EEDG/CE 6303: Testing and Testable Design

Mehrdad Nourani

Dept. of ECE
Univ. of Texas at Dallas

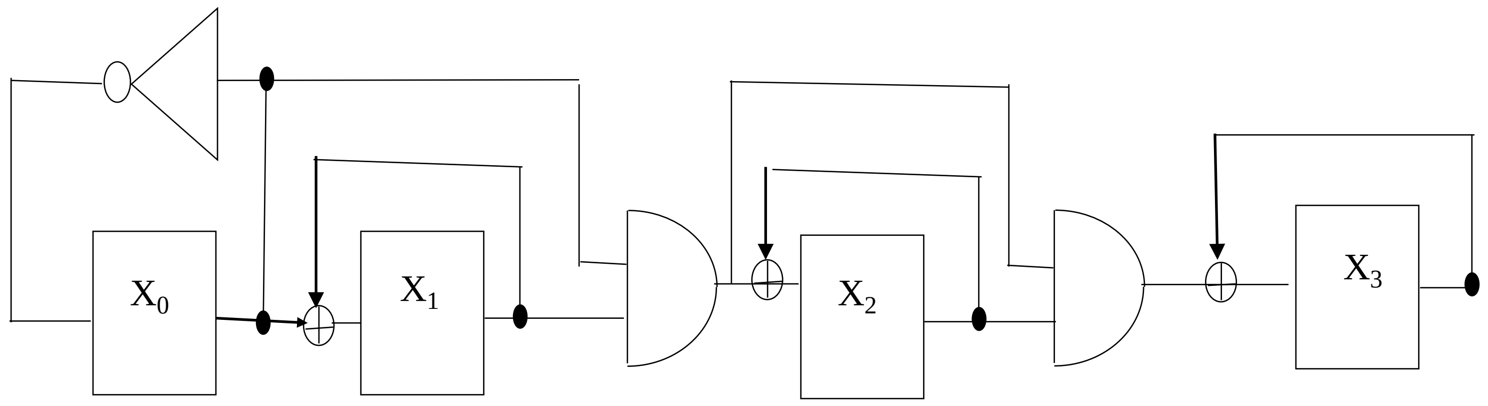
Session 10

Built-In Self-Test (BIST)

Various LFSR Architectures

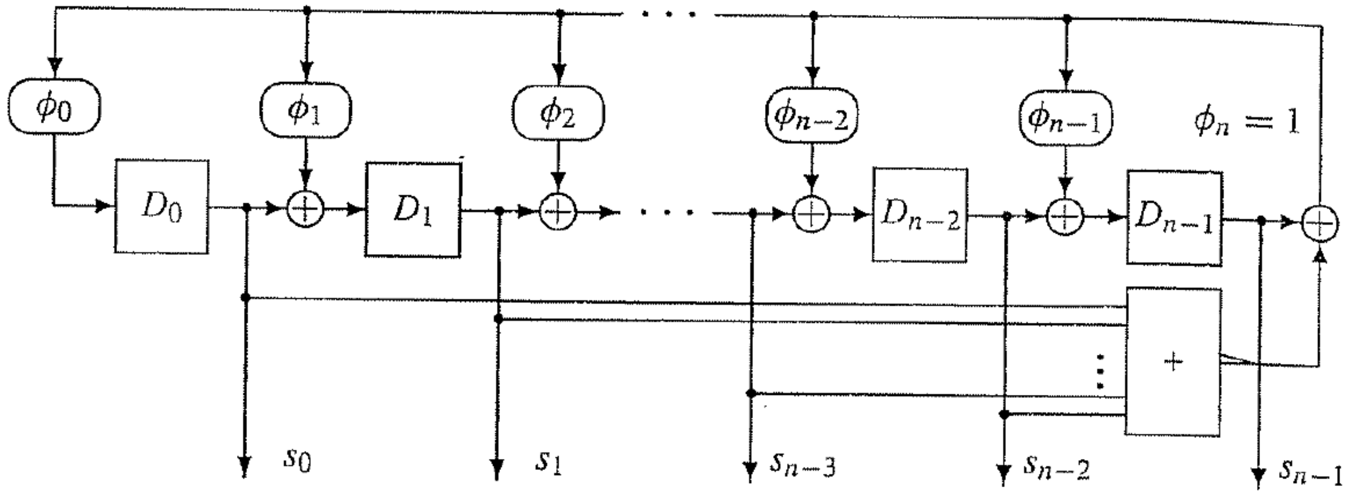
Exhaustive-Pattern LFSR

- Binary counter can be used to generate all 2^n patterns
- Example: 4-bit binary counter



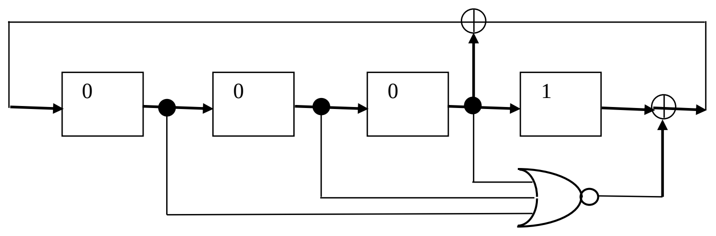
Complete LFSR

- We can easily modify the LFSR to generate the complete sequence that includes all-zero pattern
  - When LFSR has 0…001, the output of NOR is 1 and the last XOR injects a 0 . So, the next state will be all zero 0…000.
  - When LFSR has 0…000, the output of NOR is 1 again but the last XOR injects a 1. So, the next state will be out of all-zero state.

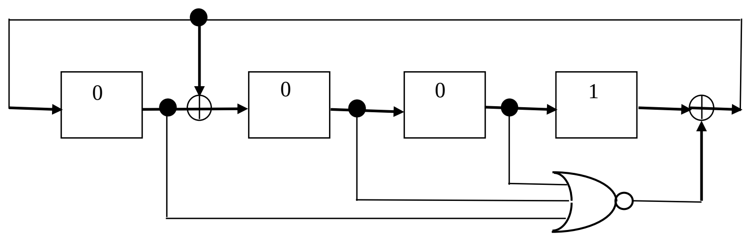


Examples of Complete LFSR

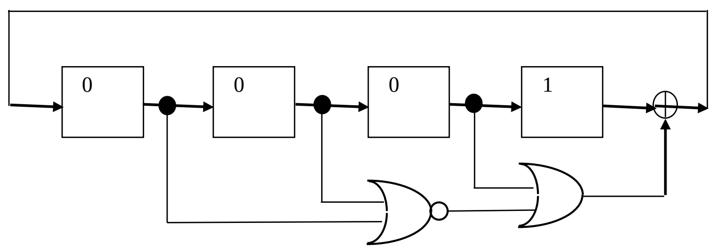
- Further optimization would be possible by optimizing NOR and XOR gates together.



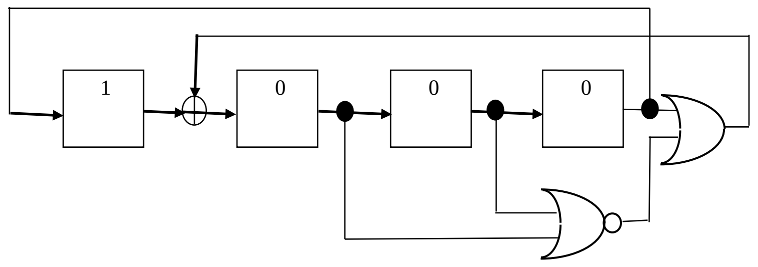
(a) 4-stage standard CFSR



(b) 4-stage modular CFSR



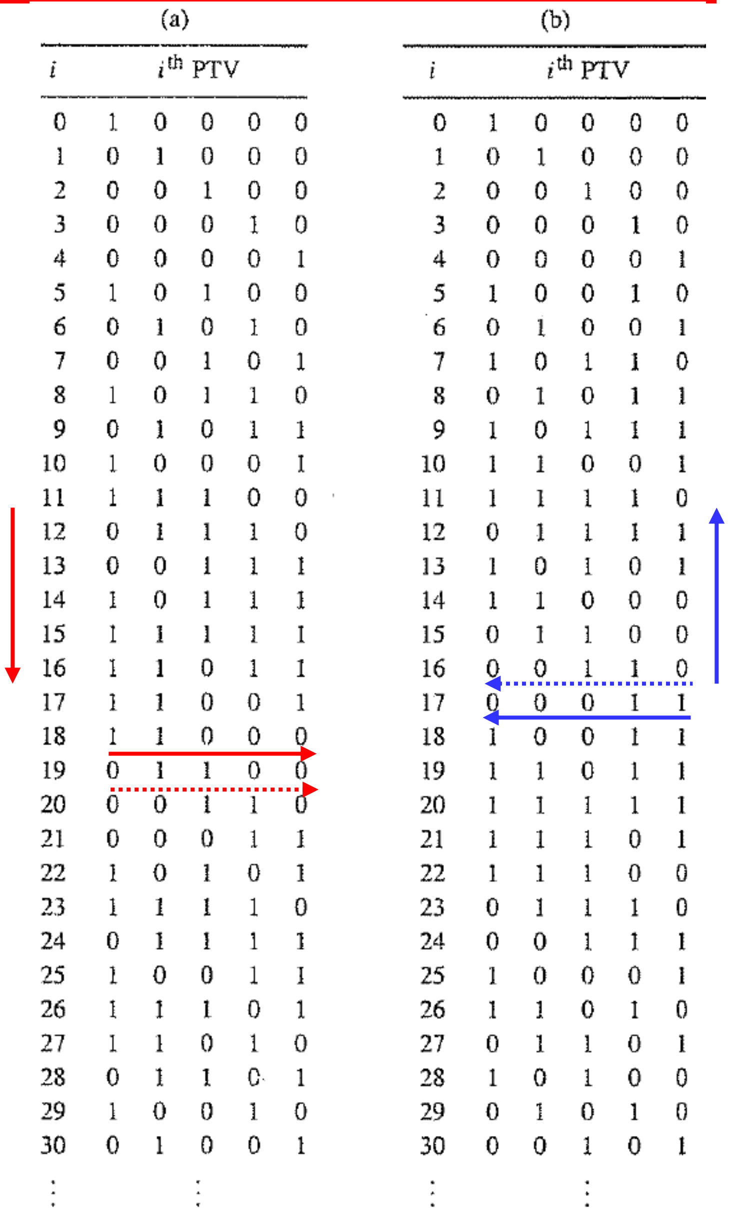
(c) A minimized version of (a)



(d) A minimized version of (b)

Reverse-Order Sequence

- In some test scenarios, a pair of test sequences are required where one contains vectors in the reverse order of the other.
- The reverse-order sequence can be generated using polynomial: $\varphi(x) = x^n \varphi(1/x)$
- Example: 5-stage internal XOR LFSR with
  - (a) $\varphi(x) = x^5 + x^2 + 1$
  - (b) $\varphi(x) = x^5 + x^3 + 1$



(a)

i	*i*th PTV				
0	1	0	0	0	0
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	0
4	0	0	0	0	1
5	1	0	1	0	0
6	0	1	0	1	0
7	0	0	1	0	1
8	1	0	1	1	0
9	0	1	0	1	1
10	1	0	0	0	1
11	1	1	1	0	0
12	0	1	1	1	0
13	0	0	1	1	1
14	1	0	1	1	1
15	1	1	1	1	1
16	1	1	0	1	1
17	1	1	0	0	1
18	1	1	0	0	0
19	0	1	1	0	0
20	0	0	1	1	0
21	0	0	0	1	1
22	1	0	1	0	1
23	1	1	1	1	0
24	0	1	1	1	1
25	1	0	0	1	1
26	1	1	1	0	1
27	1	1	0	1	0
28	0	1	1	0	1
29	1	0	0	1	0
30	0	1	0	0	1
⋮			⋮		

(b)

i	*i*th PTV				
0	1	0	0	0	0
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	0
4	0	0	0	0	1
5	1	0	0	1	0
6	0	1	0	0	1
7	1	0	1	1	0
8	0	1	0	1	1
9	1	0	1	1	1
10	1	1	0	0	1
11	1	1	1	1	0
12	0	1	1	1	1
13	1	0	1	0	1
14	1	1	0	0	0
15	0	1	1	0	0
16	0	0	1	1	0
17	0	0	0	1	1
18	1	0	0	1	1
19	1	1	0	1	1
20	1	1	1	1	1
21	1	1	1	0	1
22	1	1	1	0	0
23	0	1	1	1	0
24	0	0	1	1	1
25	1	0	0	0	1
26	1	1	0	1	0
27	0	1	1	0	1
28	1	0	1	0	0
29	0	1	0	1	0
30	0	0	1	0	1
⋮			⋮		

Hybrid LFSR

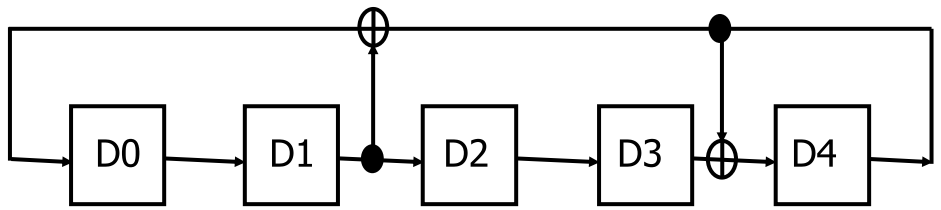
- Polynomial $f(x)=1+b(x)+c(x)$ is fully decomposable iff both $b(x)$ and $c(x)$ have no common terms and there exists an integer j such that $c(x)=x^j b(x)$, $j>=1$
- If $f(x)$ is fully decomposable, then we have *$f(x)=1+b(x)+x^j b(x)$*
- A hybrid (top-bottom) LFSR can be constructed using polynomial *$s(x)=1+\hat{}x^j+x^j b(x)$*
  - The term $\hat{}x^j$ indicates the XOR gate with one input is connected to the feedback path, not between stages

5-Stage Hybrid LFSRs

- Compared to a standard LFSR with m XOR, a hybrid LFSR can be realized using (m+1)/2 XORs and can achieve full-length 2^n-1 cycle.
  - Details in: L. Wang and E. McCluskey, “Hybrid Design Generating Maximum-Length Sequences,”, Trans. CAD, 1988.
- (a) 5-stage top-bottom LFSR

Internal XOR part:
$\varphi(x)=1+x^4+x^5$

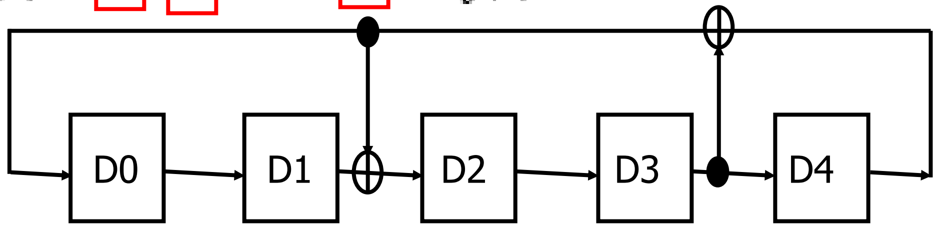
$$s(x) = \boxed{1} + {}^\wedge x^2 + \boxed{x^4} + \boxed{x^5} \text{ for } f(x) = 1+x^2+x^3+x^4+x^5$$



- (b) 5-stage bottom-top LFSR

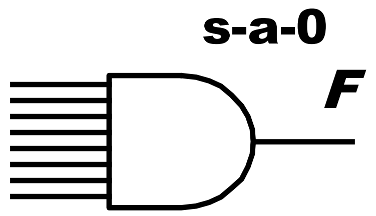
Internal XOR part:
$\varphi(x)=1+x^2+x^5$

$$s(x) = \boxed{1} + \boxed{x^2} + {}^\wedge x^4 + \boxed{x^5} \text{ for } f(x) = 1+x+x^2+x^3+x^5$$



Weighted Pseudo-Random Pattern

- Consider an AND gate with large fan-in:

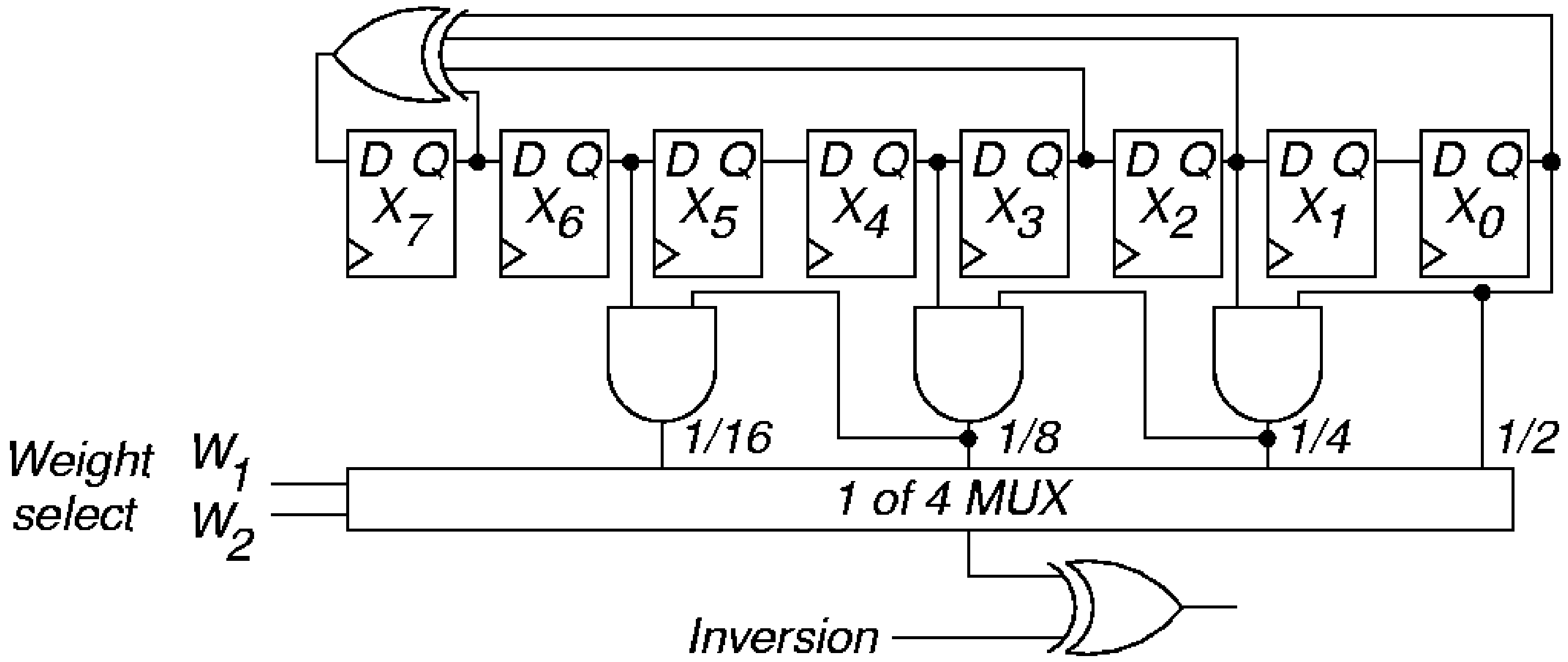


- If p(1) at all PIs is 0.5, p_F(1) = 0.5^8 = 1/256 and p_F(0) = 1-(1/256) = 255/256
- Will need enormous # of random patterns to test a stuck-at 0 fault on F -- LFSR p(1) = 0.5
  - We must not use an ordinary LFSR to test this
- IBM – holds patents on weighted pseudo-random pattern generator in ATE

Weighted LFSR

- For a regular LFSR $p(1) = 0.5$
- Solution to get a weighted LFSR
  - —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	1/2	1	0	0	1/8
0	0	1	1/2	1	0	1	7/8
0	1	0	1/4	1	1	0	1/16
0	1	1	3/4	1	1	1	15/16

Response Compaction

Motivation for Test Data Compaction

- Huge amount 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

Definitions

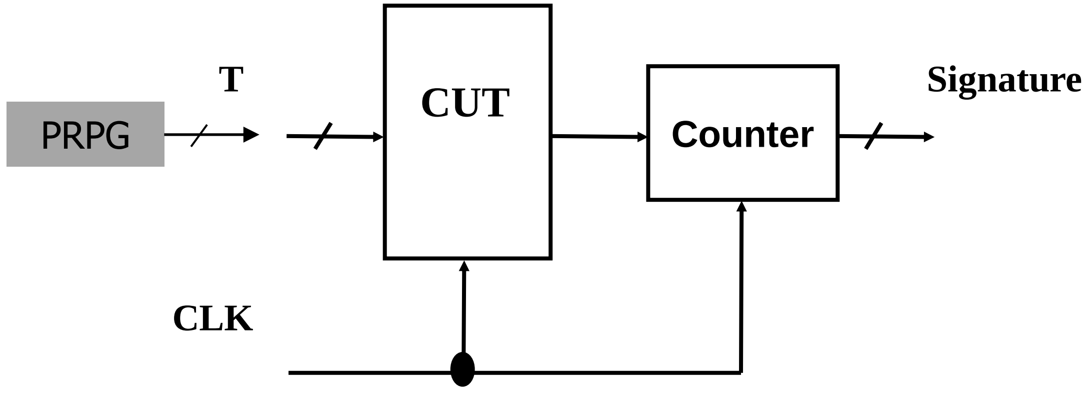
- ***Aliasing*** – Due to information loss, signatures of good and some bad machines match
- ***Compaction*** – Drastically reduce # bits in original circuit response – information is lost
- ***Compression*** – Reduce # bits in original circuit response – no information loss – fully invertible (can get back original response)
- ***Signature analysis*** – Compact good machine response into *good machine signature*. Actual signature generated during testing, and compared with good machine signature
- ***Transition Count Response Compaction*** – Count # transitions from $0 \rightarrow 1$ and $1 \rightarrow 0$ as a signature

Response Analysis Mechanisms

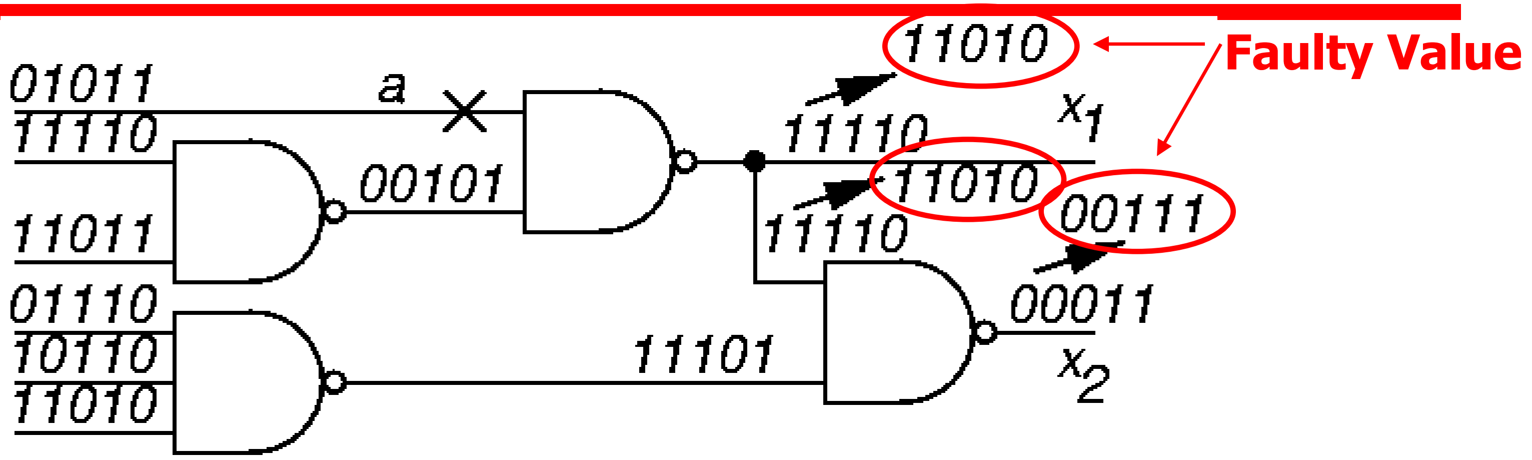
- Ones count testing
- Transition count testing
- LFSR-based signature compaction/analysis
  - Serial
  - parallel

Ones Counting

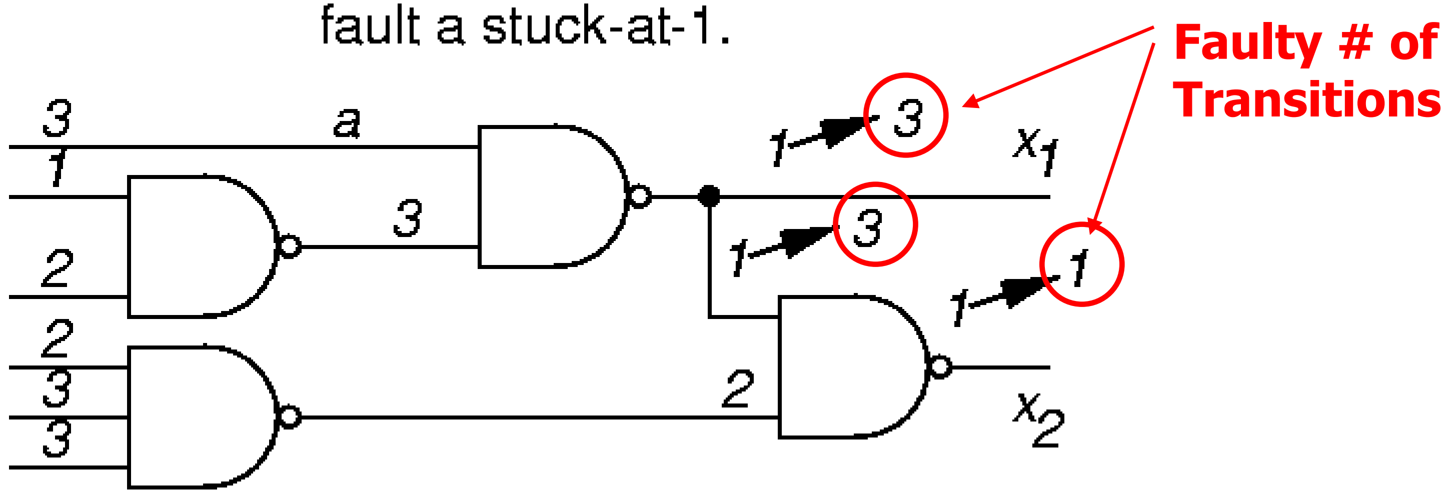
- Ones count testing will need a counter to count the number of 1s in the bit stream.
- Assume the CUT has one output and the output contains a stream of *L* bits. Let the fault-free output response be $\{r_0, r_1, \ldots, r_{L-1}\}$.
- Aliasing probability [Savir 1985] for L bit stream when the fault-free response should have m 1s
  - $P_{OC}(m) = [C(L,m)-1]/(2^L-1)$



Transition Counting



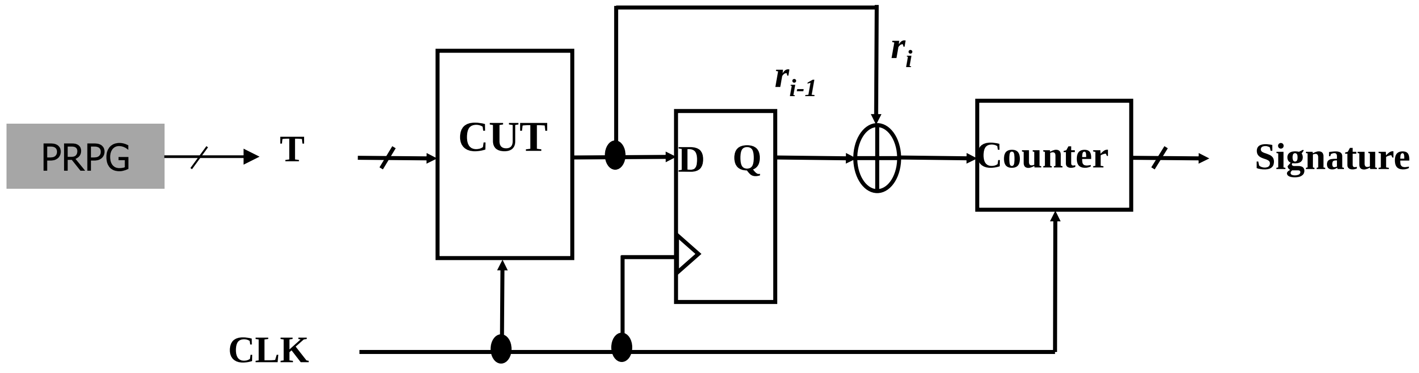
(a) Logic simulation of good machine and fault a stuck-at-1.



(b) Transition counts of good and failing machines.

Transition Counting Details

- Transition count when L bit test sequence is applied to CUT: $C(R) = \Sigma_{i=1 \text{ to } L-1} (r_i \oplus r_{i-1})$
- To maximize fault coverage:
  - Make $C(R_{fault\text{-}free})$ – good machine transition count – as large or as small as possible
- Aliasing probability [Hays 1976] for L bit stream when the fault-free response should have m transitions
  - $P_{TC}(m)=[2C(L-1,m)-1]/(2^L-1)$



Problems with OC and TC Methods

- The aliasing probability in both OC and TC methods depends on the fault-free response
- The aliasing probability is minimum (or maximum) when the fault-free response is in its minimum (or maximum) value of its respective parameter
  - P_{TC} is maximum when #of transitions $m \approx L/2$
  - P_{TC} is minimum when #of transitions $m = 0$ or $L-1$

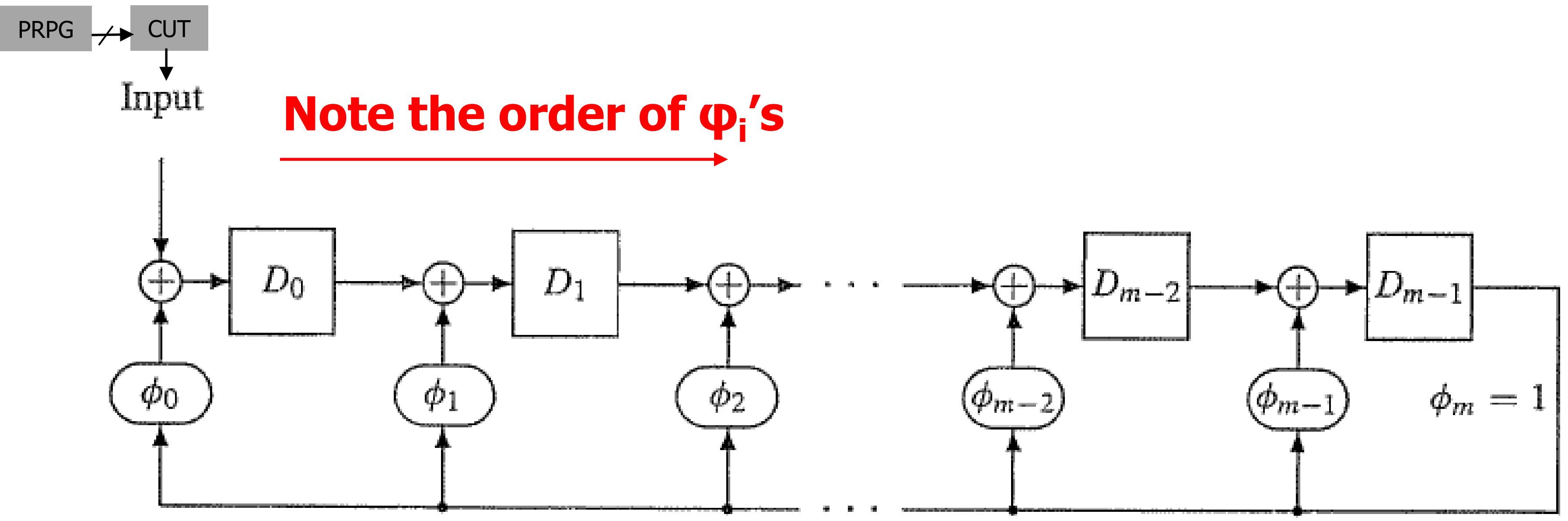
LFSR for Response Compaction

- Use *cyclic redundancy check code* (CRCC) generator (LFSR) for response compacter
- Treat data bits from circuit POs to be compacted as a decreasing order coefficient polynomial
- CRCC 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

Serial LFSR Compaction

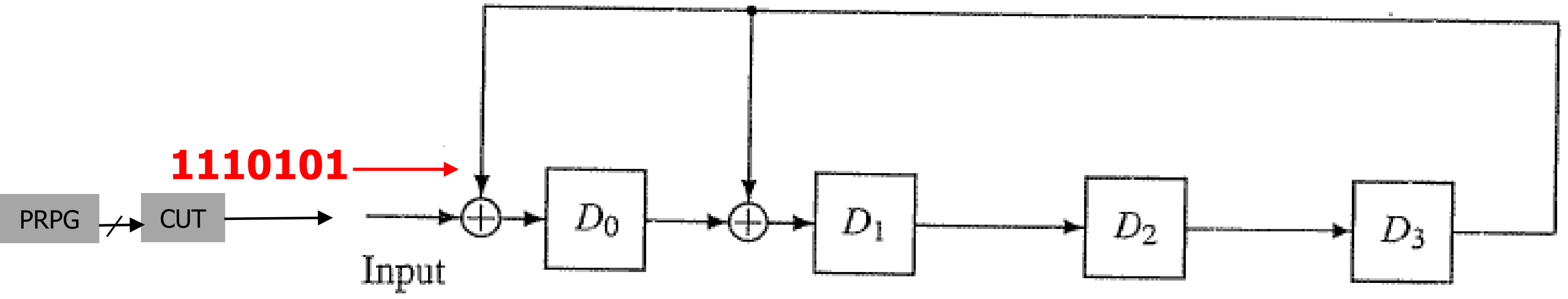
Serial LFSR Compactor

- The polynomial $\varphi(x) = \varphi_m x^m + \ldots + \varphi_1 x + \varphi_0$
- The polynomial relationship
  - L-bit sequence coming to compactor: $m_0 m_1 m_2 \ldots m_{L-1}$($m_{L-1}$ is the first bit arriving at compactor)
  - The input polynomial: $M(x)=m_0+m_1x+m_2x^2+\ldots+ m_{L-1}x^{L-1}$
  - $M(x)=q(x)\ \varphi(x)+r(x)$ where $r(x)$ the polynomial remainder of $M(x)/\varphi(x)$ will be the final compactor's response



LFSR Compactor Example I

- The polynomial: $\varphi(x) = x^4 + x + 1$
- The sequence: 1110101 that is $M(x)=1+x+x^2+x^4+x^6$
- Initial state: 0000



Polynomial Division Method

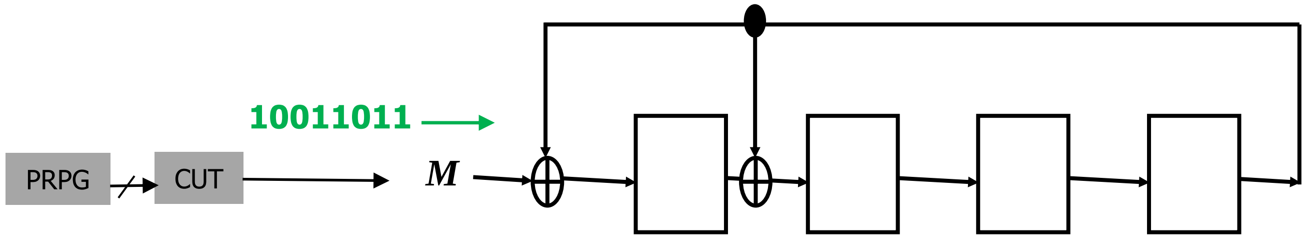
$$
\begin{array}{r|l}
 & x^2 + 1 \quad \leftarrow \text{Quotient} \\
\hline
x^4 + x + 1 & x^6 + x^4 + x^2 + x + 1 \\
 & x^6 + x^3 + x^2 \\
\hline
 & x^4 + x^3 + x + 1 \\
 & x^4 + x + 1 \\
\hline
\text{Signature} \rightarrow & \boxed{x^3} \\
\hline
\end{array}
$$

Final signature (remainder):
$D_0D_1D_2D_3$ = 0001

Tabular Method

Time	Input	LFSR state			
–	–	0	0	0	0
0	$r_1 = 1$	1	0	0	0
1	$r_2 = 0$	0	1	0	0
2	$r_3 = 1$	1	0	1	0
3	$r_4 = 0$	0	1	0	1
4	$r_5 = 1$	0	1	1	0
5	$r_6 = 1$	1	0	1	1
6	$r_7 = 1$	0	0	0	1

LFSR Compactor Example I (cont.)



Fault Free Sequence

M	r_0	r_1	r_2	r_3
1	0	0	0	0
1	1	0	0	0
0	1	1	0	0
1	0	1	1	0
1	1	0	1	1
0	0	0	0	1
0	1	1	0	0
1	0	1	1	0
R	1	0	1	1

(a) Fault-free signature

Faulty Sequence

M'	r_0	r_1	r_2	r_3
1	0	0	0	0
1	1	0	0	0
0	1	1	0	0
1	0	1	1	0
0	1	0	1	1
0	1	0	0	1
1	1	0	0	0
1	1	1	0	0
R'	1	1	1	0

Will be detected

(b) Signature for fault f_1

Faulty Sequence

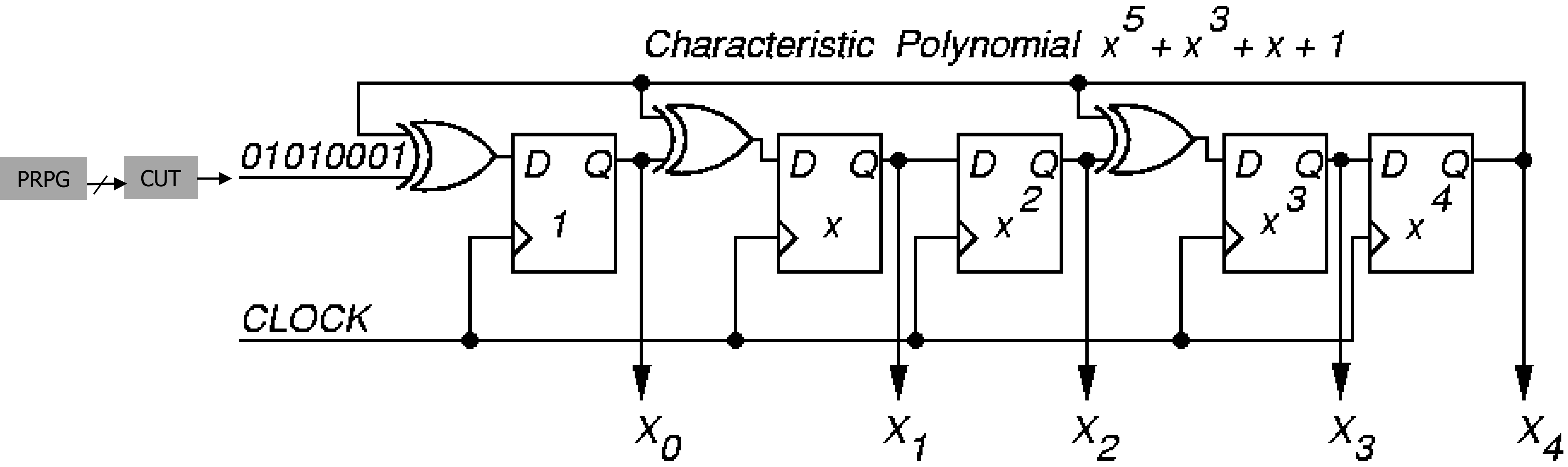
M''	r_0	r_1	r_2	r_3
1	0	0	0	0
0	1	0	0	0
1	0	1	0	0
1	1	0	1	0
0	1	1	0	1
0	1	0	1	0
1	0	1	0	1
1	0	1	1	0
R''	1	0	1	1

Cannot be detected

(c) Signature for fault f_2

LFSR Compactor Example II

- LFSR seed value is "00000"
- Input sequence: 0 1 0 1 0 0 0 1
- Symbolic polynomial: $0x^0+1x^1+0x^2+1x^3+0x^4+0x^5+0x^6+1x^7$
- Logic simulation: *Remainder* = $1 + x^2 + x^3$ (i.e. 10110)



LFSR Compactor Example II (cont.)

$$\begin{array}{r|l}
 & x^2 + 1 \\ \cline{2-2}
x^5 + x^3 + x + 1 & x^7 + x^3 + x \\
 & x^7 + x^5 + x^3 + x^2 \\ \cline{2-2}
 & x^5 + x^2 + x \\
 & x^5 + x^3 + x + 1 \\ \cline{2-2}
\textit{remainder} \longrightarrow & x^3 + x^2 + 1
\end{array}$$

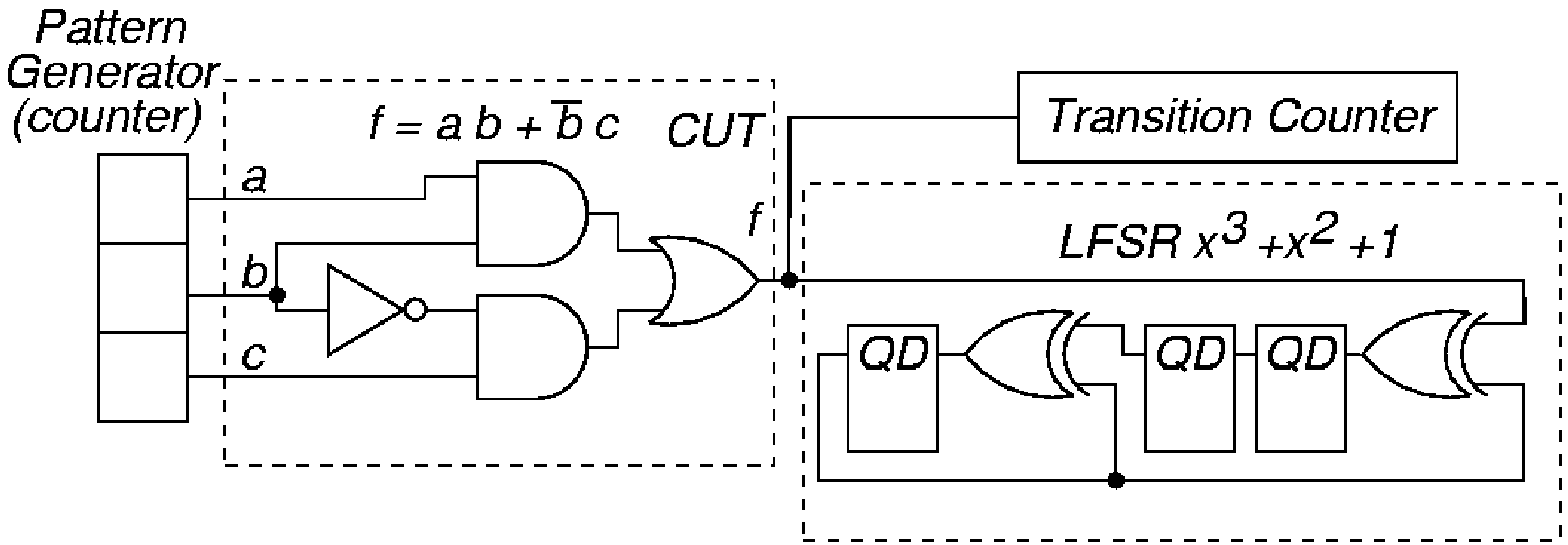
Logic Simulation:

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
0	0	0	1	0	0
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* $= 1 + x^2 + x^3$

LFSR Compactor Example III

- Suppose we use a 3-bit exhaustive binary counter for pattern generator.
- Faults may be masked here too.



LFSR Compactor Example III (cont.)

- LFSR aliases for *f* sa1, transition counter for *a* sa1

Pattern	***Responses***			
abc	***Good***	***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
LFSR	001	101	001	010

Probability of Aliasing

- Consider polynomial operations (equivalent to mod 2 where multiplication and addition are the same as AND and XOR.
  - Fault-free circuit response: $M^*(x)=q^*(x)\,\varphi(x)+r^*(x)$
  - Faulty circuit response: $M(x)=q(x)\,\varphi(x)+r(x)$
- Aliasing occurs when $r^*(x)=r(x)$ while $M^*(x)\neq M(x)$
  - $M^*(x)+M(x)=[q^*(x)+q(x)]\,\varphi(x)+(r^*(x)+r(x)]$
    $=[q^*(x)+q(x)]\,\varphi(x)$
  - Aliasing occurs when the error polynomial $E(x)=M^*(x)+M(x)$ (i.e. a bit-by-bit XOR of fault-free and faulty responses) is divisible by the LFSR feedback polynomial $\varphi(x)$.
  - Aliasing is independent of the exact value of fault-free response.

Probability of Aliasing (cont.)

- The maximum degree of $E(x)=M^*(x)+M(x)$ is $L-1$ (when L bit input streams is compacted).
- The maximum degree of $\varphi(x)$ is m
- So, the maximum degree of $[q^*(x)+q(x)]$ will be $L-1-m$.
- For a given feedback polynomial $\varphi(x)$, any polynomial of degree $\leq L-1-m$ can be used to obtain a fault-free signature.
- There exist 2^{L-m} polynomials of degree$\leq L-1-m$. If we remove all-zero polynomial, total of $2^{L-m}-1$ cases can cause aliasing.
  - $P_{serial}(m)=[2^{(L-m)}-1]/(2^L - 1) \approx 2^{-m}$

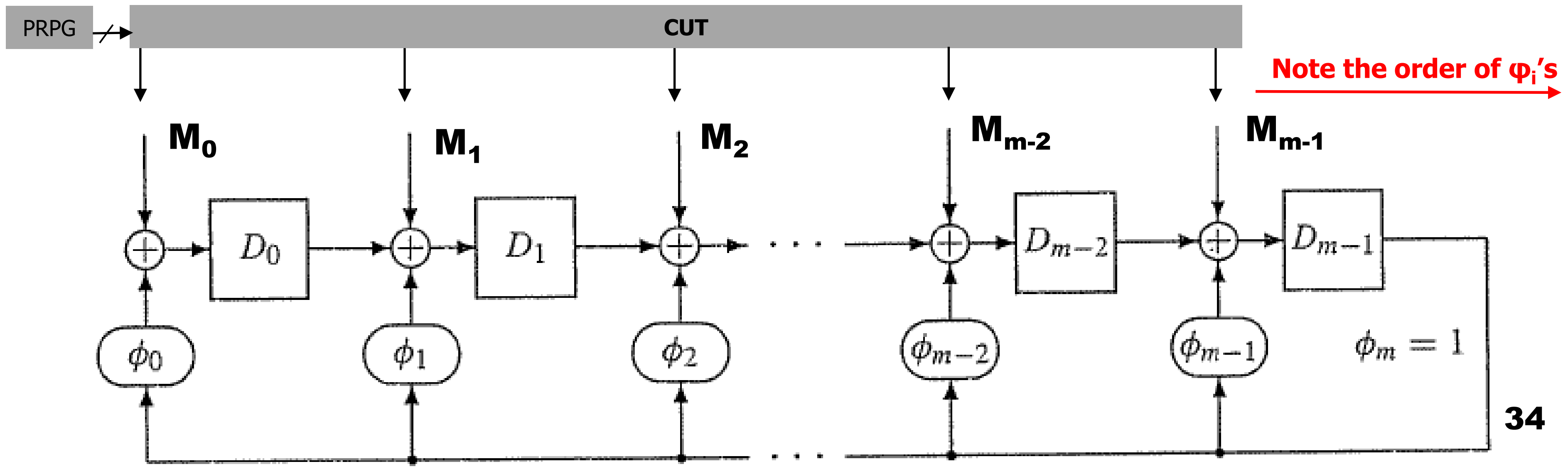
Parallel LFSR Compaction (MISR)

Multiple-Input Signature Register (MISR)

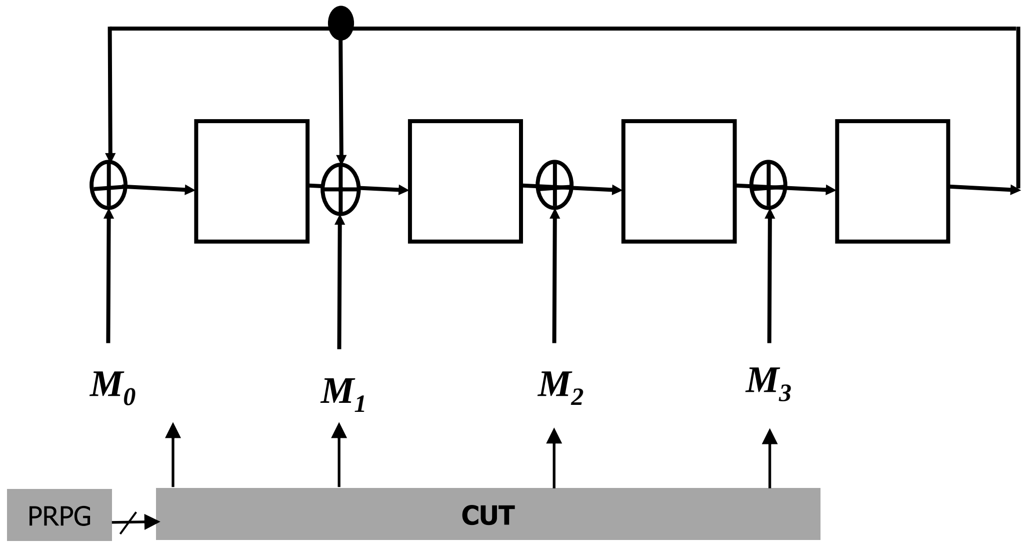
- Problem with ordinary LFSR response compacter:
  - Too much hardware if one of these is put on each *primary output* (PO)
- Solution: MISR – compacts all outputs into one LFSR
  - Works because LFSR is linear – obeys *superposition principle*
  - Superimpose all responses in one LFSR – final remainder is XOR sum of remainders of polynomial divisions of each PO by the characteristic polynomial

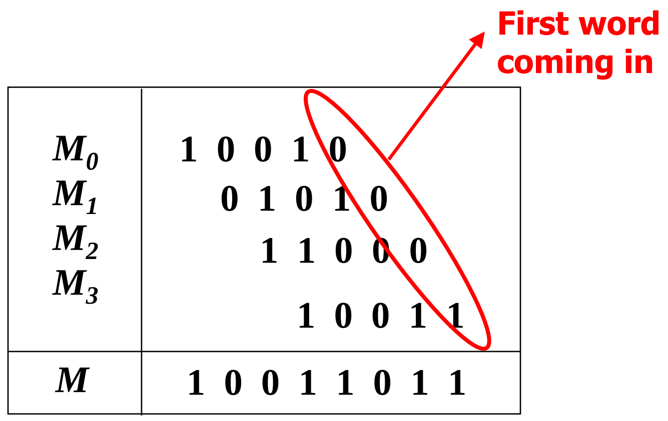
MISR Structure

- The polynomial $\varphi(x) = \varphi_m x^m + \ldots + \varphi_1 x + \varphi_0$
- The polynomial relationship
  - Up to m number of L-bit sequences coming to each input of MISR
  - The effective input polynomial will be a combination of all input polynomials: $M(x)=M_0(x)+x\, M_1(x)+ x^2\, M_2(x) +\ldots+ x^{m-1}\, M_{m-1}(x)$
  - $M(x)=q(x)*\varphi(x)+r(x)$ where $r(x)$ the polynomial remainder of $M(x)/\varphi(x)$ will be the final compactor's response



A 4-Stage MISR Example





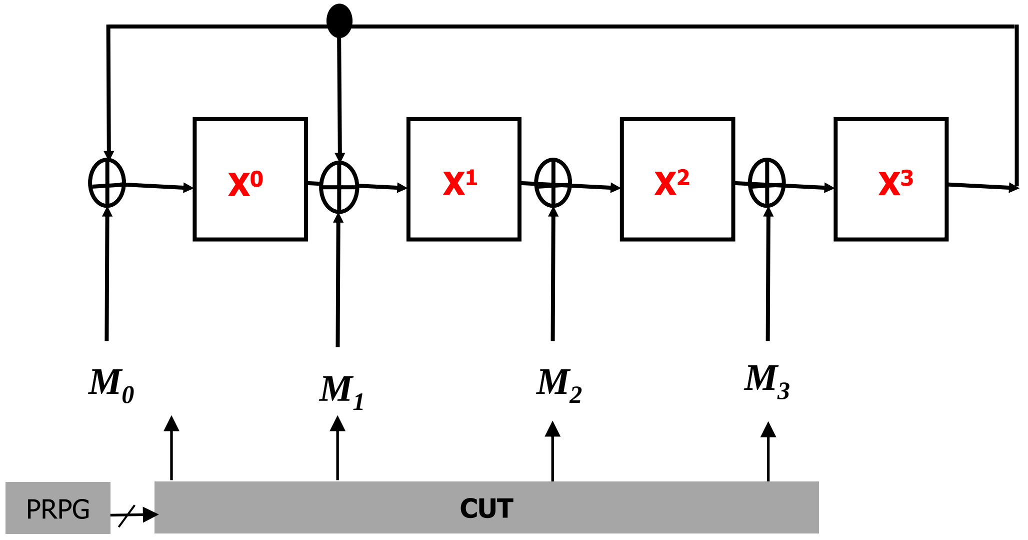
First word coming in

M_0	1	0	0	1	0			
M_1		0	1	0	1	0		
M_2			1	1	0	0	0	
M_3				1	0	0	1	1
M	1	0	0	1	1	0	1	1

- The equivalent sequence will be: $1x^0+0x^1+0x^2+1x^3+1x^4+0x^5+1x^6+1x^7$

- The aliasing probability when n out of m inputs are used
  - $P_{MISR}(m)=[2^{(nL-m)}-1]/(2^{nL}-1) \approx 2^{-m}$

MISR Division Example



$$
\begin{array}{r|l}
 & x^3+x^2 \\ \hline
x^4+x+1 & x^7+x^6 \quad +x^4+x^3 \quad\quad +x^0 \\
 & x^7 \quad\quad\;\; +x^4+x^3 \\ \hline
 & \quad\; x^6 \quad\quad\quad\quad\quad\quad\quad +x^0 \\
 & \quad\; x^6 \quad\quad +x^3+x^2 \\ \hline
 & \quad\quad\quad\quad\;\; x^3+x^2 \quad +x^0
\end{array}
$$

Remainder = Signature = X^0 X^1 X^2 X^3 = 1 0 1 1

MISR Matrix Equation

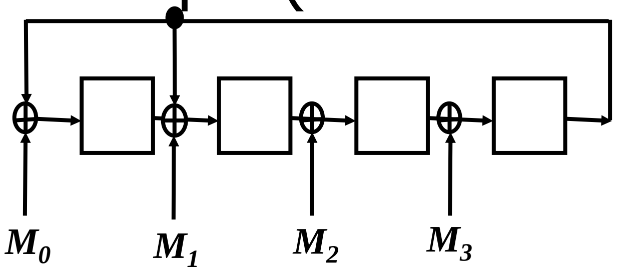
- The polynomial: $\varphi(x) = x^4 + x + 1$
- $M_i(t)$: Input streams of MISR at time t
- The transition matrix/relation:

Internal XOR Transition Matrix (A)

$$\begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & \phi_0 \\ 1 & 0 & 0 & \cdots & 0 & \phi_1 \\ 0 & 1 & 0 & \cdots & 0 & \phi_2 \\ 0 & 0 & 1 & \cdots & 0 & \phi_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \phi_{n-1} \end{pmatrix}$$

$$\begin{bmatrix} s_0(t+1) \\ s_1(t+1) \\ s_2(t+1) \\ s_3(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_0(t) \\ s_1(t) \\ s_2(t) \\ s_3(t) \end{bmatrix} + \begin{bmatrix} M_0(t) \\ M_1(t) \\ M_2(t) \\ M_3(t) \end{bmatrix}$$

- Example (note that * is AND and + is XOR)



Data Stream

M_0	M_1	M_2	M_3
0	0	0	1
1	1	0	1
0	0	0	0
0	1	1	0
1	0	1	1

$$A \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

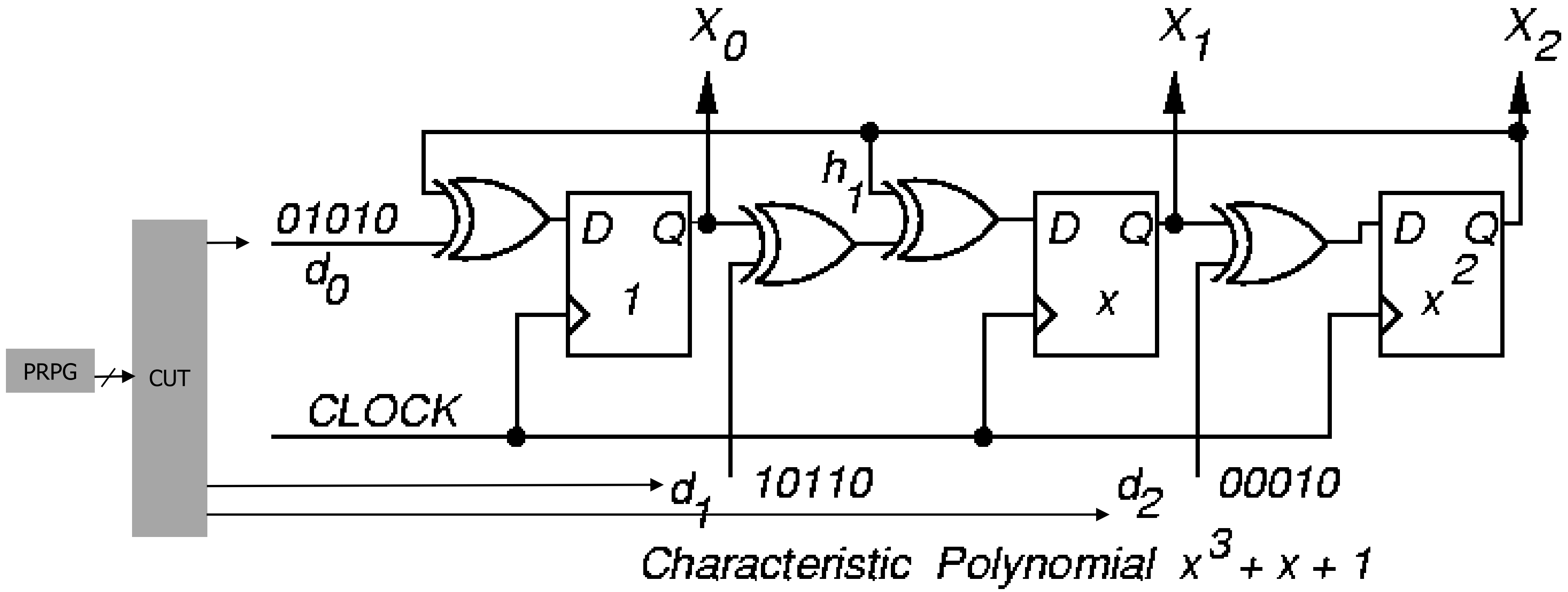
$$A \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$A \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$A \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$A \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Another MISR Example



$$\begin{bmatrix} s_0(t+1) \\ s_1(t+1) \\ s_2(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_0(t) \\ s_1(t) \\ s_2(t) \end{bmatrix} + \begin{bmatrix} M_0(t) \\ M_1(t) \\ M_2(t) \end{bmatrix}$$

Multiple Signature Checking

- Use 2 different testing sessions:
  - 1st with MISR with polynomial $\varphi_1(x)$
  - 2nd with MISR with different polynomial $\varphi_2(x)$
- 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