CMOS

Digital Integrated Circuit

Analysis & Design

Chapter 14 Design For Testablity

Why Model Faults?

- **I/O function tests inadequate for manufacturing (functionality versus component and interconnect testing)**
- **Real defects (often mechanical) too numerous and often not analyzable**
- **A fault model identifies targets for testing**
- **A fault model makes analysis possible**
- **Effectiveness measurable by experiments**

Some Real Defects in Chips

- Processing defects
  - Missing contact windows
  - Parasitic transistors
  - Oxide breakdown
- Material defects
  - Bulk defects (cracks, crystal imperfections)
  - Surface impurities (ion migration)
- Time-dependent failures
  - Dielectric breakdown
  - Electromigration
- Packaging failures
  - Contact degradation
  - Seal leaks

Physical Defect

Defects in Silicon Substrate
Photolithographic defects
Mask Contamination and scratches
Process variations and abnormalities
Oxide defects

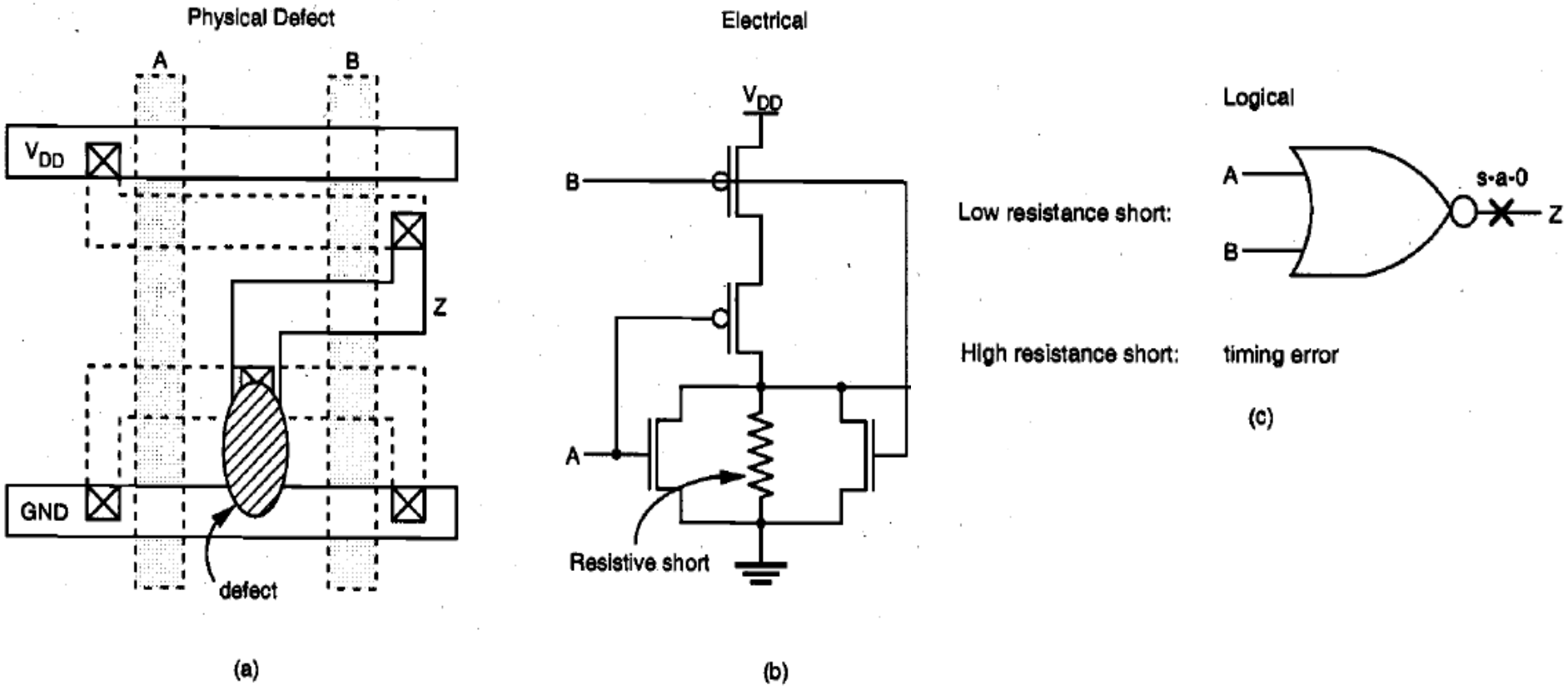
Physical defects can cause electrical faults and logical faults

Electrical Fault include

Shorts (bridging faults)
Opens
Transistor stuck-on, stuck open
Resistive short and open
Excessive change in thresold voltage
Excessive steady-state current

Logical Fault

Logical stuck-at-0 or stuck-at-1
Slower transition (Delay fault)
AND-bridging, OR-bridging



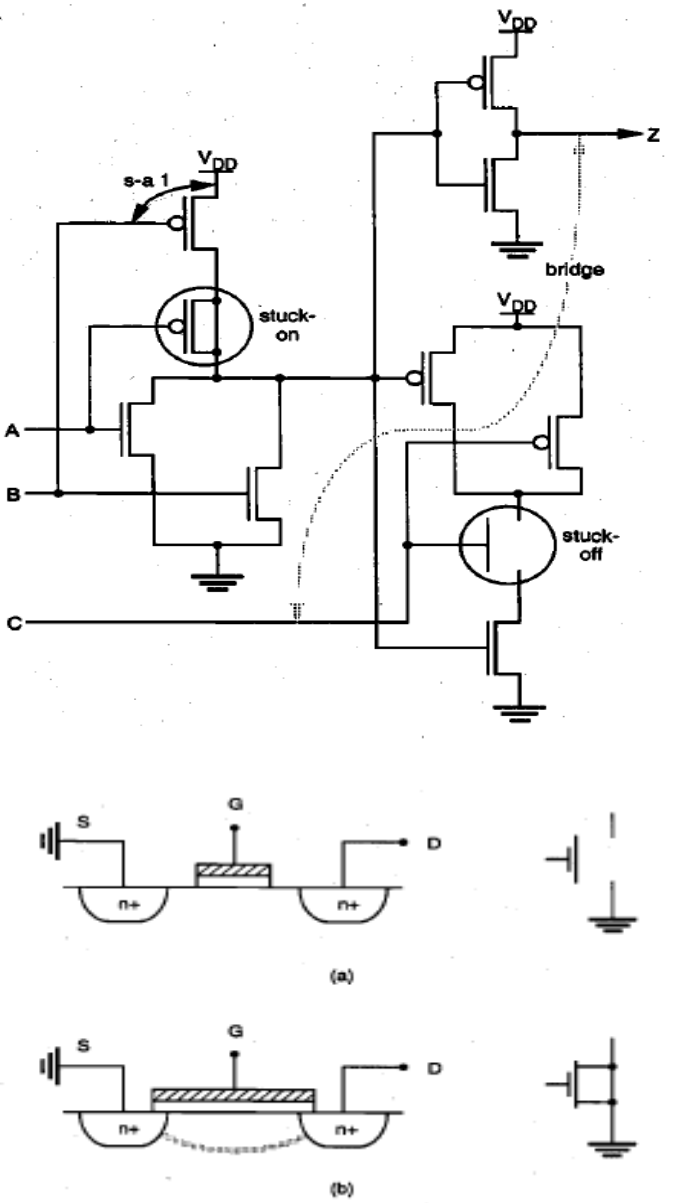
(a) Physical fault in NOR2 fabrication (b) its electrical fault model (c) its logical fault models

Figure shows other type of fault in a CMOS circuit consisting of NOR2 NAND2 and inverter gate.

Input line stuck B suck-at-1 (since input line is shorted to power line.

The PMOS transistor of the first stage NOR2 gate is stuck-on due to process problem that causes a short between its source and drain terminals.

The top nmos transistor in NAND2 gate,is stuck open due to either an incomplete contact (open) of the source or drain node or due to a large seperation of drain or source diffusion from the gate, which causes permanent turn off the transistor regartdless of the input C value.



Stuck open and stuck close

Stuck at Fault Model

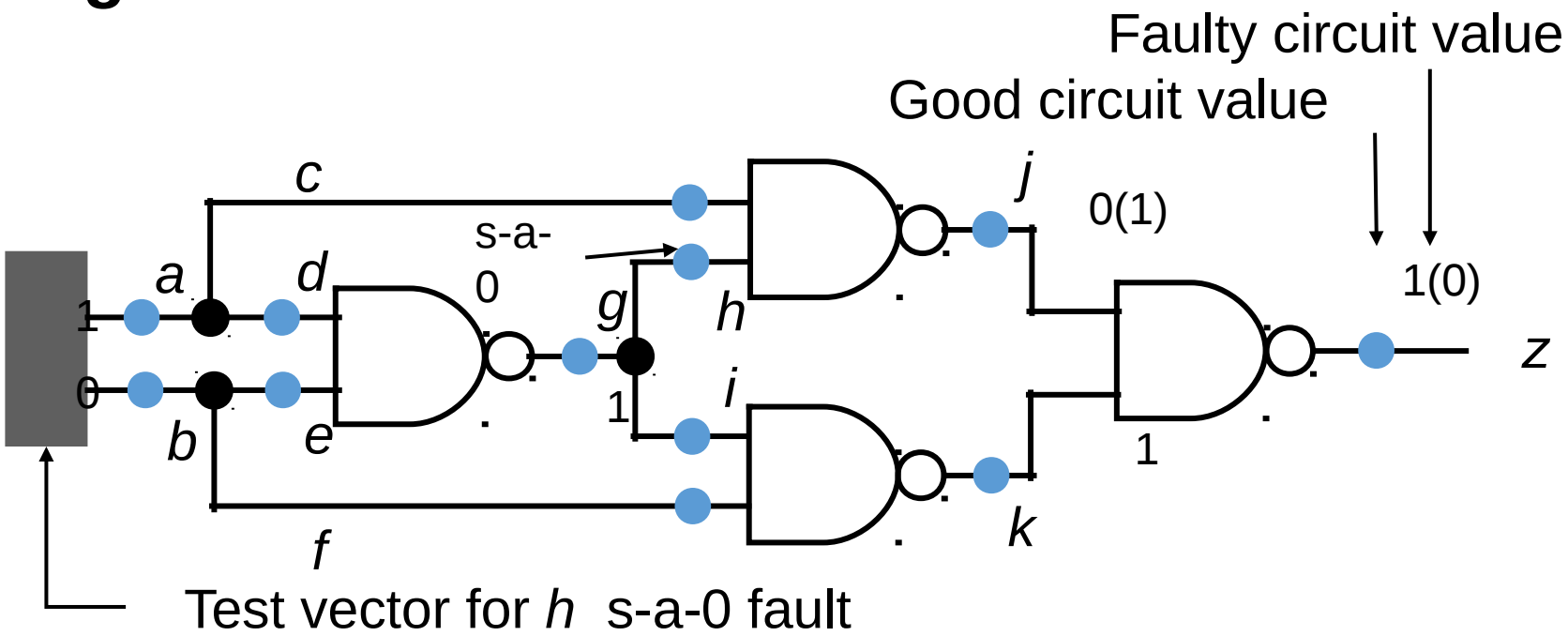
Complexity of test generation is greately reduced.
Single stuck-at fault is independent of technology, design style.
Single stuck-at test cover a large percentage of multile stuck-at faults.
Single stuck-at test cover a large percentage of unmodelled physical defects.

In a two level circuit with no redudancy, any complete test set for all single stuck-at faults can cover all stuck-at faults.

Multiple atuck-at fault model find application for fuse or anti-fuse based programmable design such as CPLD, FPGA , RAM etc.

Single Stuck-at Fault

- **Three properties define a single stuck-at fault**
  - **Only one line is faulty**
  - **The faulty line is permanently set to 0 or 1**
  - **The fault can be at an input or output of a gate**
- **Example: XOR circuit has 12 fault sites ( ) and 24 single stuck-at faults**



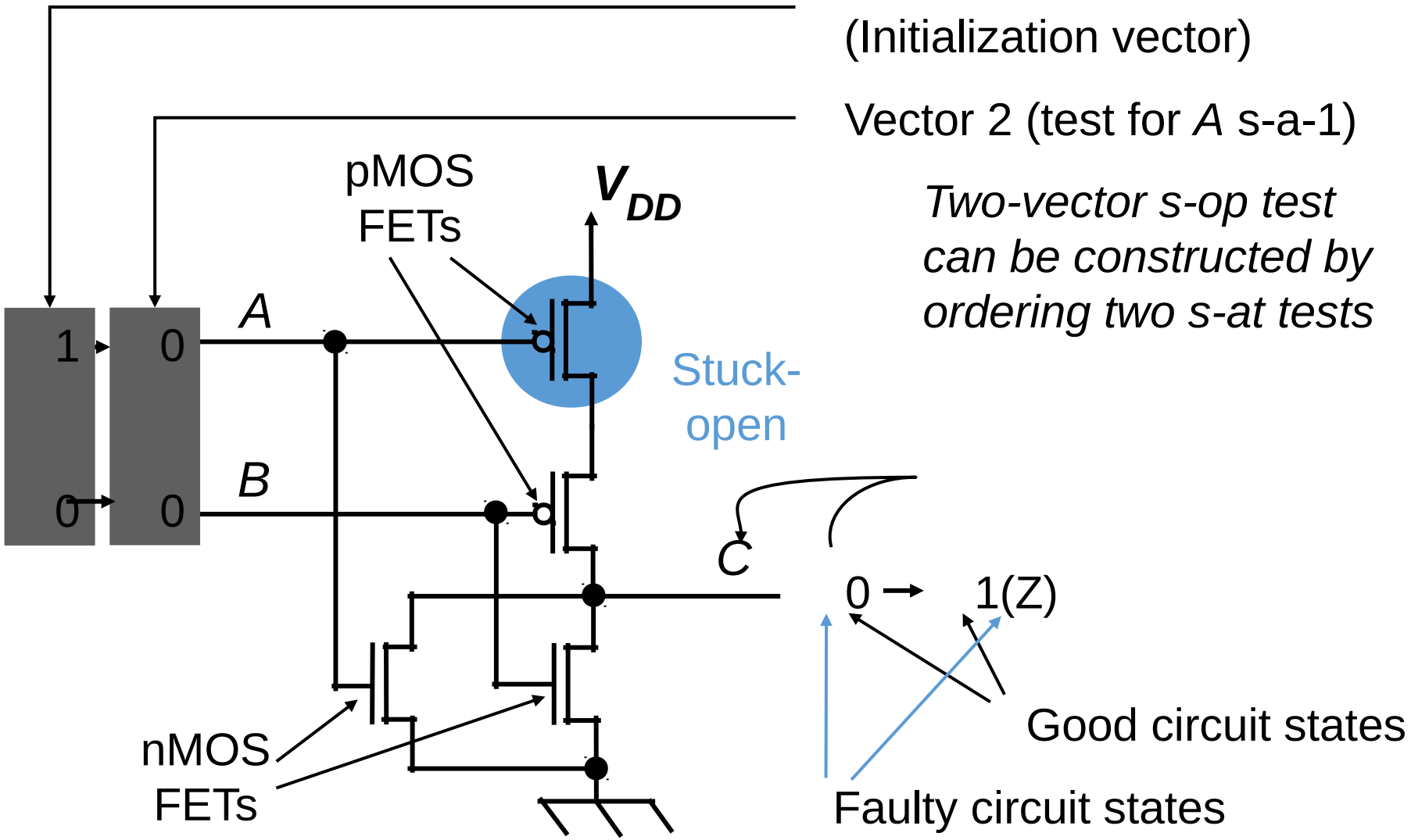
Multiple Stuck-at Faults

- **A multiple stuck-at fault means that any set of lines is stuck-at some combination of (0,1) values.**
- **The total number of single and multiple stuck-at faults in a circuit with k single fault sites is $3^k - 1$.**
- **A single fault test can fail to detect the target fault if another fault is also present, however, such masking of one fault by another is rare.**
- **Statistically, single fault tests cover a very large number of multiple faults.**

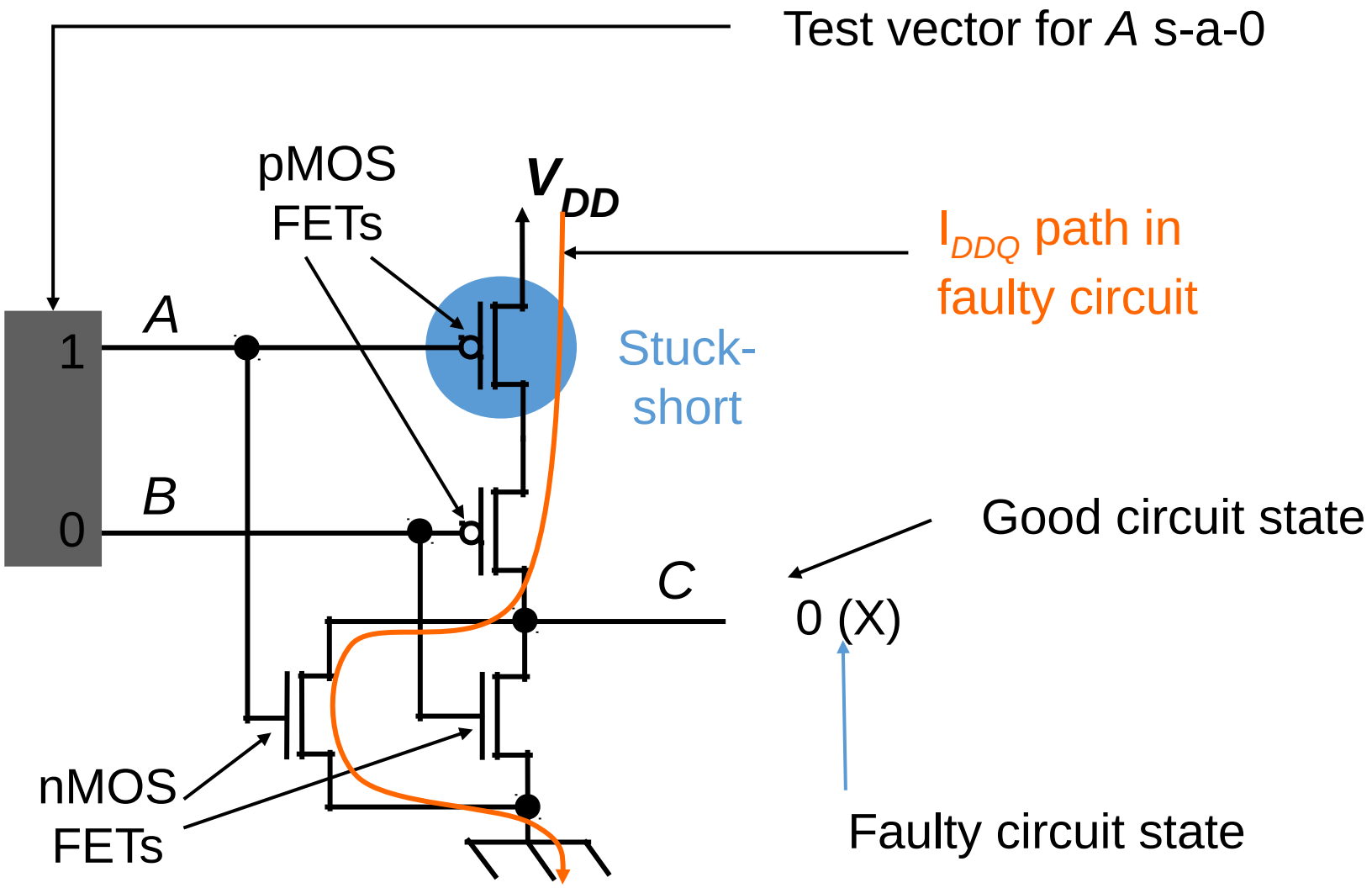
Transistor (Switch) Faults

- **MOS transistor is considered an ideal switch and two types of faults are modeled:**
  - **Stuck-open – a single transistor is permanently stuck in the open state.**
  - **Stuck-short – a single transistor is permanently shorted irrespective of its gate voltage.**
- **Detection of a stuck-open fault requires two vectors.**
- **Detection of a stuck-short fault requires the measurement of quiescent current (I_{DDQ}).**

Stuck-Open Example



Stuck-Short Example



Testability Analysis

- Determines testability measures
- Involves circuit topological analysis, but no
- test vectors (*static analysis*) and no search algorithm.
- Linear computational complexity.
- Otherwise, is pointless – might as well use automatic test-pattern generator (ATPG) and a fault simulator to calculate:
  - Exact fault coverage
  - Exact test vectors

Summary

- **Fault models are analyzable approximations of defects and are essential for a test methodology.**
- **For digital logic single stuck-at fault model offers best advantage of tools and experience.**
- **Many other faults (bridging, stuck-open and multiple stuck-at) are largely covered by stuck-at fault tests.**
- **Stuck-short and delay faults and technology-dependent faults require special tests.**
- **Memory and analog circuits need other specialized fault models and tests.**

Observability

- The *observability* of a particular circuit node is the degree to which we can observe that node at the output of an integrated circuit.
- Measure the output of a gate within a larger circuit to check whether it operates correctly.
- Limited number of nodes can be directly observed.

Controllability

- The *controllability* of an internal circuit node within a chip is a measure of the ease of setting the node to a *1* or *0* metric.
- Degree of difficulty of testing a particular signal within a circuit
- An easily controllable node would be directly settable via an input pad.

AD-HOC Testable Design Technique

1. Partition and Mux Technique

2. Initialize Sequential Circuit

3. Disable internal Clock oscillator

4. Avoid Asynchronous Logic and redudant Logic

5. Avoid delay dependent logic

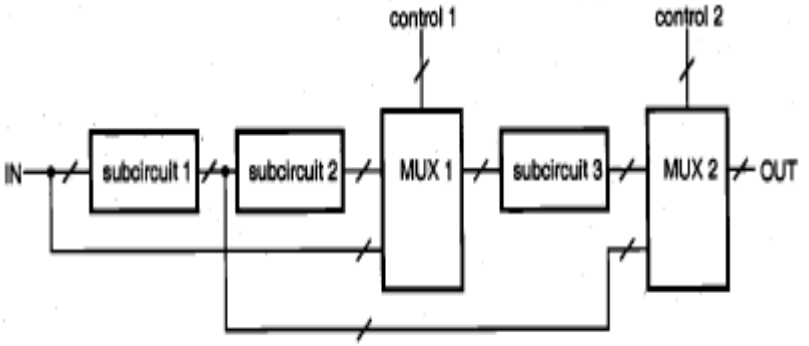
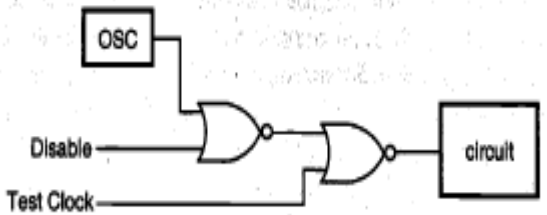


Figure 16.5. Partition-and-mux method for large circuits.



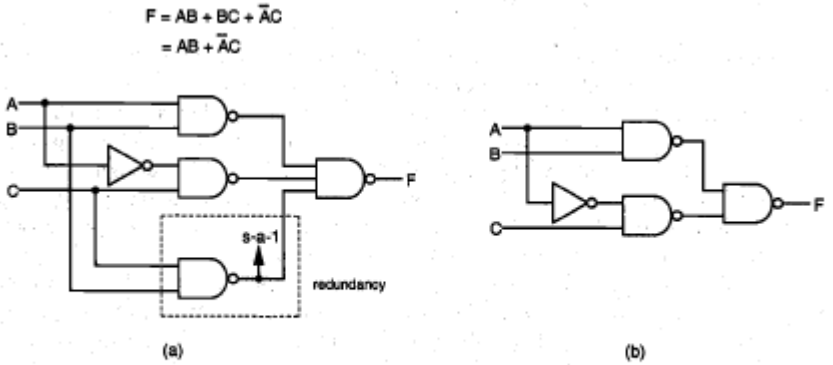


Figure 16.7. (a) A redundant logic gate example. (b) Equivalent gate with redundancy removed.

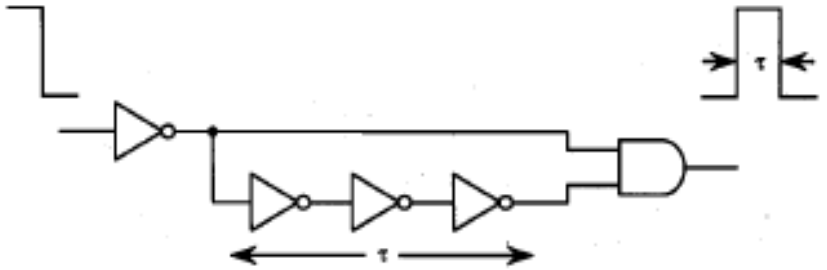


Figure 16.8. A pulse-generation circuit using a delay chain of three inverters.

When the sequential circuit is powered up, its initial state can be a random, unknown state. In this case, it is not possible to start the test sequence correctly. The state of a sequential circuit can be brought to a known state through *initialization*. In many designs, the initialization can be easily done by connecting asynchronous preset or clear-input signals from primary or controllable inputs to flip-flops or latches.

To avoid synchronization problems during testing, internal oscillators and clocks should be disabled. For example, rather than connecting the circuit directly to the on-chip oscillator, the clock signal can be ORed with a disabling signal followed by an insertion of a testing signal as shown in Fig. 16.6.

The enhancement of testability requires serious tradeoffs. The speed of an asynchronous logic circuit can be faster than that of the synchronous logic circuit counterpart. However, the design and test of an asynchronous logic circuit are more difficult than for a synchronous logic circuit, and its state transition times are difficult to predict. Also, the operation of an asynchronous logic circuit is sensitive to input test patterns, often causing race problems and hazards of having momentary signal values opposite to the expected values. Sometimes, designed-in logic redundancy is used to mask a static hazard condition for reliability. However, the redundant node cannot be observed since the primary output value cannot be made dependent on the value of the redundant node. Hence, certain faults on the redundant node cannot be tested or detected. Figure 16.7 shows that the bottom NAND2 gate is redundant and the stuck-at-1 fault on its output line cannot be detected. If a fault is undetectable, the associated line or gate can be removed without changing the logic function.

SCAN BASED TECHNIQUE

Step 1: Set the mode to *test* and let latches accept data from scan-in input.
Step 2: Verify the scan path by shifting in and out the test data.
Step 3: Scan in (shift in) the desired state vector into the shift register.
Step 4: Apply the test pattern to the primary input pins.
Step 5: Set the mode to *normal* and observe the primary outputs of the circuit after sufficient time for propagation.
Step 6: Assert the circuit clock for one machine cycle to capture the outputs of the combinational logic into the registers.
Step 7: Return to *test* mode; scan out the contents of the registers, and at the same time scan in the next pattern.
Step 8: Repeat steps 3-7 until all test patterns are applied.

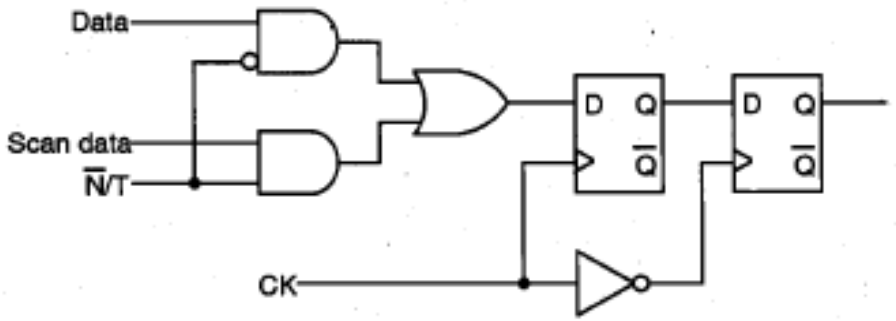
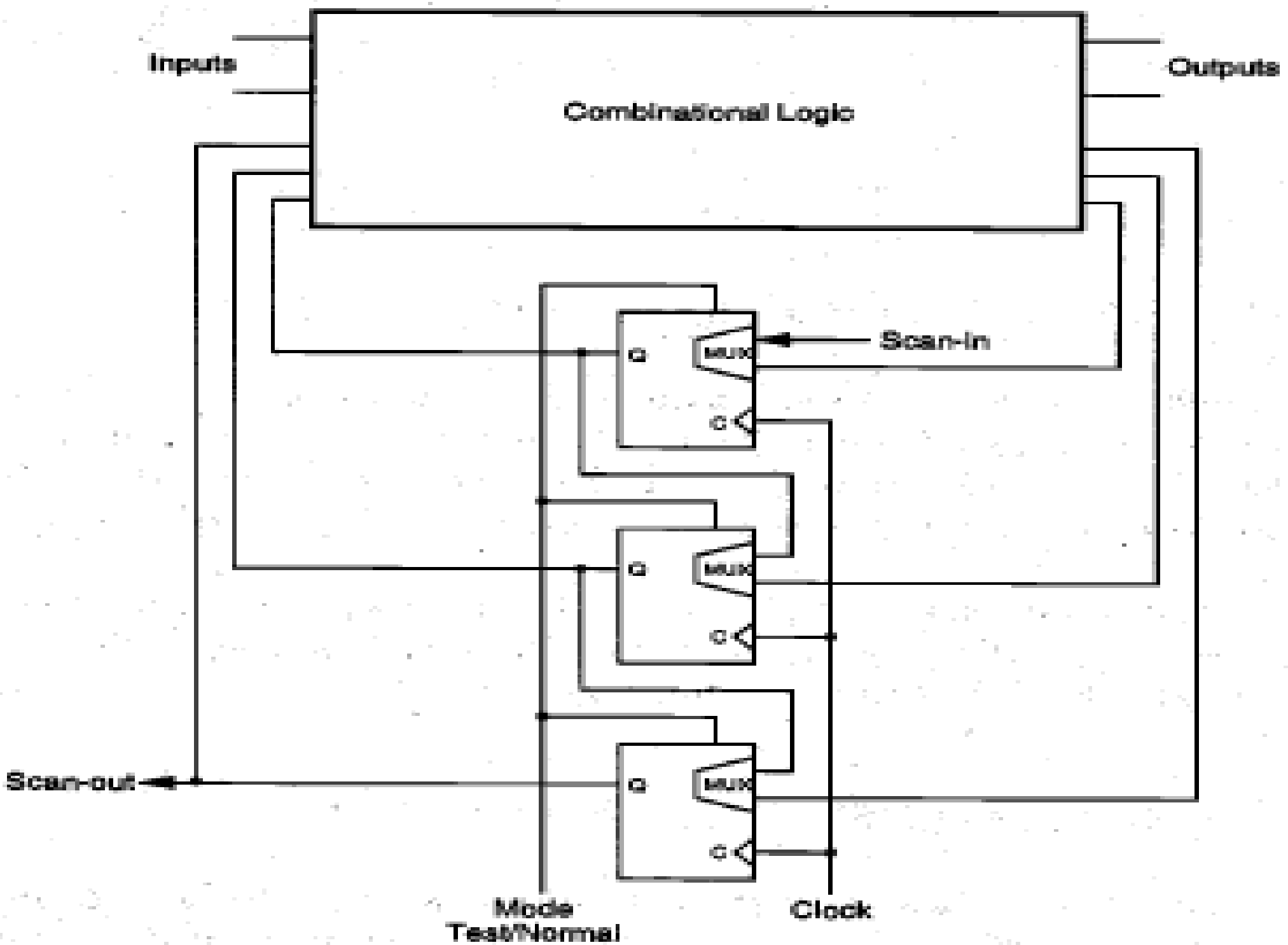
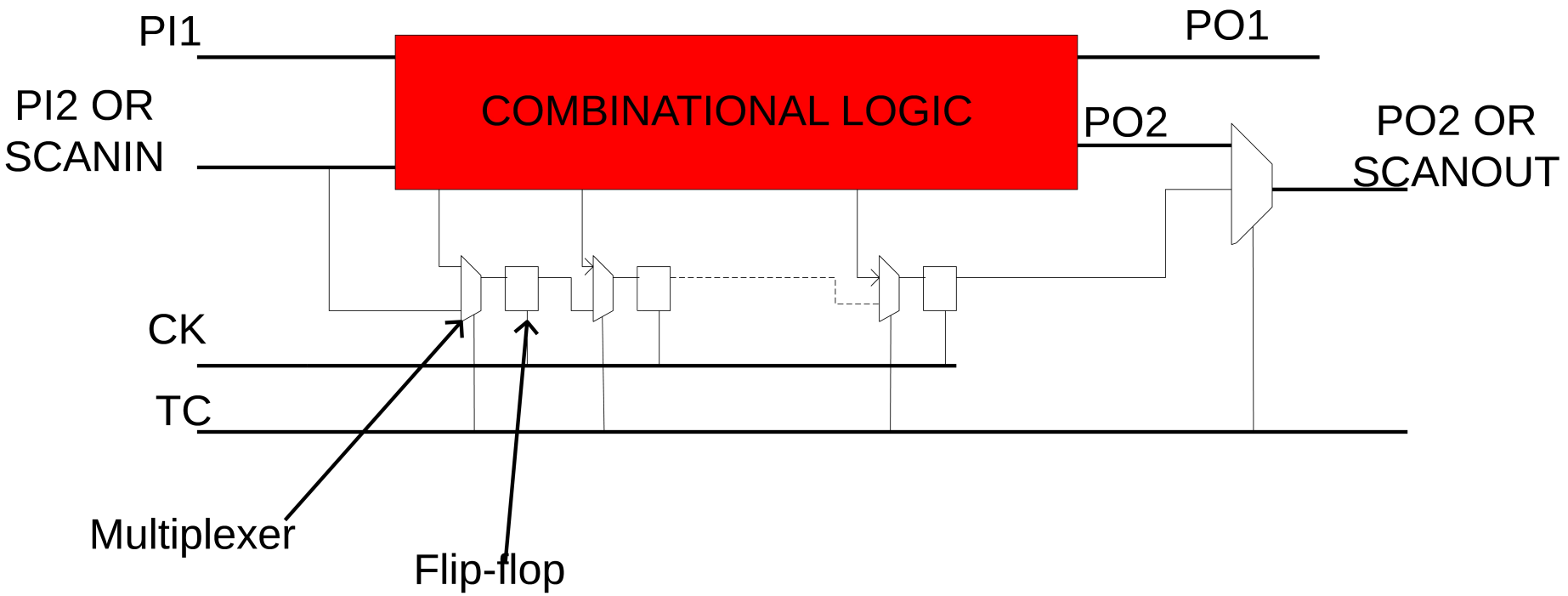


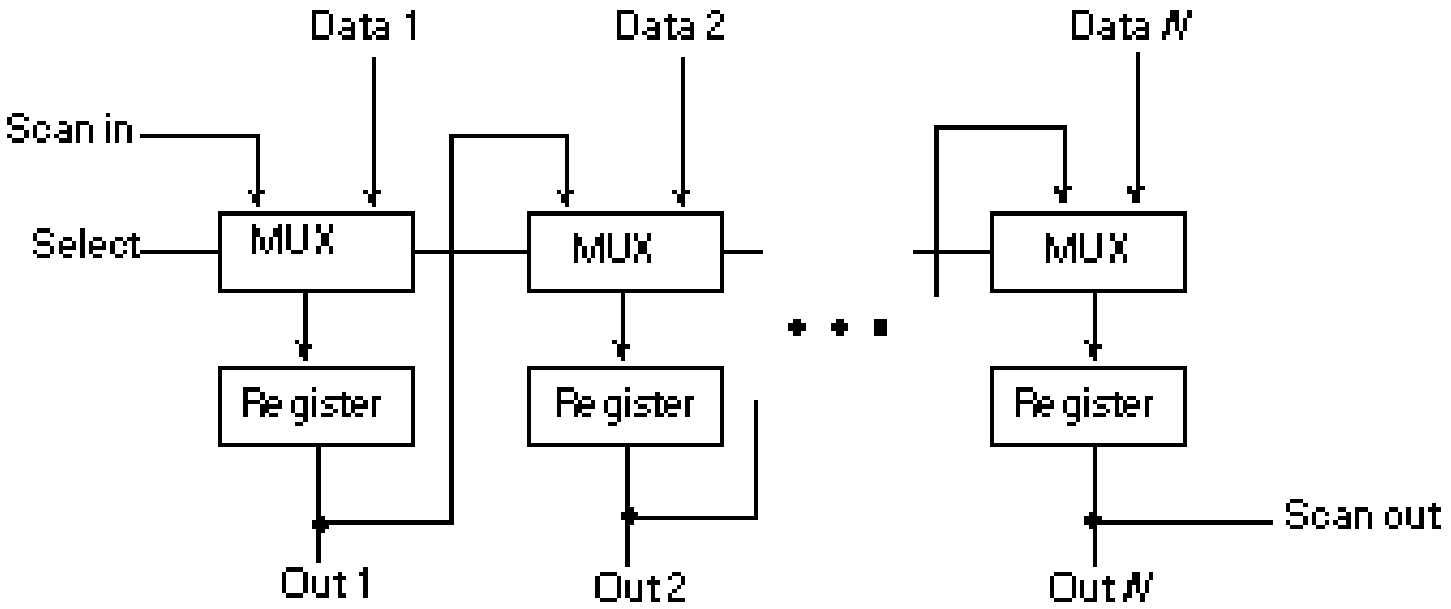
Figure 16.10. Scan-based design of an edge-triggered D flip-flop.


Inputs
Combinational Logic
Outputs
Scan-in
Q
MUX
C
Scan-out
Mode
Test/Normal
Clock

Scan Design



Scan Design



- In test mode, all flip-flops functionally form one or more shift registers
- The inputs and outputs of these shift registers are made into PI/Pos
- Using the test mode, all flip-flops can be set to any desired states
- The states of the flip-flops are observed by shifting the contents of the scan register out

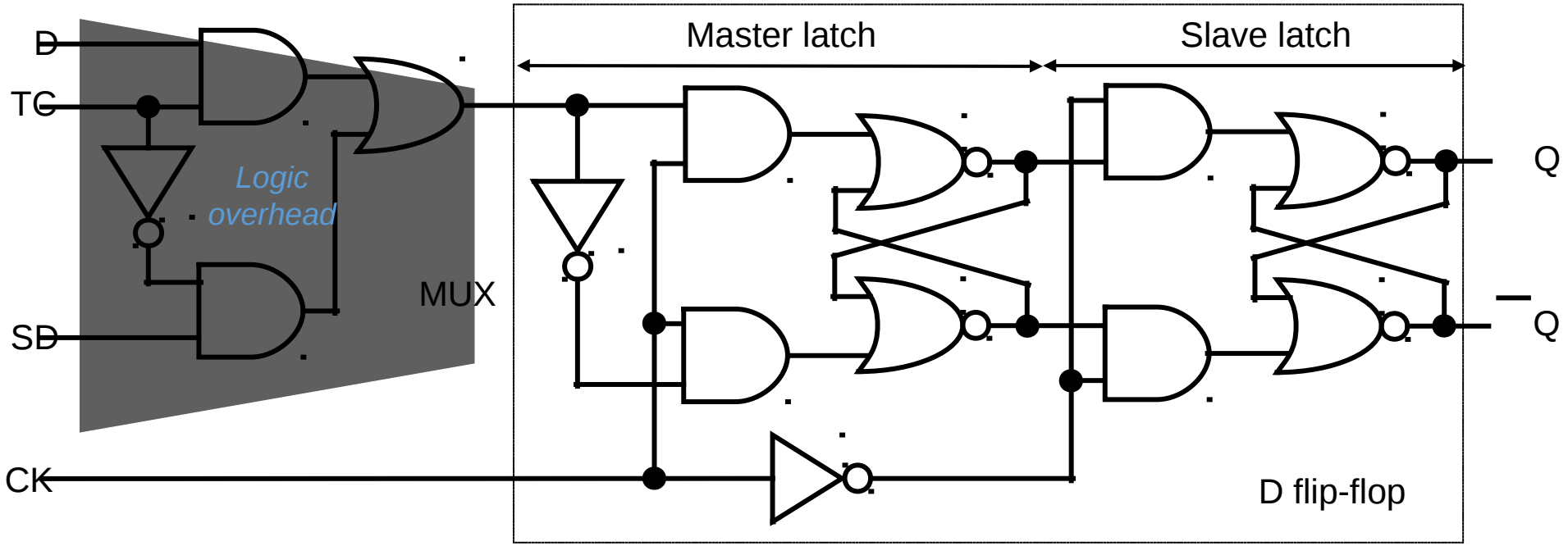
Scan Design

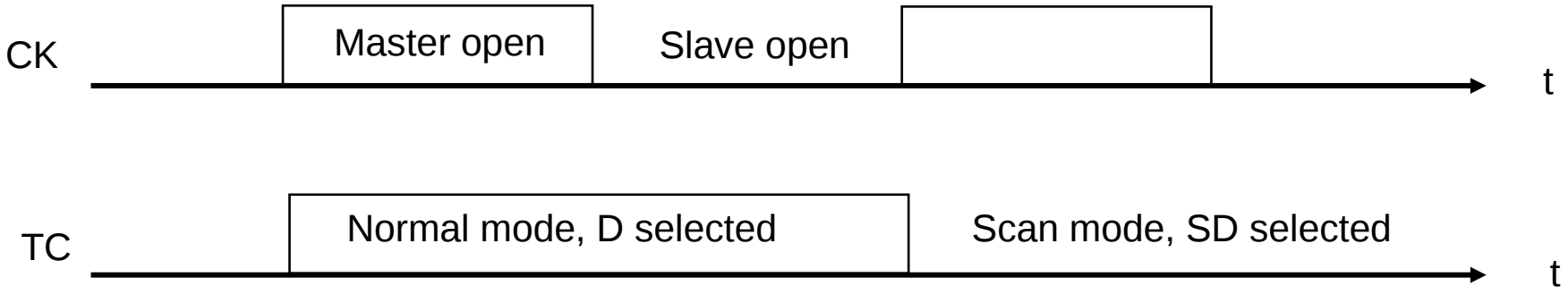
- Circuit is designed using pre-specified design rules.
- Test structure (hardware) is added to the verified design:
  - Add a *test control* (TC) primary input.
  - Replace flip-flops by *scan flip-flops* (SFF) and connect to form one or more shift registers in the test mode.
  - Make input/output of each scan shift register controllable/observable from PI/PO.
- Use combinational ATPG to obtain tests for all testable faults in the combinational logic.
- Add shift register tests and convert ATPG tests into scan sequences for use in manufacturing test.

Scan Design Rules

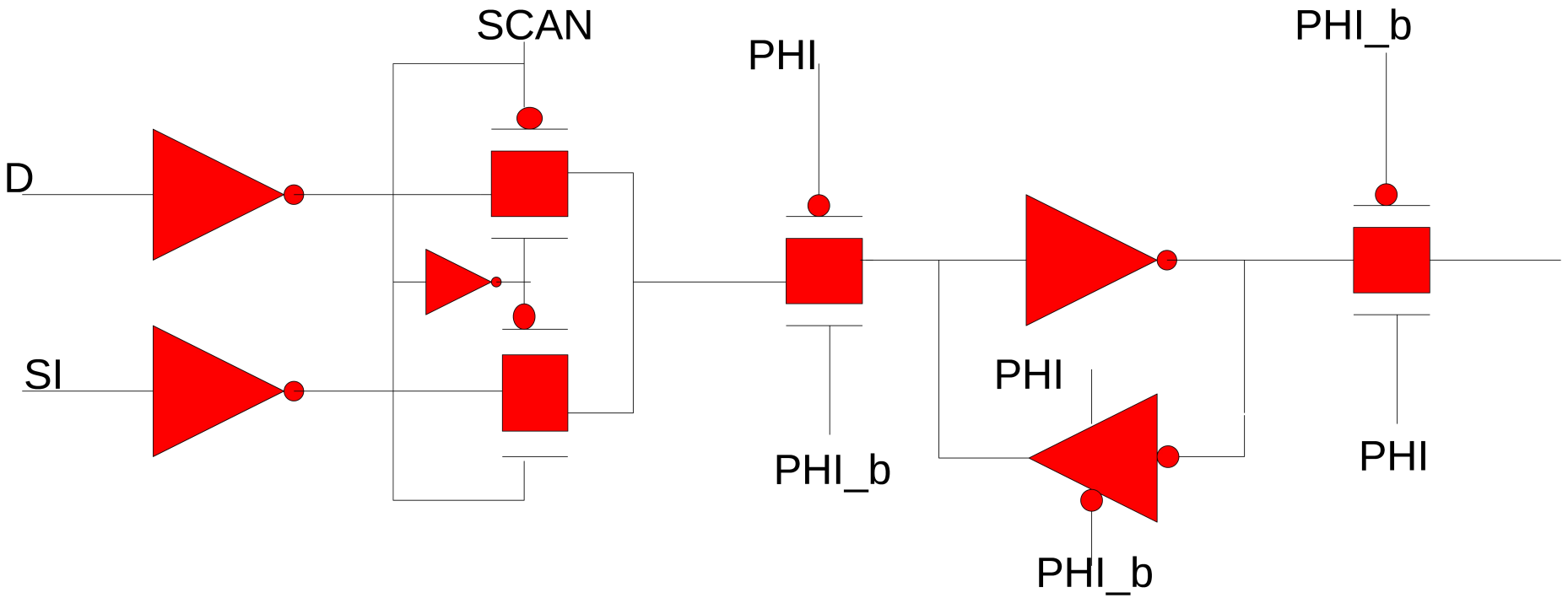
- Use only clocked D-type of flip-flops for all state variables.
- At least one PI pin must be available for test; more pins, if available, can be used.
- All clocks must be controlled from PIs.
- Clocks must not feed data inputs of flip-flops.

Scan Flip-Flop (SFF)

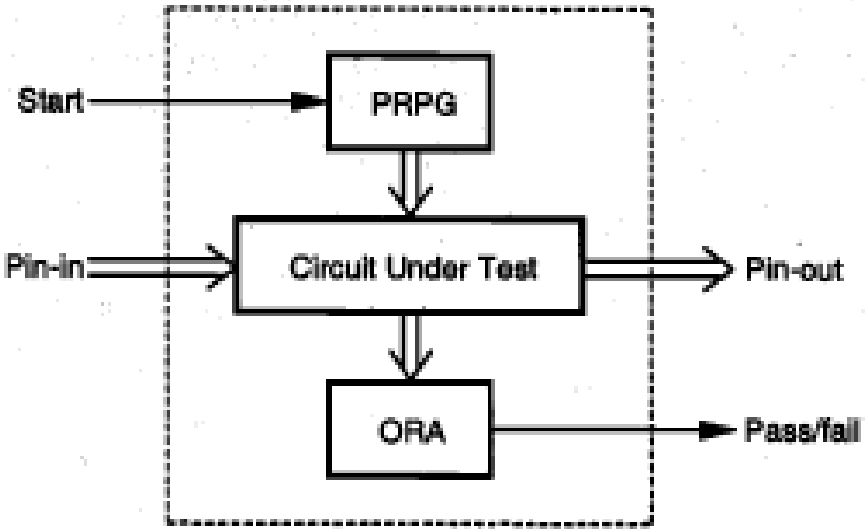




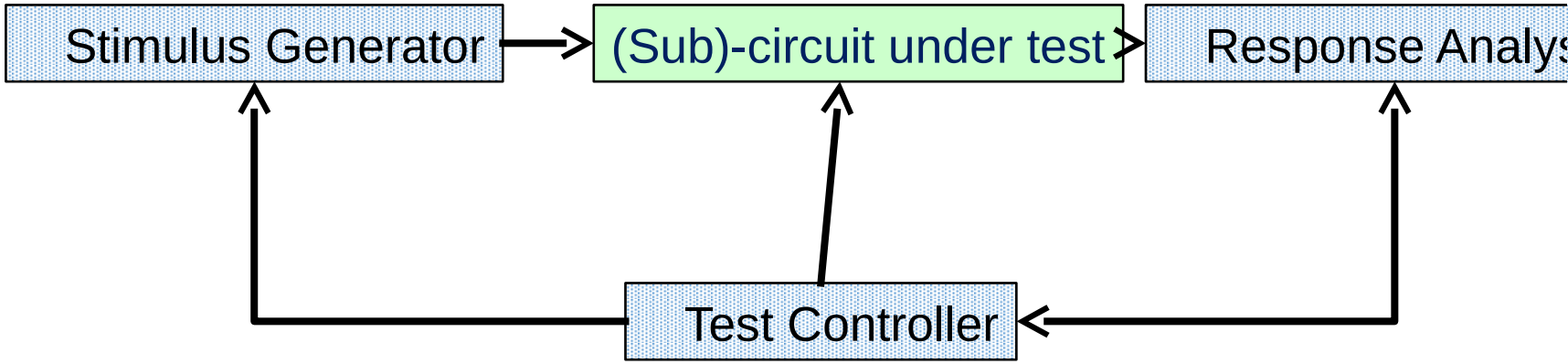
Scannable Flip-flop



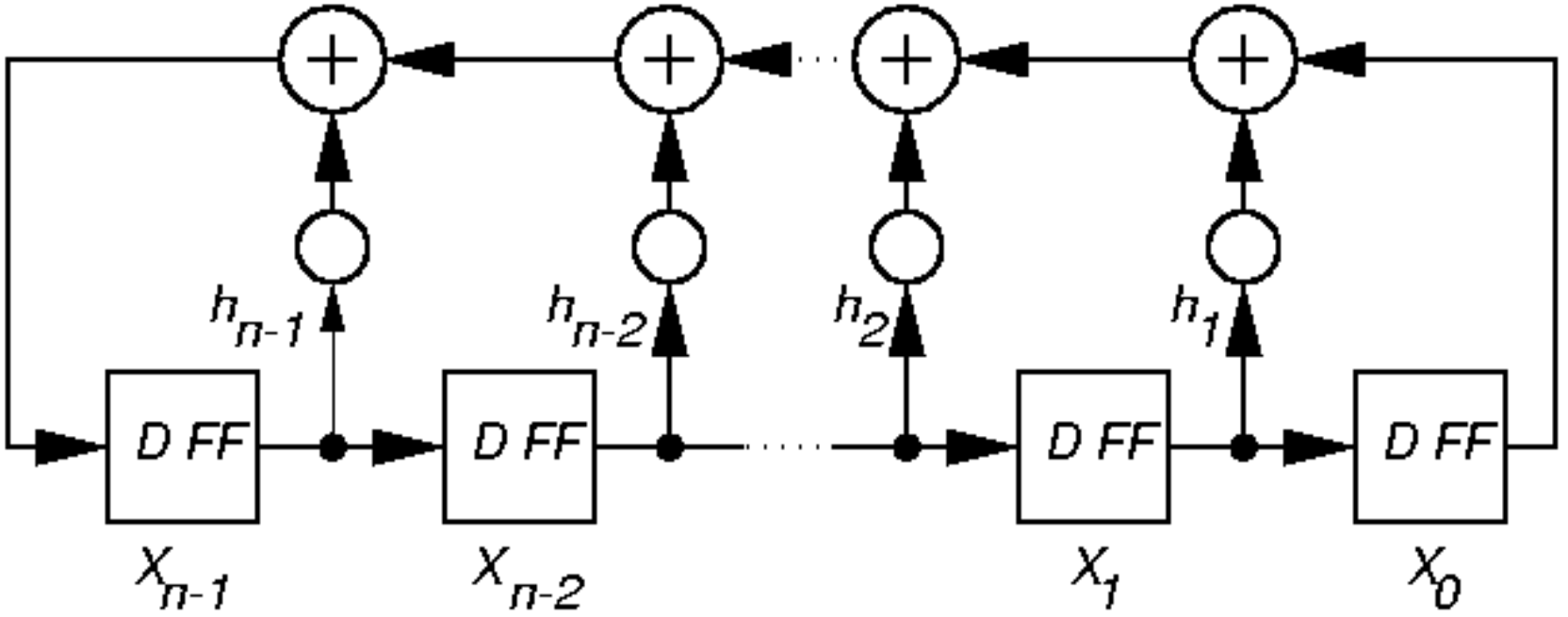
BUILT IN SELF TEST



Pseudo Random Pulse Generator
Output Response analyzer



Pseudo-Random Pattern Generation

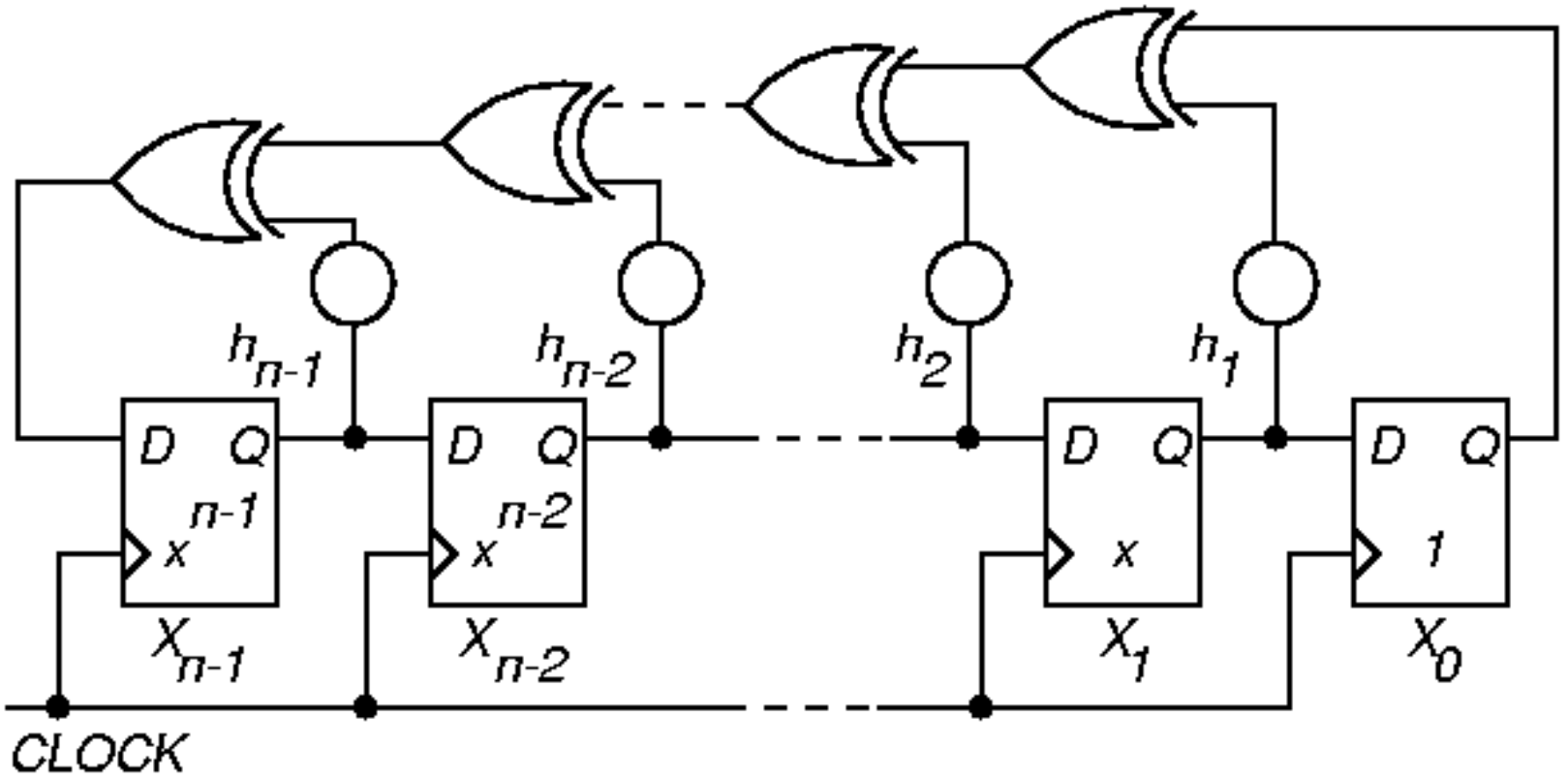


- *Standard Linear Feedback Shift Register* (LFSR)
  - Produces patterns algorithmically – repeatable
  - Has most of desirable random number properties
- Need not cover all 2^n input combinations
- Long sequences needed for good fault coverage

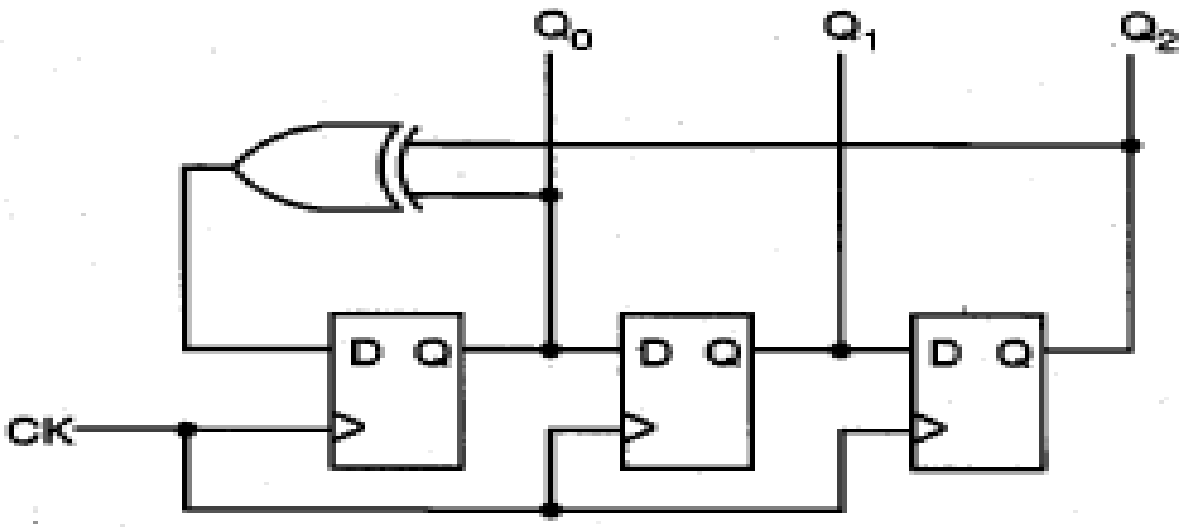
LFSR Implements a Galois Field

- Galois field (mathematical system):
  - Addition operator is XOR ($\oplus$)
- T_s companion matrix:
  - First column – all 0, except nth element which is always 1 (X_0 always feeds X_{n-1})
  - Last row – n feedback coefficients h_i
  - Rest is identity matrix I – means a right shift
- Near-exhaustive (maximal length) LFSR
  - Cycles through $2^n - 1$ states (excluding all-0)
  - 1 pattern of n 1's, one of n-1 consecutive 0's

Standard *n*-Stage LFSR Implementation

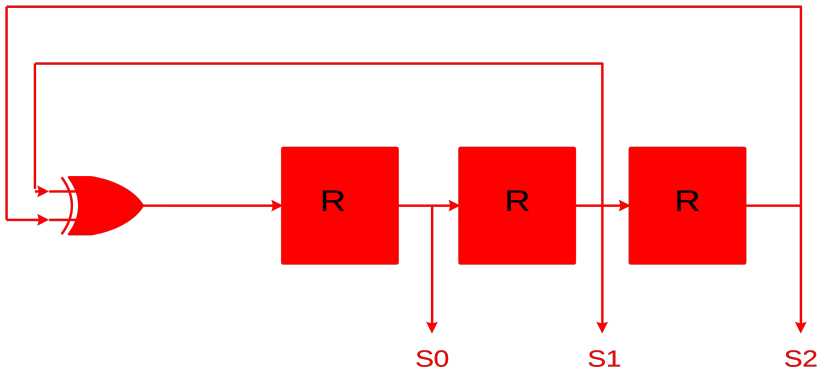


Analyze this one



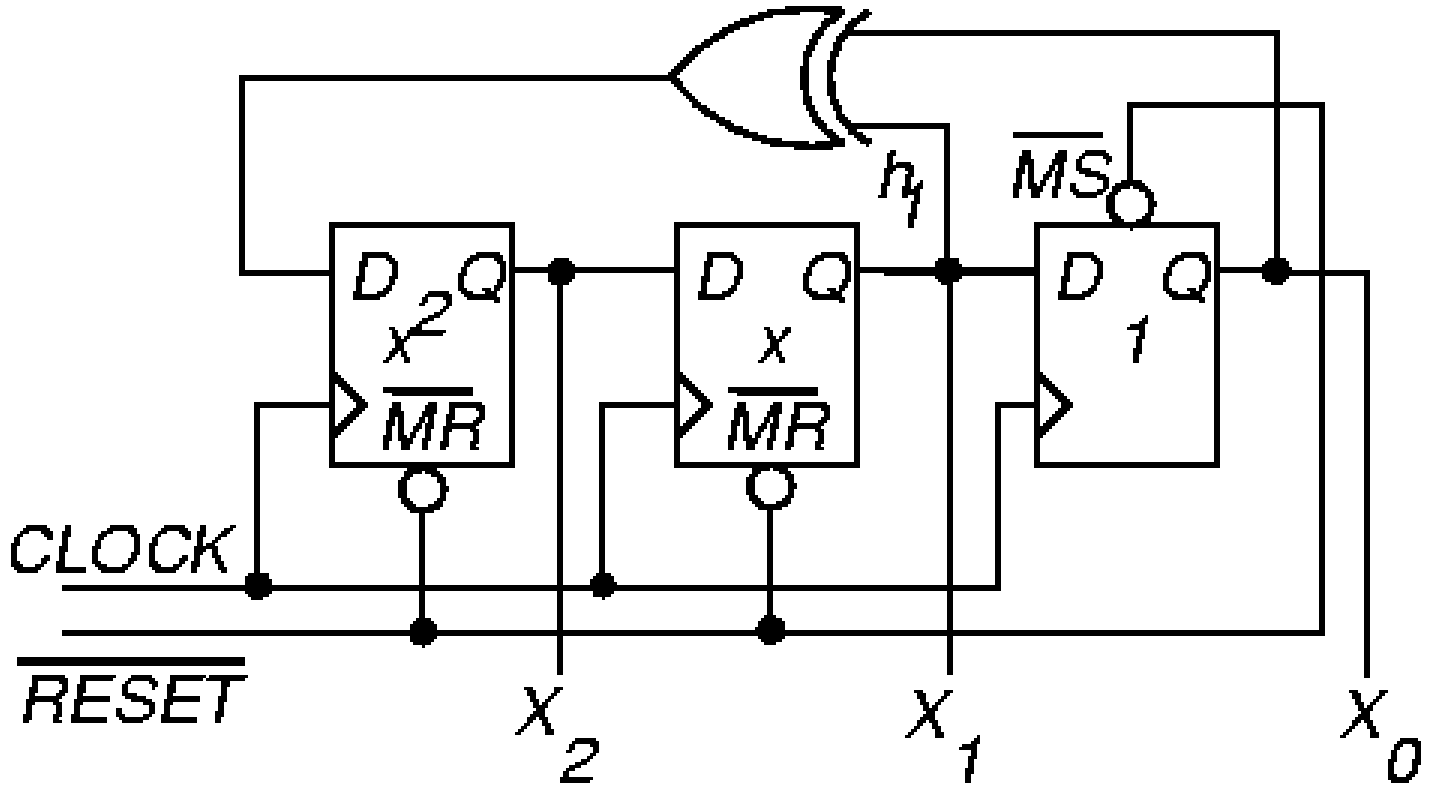
A pseudo-random sequence generator using LFSR.

Example of LFSR



S0	S1	S2
1	0	0
0	1	0
1	0	1
1	1	0
1	1	1
0	1	1
0	0	1
1	0	0

Example External XOR LFSR



- Characteristic polynomial $f(x) = 1 + x + x^3$ (read taps from right to left)

Built-In Self-Testing
Response Compaction

- Motivation and economics
- Definitions
- BIST response compaction (RC)
- BILBO
- Example
- Summary

Response Compaction

- Large 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.

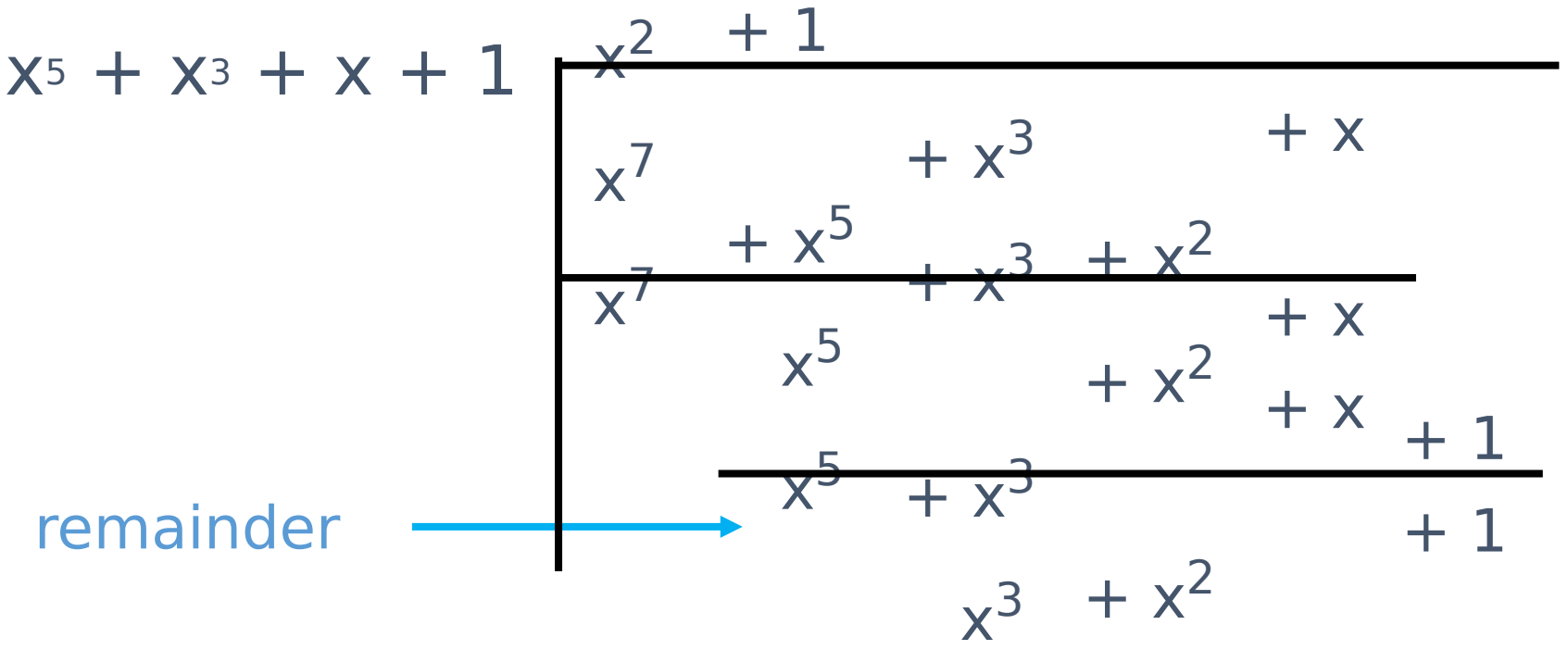
Definitions

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

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.

Symbolic Polynomial Division

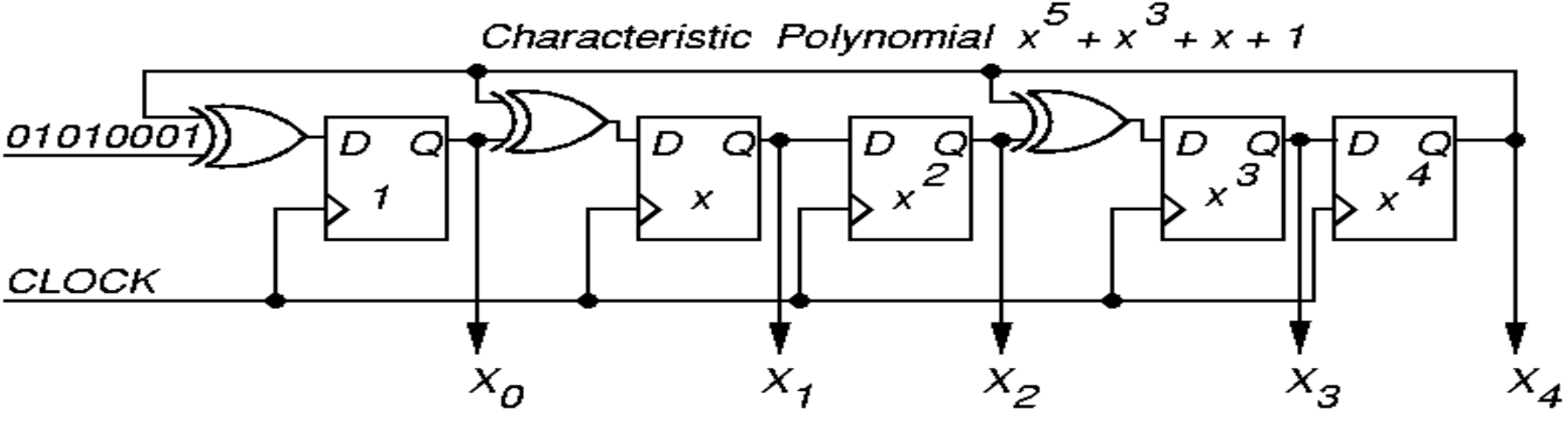


Remainder matches that from logic simulation of the response compactor!

Example: Modular LFSR Response Compactor for Signature Analysis

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



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

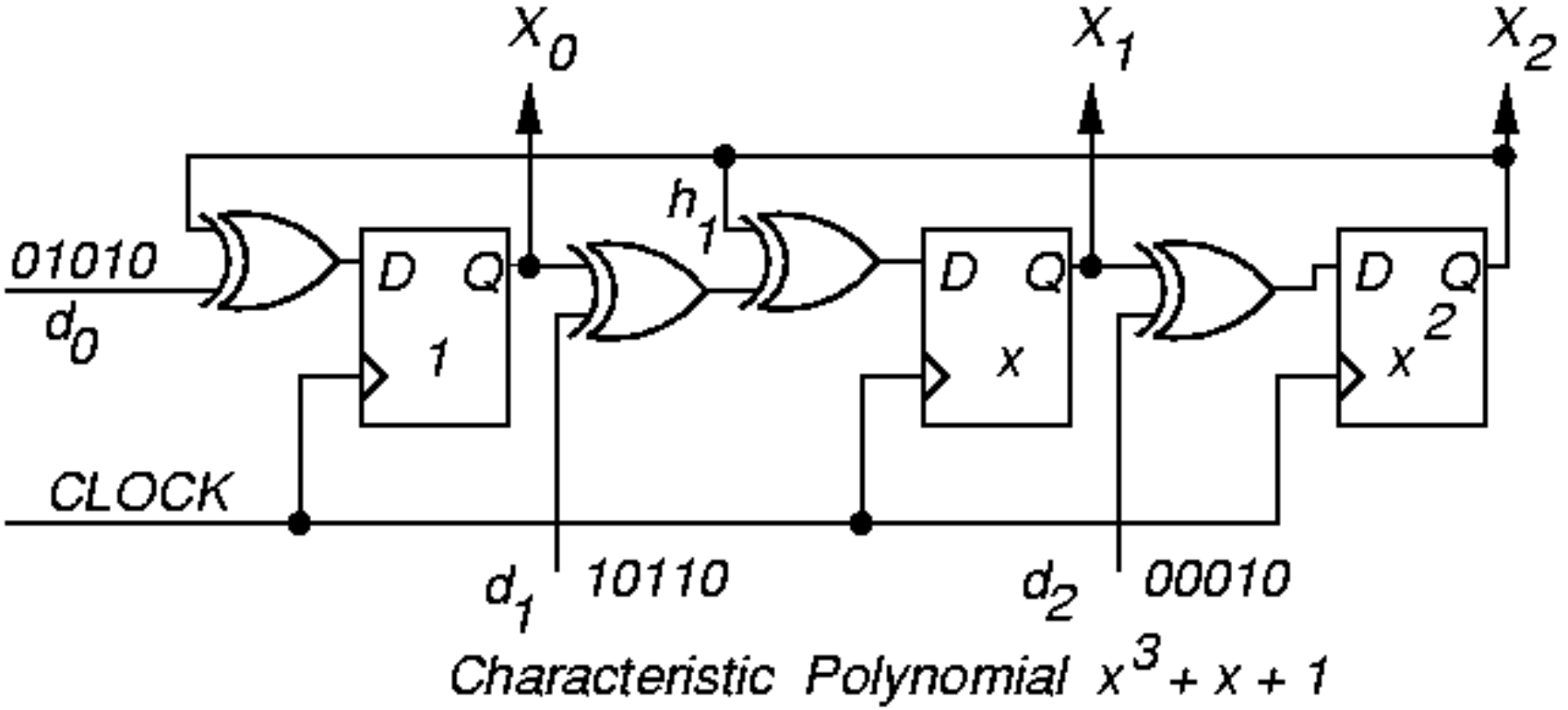
0 1 0 1 0 0 0 1

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

Multiple-Input Signature Register (MISR)

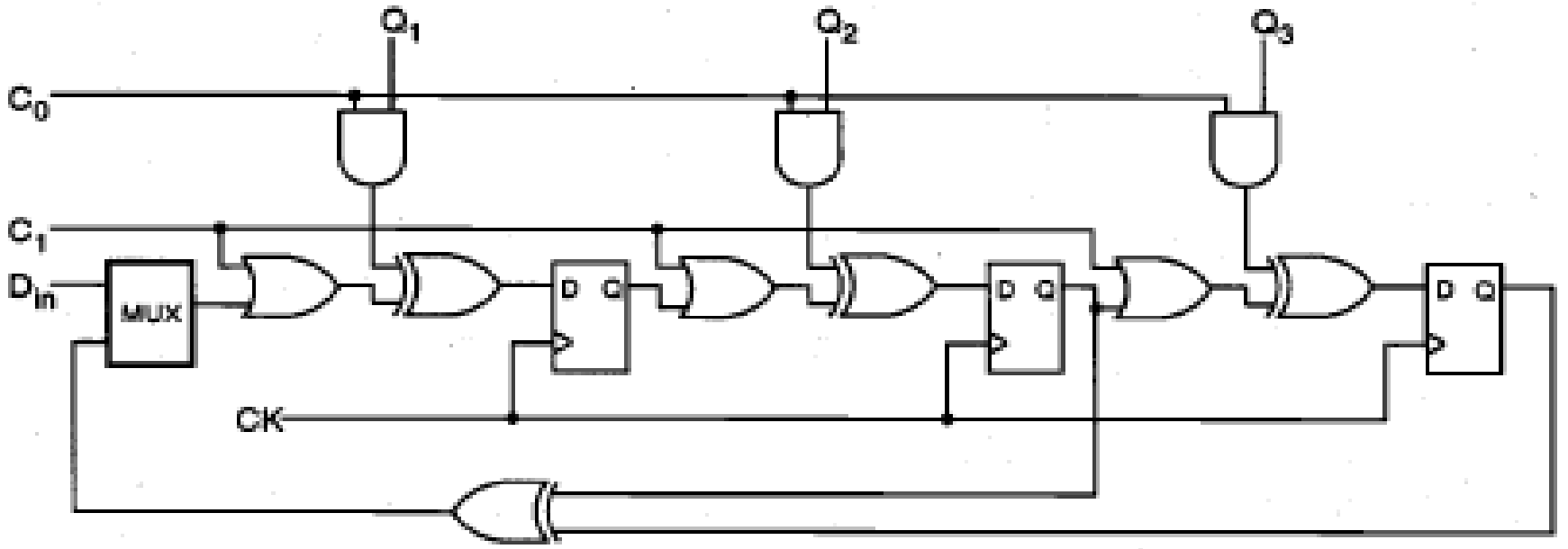
- Problem with ordinary LFSR response compactor:
  - Too much hardware if one of these is put on each primary output (PO)
- Solution: MISR – compacts all outputs into one LFSR
  - Works because LFSR is linear – obeys *superposition principle.*
  - Superimpose all responses in one LFSR – final remainder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

Modular MISR Example



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

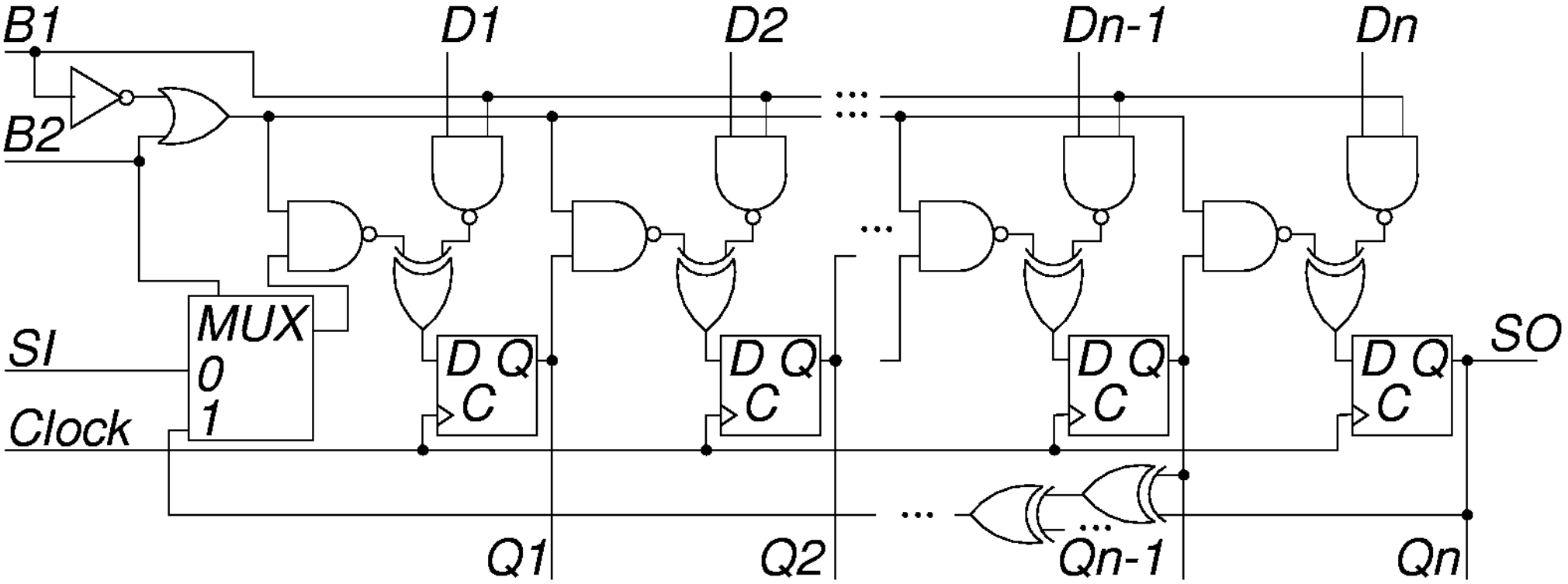
3-Bit built in Logic Observer (BILBO)



C_0	C_1	Mode
0	0	linear shift
1	0	signature analysis
1	1	data (complemented) latch
0	1	reset

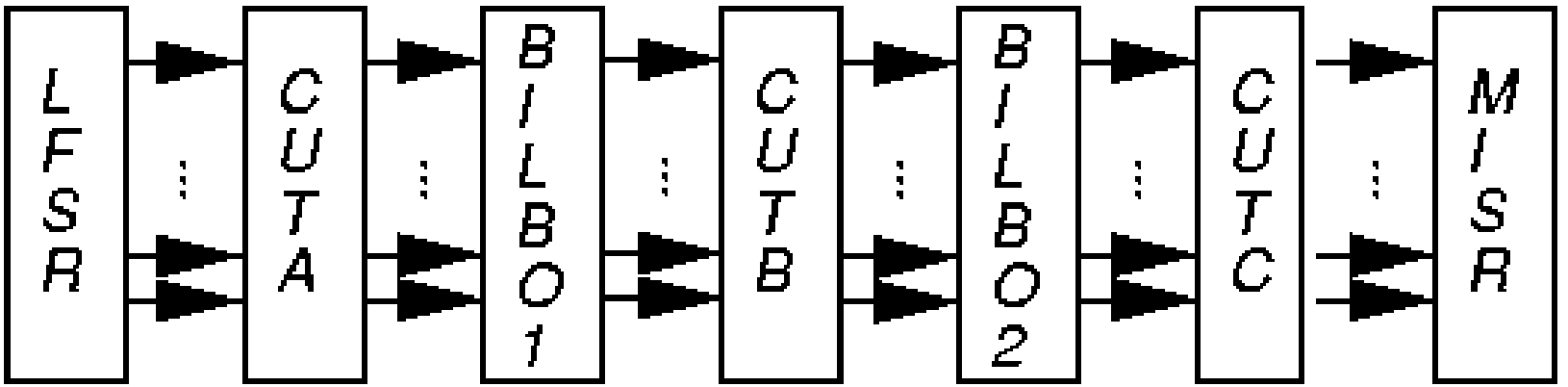
Built-in Logic Block Observer (BILBO)

- Combined functionality of D flip-flop, pattern generator, response compactor and scan chain
  - Reset all FFs to 0 by scanning in zeros.



Example: BILBO Usage

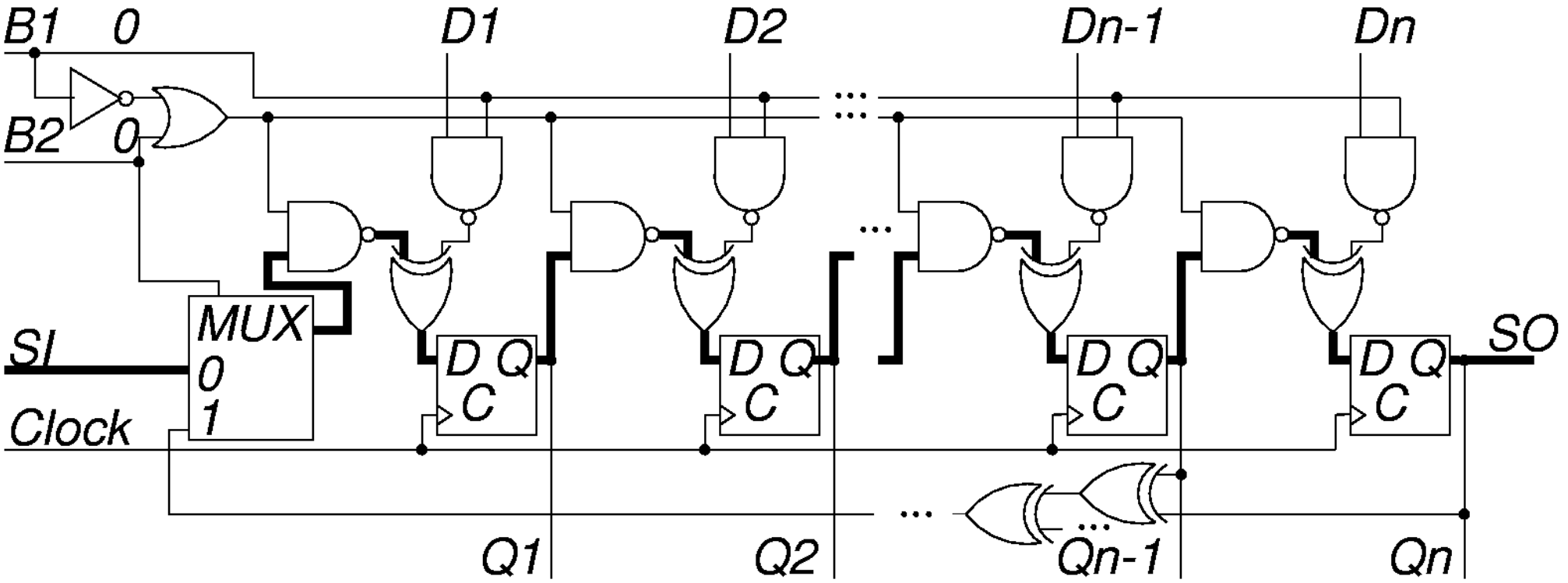
- SI – Scan In
- SO – Scan Out
  - Characteristic polynomial: $1 + x + \ldots + x^n$
  - CUTs A and C: BILBO1 is MISR, BILBO2 is LFSR
  - CUT B: BILBO1 is LFSR, BILBO2 is MISR



(a) Example test configuration.

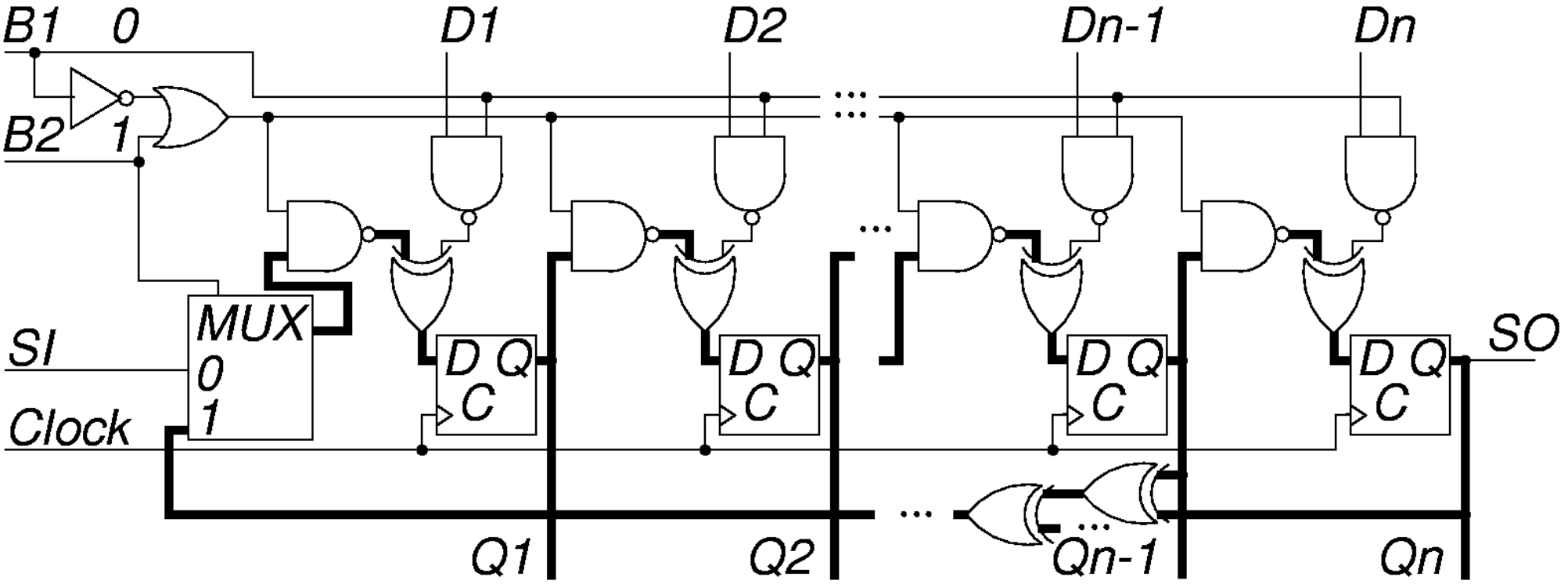
BILBO Serial Scan Mode

- *B1 B2* = “00”
  - **Dark lines** show enabled data paths.



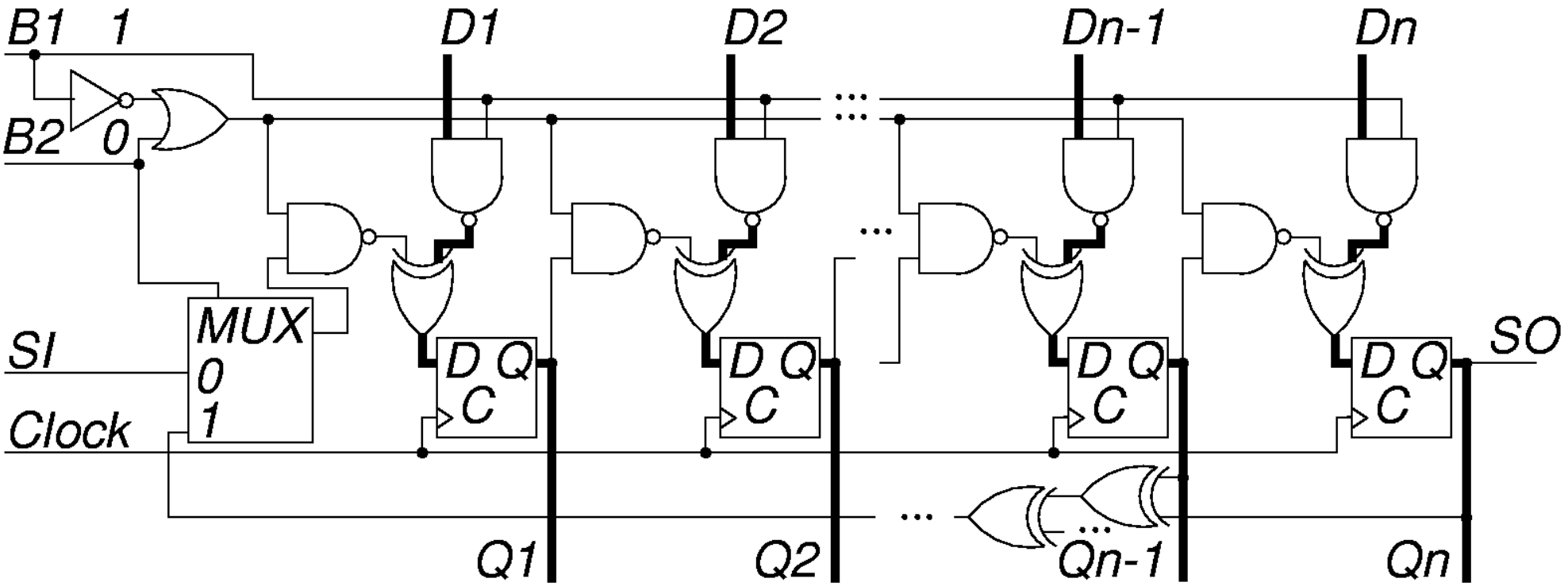
BILBO LFSR Pattern Generator Mode

- *B*1 *B*2 = “01”



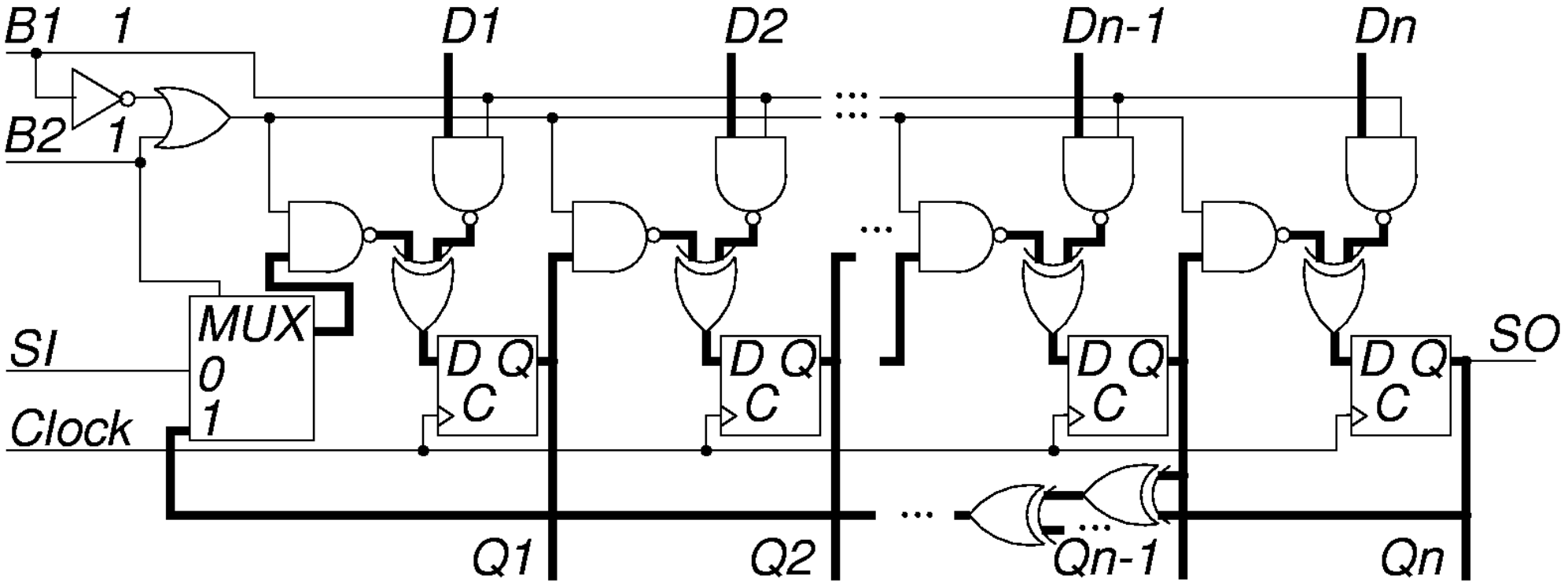
BILBO in D FF (Normal) Mode

- *B*1 *B*2 = “10”



BILBO in MISR Mode

- *B*1 *B*2 = “11”



Summary

- LFSR pattern generator and MISR response compactor – preferred BIST methods
- BIST has overheads: test controller, extra circuit delay, input MUX, pattern generator, response compactor, DFT to initialise 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