ECE 715
System on Chip Design and Test

Lecture 22

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

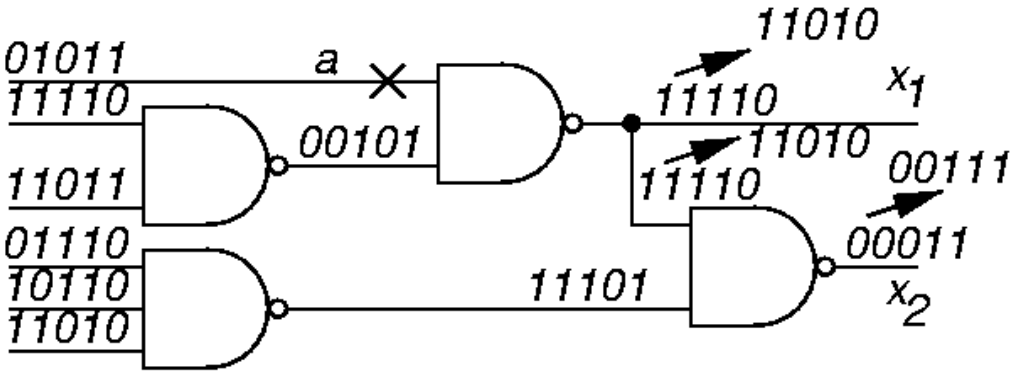
Response Compaction

- 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

Definitions

- *Aliasing* – Due to information loss, signatures of good and some bad machines match
- *Compaction* – Drastically reduce # bits in original circuit response – lose 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
- *Transition Count Response Compaction* – Count # transitions from 0 → 1 and 1 → 0 as a signature

Transition Counting



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

Faulty machine response is shown above the good machine response

Transition Counting Details

- **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**

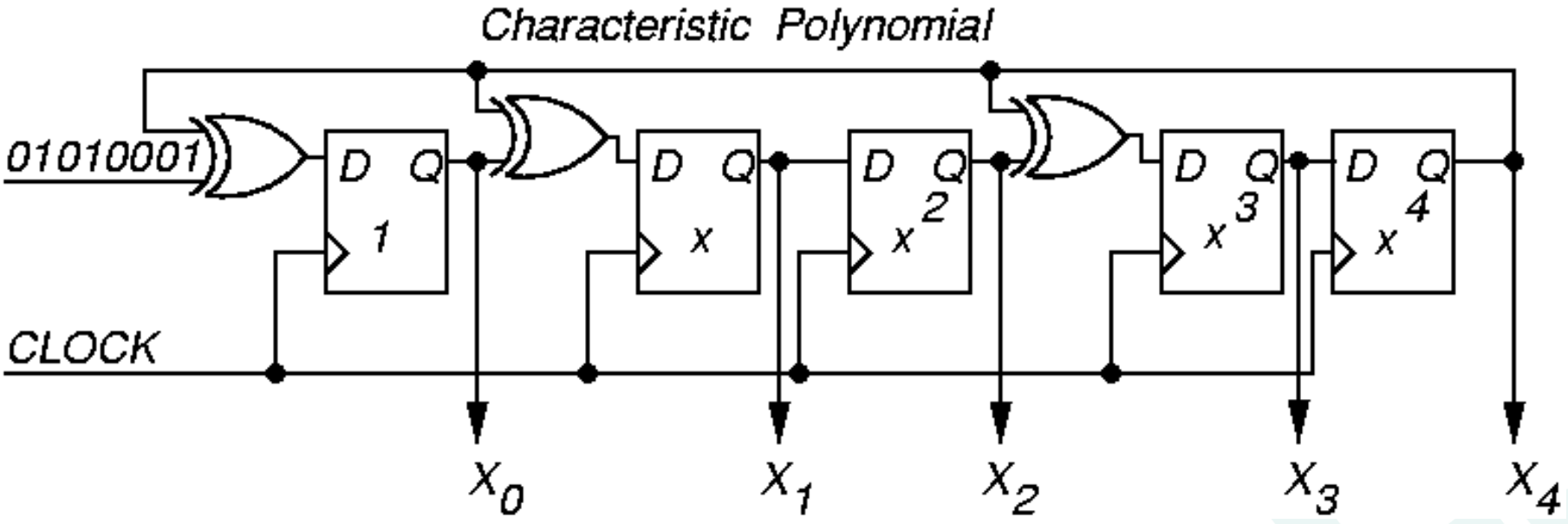
Response Compaction

- Obtain a response sequence *R* for a given order of test vectors from a gold CUT or a simulator.
- Use a compaction function C to produce a vector or a set of vectors *C(R).*
- The number of bits in *C(R)* to be far fewer than the number in *R.*
- Store the compacted vectors on chip or off chip, and, during BIST, use the compaction function *C* to, compact the CUT's actual responses *R** to provide *C(R**).
- Finally, to determine the CUT'S status (fault-free or faulty), we compare *C(R)* and *C(R **).
- We declare the CUT fault-free if these two values are identical.

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

Example Modular LFSR Response Compacter



Polynomial Division

- An LFSR modified to accept an external input, acts as a polynomial divider.
- It divides the input sequence, represented by a polynomial, by the characteristic polynomial g (x) of the LFSR.
- As this division proceeds bit by bit, the quotient sequence appears at the output of the LFSR and the remainder appears in the LFSR with every shift of the input sequence into the LFSR.

Polynomial Division

Logic Simulation:

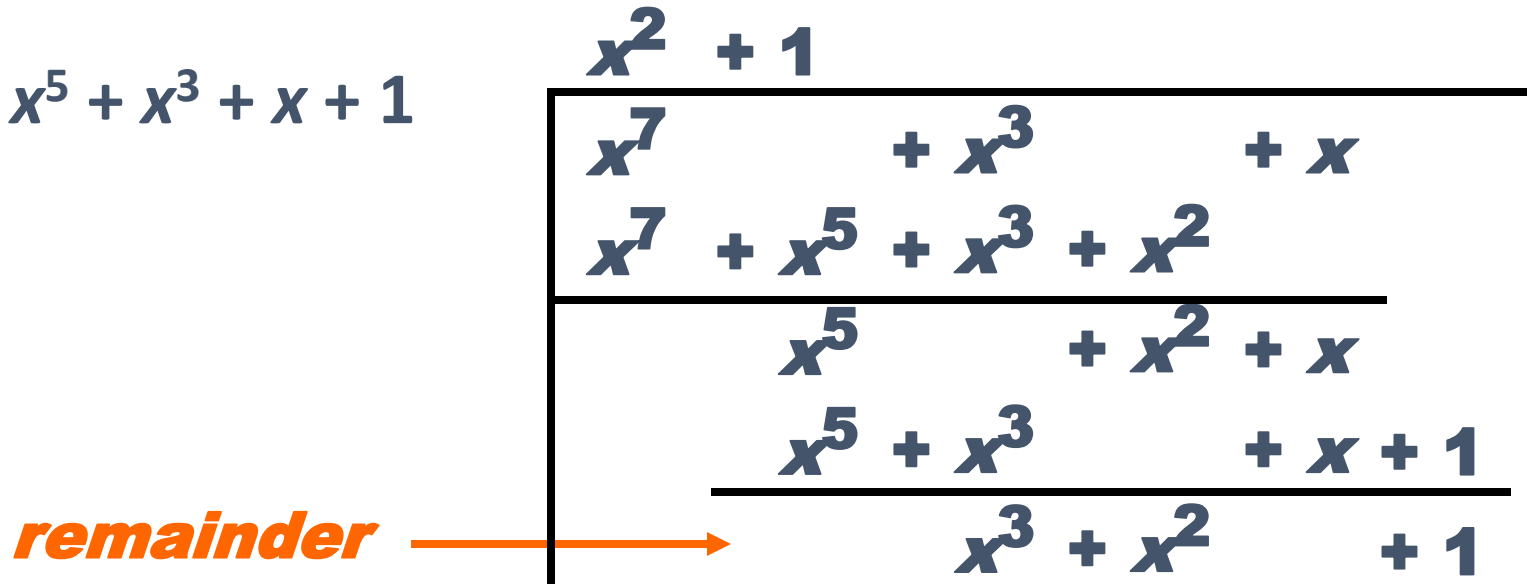
Inputs	x^0	x^1	x^2	x^3	x^4
Initial State	0	0	0	0	0
1	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
1	1	0	0	0	1
0	1	0	0	1	0
1	1	1	0	0	1
0	1	0	1	1	0

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

0 1 0 1 0 0 0 1

$0 \quad x^0 + 1 \quad x^1 + 0 \quad x^2 + 1 \quad x^3 + 0 \quad x^4 + 0 \quad x^5 + 0 \quad x^6 + 1 \quad x^7$

Symbolic Polynomial Division



Remainder matches that from logic simulation of the response compacter!

Multiple-Input Signature Register (MISR)

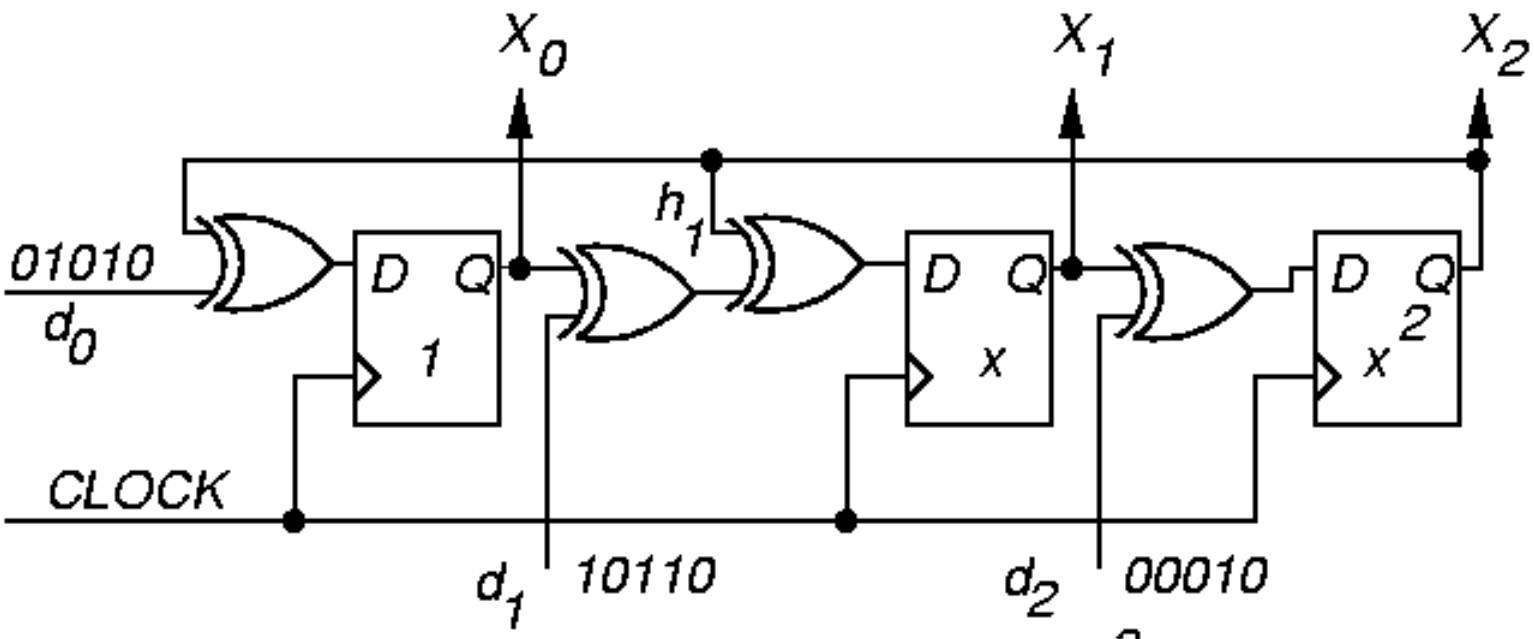
- **Problem with ordinary LFSR response compacter:**
  - **Too much hardware if one of these is put on each *primary output* (PO)**
- **Solution: MISR – compacts all outputs into one LFSR**
  - **Works because LFSR is linear – obeys *superposition principle***
  - **Superimpose all responses in one LFSR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

MISR Matrix Equation

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

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

Modular MISR Example

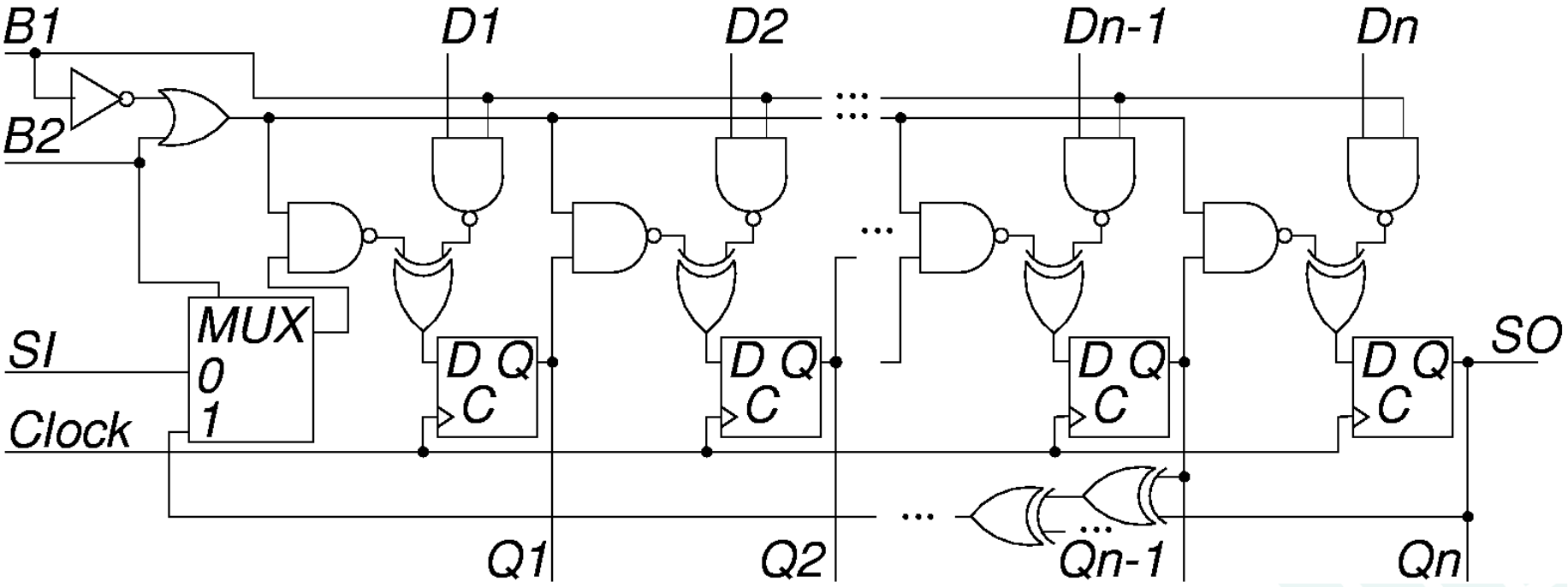


Characteristic Polynomial $x^3 + x + 1$

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \end{bmatrix} + \begin{bmatrix} d_0(t) \\ d_1(t) \\ d_2(t) \end{bmatrix}$$

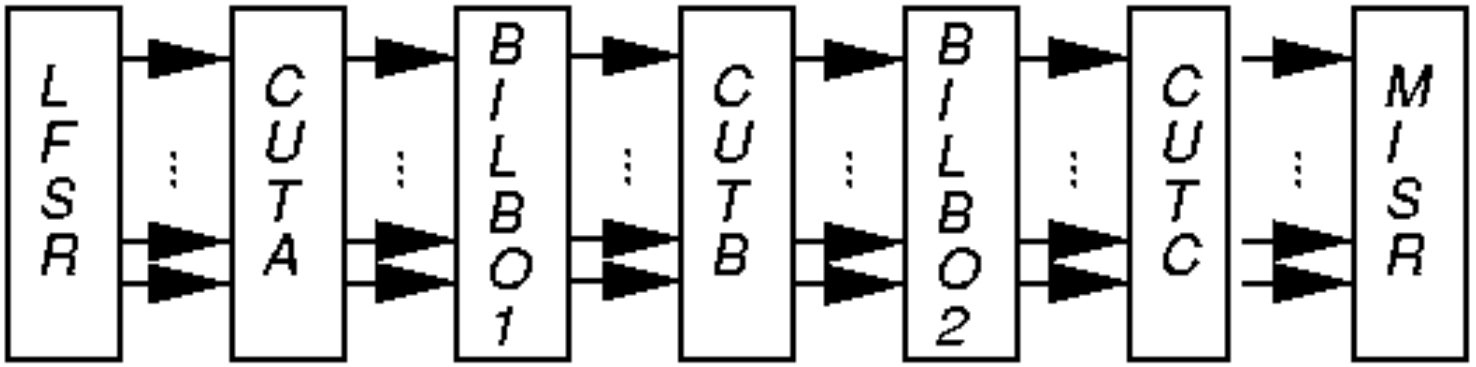
Built-in Logic Block Observer (BILBO)

- **Combined functionality of D flip-flop, *pattern generator*, *response compacter*, & *scan chain***
  - **Reset all FFs to 0 by scanning in zeros**



Example BILBO Usage

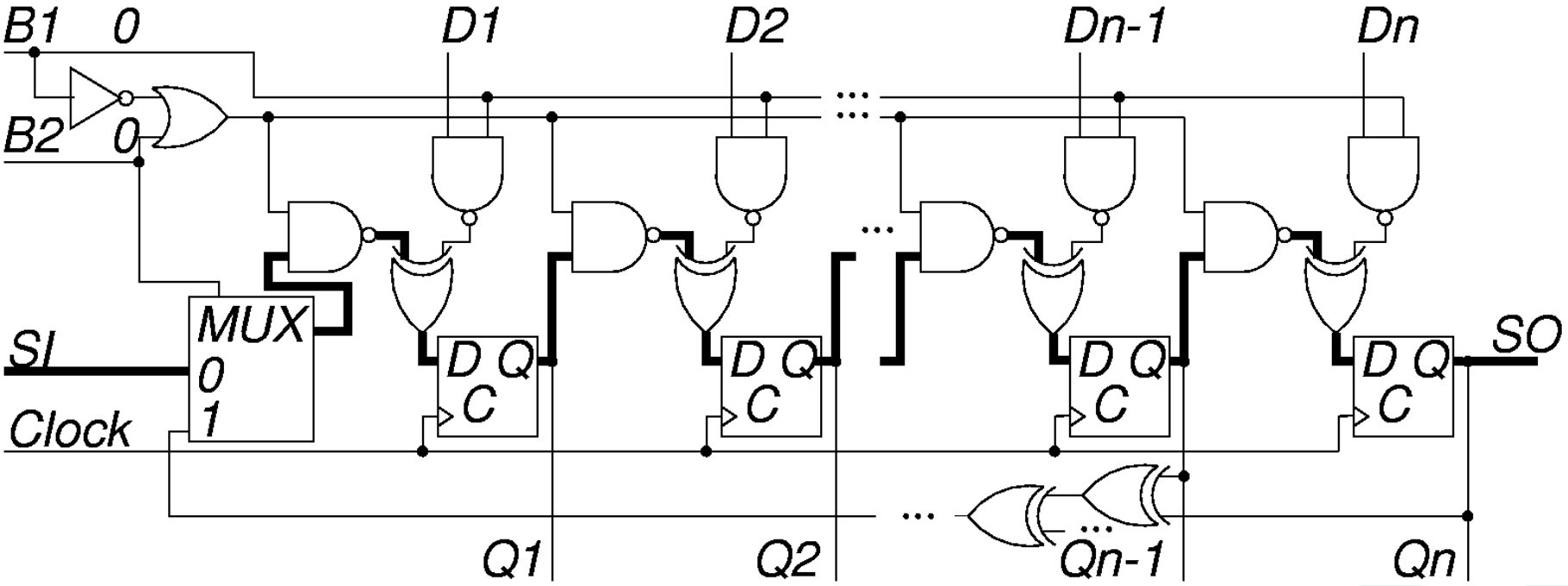
- *SI – Scan In*
- *SO – Scan Out*
- *Characteristic polynomial*: $1 + x + \ldots + x^n$
- CUTs A and C: BILBO1 is MISR, BILBO2 is LFSR
- CUT B: BILBO1 is LFSR, BILBO2 is MISR



(a) Example test configuration.

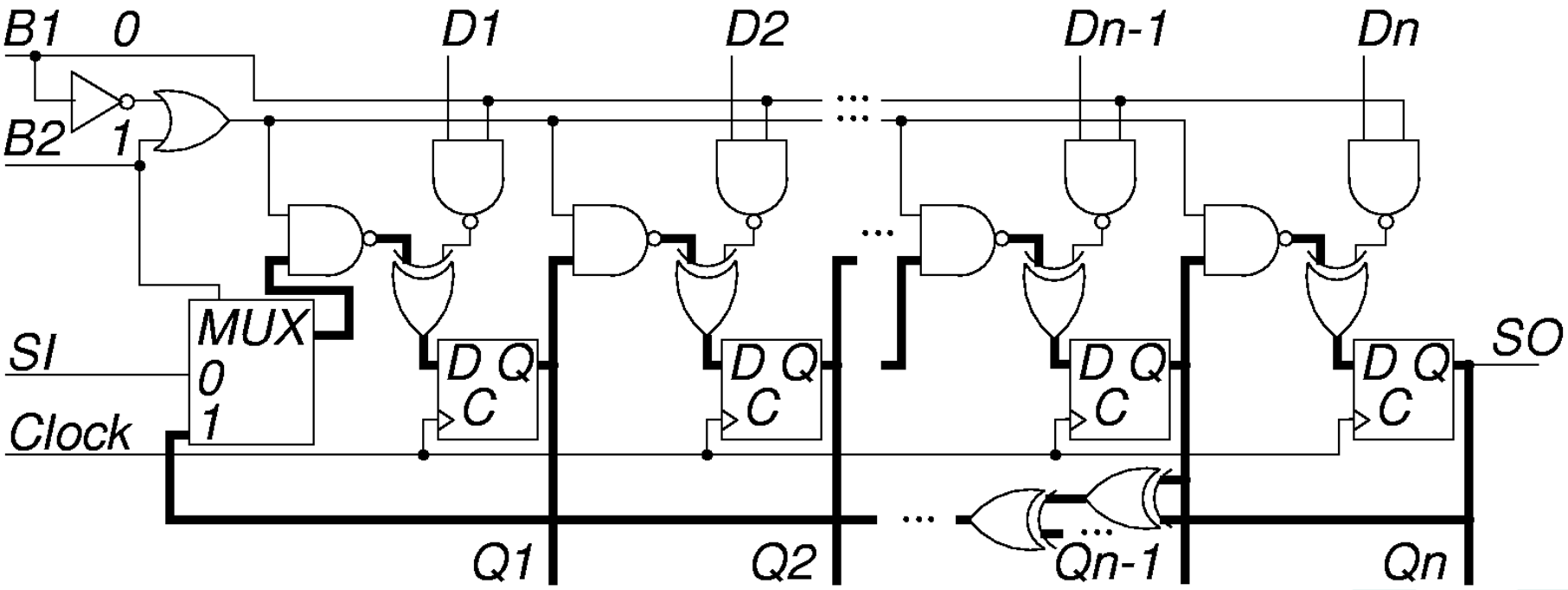
BILBO Serial Scan Mode

- *B*1 *B*2 = “00”
- Dark lines show enabled data paths



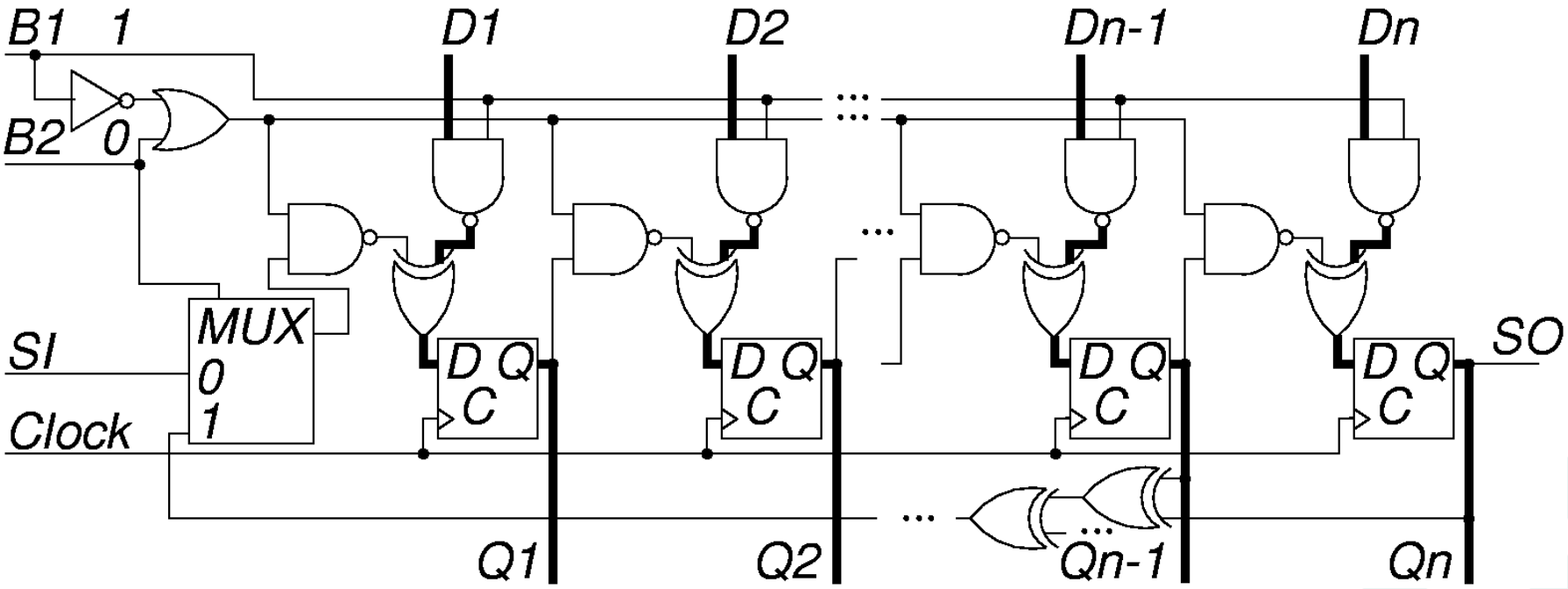
BILBO LFSR Pattern Generator Mode

- *B1 B2* = “01”



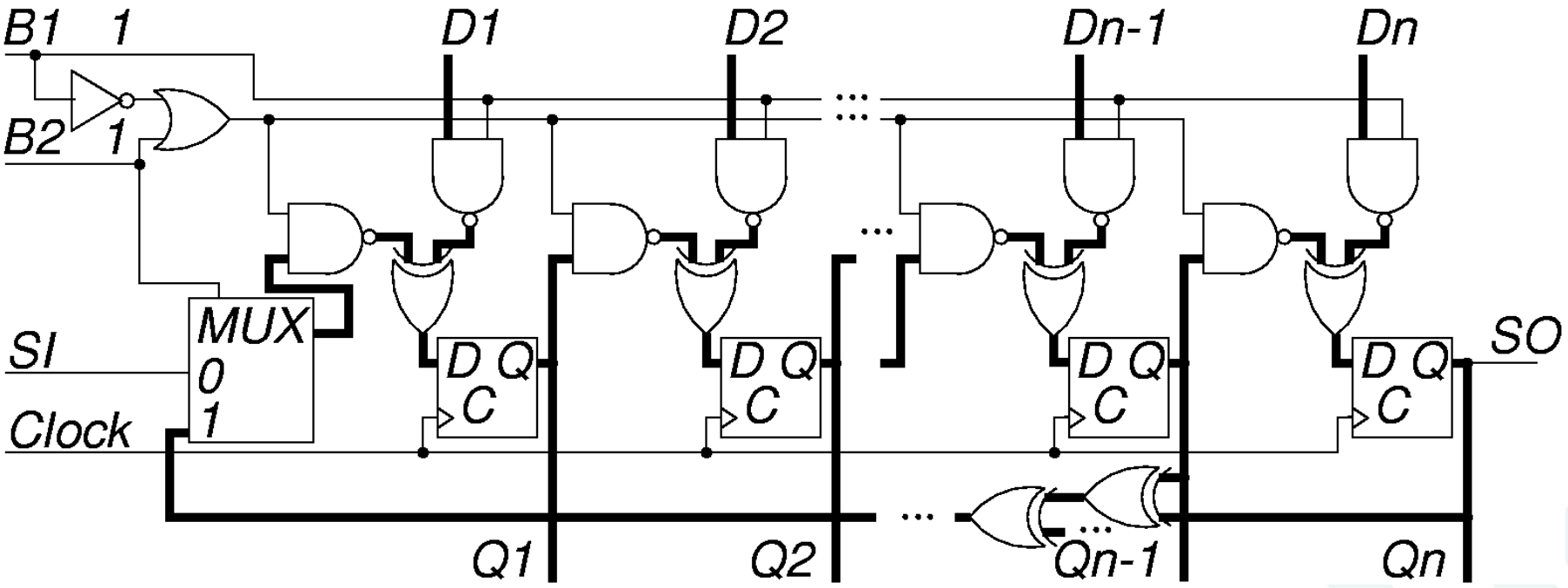
BILBO in D FF (Normal) Mode

- *B*1 *B*2 = “10”



BILBO in MISR Mode

- ***B1 B2*** **= “11”**

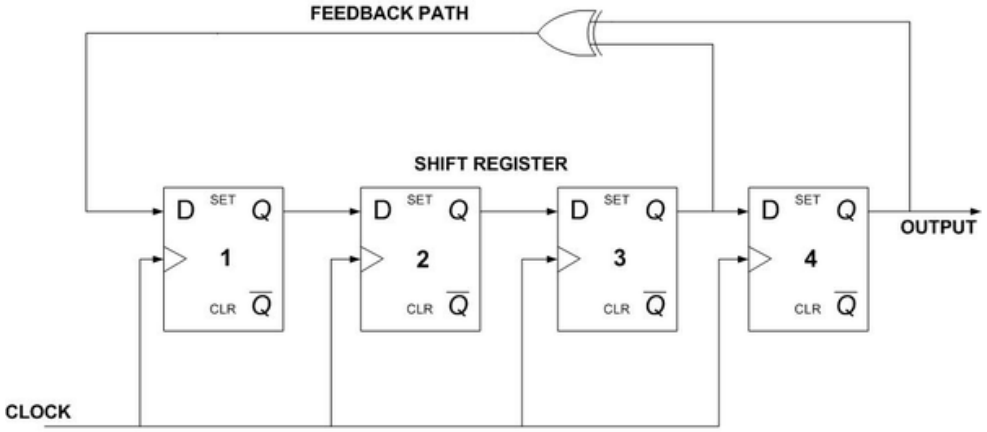


Summary

- 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:
  - Drastic ATE cost reduction
  - Field test capability
  - Faster diagnosis during system test
  - Less effort to design testing process
  - Shorter test application times

Quiz VI

- Consider the following LFSR. What is the characteristic polynomial?

1. 

2.