSR Example



$$f(x) = x^4 + x^1 + 1$$

D_4	D_3	D_2	D_1
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

LFSR - Recurrence Relation



I_S	a_{-1}	a_{-2}	a_{-3}	...	a_{-n+1}	a_{-n}
C_S	a_{m-1}	a_{m-2}	a_{m-3}	...	a_{m-n+1}	a_{m-n}

- **Generating Function**

$$G(x) = \sum_{m=0}^{\infty} a_m x^m$$

- **Characteristic polynomial**

$$f(x) = \sum_{i=1}^{n} c_i x^i + 1$$

LFSR - Recurrence Relation (continue)

$$a_m = \sum_{i=1}^{n} c_i \, a_{m-i}$$

$$G(x) = \sum_{m=0}^{\infty} a_m x^m$$

$$= \sum_{m=0}^{\infty} \sum_{i=1}^{n} c_i \, a_{m-i} x^m = \sum_{i=1}^{n} c_i \sum_{m=0}^{\infty} a_{m-i} x^{m-i}$$

$$= \sum_{i=1}^{n} c_i x^i \left[a_{-i} x^{-i} + \ldots + a_{-1} x^{-1} + \sum_{m=0}^{\infty} a_m x^m \right]$$

$$= \sum_{i=1}^{n} c_i x^i \left[a_{-i} x^{-i} + \ldots + a_{-1} x^{-1} + G(x) \right]$$

LFSR - Recurrence Relation (continue)

$$\Rightarrow G(x) = \sum_{i=1}^{n} c_i x^i G(x) + \sum_{i=1}^{n} c_i x^i \left(a_{-i} x^{-i} + \cdots + a_{-1} x^{-1} \right)$$

$$\Rightarrow G(x) = \frac{\sum_{i=1}^{n} c_i x^i \left(a_{-i} x^{-i} + \cdots + a_{-1} x^{-1} \right)}{1 + \sum_{i=1}^{n} c_i x^i}$$

$$G(x) = \frac{\sum_{i=1}^{n} c_i x^i \left(a_{-i} x^{-i} + \cdots + a_{-1} x^{-1} \right)}{f(x)}$$

if $a_{-1} = a_{-2} = \cdots = a_{-n+1} = 0$ and $a_{-n} = 1 \Rightarrow G(x) = \frac{1}{f(x)}$

$G(x)$ is function of initial state and $g(x)$

LFSR - Definitions

- If the sequence generated by an n-stage LFSR has period 2^n-1, then it is called a *maximum-length sequence* or *m-sequence.*
- The characteristic polynomial associated with maximum-length sequence is called a *primitive polynomial.*
- An *irreducible polynomial* is one that cannot be factored; i.e., it is not divisible by any other polynomial other than 1 and itself.

Example Primitive Polynomials

3: 1 0 → $x^3 + x + 1$

4: 1 0

5: 2 0

6: 1 0

7: 1 0

8: 6 5 1 0

16: 5 3 2 0

32: 28 27 1 0

64: 4 3 1 0

LFSR - Theories

- If the initial state of an LFSR is

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

  then the LFSR sequence $\{a_m\}$ is periodic with a period that is the smallest integer k for which $f(x)$ divides $(1+x^k)$.
- An *irreducible polynomial f(x)* satisfying the following two conditions is a *primitive polynomial*:
  - It has an odd number of terms including the 1 term.
  - If its degree n is greater than 3, then $f(x)$ must divide $(1 + x^k)$, where $k = 2^n - 1$

Properties of m-sequences

1. The period of $\{a_n\}$ is $p=2^n-1$, that is, $a_{p+l} = a_i$, for all $i \geq 0$.
2. Starting from any nonzero state, the LFSR that generates $\{a_n\}$ goes through all 2^n-1 states before repeating.
3. The number of 1's differs from the number of 0's by one.
4. If a window of width n is slid along an m-sequence, then each of the 2^n-1 nonzero binary n-tuples is seen exactly once in a period.
5. In every period of an *m*-sequence, one-half the runs have length 1, one-fourth have length 2, one-eighth have length 3, and so on.

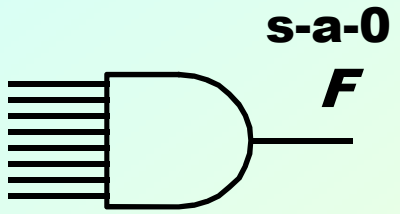
Randomness Properties of m-sequence

- **m-sequences generated by LFSRs are called pseudo random sequence.**
  - **The autocorrelation of any output bit is very close to zero.**
  - **The correlation of any two output bits is very close to zero.**

LFSR as Pseudo-Random Pattern Generator

- Standard LFSR
  - Produces patterns algorithmically – repeatable.
  - Has most of desirable randomness properties.
- Need not cover all 2^n input combinations.
- Long sequences needed for good fault coverage.

Weighted Pseudo-Random Pattern Generation



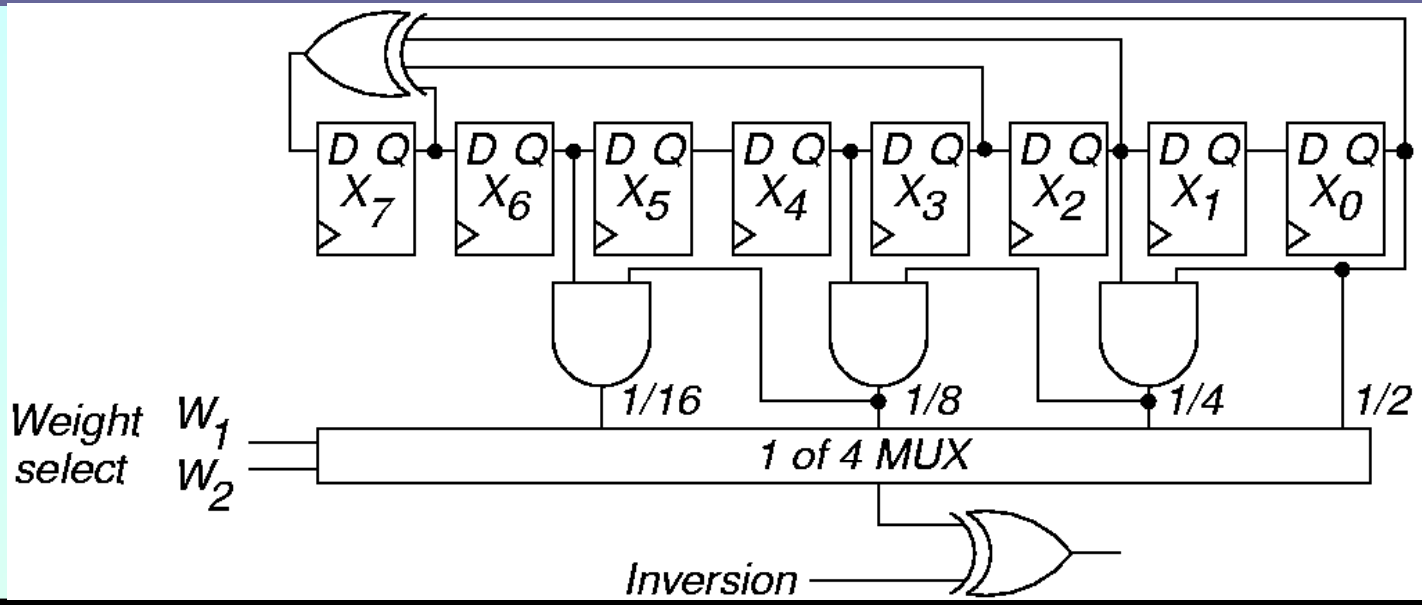
- If $p(1)$ at all PIs is 0.5, $p_F(1) = 0.5^8 = \frac{1}{256}$

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

- Will need enormous # of random patterns to test a stuck-at 0 fault on *F*.
- We must not use an ordinary LFSR to test this.
- IBM holds patents on weighted pseudo-random pattern generator in ATE.

- 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 Generator



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

How to compute weights?

- Assume p(1) of primary output(s) to be 0.5.
- Systematically backtrace and compute the p(1) values of all other lines.
- Finally obtain the p(1) values of the primary input lines.

Cellular Automata (CA)

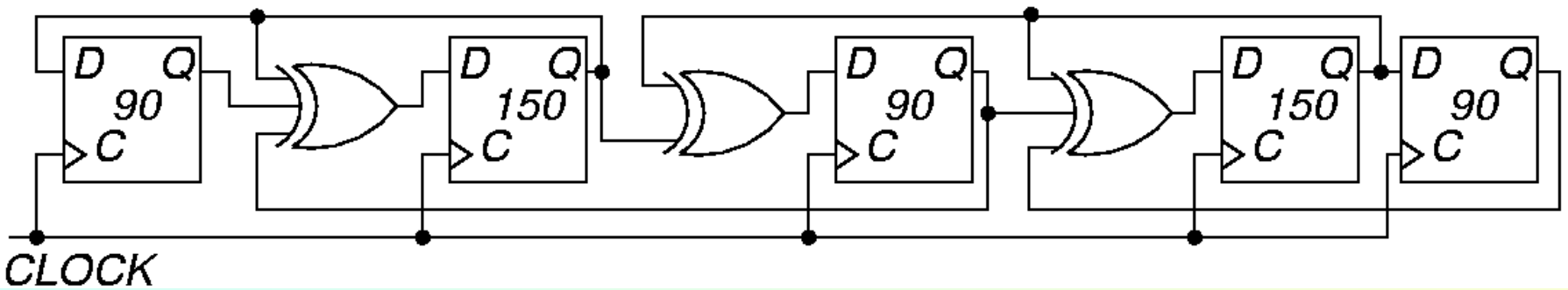
- Superior to LFSR – even "more" random
  - No shift-induced bit value correlation
  - Can make LFSR more random with linear phase shifter
- Regular connections – each cell only connects to local neighbors

	$x_{c-1}(t)$ $x_c(t)$ $x_{c+1}(t)$ Gives CA cell connections							
	111	110	101	100	011	010	001	000
$x_c(t+1)$	0	1	0	1	1	0	1	0

$2^6 + 2^4 + 2^3 + 2^1 = 90$ Called *Rule 90*

$x_c(t+1) = x_{c-1}(t) \oplus x_{c+1}(t)$

Cellular Automata Example



- Five-stage hybrid cellular automaton
- Rule 150: $x_c(t+1) = x_{c-1}(t) \oplus x_c(t) \oplus x_{c+1}(t)$
- Alternate Rule 90 and Rule 150 CA

Test Pattern Augmentation

- Secondary ROM – to get LFSR to 100% stuck-at fault coverage.
  - Add a small ROM with missing test patterns.
  - Add extra circuit mode to input MUX – shift to ROM patterns after LFSR done.
  - LFSR reseeding is another alternative.
- 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.

Test Response Compaction

Response Compaction

- Huge volume of data in CUT response:
  - An 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 circuits match.
- **Compaction**
  - Drastically reduce # bits in original circuit response.
  - Loss of 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
- **Ones Count (Syndrome) Compaction.**
  - Count # of 1's
- **Transition Count Response Compaction**
  - Count # of transitions from 0 → 1 and 1 → 0 as a signature.

BIST - Response Compression

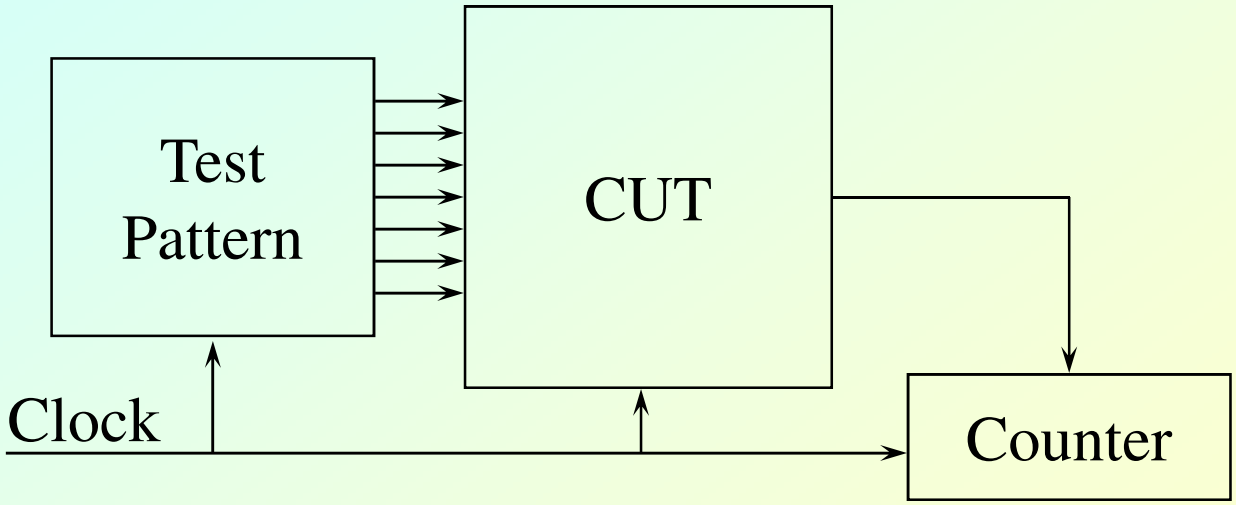
- Introduction
- Ones-Count Compression
- Transition-Count Compression
- Syndrome-Count Compression
- Signature Analysis
- Space Compression

Some Points

- Bit-to-bit comparison is infeasible for BIST.
- General principle:
  - Compress a very long output sequence into a single signature.
  - Compare the compressed word with the prestored *golden signature* to determine the correctness of the circuit.
- Problem of aliasing:
  - Many output sequences may have the same signature after the compression.
- Poor diagnosis resolution after compression.

Ones-Count - Hardware

- Apply predetermined patterns.
- Count the number of ones in the output sequence.



Ones Counter - Aliasing

- **Aliasing Probability**

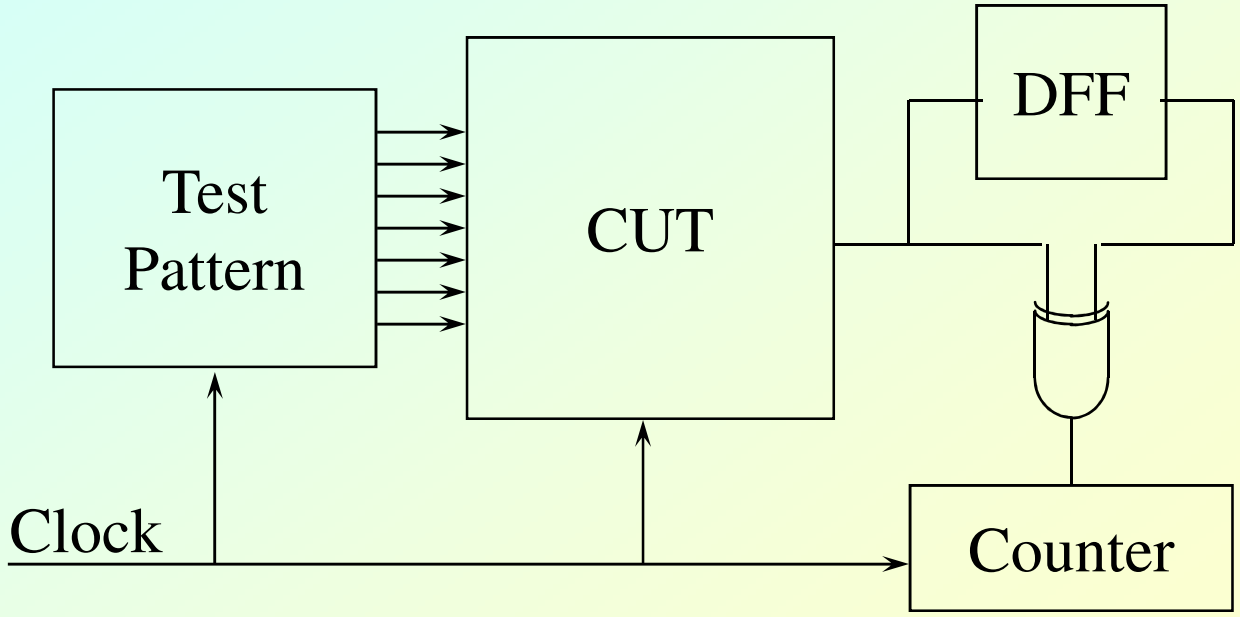
$$P_{OC} = \frac{\begin{bmatrix} m \\ r \end{bmatrix} - 1}{2^m - 1} \cong (\pi m)^{1/2}$$

m : the test length
r : the number of ones

- *r*=*m*/2 :: the case with the highest aliasing prob.
- *r*=*m* and *r*=*0* :: no aliasing probability
- For combinational circuits, the input sequence can be permuted without changing the count.

Transition Count - Hardware

- Apply predetermined patterns
- Count the number of the transitions (0 →1 and 1 → 0).



Transition Count

- **Aliasing Probability**

$$P_{TC} = \frac{2\left[\begin{smallmatrix} m-1 \\ r \end{smallmatrix}\right] - 1}{2^m - 1} \cong (\pi m)^{1/2}$$

m : the test length
r : the number of transitions

- *r=m/2* :: highest aliasing probability
- *r=0* and *r=m* :: no aliasing probability

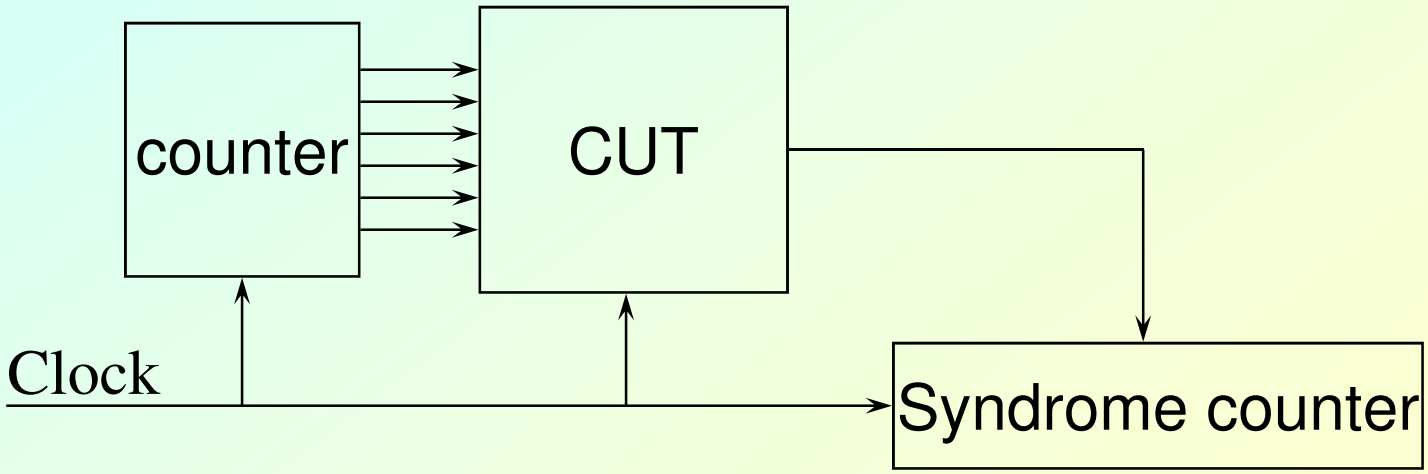
- **Transition count:**

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

- **To maximize fault coverage:**
  - **Make $C(R0)$ – good machine transition count – as large or as small as possible**

Syndrome Testing

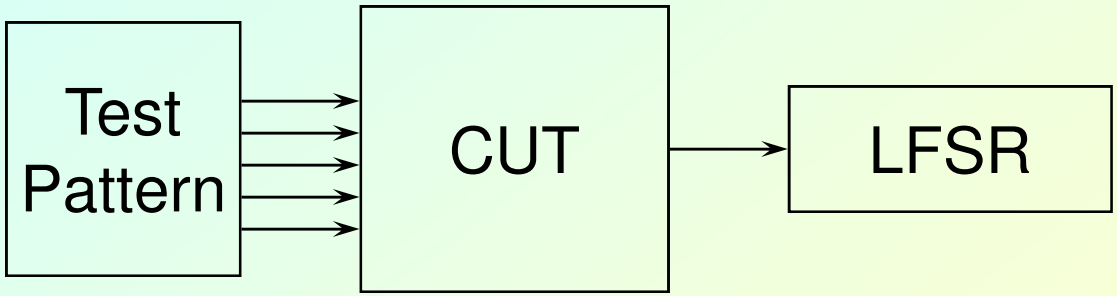
- Apply exhaustive test patterns.
- Count the number of 1's in the output.
- Normalize by dividing with number of minterms.



Analysis of Syndrome Testing

Signature Analysis

- Apply predetermined test patterns.
- Compress the output sequence by LFSR.
  - Compressed value is called *signature*.



Signature Analysis

- **Aliasing Probability**

$$P_{SA} = \frac{2^{m-n} - 1}{2^m - 1} \cong 2^{-n}$$

m: **test length,** ***n***: **length of LFSR**

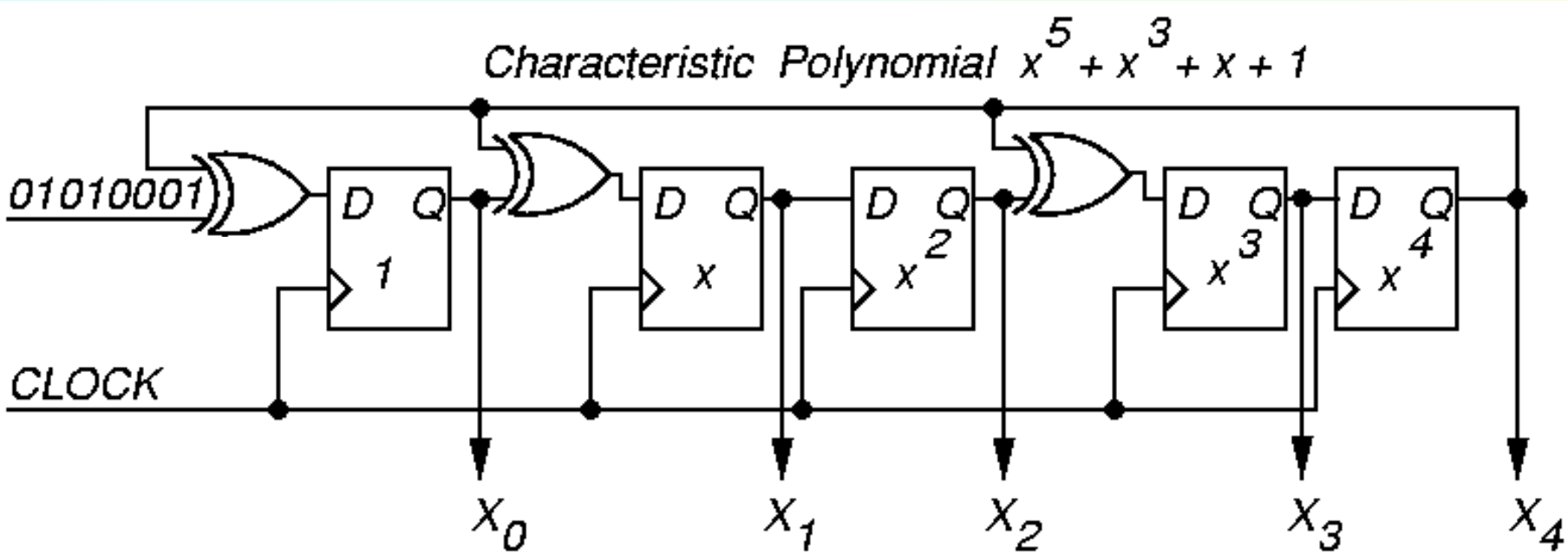
- **Aliasing probability is output independent.**
- **An LFSR with two or more nonzero coefficients detect any single faults.**
- **An LFSR with primitive polynomial detect any double faults separated less than 2^n-1.**

LFSR Based Response Compaction

LFSR for Response Compaction

- Use LFSR based CRC generator as response compacter.
- Treat data bits from circuit POs to be compacted as a decreasing order coefficient polynomial.
- CRC 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

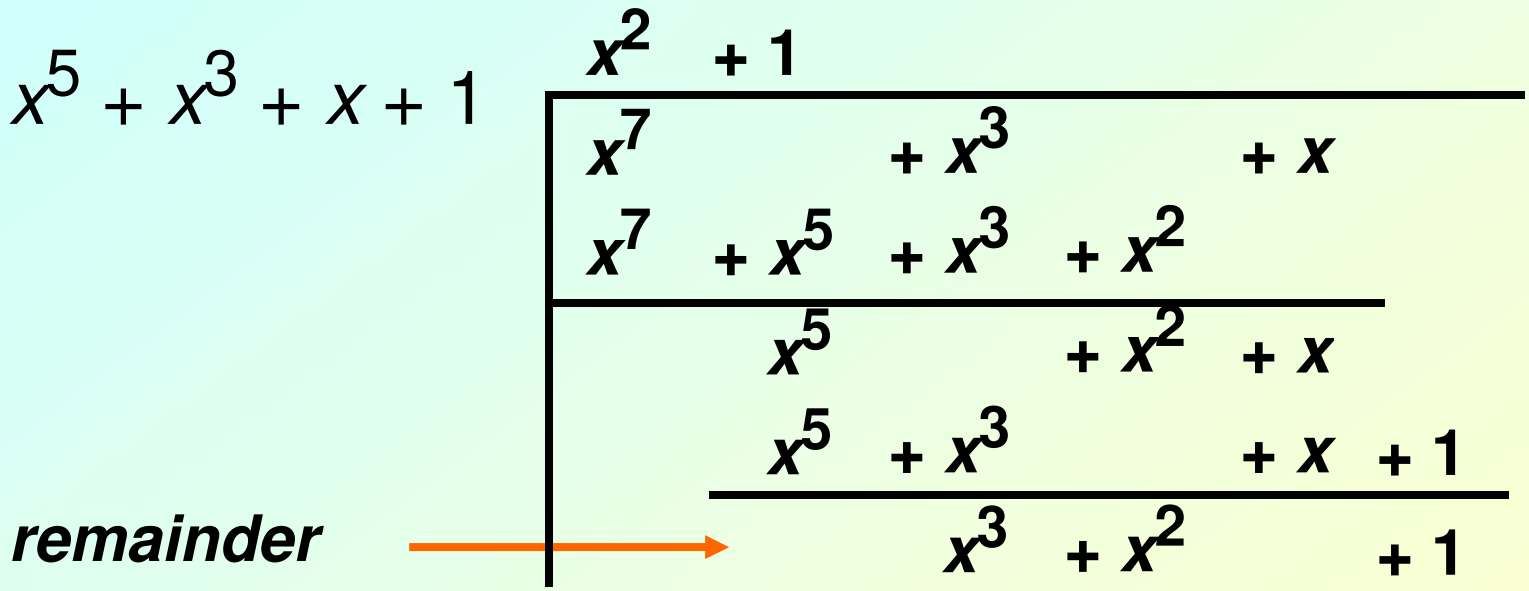
	Inputs	X^0	X^1	X^2	X^3	X^4
	Initial State	0	0	0	0	0
	1	1	0	0	0	0
	0	0	1	0	0	0
Logic	0	0	0	1	0	0
Simulation:	0	0	0	0	1	0
	1	1	0	0	0	1
	0	1	0	0	1	0
	1	1	1	0	0	1
	0	1	0	1	1	0

Logic simulation: ***Remainder*** $= 1 + x^2 + x^3$

0 1 0 1 0 0 0 1

Input polynomial: $x^1 + x^3 + x^7$

Symbolic Polynomial Division

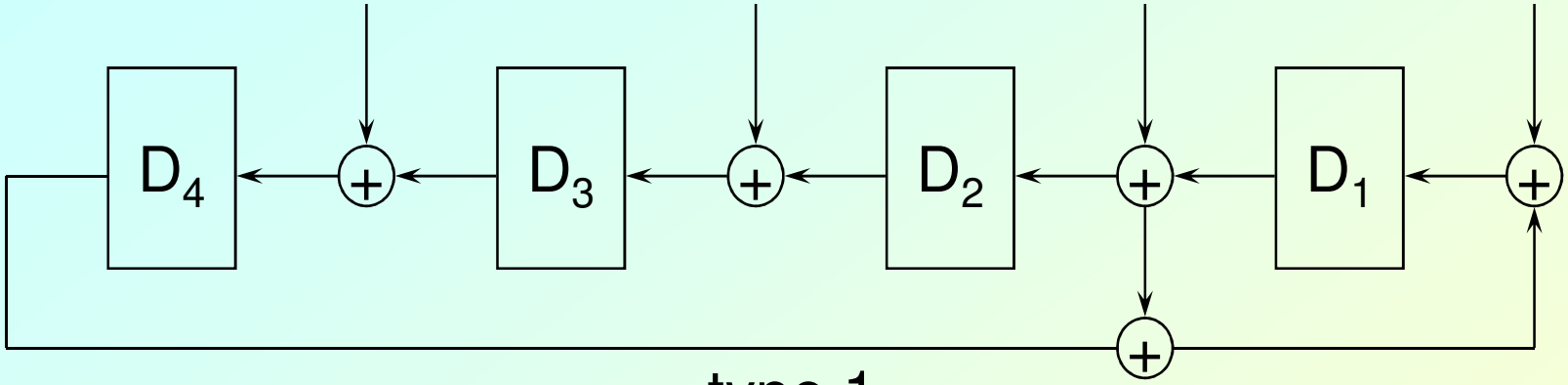


Remainder matches that from logic simulation of the response compacter!

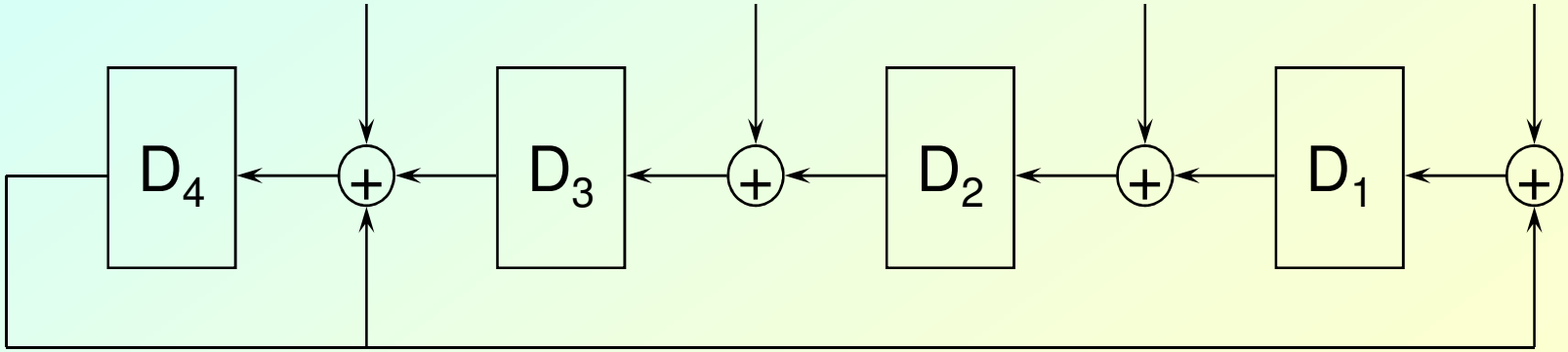
Multiple-Input Signature Register (MISR)

- 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

Multiple Input Signature Register (MISR)



type 1



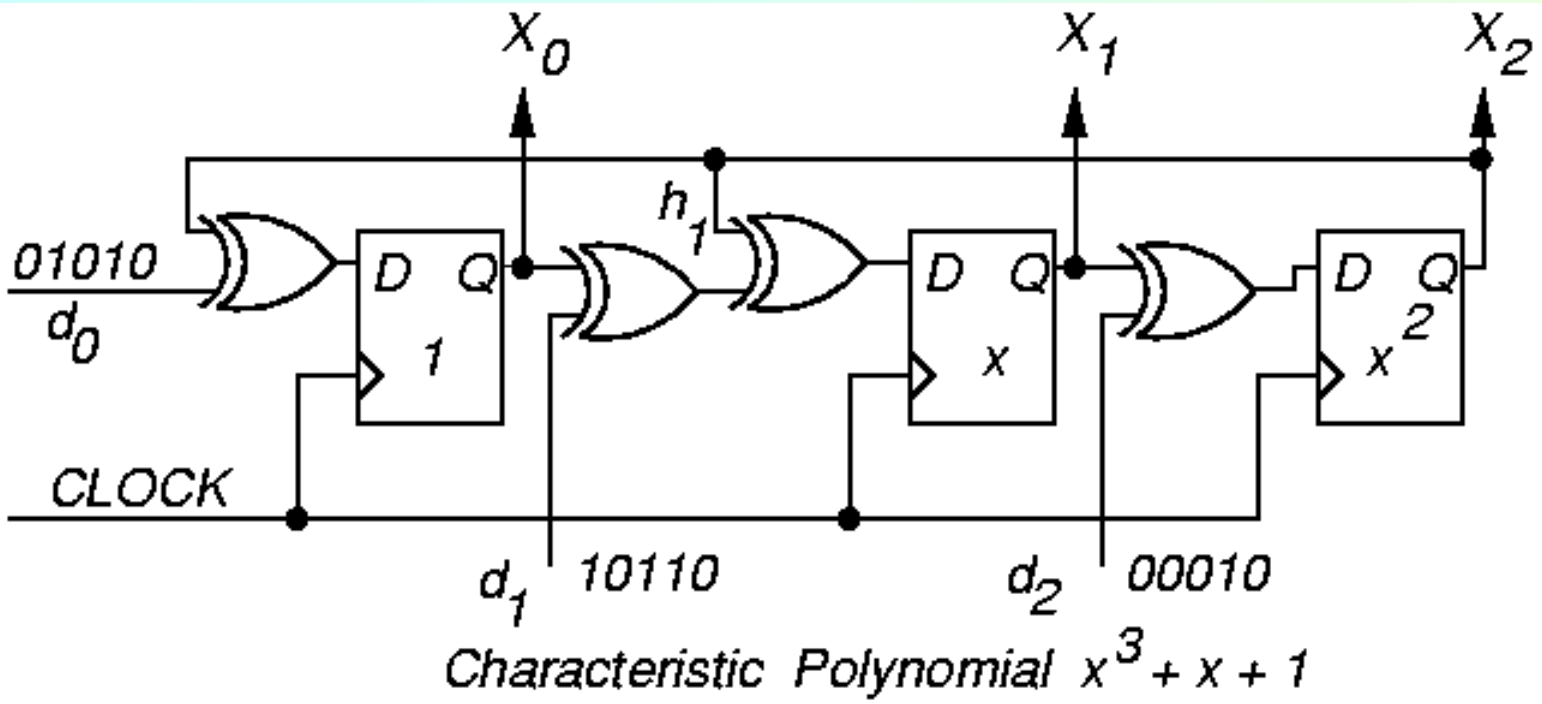
type 2

MISR Matrix Equation

- $d_i\,(t)$ – output response on PO_i at time t

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

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

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