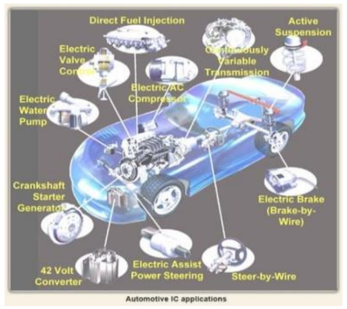
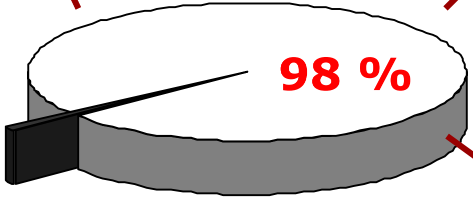
Design for Testability (DFT)

Computers and Embedded Systems

Universal computers 2%


Direct Fuel Injection
Active Suspension
Electric Valve Control
Continuously Variable Transmission
Electric AC Compressor
Electric Water Pump
Crankshaft Starter Generator
Electric Brake (Brake-by-Wire)
42 Volt Converter
Electric Assist Power Steering
Steer-by-Wire
Automotive IC applications

Embedded systems 98%



Microprocessor market shares

We notice our dependency on electronics only when it suddenly gives up to work

DFT

- A fault is **testable** if there exists a well-specified procedure
  - (e.g., test pattern generation, evaluation, and application) to expose it, and the procedure is implementable with a reasonable cost using current technologies.
- DFT techniques improve the controllability and observability of internal nodes, so that embedded functions can be tested.

1) **Controllability**, which is a measure of the difficulty of setting internal circuit nodes to 0 or 1 by assigning values to primary inputs (PIs), and
2) **Observability**, which is a measure of the difficulty of propagating a node's value to a primary output (PO)

- For sequential circuits, **predictability**, which represents the ability to obtain known output values in response to given input stimuli.

Definition

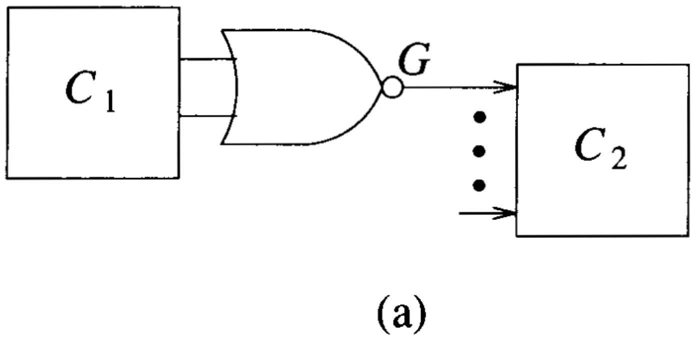
- *Design for testability* (DFT) refers to those design techniques that make the task of subsequent testing easier.
- DFT methods for digital circuits:
  - Ad-hoc methods
  - Structured methods:
    - *Scan*
    - *Partial Scan*
    - *Built-in self-test* (BIST)
    - *Boundary scan*
- DFT method for mixed-signal circuits:
    - Analog test bus

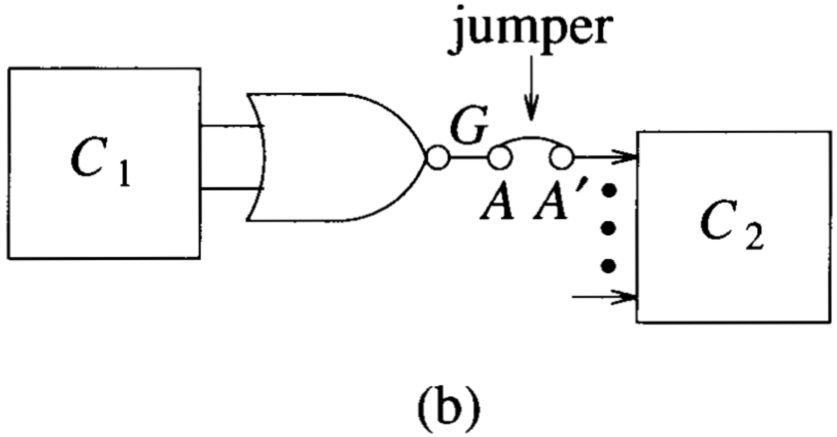
Ad-Hoc DFT Methods

- Good design practices learnt through experience are used as guidelines:
  - Avoid asynchronous (unclocked) feedback.
  - Make flip-flops initializable.
  - Avoid redundant gates. Avoid large fanin gates.
  - Provide test control for difficult-to-control signals.
  - Avoid gated clocks.
- Design reviews conducted by experts or design auditing tools.
- Disadvantages of ad-hoc DFT methods:
  - Experts and tools not always available.
  - Test generation is often manual with no guarantee of high fault coverage.
  - Design iterations may be necessary.

Ad-Hoc DFT Methods

- Test points
  - Employ test points to increase control and observability
    - CP – Control point
    - OP – Observable point

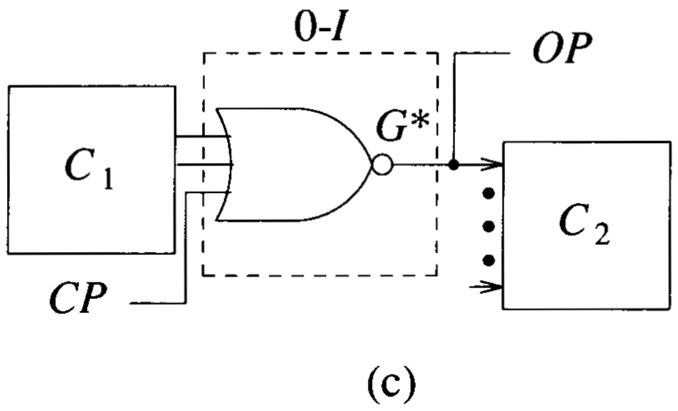


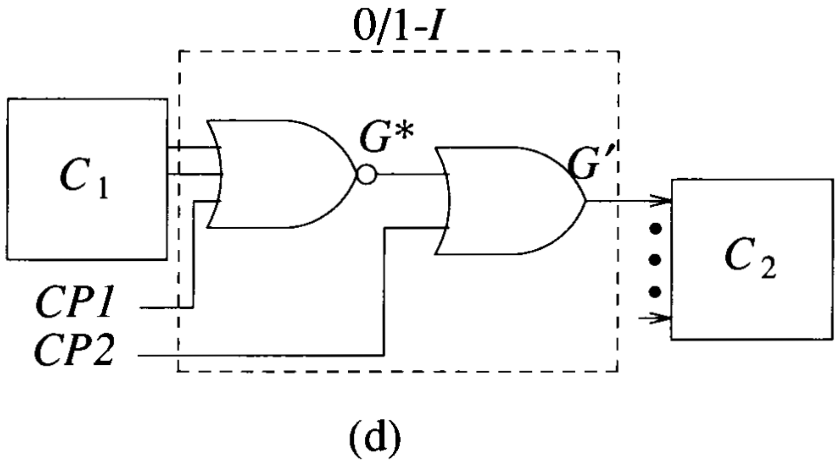


Employing test points (a) Original circuit (b) Using edge pins to achieve controllability and observability

Ad-Hoc DFT Methods

- Test points
  - Employ test points to increase control and observability
    - CP – Control point
    - OP – Observable point



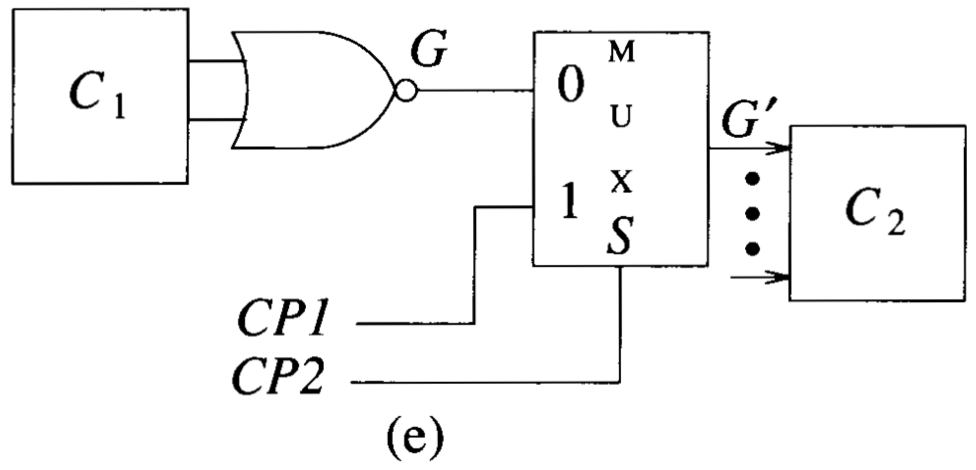


(c) Using a CP for 0-injection and an OP for observability

(d) Using a 0/1 injection circuit

Ad-Hoc DFT Methods

- Test points
  - Employ test points to increase control and observability
    - CP – Control point
    - OP – Observable point

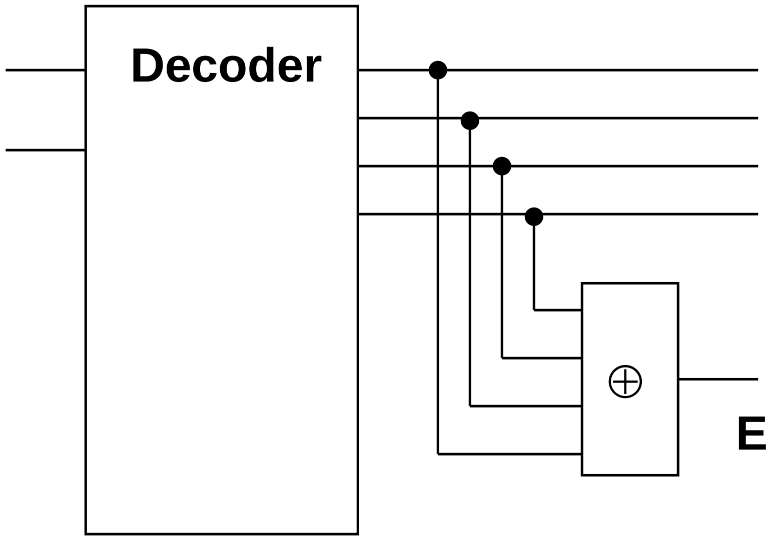


(e) Using a MUX for 0/1 injection

Ad-Hoc DFT Methods

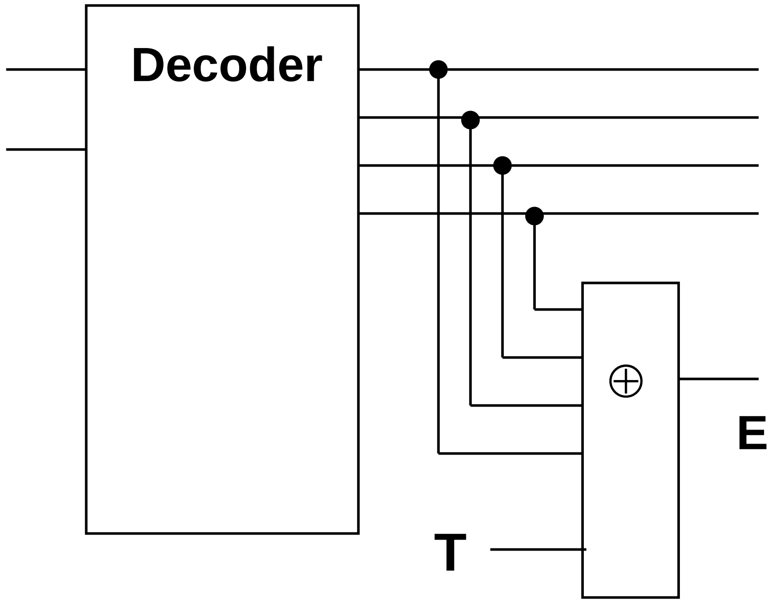
Fault redundancy:

Error control circuitry:



E = 1 if decoder is fault-free
Fault ≡ 1 not testable

Testable error control circuitry:

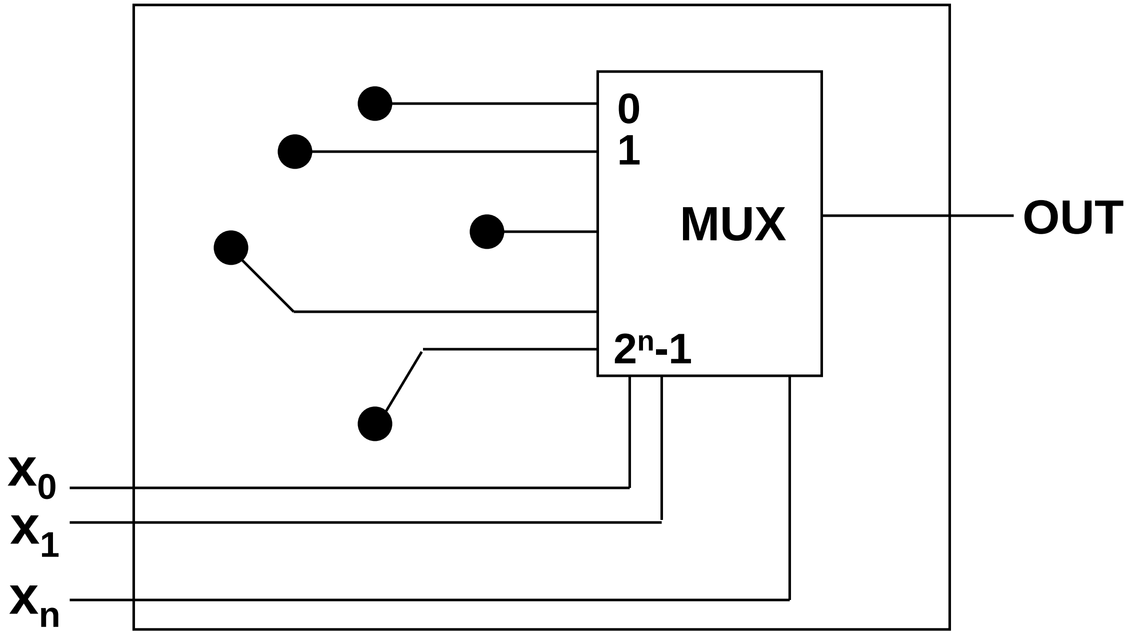


T ≡ 0 - normal working mode
T = 1 - testing mode

Ad-Hoc DFT Methods

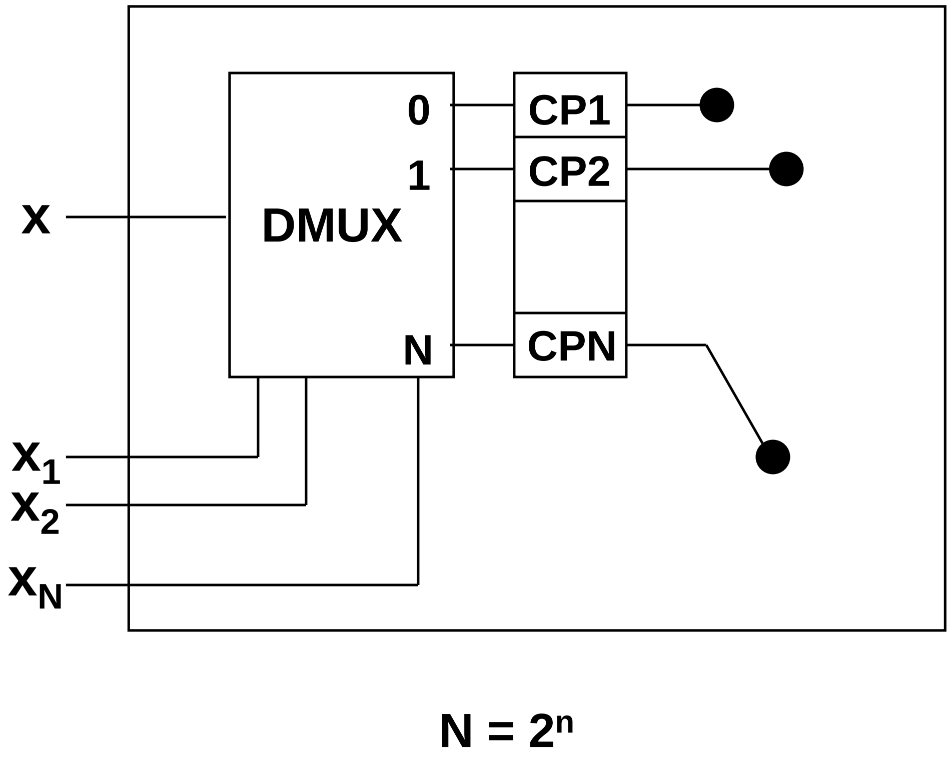
Multiplexing monitor points:

- **To reduce the number of output pins for observing monitor points, multiplexer can be used:**
- **2^n observation points are replaced by a single output and n inputs to address a selected observation point**
- <u>*Disadvantage:*</u>
- **only one observation point can be observed at a time**



Ad-Hoc DFT Methods

De-multiplexer for implementing control points:



- **To reduce the number of input pins for controlling test-points, de-multiplexer and a latch register can be used.**

Scan Design

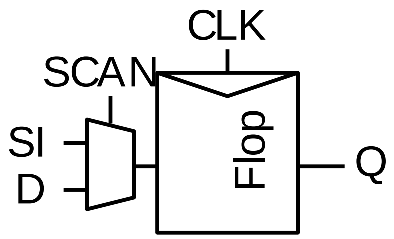
- Scan design is implemented to provide controllability and observability of internal state variables for testing a circuit.
- It is also effective for circuit partitioning.
- A scan design with full controllability and observability turns the sequential test problem into a combinational one.

Scan Design Requirements

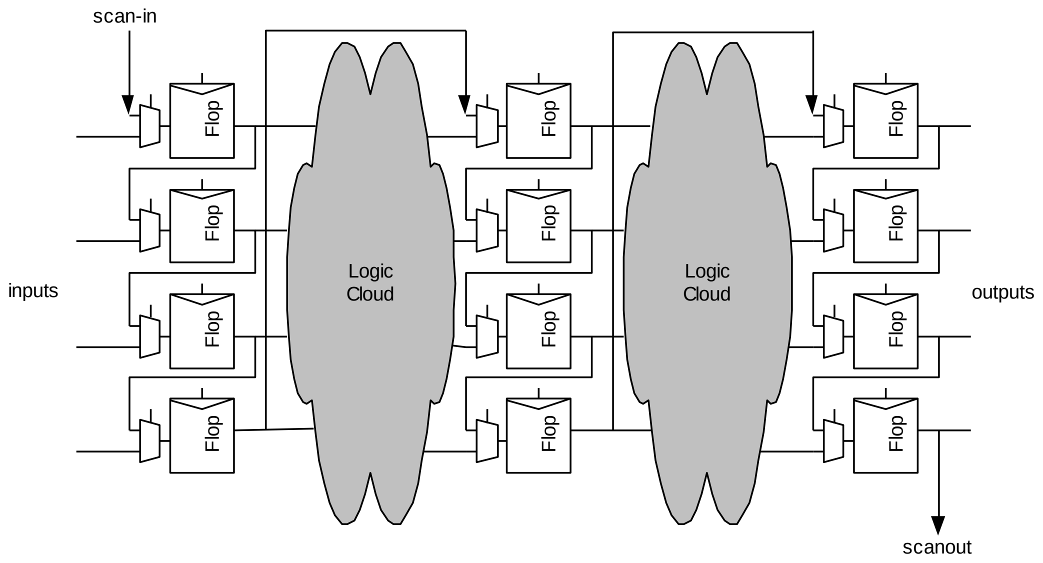
- Circuit is designed using pre-specified design rules.
- Test structure (hardware) is added to the verified design:
  - Add a *test control* (TC) primary input.
  - Add flip-flops called as *scan flip-flops* (SFF) and connect to form one or more shift registers in the test mode.
  - Make input/output of each scan shift register controllable/observable from PI/PO.

Adding Scan Structure

- Convert each flip-flop to a scan register
  - Only costs one extra multiplexer
- Normal mode: flip-flops behave as usual
- Scan mode: flip-flops behave as shift register



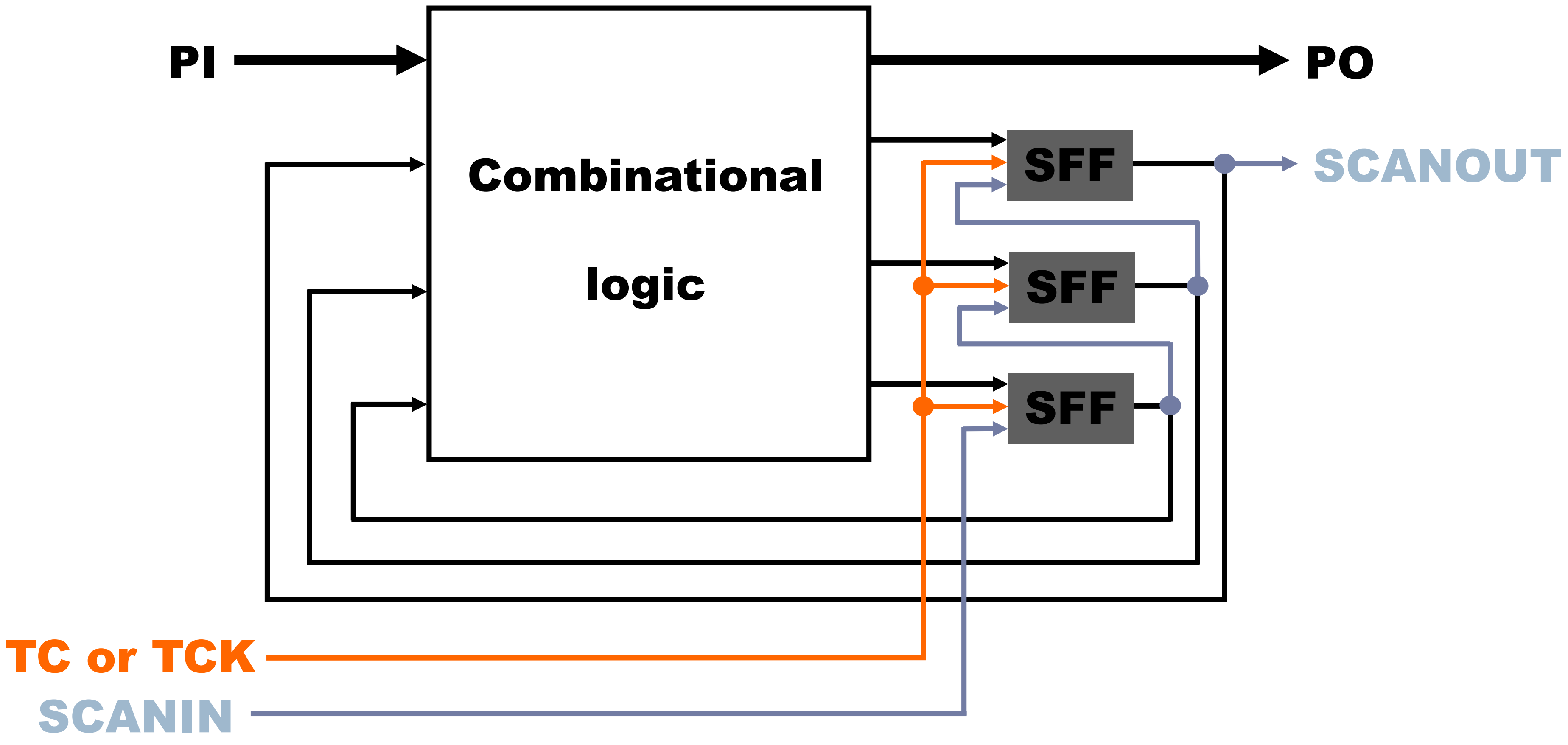
- Contents of flops can be scanned out and new values scanned in



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.

Adding Scan Structure



- **Built-In Self-Testing**

  Pattern Generation and Response Compaction

BIST Motivation

- **Useful for field test and diagnosis**

- **Hardware BIST benefits:**
  - **Lower system test effort**
  - **Improved system maintenance and repair**
  - **Improved component repair**
  - **Better diagnosis**

Costly Test Problems Alleviated by BIST

- Increasing chip logic-to-pin ratio – harder observability
- Increasingly dense devices and faster clocks
- Increasing test generation and application times
- Increasing size of test vectors stored in ATE
- Expensive ATE needed for 1 GHz clocking chips
- Hard testability insertion – designers unfamiliar with gate-level logic, since they design at behavioral level
- *In-circuit testing* no longer technically feasible
- Shortage of test engineers
- Circuit testing cannot be easily partitioned

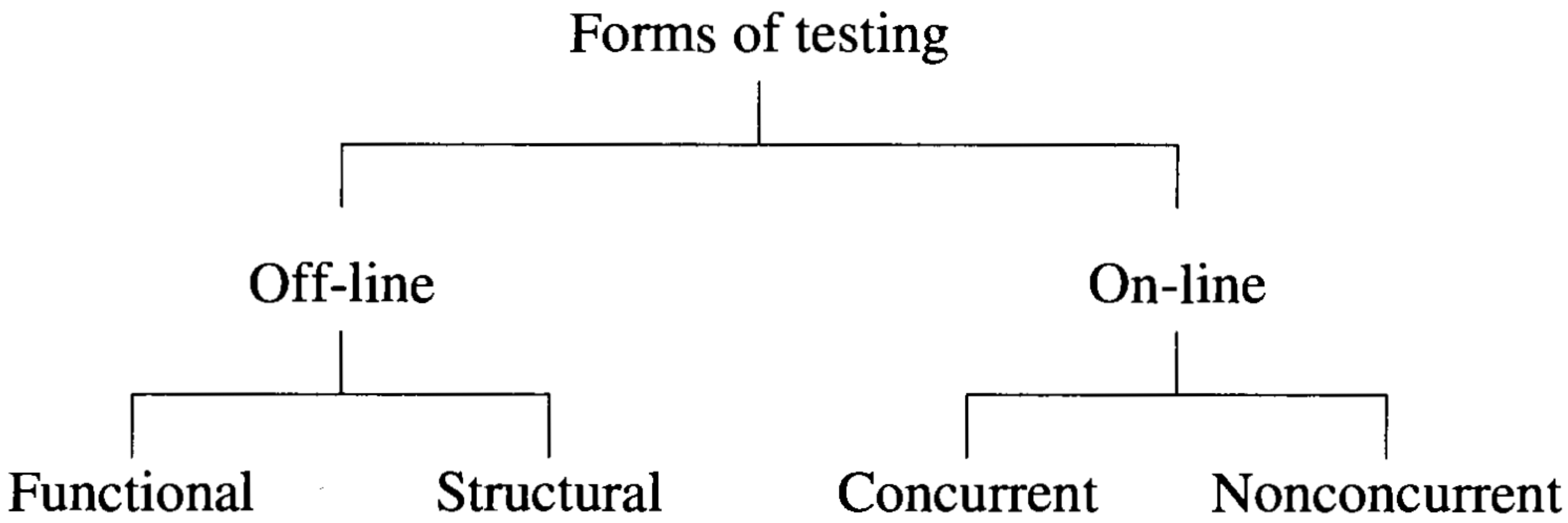
Economics – BIST Costs

- Chip area overhead for:
  - Test controller
  - Hardware pattern generator
  - Hardware response compacter
- 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 or more chips In system because of BIST
- 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

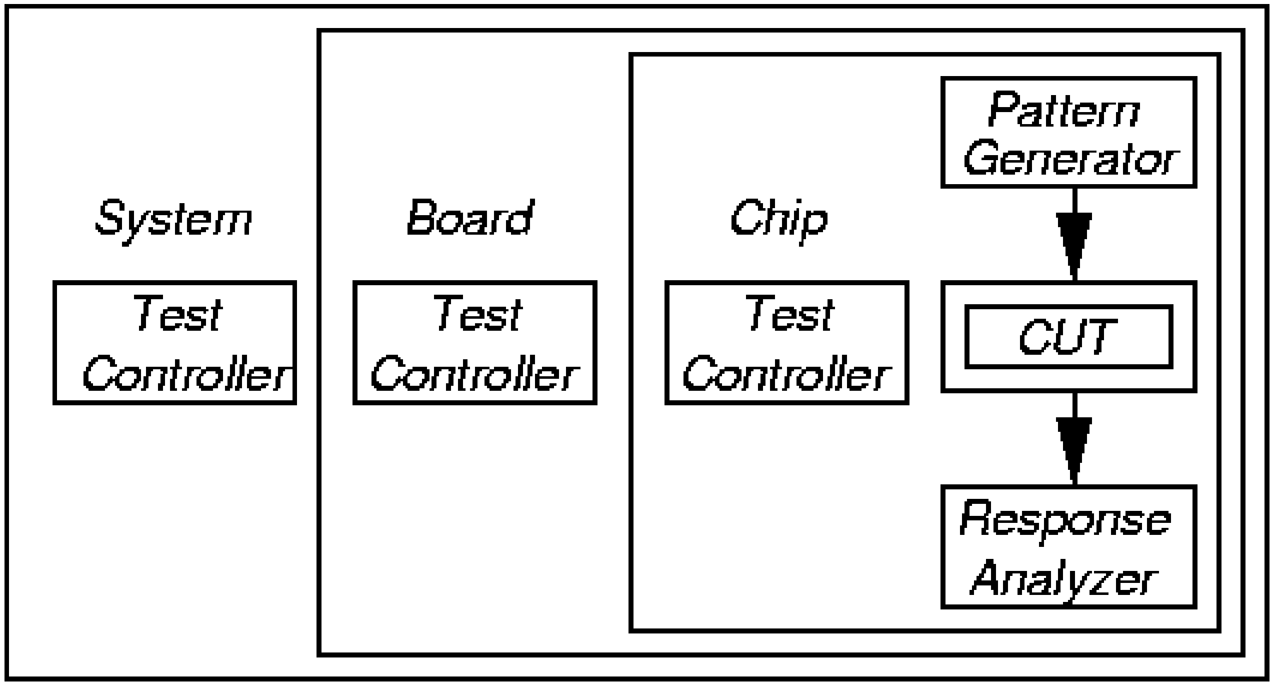
BIST – Forms of Testing



Definitions

- **BILBO** – Built-in logic block observer, extra hardware added to flip-flops so they can be reconfigured as an LFSR pattern generator or response compacter, a scan chain, or as flip-flops
- **Concurrent testing** – Testing process that detects faults during normal system operation
- **CUT** – Circuit-under-test
- **Exhaustive testing** – Apply all possible 2^n patterns to a circuit with n inputs
- **LFSR** – Linear feedback shift register, hardware that generates pseudo-random pattern sequence
- ***Signature*** **–** Any statistical circuit property distinguishing between bad and good circuits

BIST Process



- ***Test controller* – Hardware that activates self-test simultaneously on all PCBs**

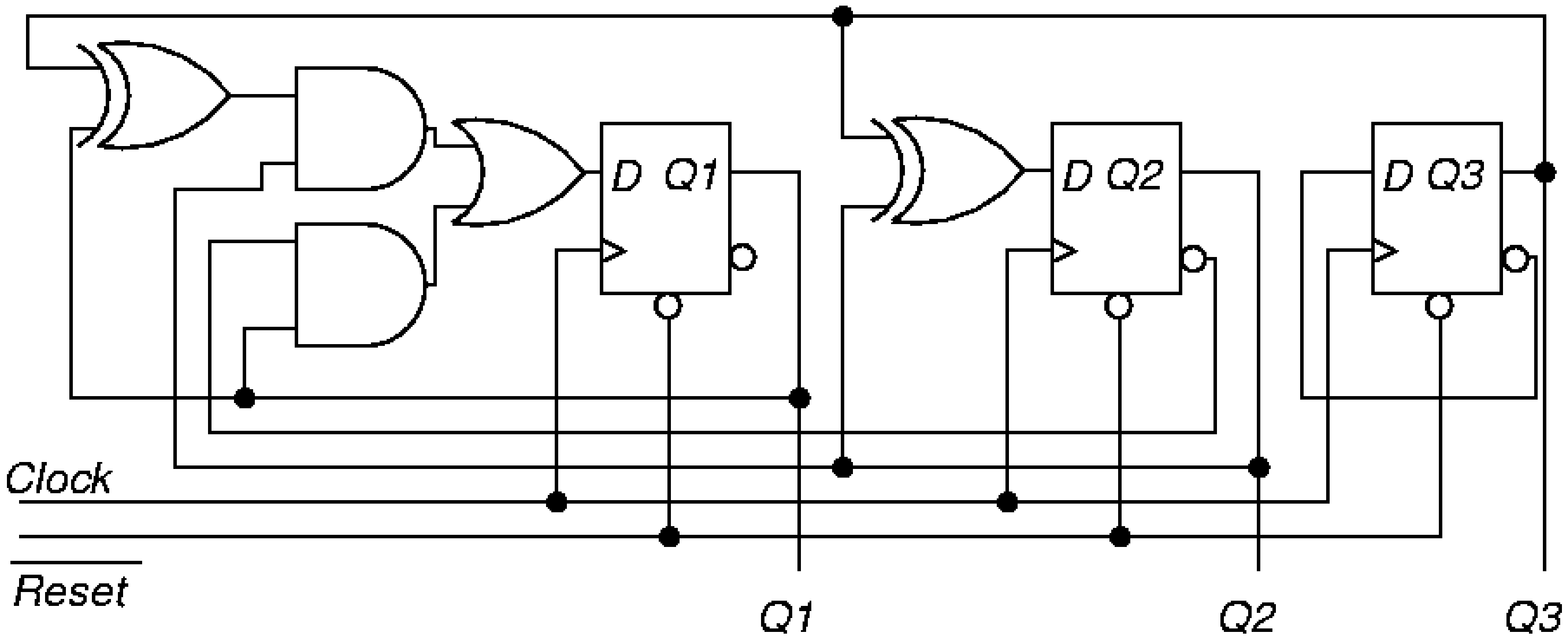
Pattern Generation

- **Store in ROM – too expensive**
- ***Exhaustive***
- ***Pseudo-exhaustive***
- ***Pseudo-random* (LFSR) – Preferred method**

Stored patterns

- An automatic test pattern generation (ATPG) and fault simulation technique is used to generate the test patterns.
- A good test pattern set is stored in a ROM on the chip.
- When BIST is activated, test patterns are applied to the CUT and the responses are compared with the corresponding stored patterns.
- Although stored-pattern BIST can provide excellent fault coverage, it has limited applicability due to its high area overhead.

Exhaustive Pattern Generation

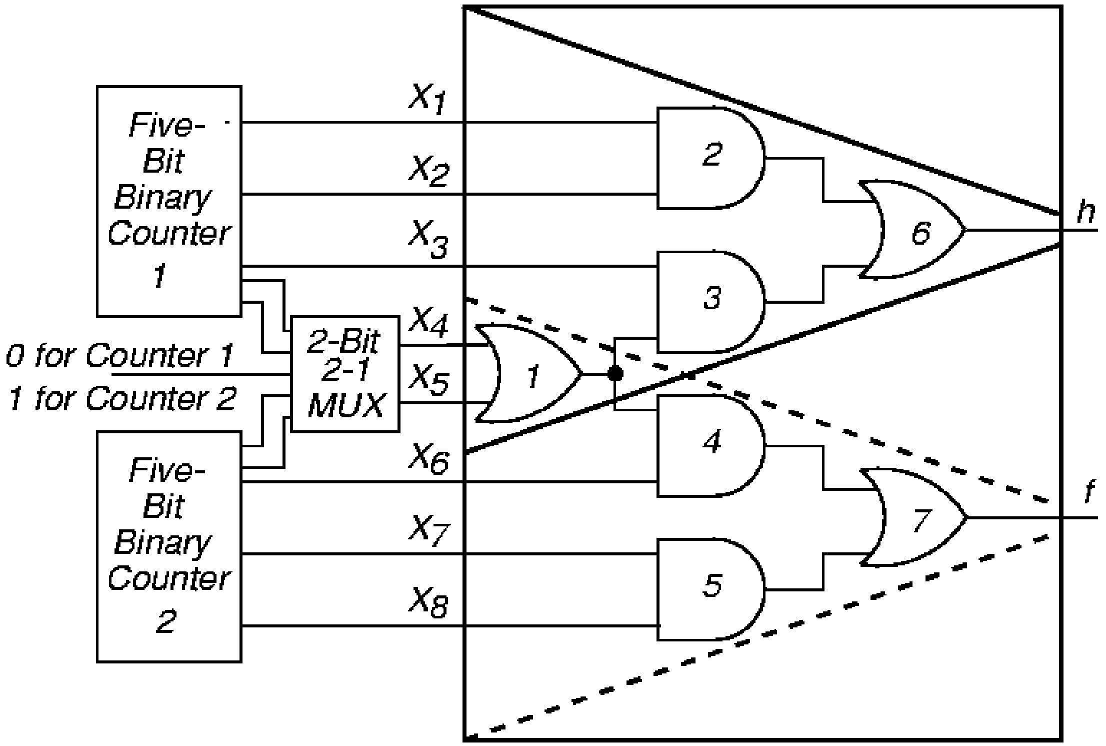


- Exhaustive pattern BIST eliminates the test generation process and has very high fault coverage.
- To test an n-input block of combinational logic, it applies all possible $\mathbf{2}^n$ -input patterns to the block.
- Impractical for $n > 20$

Pseudo-Exhaustive Method

- Partition large circuit into *fanin cones*
  - Backtrace from each PO to PIs influencing it
  - Test fanin cones in parallel

Pseudo-Exhaustive Pattern Generation



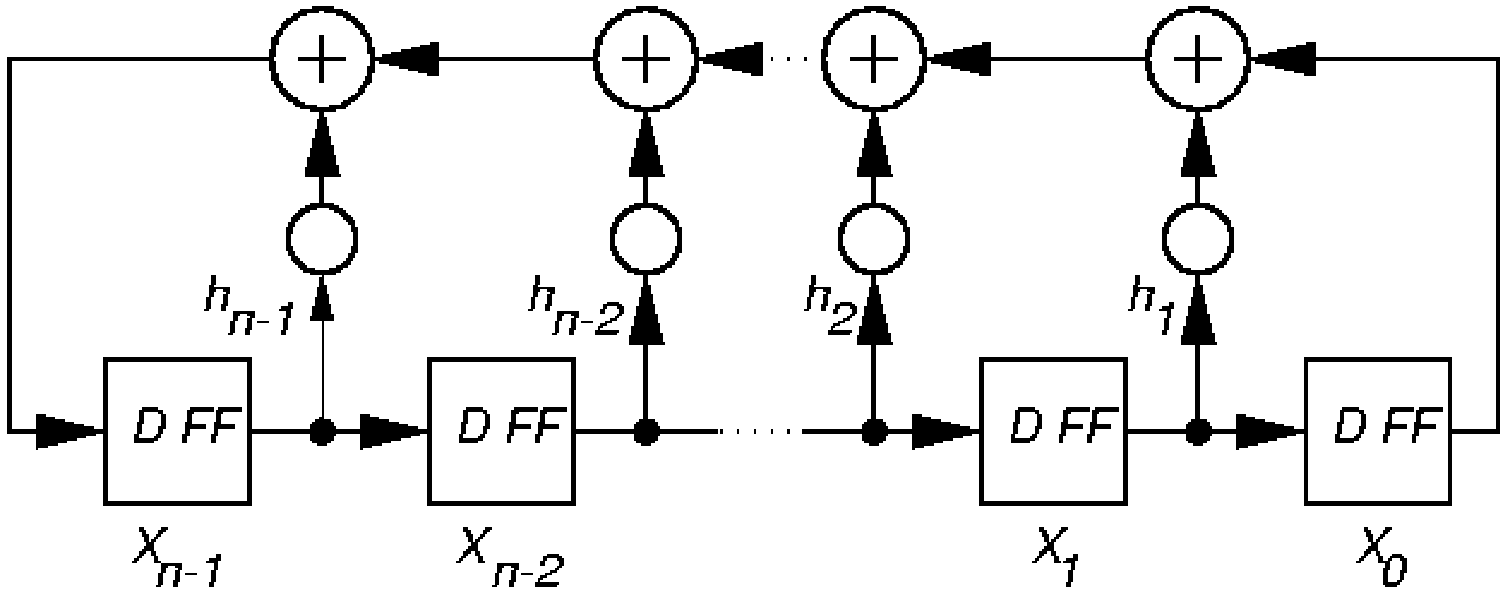
- Reduced # of tests from $2^8 = 256$ to $2^5 \times 2 = 64$

The total circuit is divided into two cones based on the cones of influence

Pseudo-Random Pattern Generation

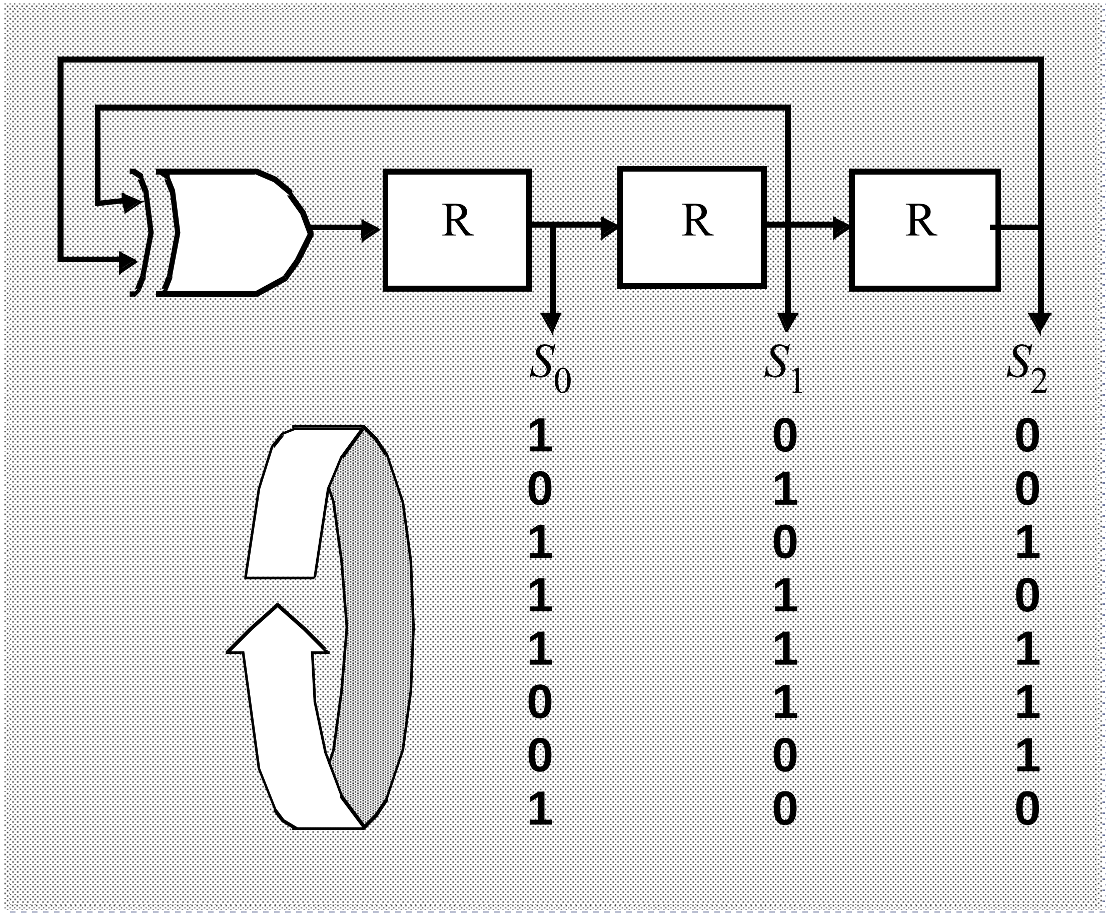
- A string of 0's and 1's is called a pseudo-random binary sequence when the bits appear to be random in the local sense, but they are in someway repeatable.
- The linear feedback shift register (LFSR) pattern generator is most commonly used for pseudo-random pattern generation

Pseudo-Random Pattern Generation



- standard, external exclusive-OR linear feedback shift register
- There are n flip-flops (Xn-1,……X0) and this is called n-stage LFSR
- This is known as a maximal length LFSR

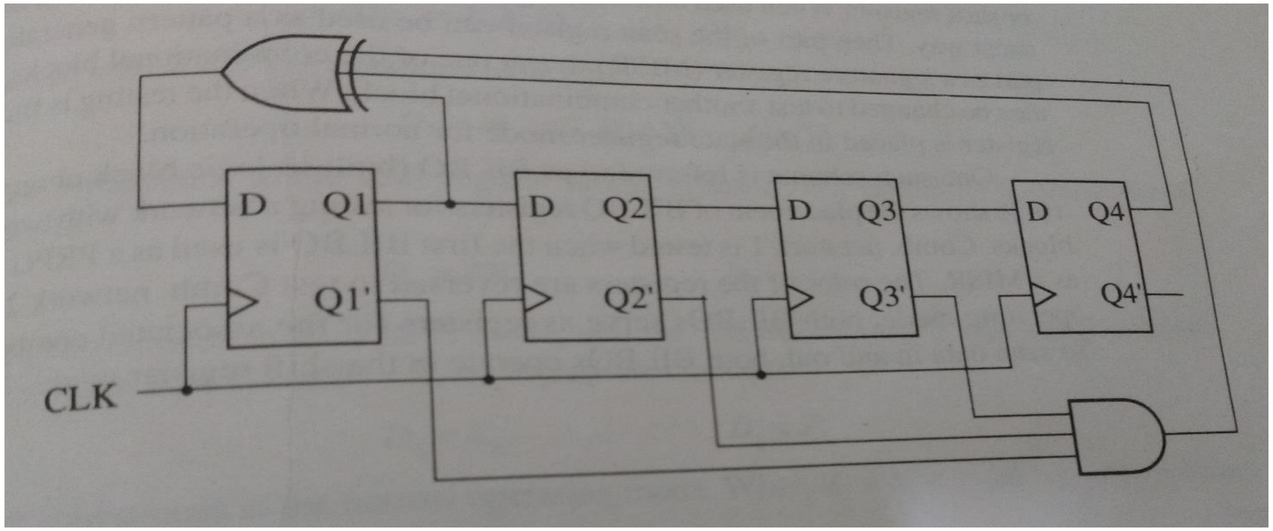
LFSR Example



- These patterns have no obvious order, and they have certain randomness properties.
- Such a LFSR known as pseudo random pattern generator.

Modified LFSR with 0000 state

- If the all 0’s test pattern is required, an n bit LFSR can be modified by adding an AND gate with n-1 inputs.
- When in state 0001, the next state is 0000; when in state 0000, the next state is 1000; otherwise the sequence is same.


D
Q1
Q1'
D
Q2
Q2'
D
Q3
Q3'
D
Q4
Q4'
CLK