Chapter 6
Design for Testability and Built-In Self-Test

Jin-Fu Li
Advanced Reliable Systems (ARES) Lab.
Department of Electrical Engineering
National Central University
Jungli, Taiwan

Outline

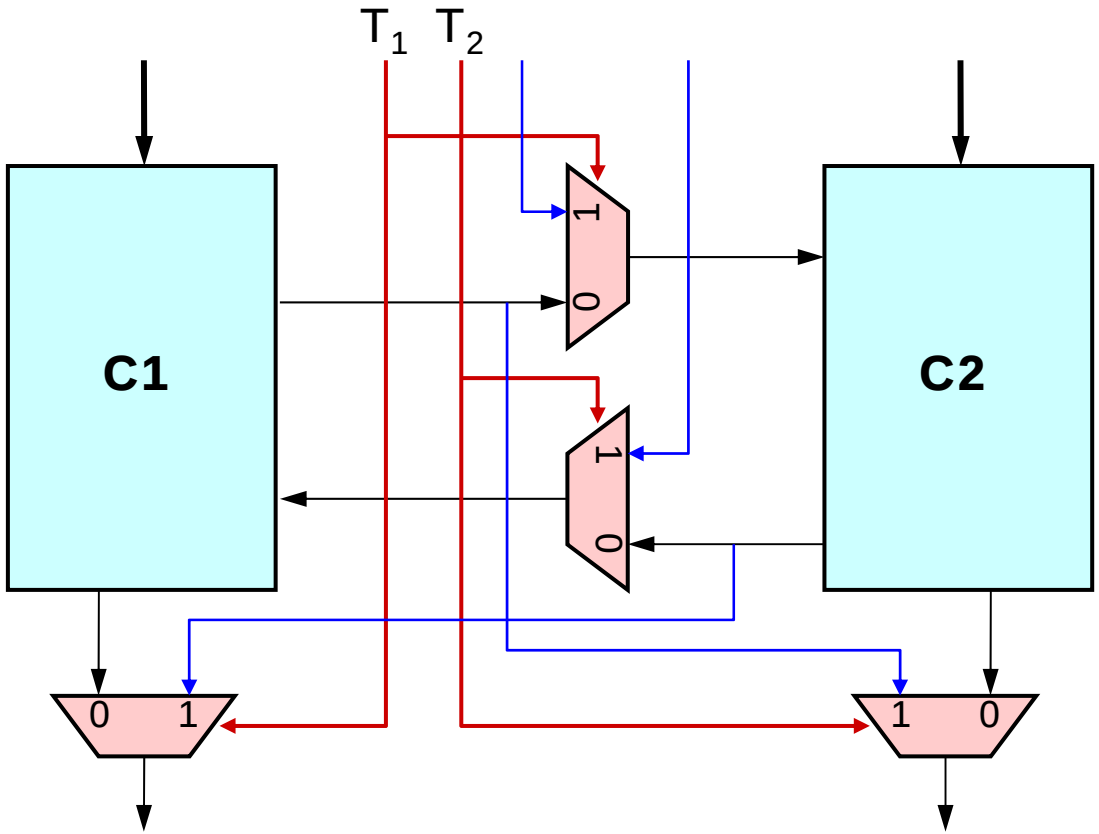
- Basics
- Design-for-Testability (DFT) Techniques
  - Ad Hoc DFT
  - Structural Methods
    - Scan
    - Partial Scan
    - BIST
    - Boundary Scan
    - Syndrome-Testable Design
    - C-Testable Design
- Built-In Self-Test (BIST) Techniques
  - Signature Analysis
  - Pseudorandom Pattern Generator (PRPG)
  - Built-In Logic Block Observer (BILBO)
- Summary

Definitions

- Definition
  - A fault is ***testable*** if there exists a well-specified procedure to expose it, which is implementable with a reasonable cost using current technologies. A circuit is ***testable with respect to a fault set*** when each and every fault in this set is testable
- Definition
  - ***Design for testability*** **(DFT)** refers to those design techniques that make test generation and test application cost-effective
- Electronic systems contain three types of components: (a) digital logic, (b) memory blocks, and (c) analog or mixed-signal circuits
- In this chapter, we discuss DFT techniques for digital logic

Ad Hoc DFT Guidelines

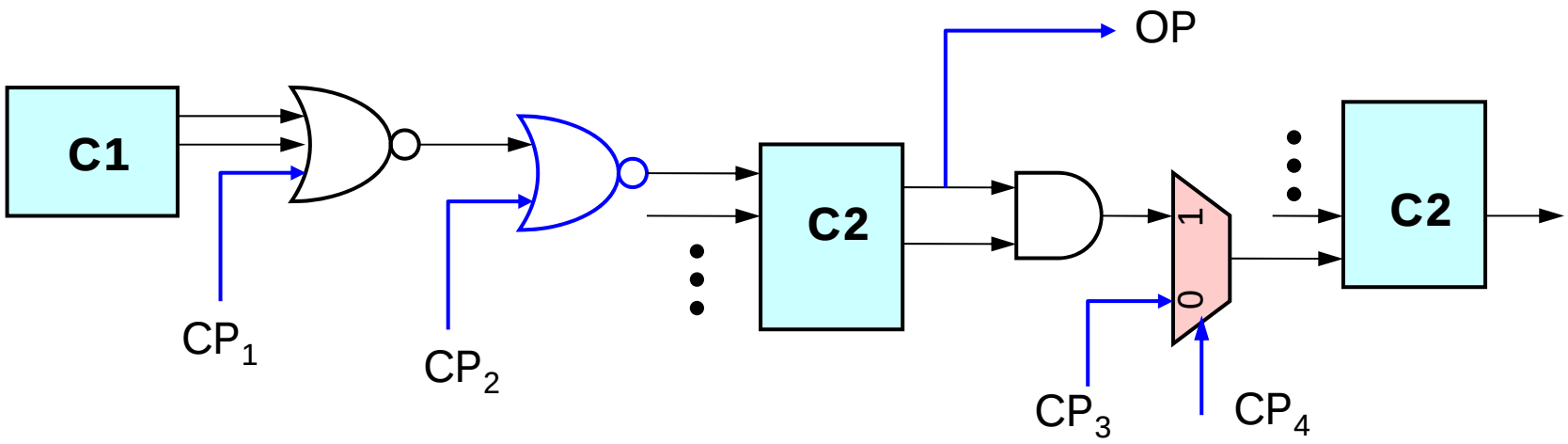
- Partition large circuits into smaller subcircuits to reduce test generation cost (using MUXed and/or scan chains)



Mode	T_1	T_2
Normal	0	0
Test C1	0	1
Test C2	1	0

Ad Hoc DFT Guidelines

- Insert test points to enhance controllability & observability
  - Test points: control points & observation points



Ad Hoc DFT Guidelines

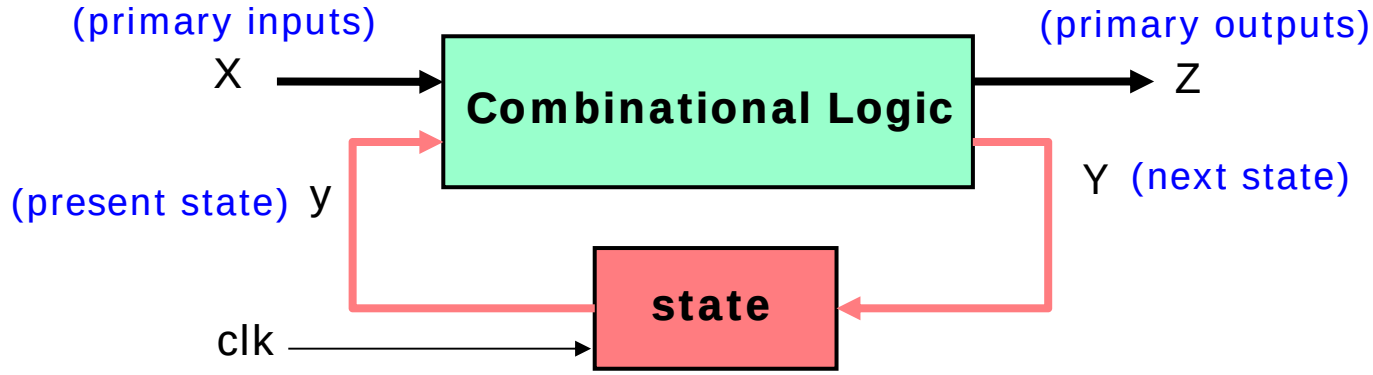
- Design circuits to be easily initializable
- Provide logic to break global feedback paths
- Partition large counter into smaller ones
- Avoid the use of redundant logic
- Keep analog and digital circuits physically apart
- Avoid the use of asynchronous logic
- Consider tester requirements (pin limitation, etc)
- Etc

Scan Design Approaches

- They are effective for circuit partitioning
- They provide controllability and observability of internal state variables for testing
- They turn the sequential test problem into a combinational one
- Four major approaches
  - Shift-register modification
  - Scan path
  - Level-sensitive scan design (LSSD)
  - Random access
- Circuit is designed using pre-specified design rules.

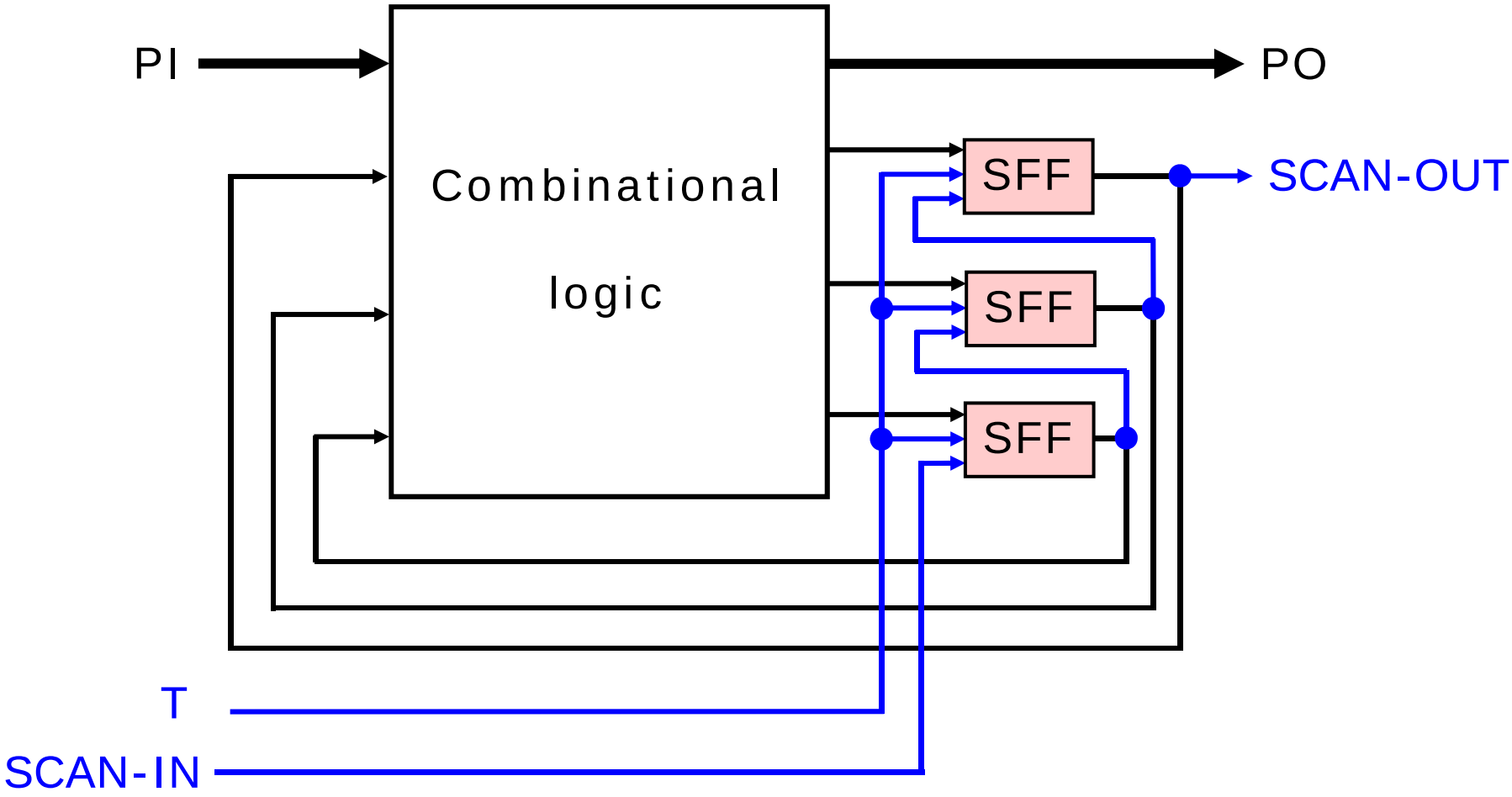
Scan Design Approaches

- Consider a representation of sequential circuits



- To make elements of state vector controllable and observable, we add
  - A TEST mode pin (T)
  - A SCAN-IN pin (SI)
  - A SCAN-OUT pin (SO)
  - A MUX (switch) in front of each FF (M)

Adding Scan Structure

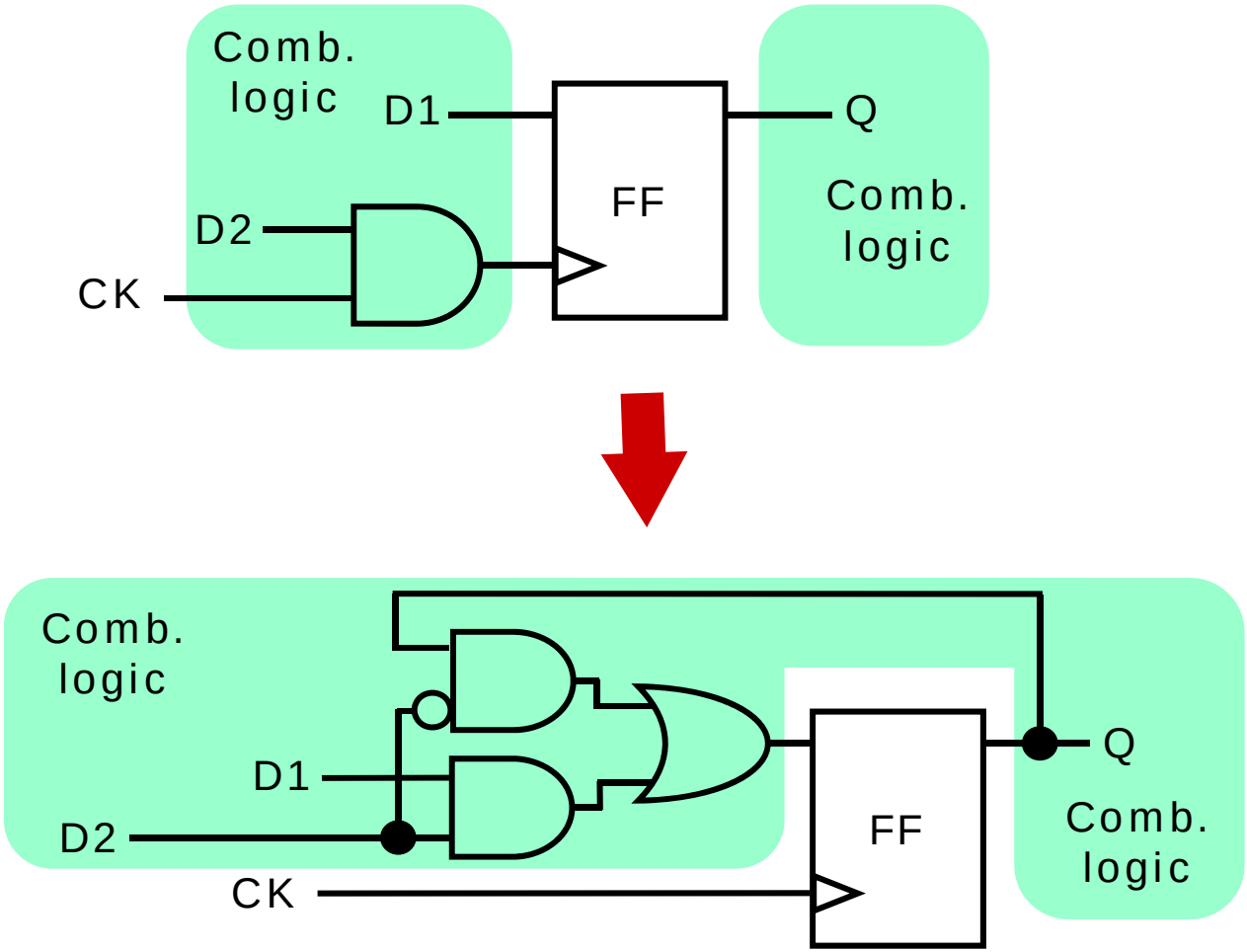


Scan Test Generation & Design Rules

- Test pattern generation
  - 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

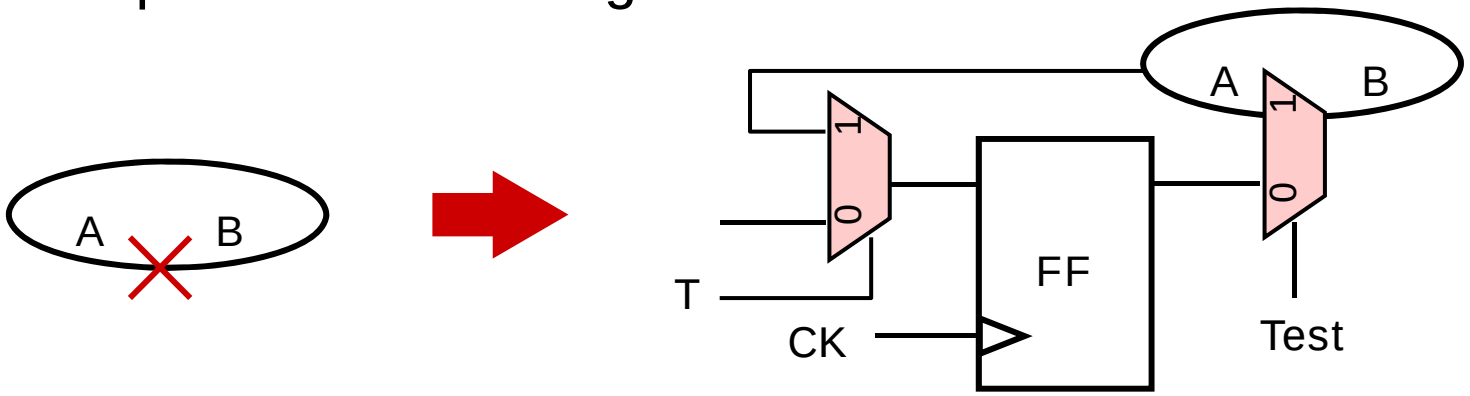
Correcting a Rule Violation

- All clocks must be controlled from PIs

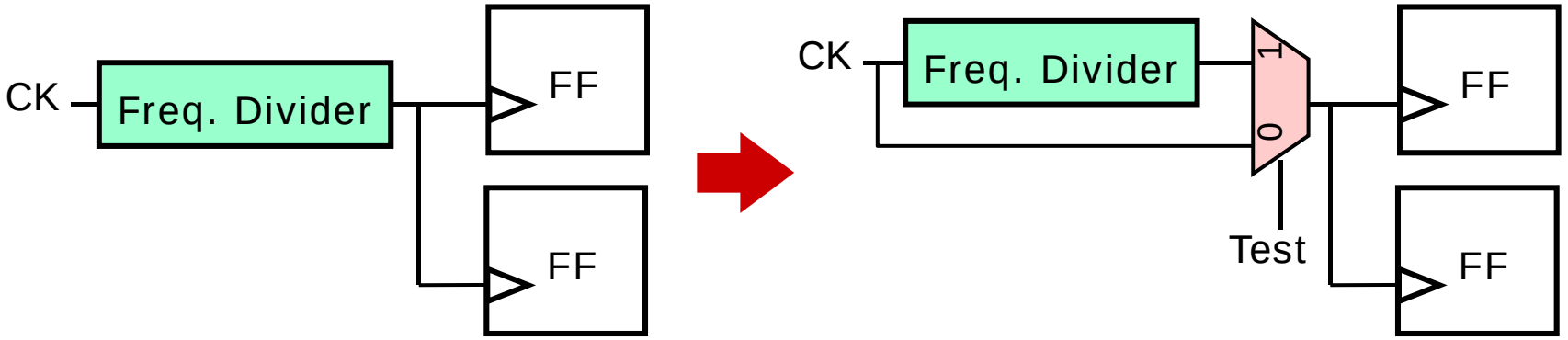


Correcting a Rule Violation

- Adding a scan FF and a mux allows a feedback loop to be opened for testing

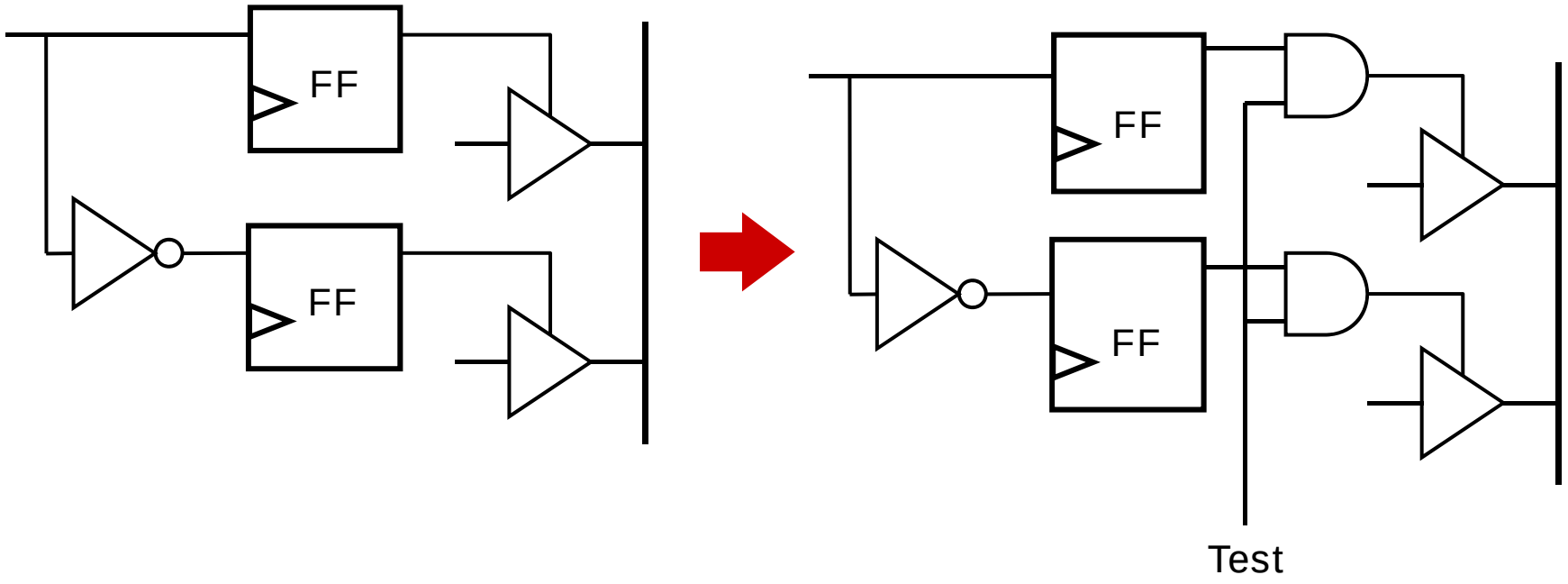


- Testing derived clocks requires the use of a mux to bypass the division stages



Correcting a Rule Violation

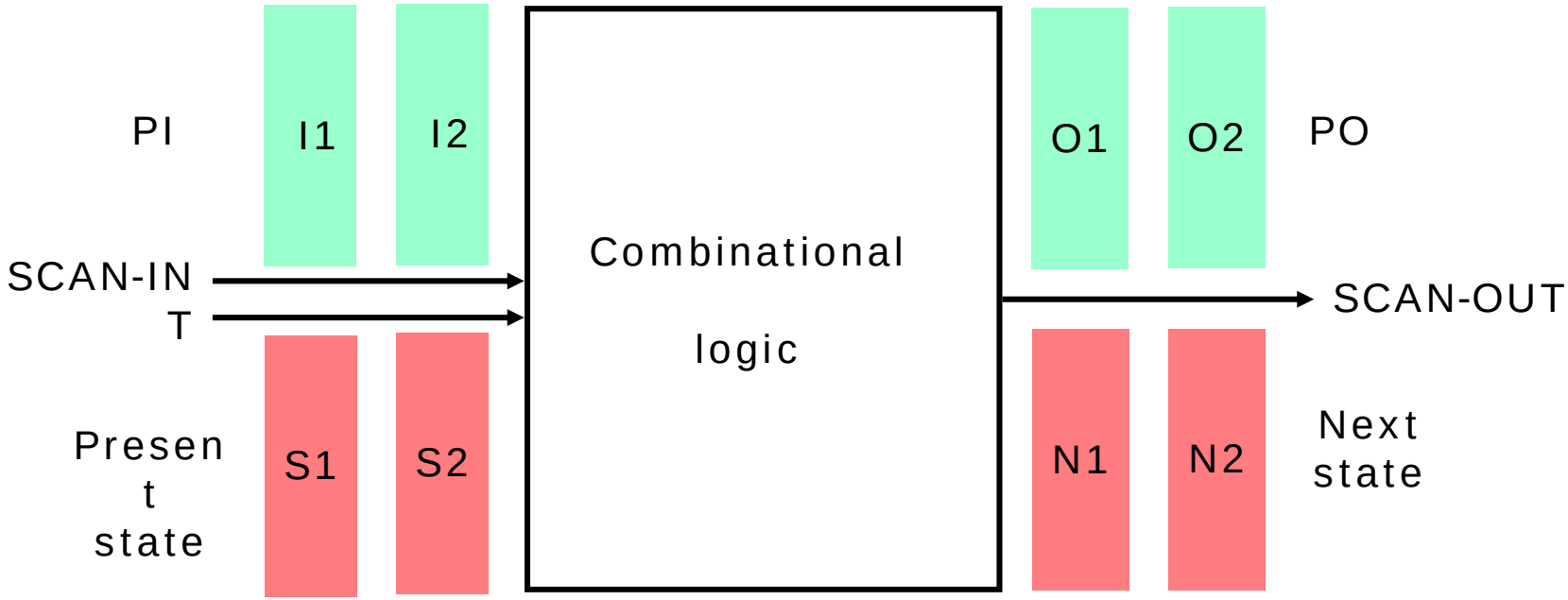
- The AND gates keep the bus drivers from being activated by the normal logic during testing



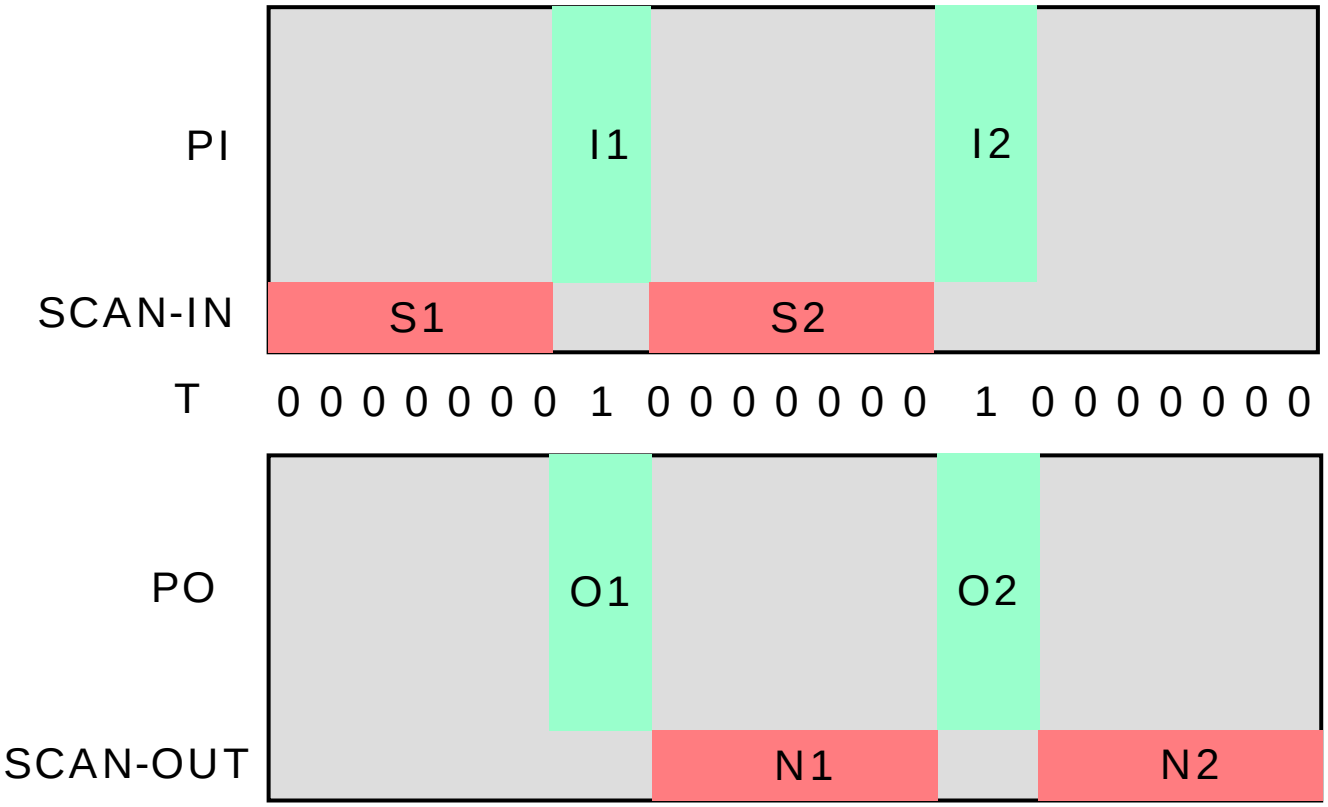
Scan Test Procedure

- **Step 1**: Switch to the shift-register mode and check the SR operation by shifting in an alternating sequence of 1s and 0s, e.g., 00110 (functional test)
- **Step 2**: Initialize the SR---load the first pattern
- **Step 3**: Return to the normal mode and apply the test pattern
- **Step 4**: Switch to the SR mode and shift out the final state while setting the starting state for the next test. Go to **Step 3**

Combining Test Vectors



Combining Test Vectors



Don't care or random bits

$$Sequence\ length = (n_{comb} + 1)\, n_{sff} + n_{comb}\ clock\ periods$$

n_{comb} = number of combinational vectors

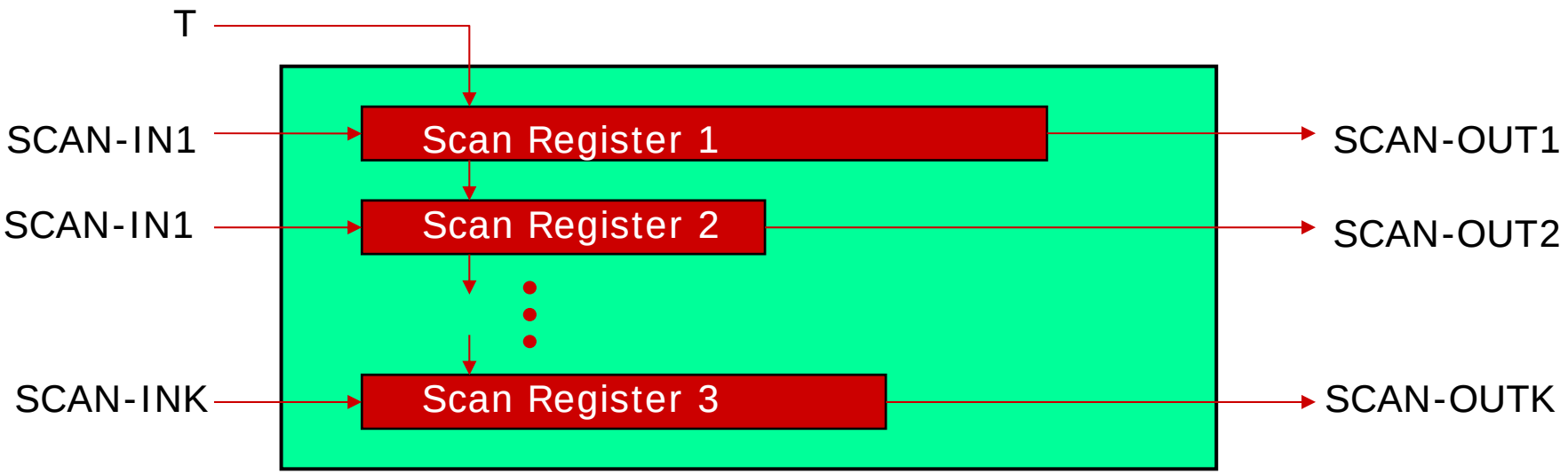
n_{sff} = number of scan flip-flops

Testing Scan Register

- Scan register must be tested prior to application of scan test sequences
- A shift sequence 00110011 . . . of length $n_{sff}+4$ in scan mode (TC=0) produces 00, 01, 11 and 10 transitions in all flip-flops and observes the result at SCAN-OUT output
- Total scan test length: $(n_{comb}+2)n_{sff}+n_{comb}+4$ *clock periods*
- Example: 2,000 scan flip-flops, 500 comb. vectors, total scan test length ~ 10^6 clocks
- Multiple scan registers reduce test length

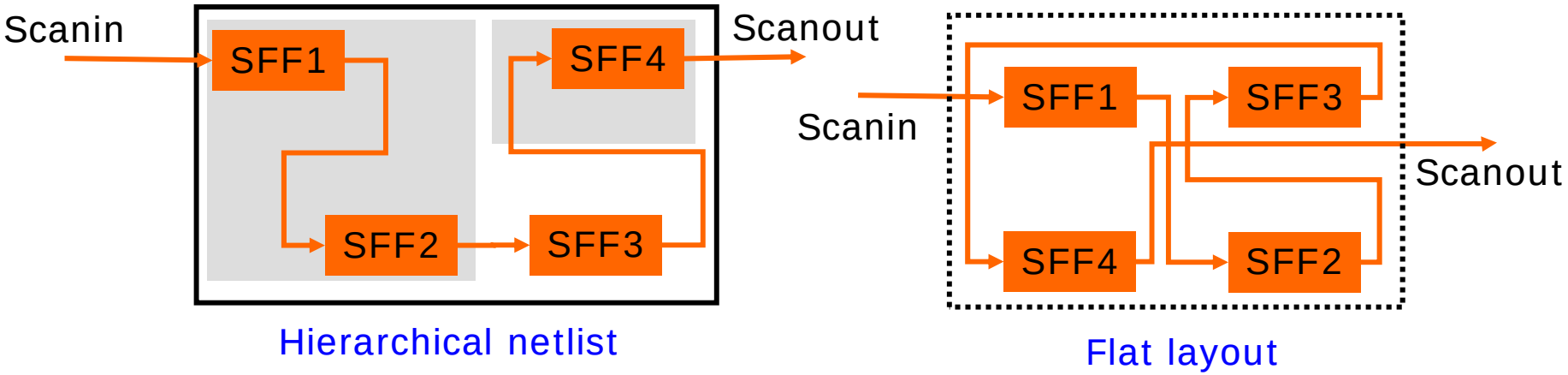
Multiple Scan Registers

- Scan flip-flops can be distributed among any number of shift registers, each having a separate *SCAN-IN* and *SCAN-OUT* pin
- Test sequence length is determined by the longest scan shift register
- Just one *test control* (TC) pin is essential

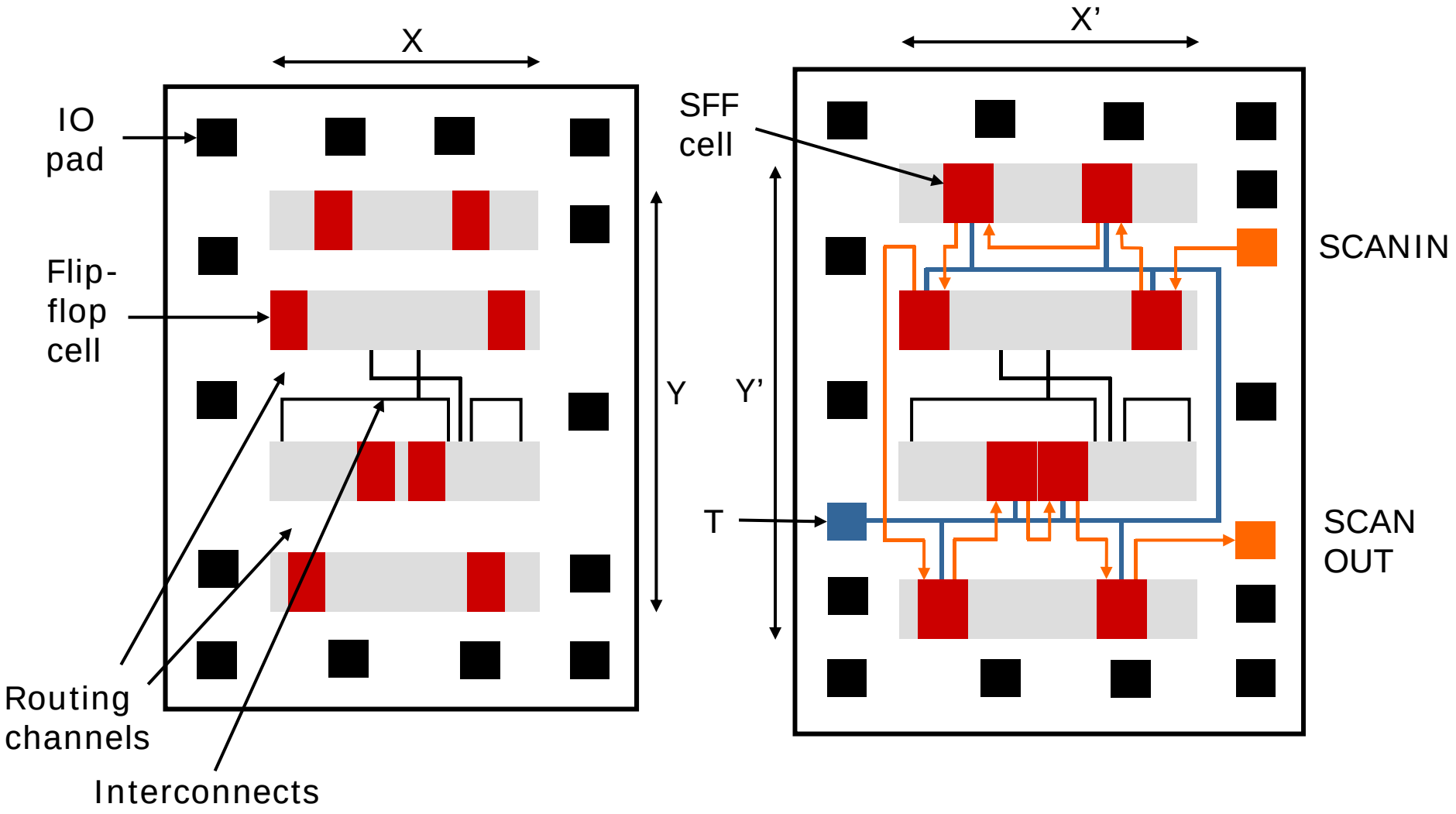


Hierarchical Scan

- Scan flip-flops are chained within subnetworks before chaining subnetworks
- Advantages:
  - Automatic scan insertion in netlist
  - Circuit hierarchy preserved – helps in debugging and design changes
- Disadvantage: Non-optimum chip layout

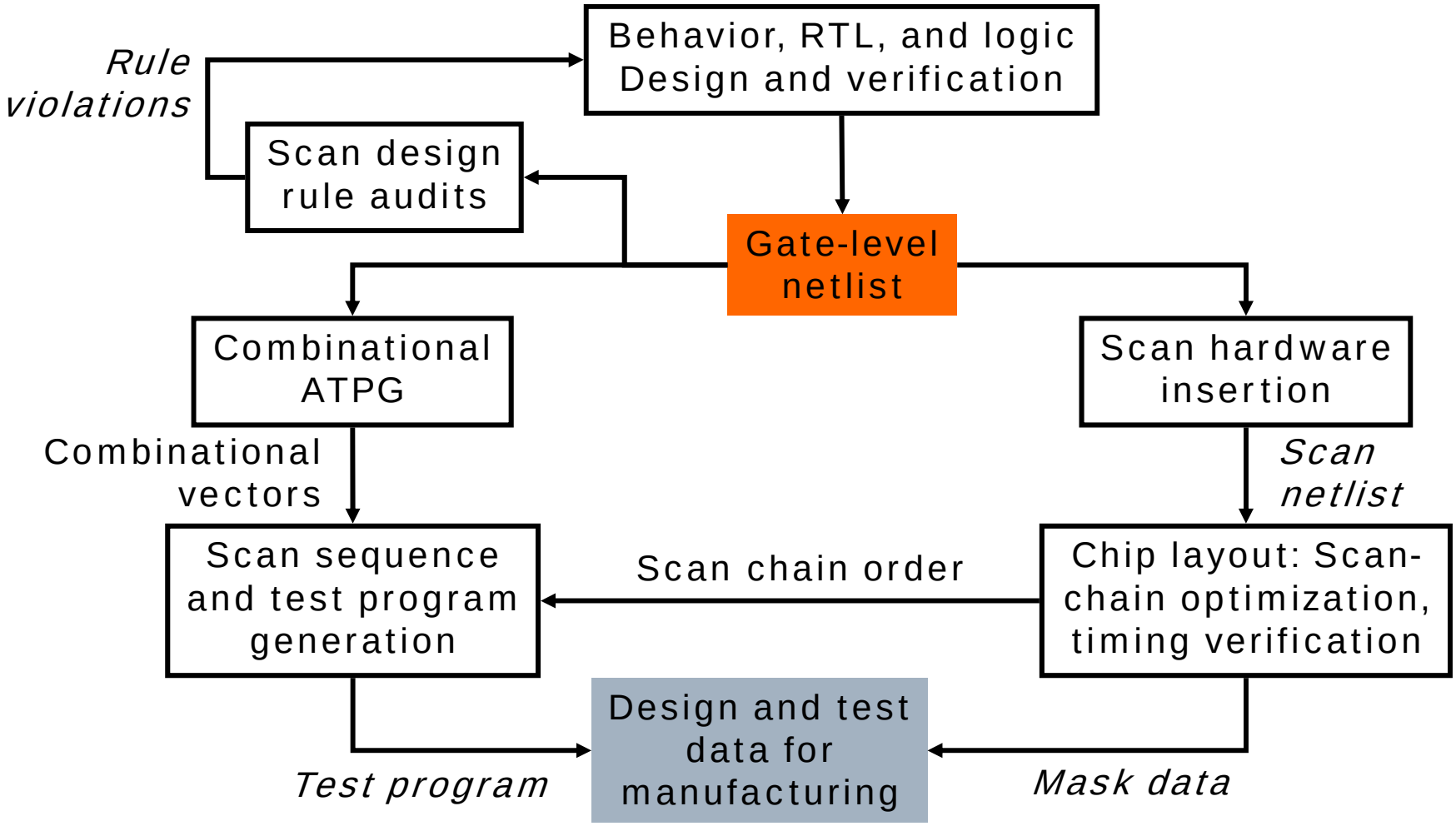


Optimum Scan Layout

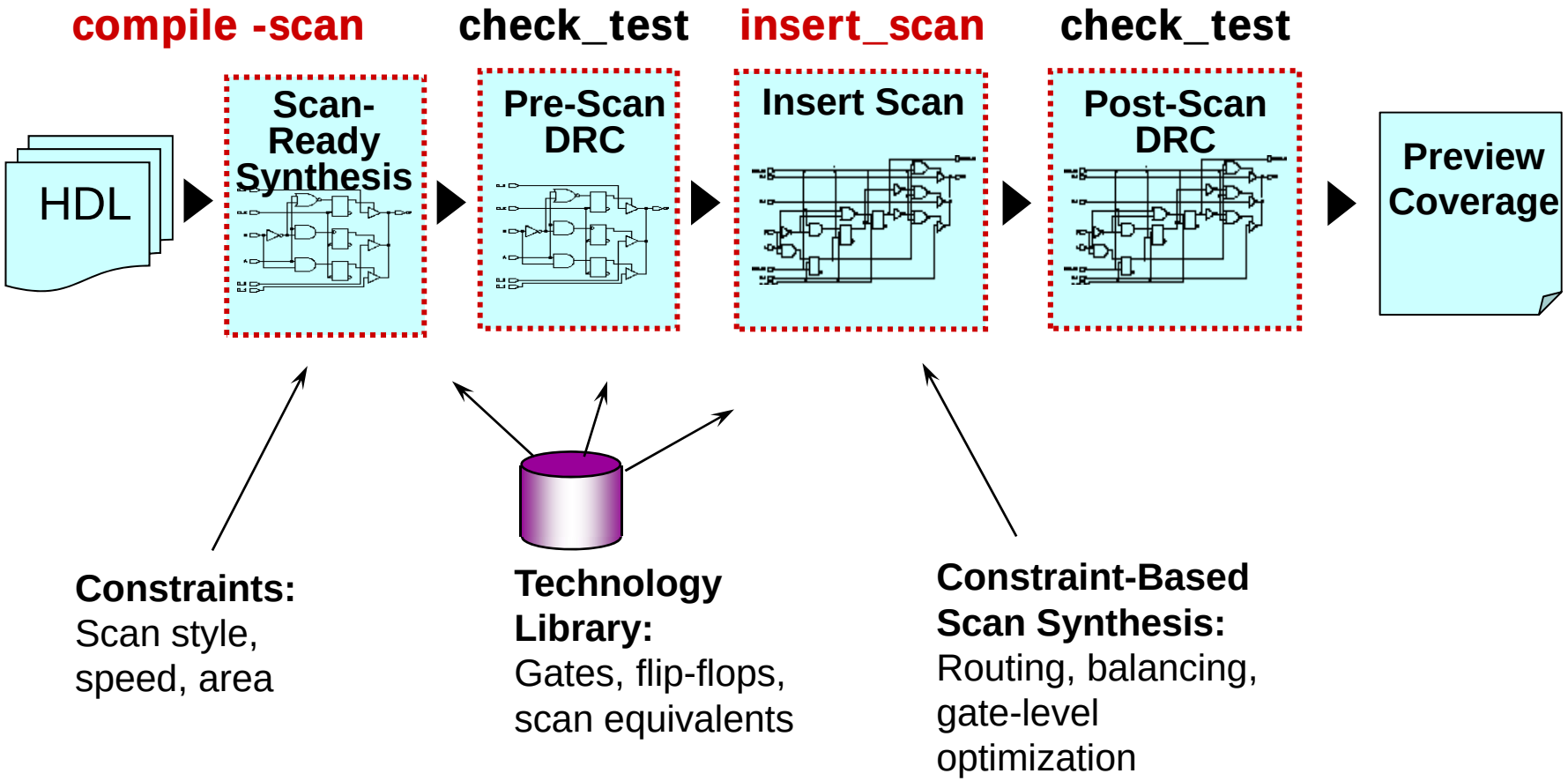


Active areas: XY and X'Y'

Automated Scan Design



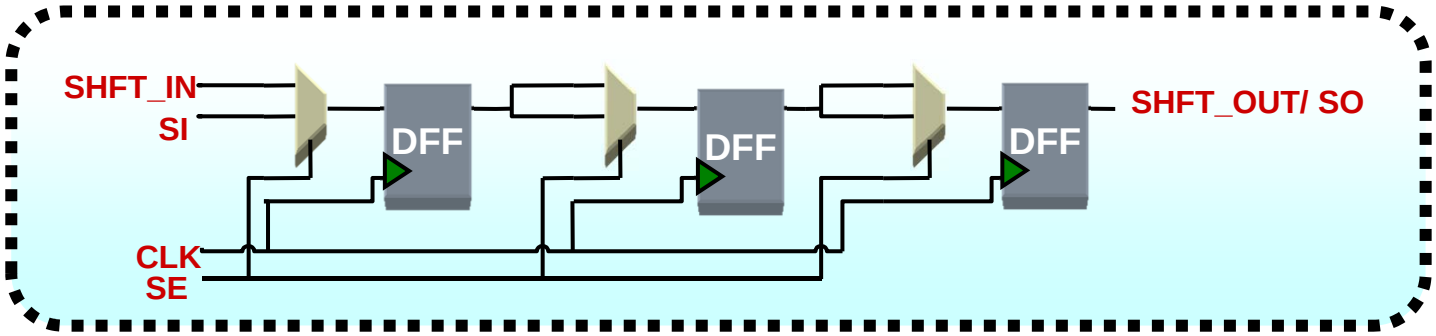
An Example of DFT Compiler Flow



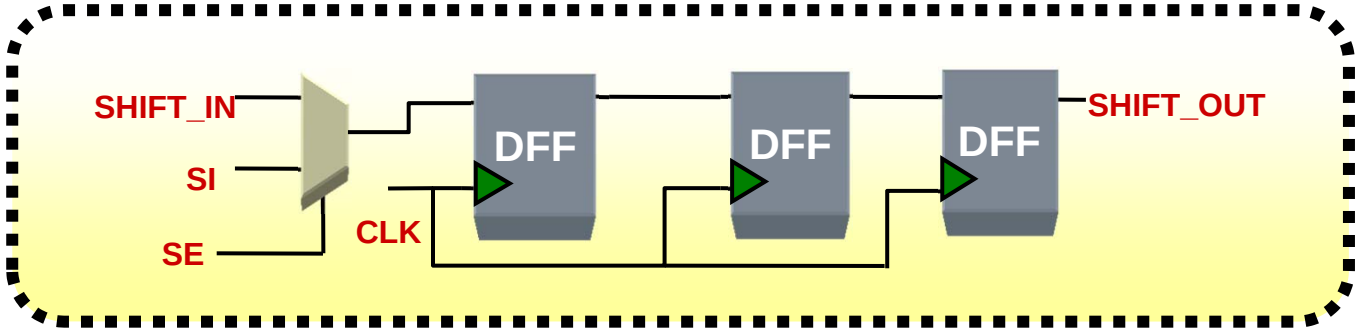
Source: H.-J. Huang, CIC

Shift Registers

Scan added:

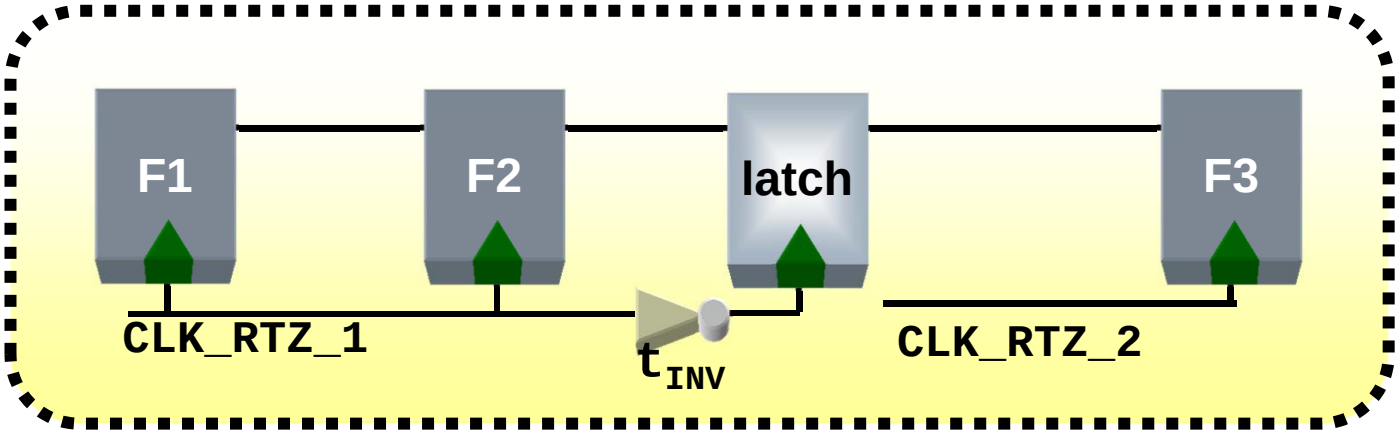


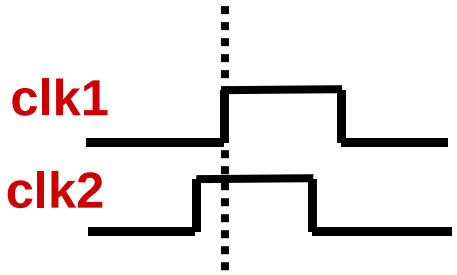
Revised:



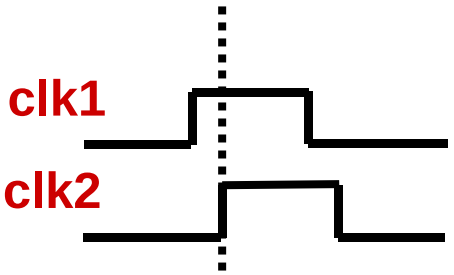
Source: H.-J. Huang, CIC

Lockup Latch Insertion





OK!

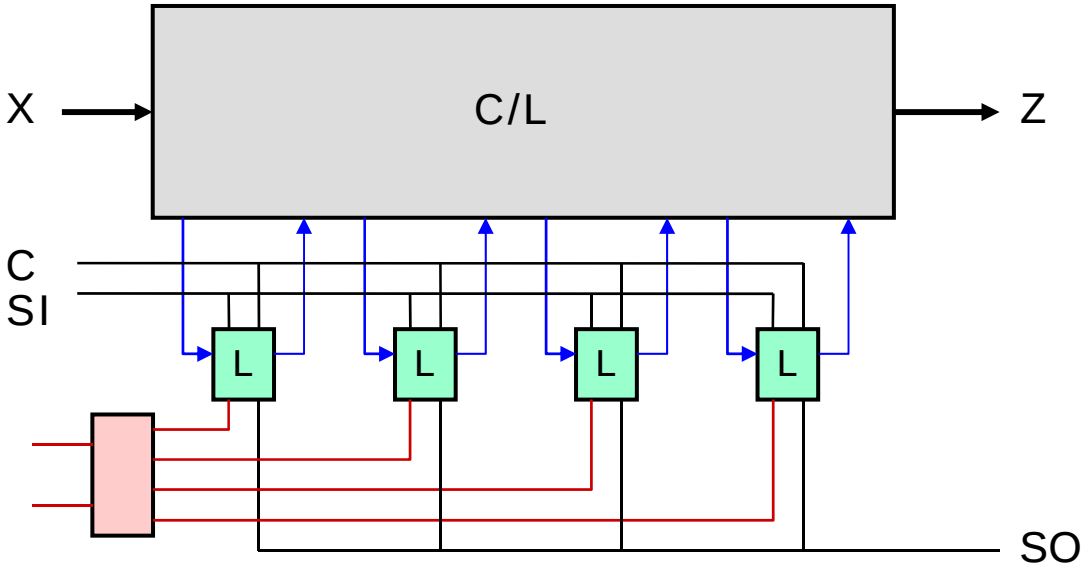


Big Problem !!
Rearrange clock domain or
insert lockup latch

Source: H.-J. Huang, CIC

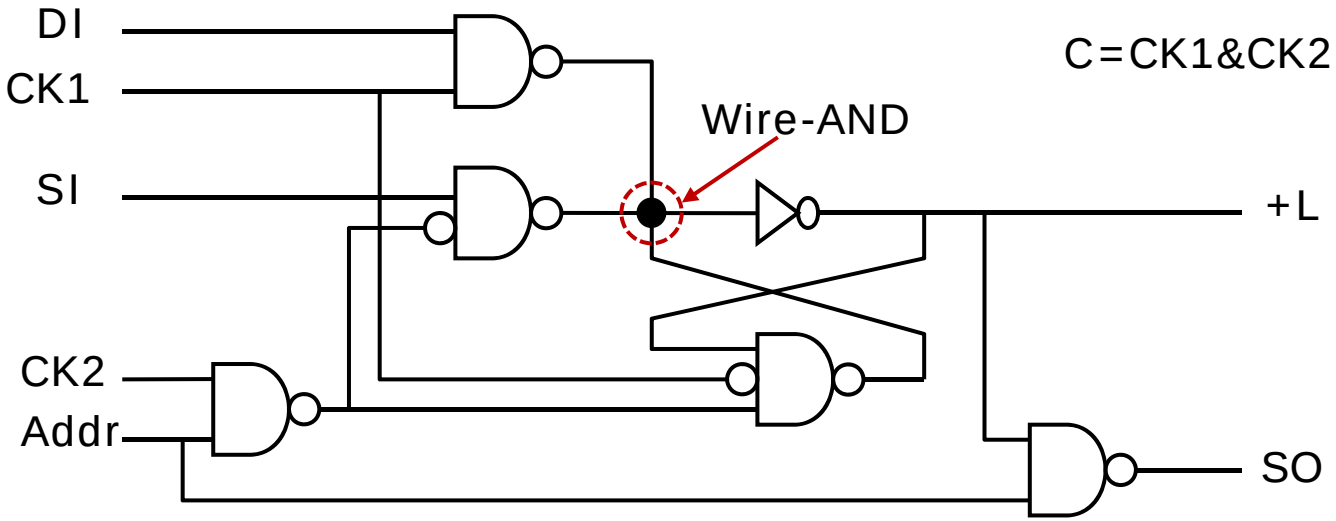
Random Access Scan

- Uses addressable latches
- Provides random access to FFs via multiplexing—address selection



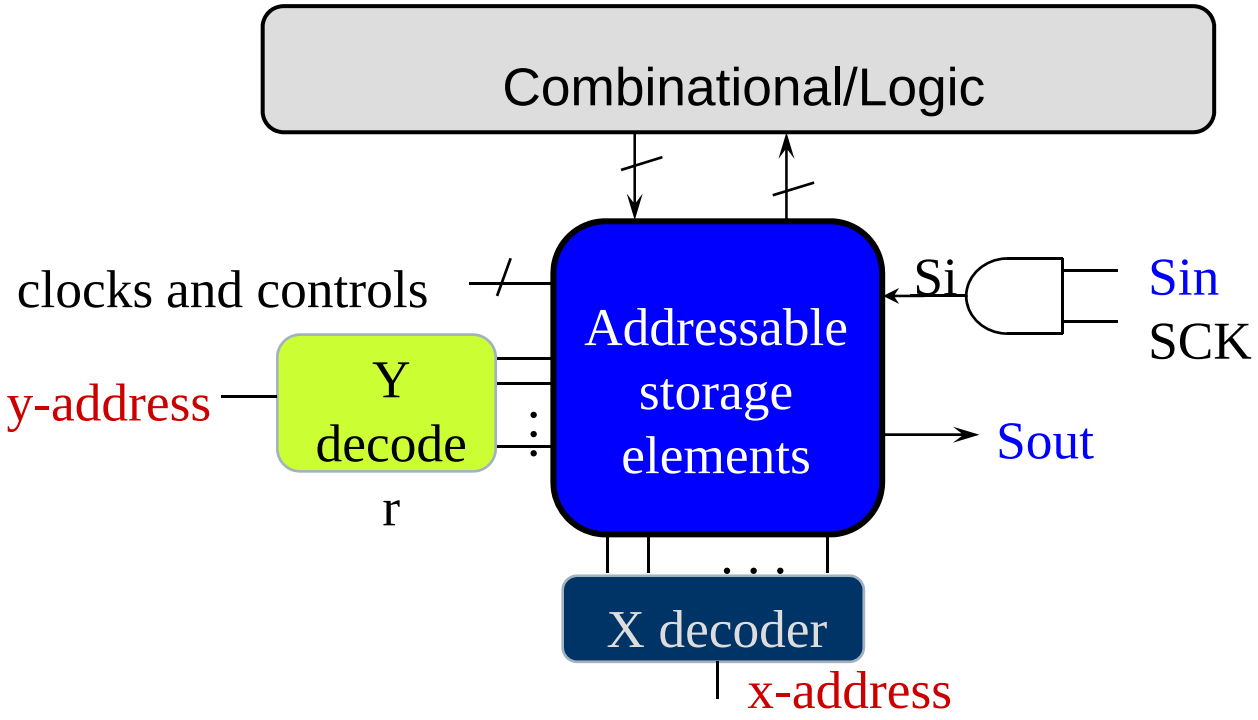
Random Access Scan

- Random access scan cell



- Advantages
  - Fast; minimal impact on normal path
  - Fast for testing—random access
  - Ability to 'watch' a node in normal operation mode
- Disadvantages
  - Hardware cost is large; more pins added

Random Access Architecture



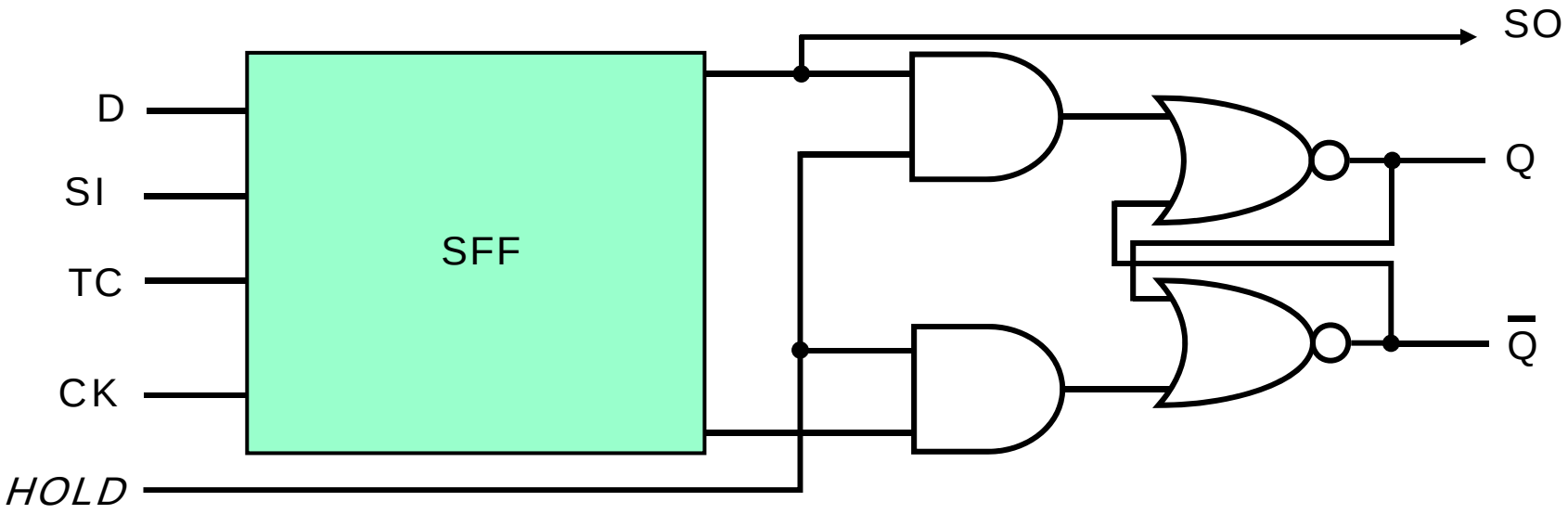
- During normal operation the storage cells operate in their parallel-load mode
- To scan in a bit, the appropriate cell is addressed, the data are applied to s_{in}

Test Procedure

1. Set test input to all test points
2. Apply the master reset signal to initialize all memory elements
3. Set scan-in address and data, and then apply the scan clock
4. Repeat step 3 until all internal test inputs are scanned in
5. Clock once for normal operation
6. Check states of the output points
7. Read the scan-out states of all memory elements by applying appropriate X-Y signals

Scan-Hold FFs (SHFFs)

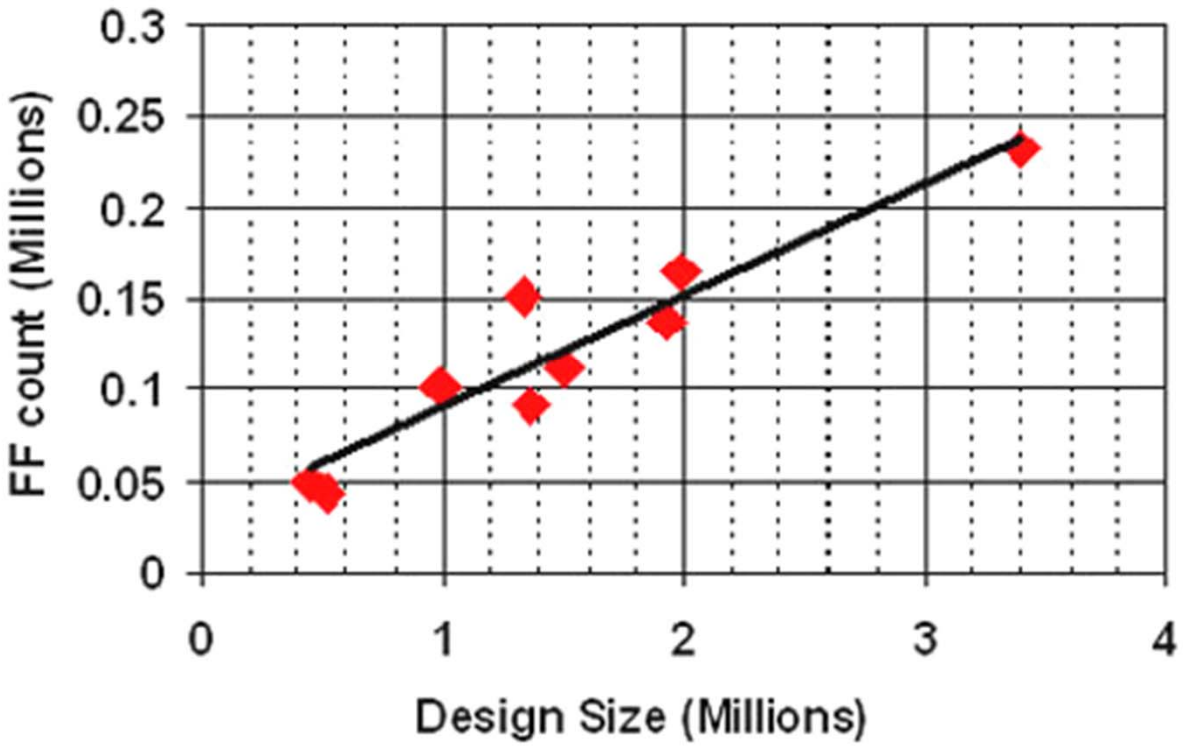
- HOLD=0→Q & Q' are fixed



- The control input *HOLD* keeps the output steady at previous state of flip-flop
- Applications
  - Reduce power dissipation during scan, etc.

Scan Enters the Nanometer Era

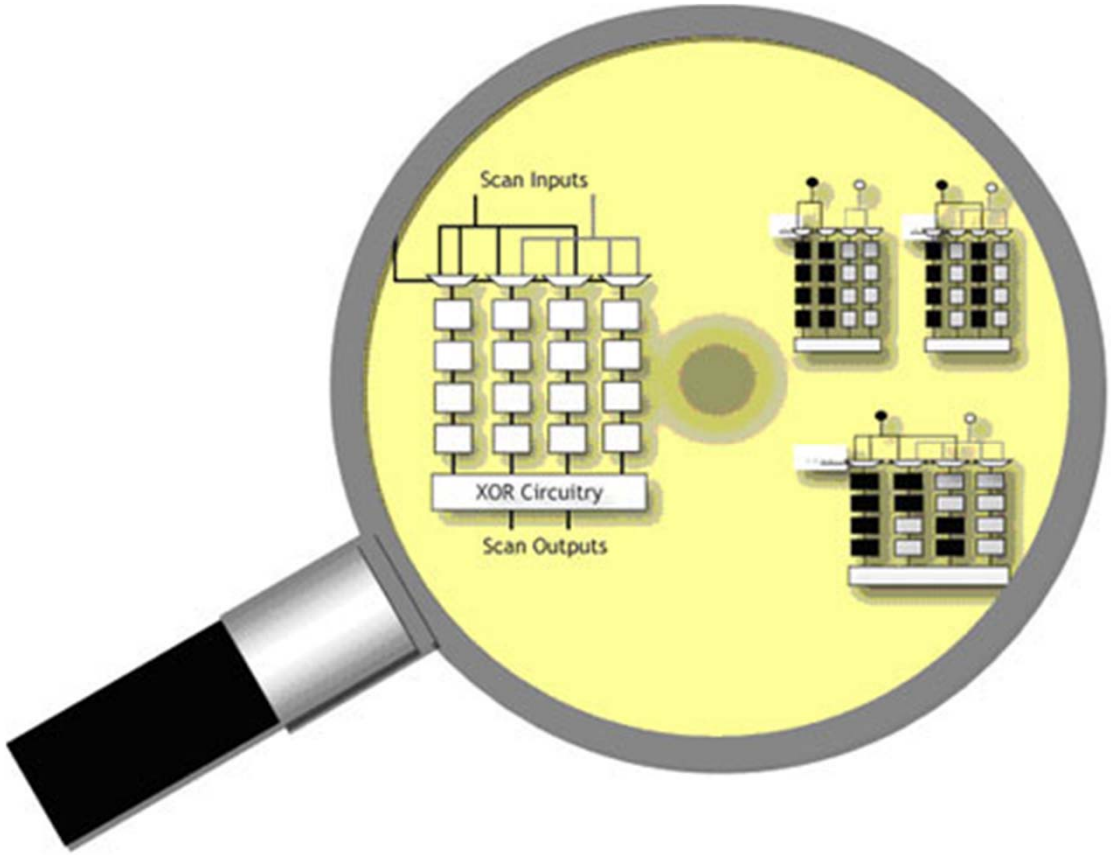
- Trend in flip flop count with design size



[Source: EE Times]

Scan Enters the Nanometer Era

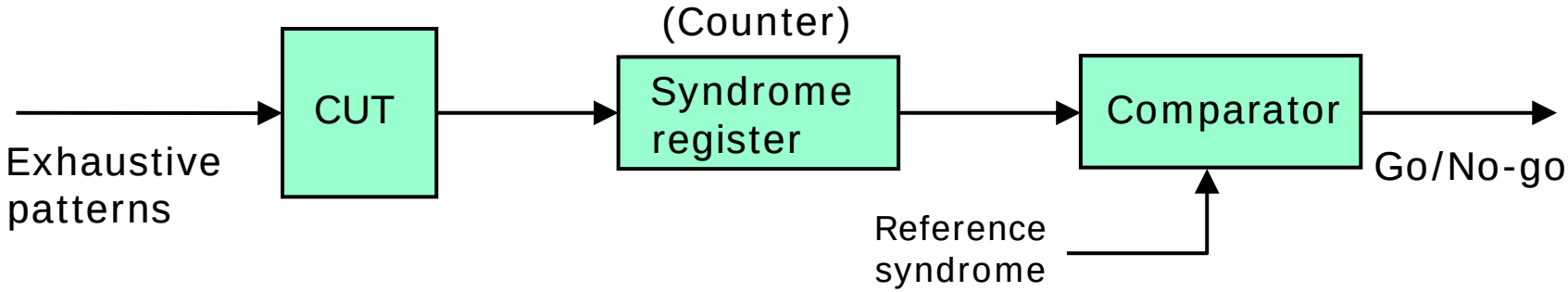
- Adaptive scan architecture



[Source: EE Times]

Syndrome-Testable Design

- Definition
  - The syndrome of a Boolean function f is $S(f) \equiv \frac{k(f)}{2^n}$, where k is the number of 1s (minterms) in f and n is the number of independent input variables
  - A typical syndrome testing set-up



  - $0 \le S(f) \le 1$
  - A circuit is syndrome testable iff $\forall$ fault α, $S(f) \neq S(f_\alpha)$
  - Syndromes of logic gates

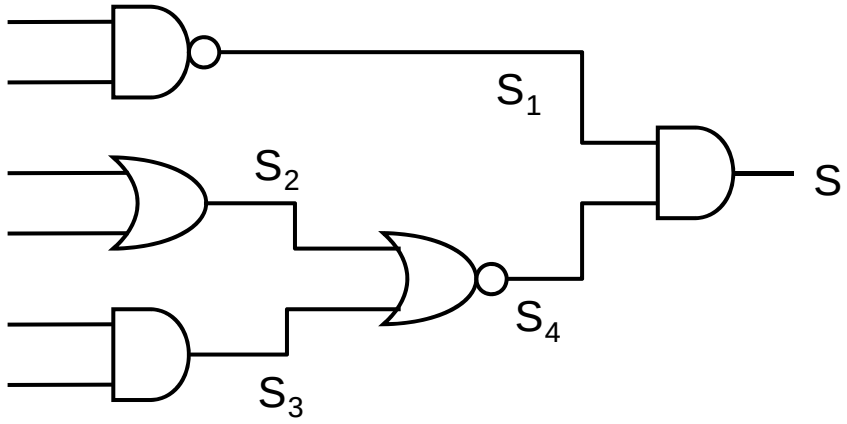
Gate	AND_n	OR_n	XOR_n	NOT
S	$1/2^n$	$1-(1/2^n)$	$1/2$	$1/2$

Syndrome Computation

- Consider a circuit having 2 blocks, f and g, with unshared inputs

O/P Gate	*OR*	*AND*	*XOR*	*NAND*	*NOR*
S	$S_f + S_g - S_f S_g$	$S_f S_g$	$S_f + S_g - 2S_f S_g$	$1 - S_f S_g$	$1 - S_f - S_g + S_f S_g$

- Example
  - Calculate the syndrome of the following circuit



$S_1 = 1\text{-}1/4 = 3/4$

$S_2 = 1\text{-}1/4 = 3/4$

$S_3 = 1/8$

$S_4 = 1 - S_2 - S_3 + S_2S_3 = 7/32$

$S = S_1S_4 = 21/128$

Syndrome-Testable Design

- Consider the function $f = xz + y\bar{z}$. The circuit is syndrome untestable
  - $S_f = 1/2$
  - If the circuit has a fault $\alpha \equiv z/0$, then the corresponding syndrome of the faulty circuit is $S_f' = 1/2$
  - Thus the circuit is syndrome untestable
- A realization C of a function *f* is said to be syndrome-testable if no single stuck-at fault causes the circuit to have the same syndrome as the fault-free circuit
- Syndrome is a property of ***function***, not of ***implementation***

Syndrome-Testable Design

- Definition
  - A logic function is ***unate*** in a variable x_i if it can be represented as an SOP or POS expression in which the variable x_i appears either only in an uncomplemented form or only in a complemented form
  - For example:
    - $f(x_1, x_2) = x_1\overline{x_2} + \overline{x_1}x_2$ no unate
    - $f(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_1x_3$ unate in x_2, x_3, not unate in x_1
- Theorem
  - A 2-level irredundant circuit realizing a unate function in all its variables is syndrome testable

Syndrome-Testable Design

- Theorem
  - Any 2-level irredundant circuit can be made syndrome-testable by adding control inputs to the AND gates
- For example
  - The function $f = xz + y\bar{z}$ is syndrome untestable
  - Now add a control input c э $f' = cxz + y\bar{z}$, where

$$C \equiv \begin{cases} 1 & \text{when in normal operation mode} \\ \text{normal i/p} & \text{when in test mode} \end{cases}$$

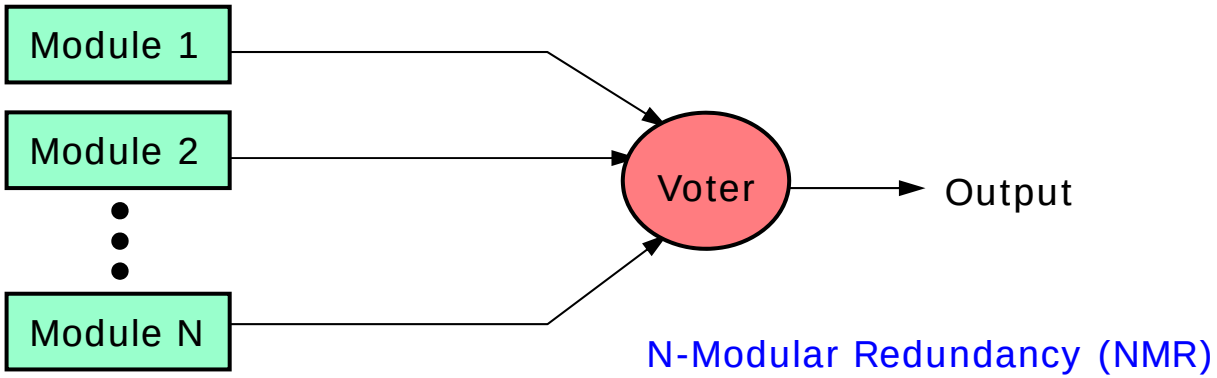
  - $S' = 3/8$, $f'_\alpha = y$, and $S'_\alpha = 1/2 \neq S'$ → Syndrome testable
- Drawbacks
  - Only for combinational logic
  - Exhaustive; modification doubles test set size

Introduction to Built-In Self-Test

- Built-in self-test (BIST):
  - The capability of a circuit (chip/board/system) to test itself
- Advantages of BIST
  - Test patterns generated on-chip →controllability increased
  - (Compressed) response evaluated on-chip →observability increased
  - Test can be on-line (concurrent) or off-line
  - Test can run at circuit speed →more realistic; shorter test time; easier delay testing
  - External test equipment greatly simplified, or even totally eliminated
  - Easily adopting to engineering changes

Introduction to Built-In Self-Test

- On-line BIST
  - Concurrent (EDAC, NMR, totally self-checking checkers, etc.):
    - Coding or modular redundancy techniques (fault tolerance)

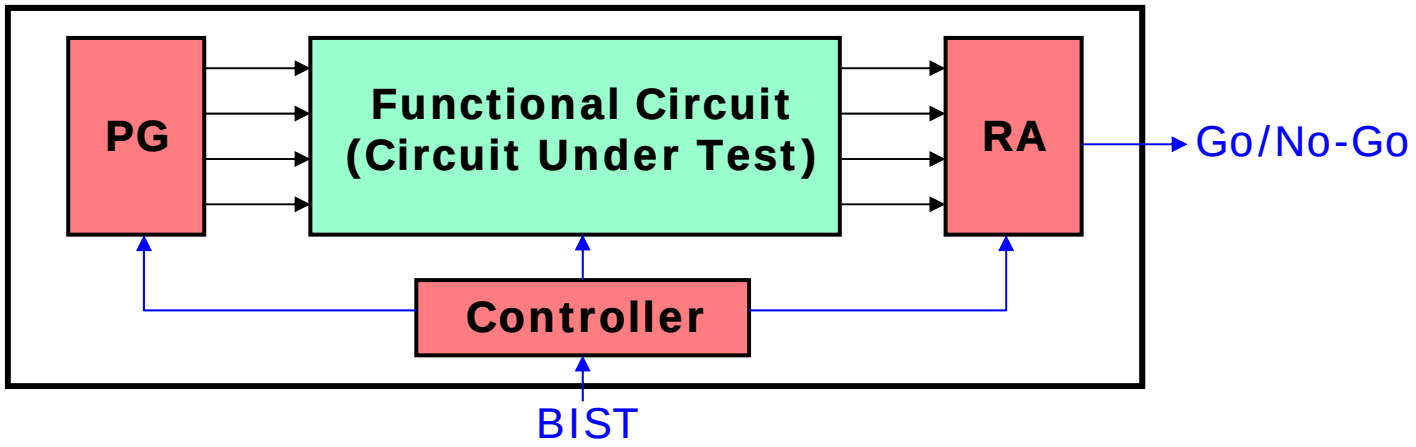


N-Modular Redundancy (NMR)

    - Instantaneous correction of errors caused by temporary or permanent faults
  - Nonconcurrent (diagnostic routines):
    - Carried out while a system is in an idle state

Introduction to Built-In Self-Test

- Off-line BIST
  - A typical BIST architecture



  - Test generation
    - Prestored TPG, e.g., ROM or shift register
    - Exhaustive TPG, e.g., binary counter
    - Pseudo-exhaustive TPG, e.g., constant-weight counter, combined LFSR and SR
    - Pseudo-random pattern generator, e.g., LFSR

Introduction to Built-In Self-Test

- Response analysis
  - Check-sum
  - Ones counting
  - Transition counting
  - Parity checking
  - Syndrome analysis
  - Etc.
- Linear feedback shift register (LFSR) can be both the test generator and response analyzer
- We need a gold unit to generate the good signature or a simulator

Signature Analysis

- A compression technique based on the concept of cyclic redundancy checking (CRC) and realized in hardware using linear feedback shift registers
- Definition
  - A function $f(x_1,x_2,\ldots,x_n)$ is said to be linear if it can be expressed in the form

$$f = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_n x_n$$

    where $a_i \in \{0,1\} \forall i = 0,1,\cdots,n$

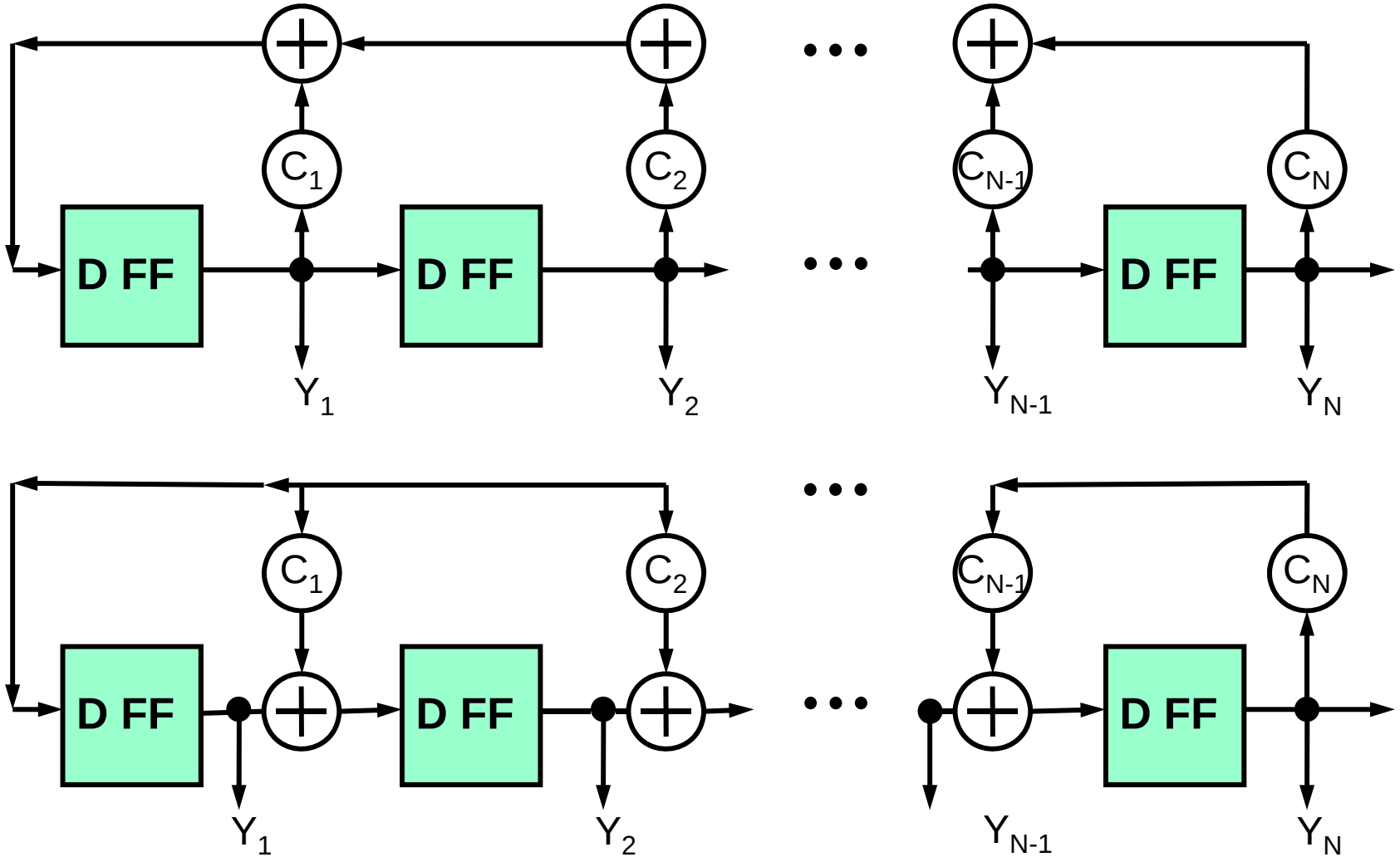
    - There are 2^{n+1} linear functions of n variables
    - Linear operations: modulo addition, module scalar multiplication, & delay
    - Nonlinear operations: AND, OR, NAND, NOR, etc.

Linear Feedback Shift Register

- Definition
  - A linear feedback shift register is a shift register with feedback paths which consist only of unit delays and XOR operators
- Let M=fault-free circuit response, B=faulty circuit response, and E=error syndrome (Hamming), where $E=M \oplus B \rightarrow$ thus $M=B \oplus E$ and $B=M \oplus E$
  - We need a circuit to take B as input and compact it but still be able to tell if M!=B
- LFSR is considered as a popular approach for test response compaction

Structures of LFSR

- Two types of generic standard LFSRs



Mathematical Foundation of LFSR

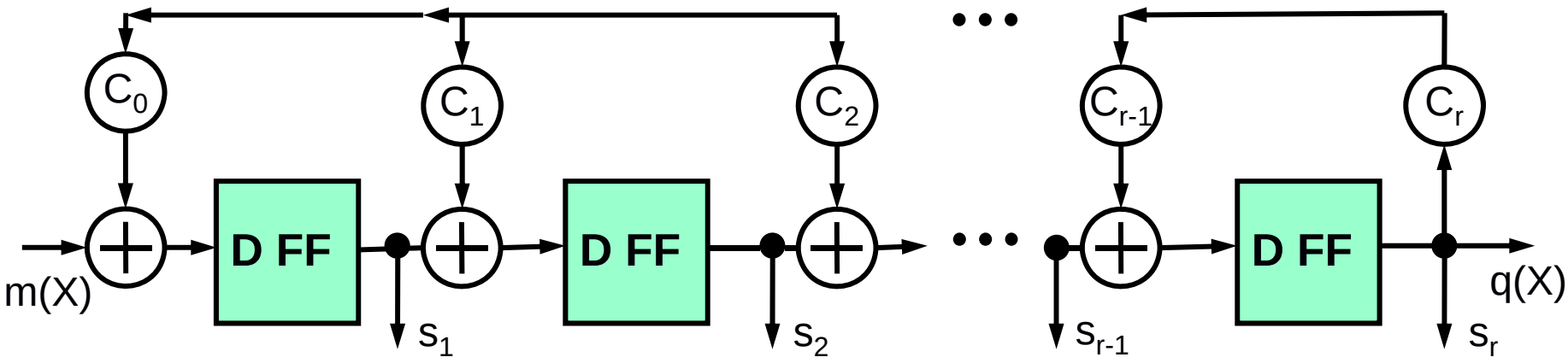
- As a function of time, Y_j can be expressed as
  - $Y_j(t) = Y_{j-1}(t-1)$ for $j \neq 0$
  - Hence $Y_j(t) = Y_0(t-j)$
- If we denote the translation operator as X^k, where k is the time translation unit
  - $Y_j(t) = Y_0(t)X^j$
- On the other hand, $Y_0(t)$ can be expressed as
  - $Y_0(t) = \sum_{j=1}^{N} C_j Y_j(t)$
- Then
  - $Y_0(t) = \sum_{j=1}^{N} C_j Y_0(t) X^j$ for $1 \leq j \leq N$

Mathematical Foundation of LFSR

- We can rewrite the $Y_0(t)$ as
  - $Y_0(t) = Y_0(t)\sum_{j=1}^{N} C_j X^j$
  - Also, $Y_0(t)(\sum_{j=1}^{N} C_j X^j + 1) = 0$
- We can rewrite this expression as $Y_0(t)P_N(X) = 0$
  - For nontrivial solution, $Y_0(t) \neq 0$, we must have $P_N(X)=0$
    where $P_N(X) = 1 + \sum_{j=1}^{N} C_j X^j$
  - $P_N(X)$ is called the *characteristic polynomial* of the LFSR

LFSR for Signature Analysis

- A serial input stream m_n, m_{n-1},..., m_1, m_0 entering the LFSR can be considered as the coefficients of a polynomial
  - $m(X) = m_n X^n + m_{n-1} X^{n-1} + \cdots + m_1 X + m_0$



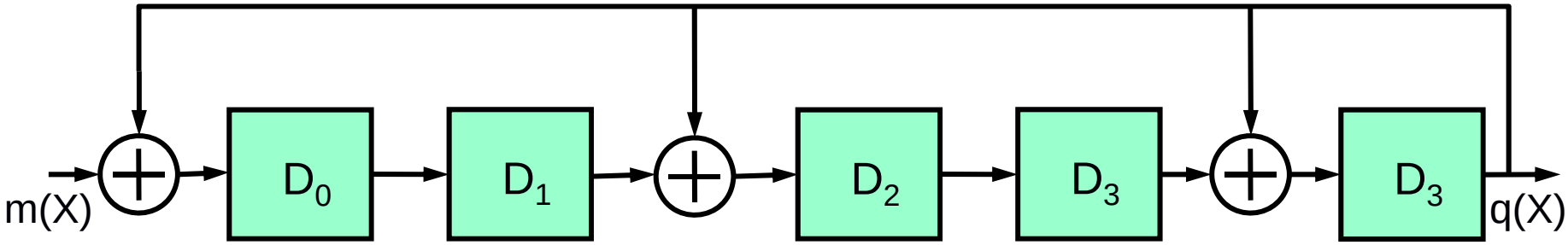
  - The LFSR is said to have a characteristic polynomial defined as follows
    - $c(X) = c_r X^r + c_{r-1} X^{r-1} + \cdots + c_1 X + c_0$

LFSR for Signature Analysis

- Assume that the initial state of the LFSR is $D_i=0$, i=0,...,r-1, then the LFSR effectively divides any m(X) by c(X), i.e.,
  - $m(X) = q(X) \bullet C(X) + s(X)$
- The quotient q(X) appears serially at the output of the SR. The remainder s(X) is in the SR after n+1 shifts:
  - $s(X) = s_r X^r + s_{r-1} X^{r-1} + \cdots + s_1 X + s_0$

An Example

- The following LFSR divides any m(X) by $c(X)=X^5+X^4+X^2+1$



  - Suppose $m(x)=X^7+X^6+X^5+X^4+X^2+1$, then $q(X)=X^2+1$, and $s(X)=X^4+X^2$

I/P	D_0	D_1	D_2	D_3	D_4	O/P
10101111	0	0	0	0	0	-
101	0	1	1	1	1	-
10	0	0	0	1	0	1
1	0	0	0	0	1	01
-	0	0	1	0	1	101

Signature Analysis

- Let m(X) be the input polynomial of degree k-1, q(X) the quotient, and s(X) the signature (remainder). Then
  - $m(X)=q(X)c(X)+s(X)$
- The error syndrome can be represented as a polynomial e(X)
  - E.g., le $m(X)=X^4+X^3+1(11001)$, and an erroneous input $b(X)=X^3+X+1(01011)$, then the error syndrome is $11001 \oplus 01011=10010$, and is represented by $e(X)=X^4+X$
- In general, an erroneous input polynomial can be represented by
  - $B(X)=m(X)+e(X)$

Signature Analysis

- Theorem1: *Input streams m(X) and b(X) have the same signature iff e(X) is a multiple of c(X)*
  - Proof: an error is not detected when m(X) and b(X) have the same signature, i.e., b(x)=q'(X)c(X)+s(X). Since m(X)=q(X)c(X)+s(X), we obtain e(X)=m(X)+b(X)=c(X)(q'(X)-q(X))
- Theorem2: *Undetected errors correspond to error patterns which are multiples of c(X)*
- Theorem3: *If c(X) has 2 or more nonzero coefficients—i.e., at least 1 feedback term—then it can detect all single-bit errors*
  - Proof: all nonzero multiples of c(X) must have at least 2 nonzero coefficients. Therefore, any error with only 1 nonzero coefficient cannot be a multiple of c(X) and must be detectable.

Aliasing Probability

- Theorem4: for a *k*-bit response sequence, if all possible error patterns are equally likely, then the probability of failing to detect an error (i.e., the *aliasing probability*) by the LFSR of length *r* is

$$P_{al} = \frac{2^{k-r} - 1}{2^k - 1}$$

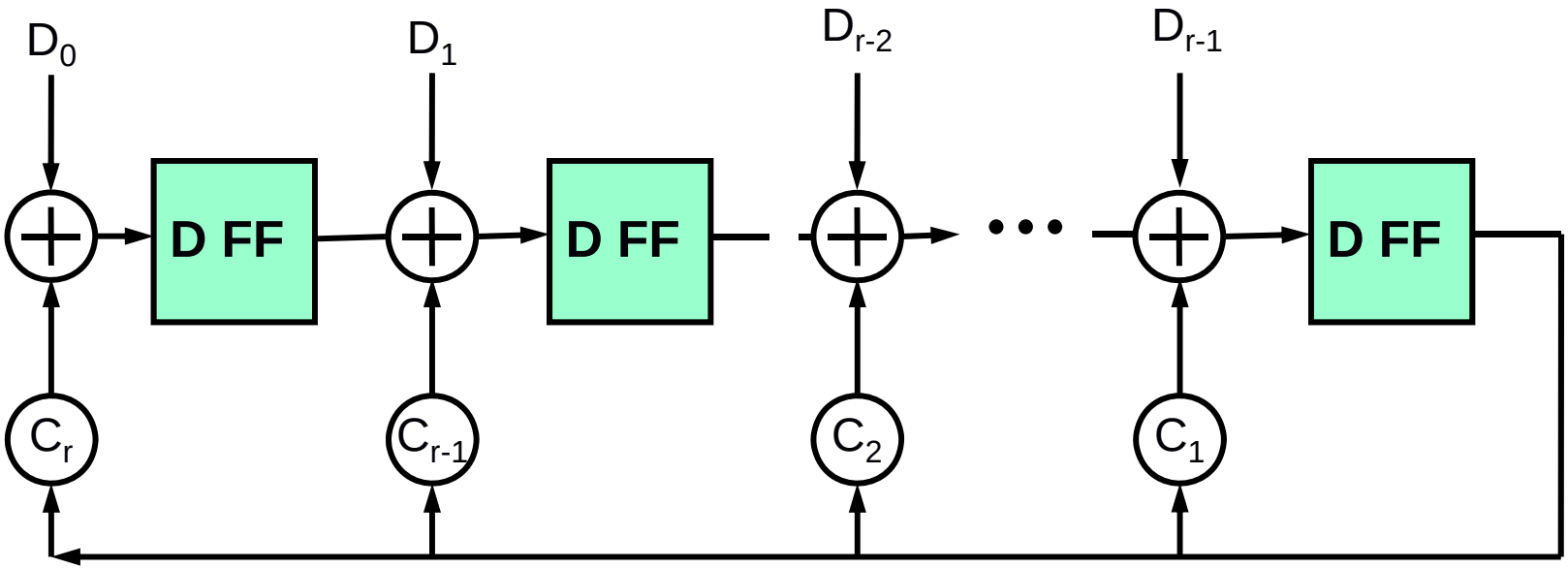
  - Proof: For a k-bit response, deg(m(X))=*k*-1, and deg(e(X))<=*k*-1. Therefore, the number of possible error polynomial is represented by e(X)=c(X)p(X) for some nonzero p(X). Since deg(c(X))=*r*, the number of possible p(X)'s is 2^{k-r}-1. Thus

$$P_{al} = \frac{2^{k-r} - 1}{2^k - 1}$$

  - For a long sequence, k>>r→$P_{al} \sim 1/2^r$

Multiple-Input Signature Register

- The structure of multiple-input signature register (MISR)



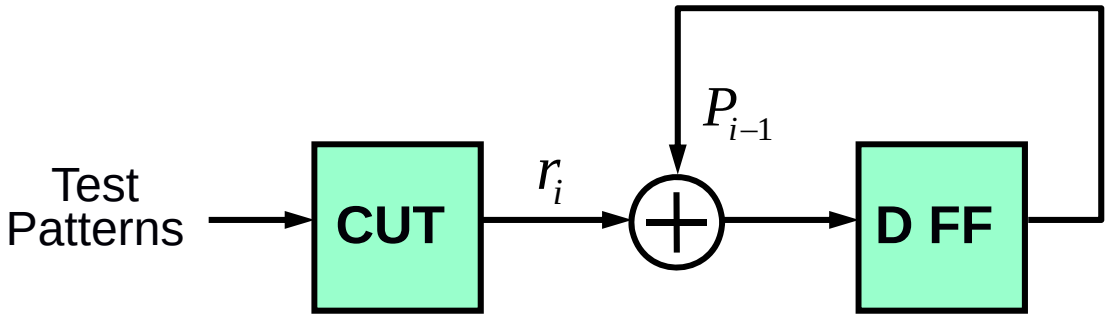
  - The mathematical theory is a direct extension of the results shown above
  - For equally likely error patterns and long data streams, the aliasing probability for an MISR of *r* stages also is $P_{al} \approx 1/2^r$.

Response Compaction

- Usually, we think of *data compression* as a process that preserves data integrity. This is why we given more attention here to *data compaction*, which may result in some losses
- There are several compaction testing techniques
  - Parity testing
  - One counting
  - Transition counting
  - Syndrome calculation
  - Signature analysis

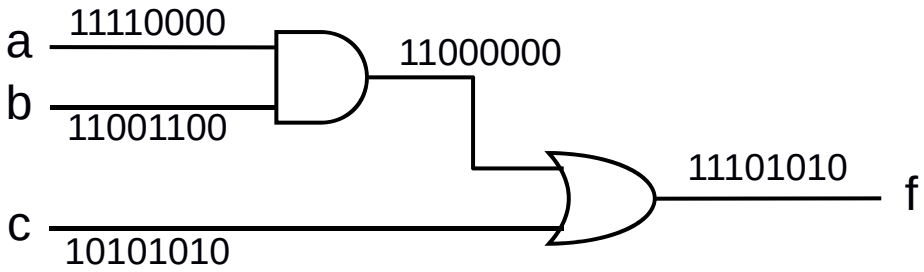
Parity Testing

- This is the simplest of all techniques but also the most lossy
- The parity of responses to the test patterns is calculated as
  - $P = \sum_{i=1}^{i=L} r_i$, where L is the length of the test and r_i is the response for the *i*th test pattern
- The response of the circuit under test (CUT) to pattern *i* and the partial product P_{i-1} is illustrated as below



One Counting

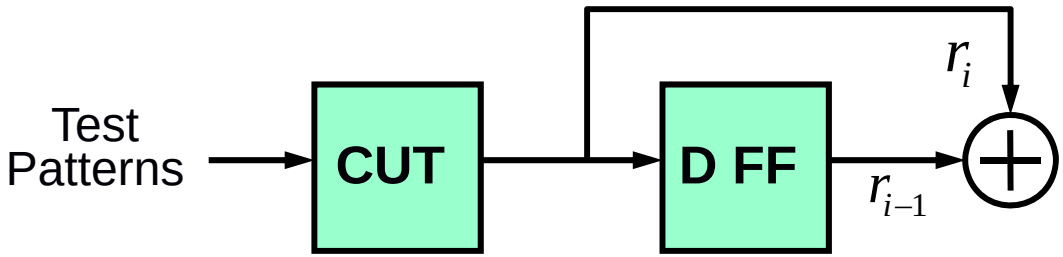
- The number of 1's in the response stream is calculated and compared to the number of 1's in the fault-free responses
- Consider the circuit shown below



- If we have a test of length L and the fault-free count is m, the possibility of aliasing is [C(L,m)-1] patterns out of a total number of possible strings of length L, (2^L-1)

Transition Counting

- In transition counting compaction, it is only the number of transition 0→1 and 1→0 that are counted. Thus the signature is given by
  - $\sum_{i=1}^{i=L-1} r_i \oplus r_{i+1}$, where the summation is ordinary addition and $\oplus$ is XOR operation
- The compaction scheme is shown below

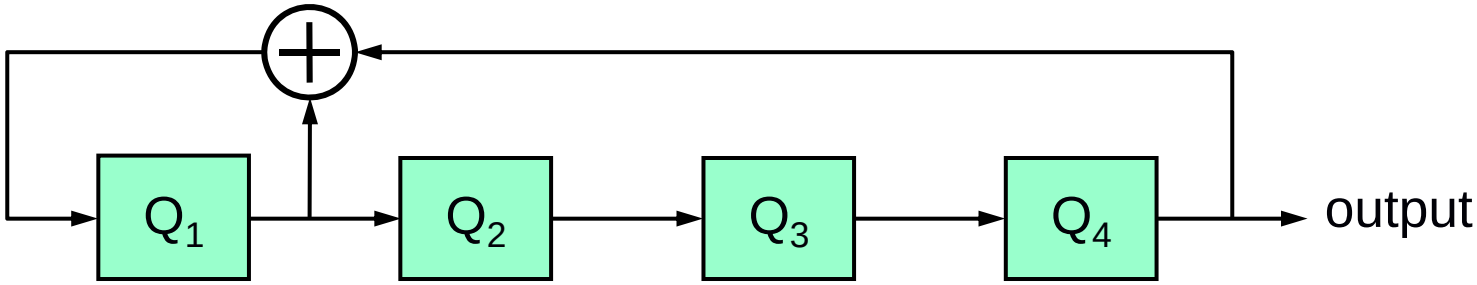


Pseudorandom Pattern Generator

- Logic BIST uses mostly pseudorandom (PR) tests. They are usually much longer than deterministic tests, but are definitely less costly to generate
- PR tests are generated using a LFSR or cellular automata
- By means of a simple circuit called an *autonomous linear feedback shift register* (ALFSR)
- Definition: an ALFSR is a LFSR with no external inputs
- Faults that are hard to detect with PR tests are called *random pattern resistant* faults

Pseudorandom Pattern Generator (PRPG)

- Example: the following ALFSR generates the pseudorandom sequence shown in the table below

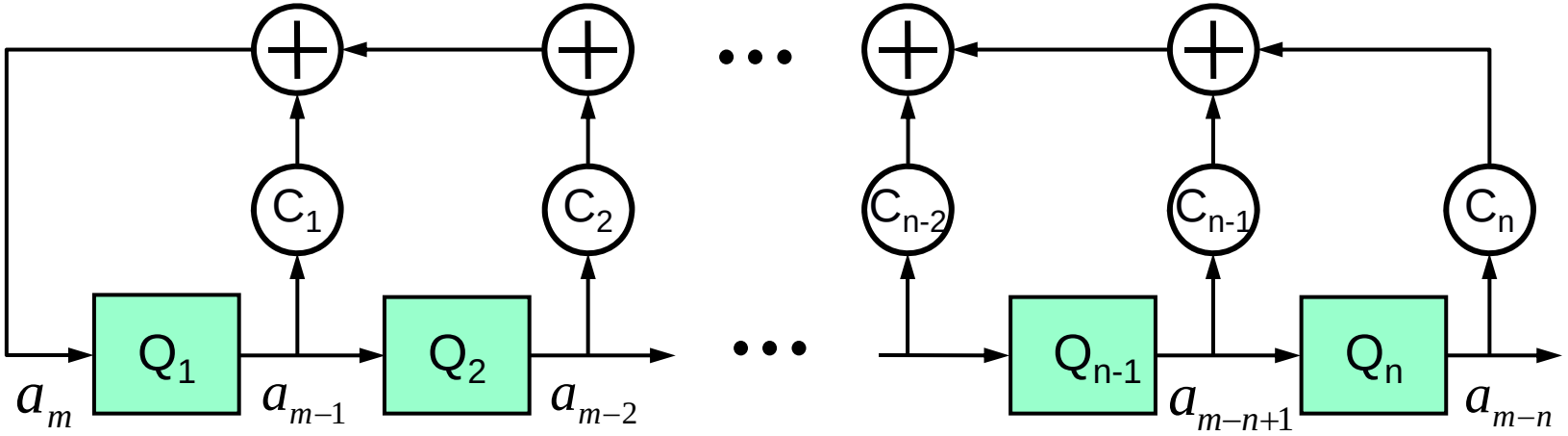


State	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15=0
Q_1	1	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1
Q_2	0	1	1	1	1	0	1	0	1	1	0	0	1	0	0	0
Q_3	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0	0
Q_4	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0

- The output sequence is 000111101011001, which repeats after $15(2^n-1)$ clocks
- Max period for an n-stage ALFSR=2^n-1
- All-0 state of the register cannot occur in the max-length cycle

Mathematical Foundation of PRPG

- A generic structure of ALFSR



- A sequence of bits $\{a_m\}=a_0,a_1,\ldots,a_m,\ldots$ can be associated with a polynomial—its ***generation function***:

$$G(X) \equiv a_0 + a_1 X + \cdots + a_m X^m + \cdots = \sum_{m=0}^{\infty} a_m X^m$$

- In the above figure, assume that the current state of Q_i is a_{m-i}, i=1,2,…,n, and the initial state of Q_i is $a_{-i}=0$, i=1,2,…,n, but $a_{-n}=1$, then $a_m = \sum_{i=1}^{n} c_i a_{m-i}$

Mathematical Foundation of PRPG

$$G(X) = \sum_{m=0}^{\infty} a_m X^m = \sum_{m=0}^{\infty} \left(\sum_{i=1}^{n} c_i a_{m-i} \right) X^m = \sum_{i=1}^{n} c_i X^i \sum_{m=0}^{\infty} a_{m-i} X^{m-i}$$

$$= \sum_{i=1}^{n} c_i X^i \left[a_{-i} X^{-i} + \cdots + a_{-1} X^{-1} + \sum_{m=i}^{\infty} a_{m-i} X^{m-i} \right]$$

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

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

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

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

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

Mathematical Foundation of PRPG

- Now $c(X) = 1 + \sum_{i=1}^{n} c_i X^i$ is the characteristic polynomial of the LFSR as defined above. Since $a_{-1}=0$, i=1,2,...,n-1, and $a_{-n}=1$, we have

$$G(X) = \frac{1}{c(X)} = \sum_{m=0}^{\infty} a_m X^m$$

- The sequence $\{a_m\}$ is cyclic, with the period assumed to be p

$$\begin{aligned}
\therefore G(X) = \frac{1}{c(X)} &= (a_0 + a_1 X + \cdots + a_{p-1} X^{p-1}) \\
&\quad + X^p (a_0 + a_1 X + \cdots + a_{p-1} X^{p-1}) \\
&\quad + X^{2p} (a_0 + a_1 X + \cdots + a_{p-1} X^{p-1}) \\
&\quad + \cdots \\
&= (a_0 + a_1 X + \cdots + a_{p-1} X^{p-1})(1 + X^p + X^{2p} + \cdots) \\
&= \frac{(a_0 + a_1 X + \cdots + a_{p-1} X^{p-1})}{1 - X^p}
\end{aligned}$$

$$\therefore \frac{1 - X^p}{c(X)} = a_0 + a_1 X + \cdots + a_{p-1} X^{p-1}$$ i.e., c(X) evenly divides into $1-X^p$

Theorems

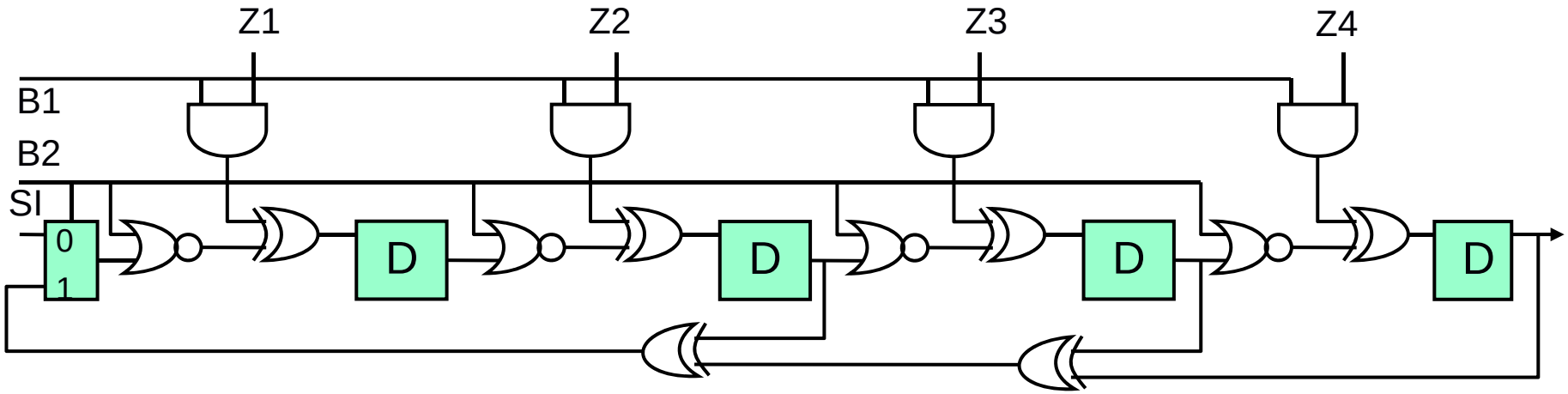
- Theorem: If the initial state of an n-stage LFSR is $a_{-i}=0$, $i=1,2,\ldots,n-1$, and $a_{-n}=1$, then the LFSR sequence $\{a_m\}$ is periodic with a period that is the smallest integer p for which c(X) divides $1-X^p$
  - The period $p<=2^n-1$
  - For a given n, we want to find a c(X) that maximizes p
- Definition: The sequences produced by max-length LFSRs are called *pseudorandom sequences* or *m-sequences*. The characteristic polynomial associated with an m-sequence is called a *primitive polynomial*. An *irreducible polynomial* is one that cannot be factored
  - Pseudorandom sequences (or m-sequences) are not really random since they are produced by a fixed circuit.

Theorems

- Theorem: An *irreducible* polynomial c(X) satisfies the following 2 conditions:
  - It has an odd number of terms including the constant term
  - If its degree n>3, then c(X) must divide $1+X^p$, where $p=2^n-1$
- Theorem: A primitive polynomial is irreducible if the smallest positive integer p that allows the polynomial to divide evenly into $1+X^p$ occurs for $p=2^n-1$, where n is the degree of the polynomial

Built-In-Logic-Block-Observer (BILBO)

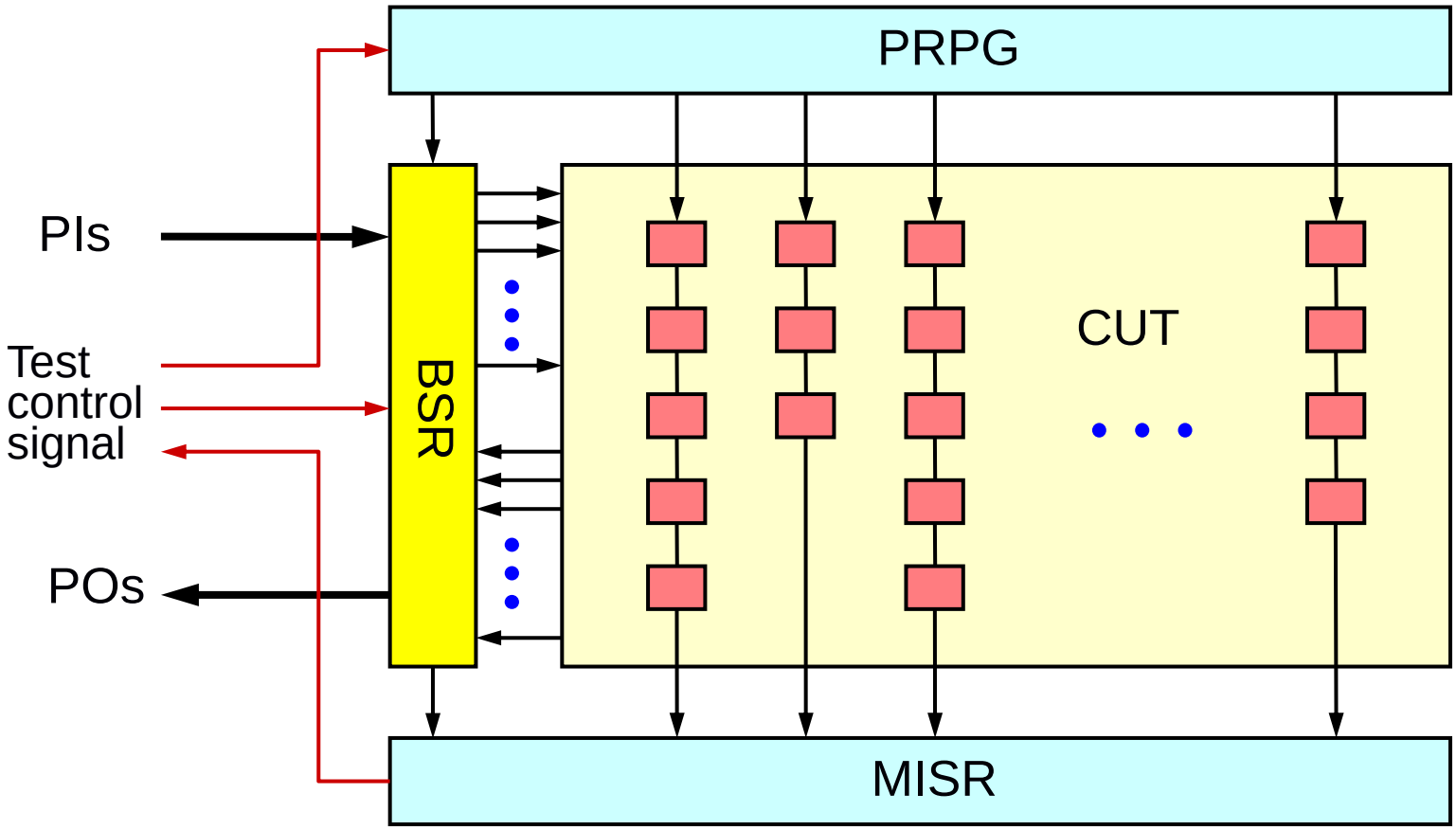
- A BILBO is a multi-purpose test module which serves as a test generator or a signature analyzer. It is composed of a row of FFs and some additional gates for shift and feedback operation



B1	B2	Function
0	1	All FFs are reset
1	1	Behaves as separate latches—normal mode
0	0	A linear shift register—SR mode
1	0	MISR/PRPG—test mode

STUMPS Architecture

- Logic BIST with STUMPS architecture



Summary

- Design-for-testability techniques
  - Ad-hoc techniques
  - Scan
  - LSSD
  - Random access scan
  - Syndrome-testable
  - C-testability
- Scan is a popular DFT technique in modern IC design
- DFT can increase the controllability and observability of the circuit under test

Summary

- Built-in self-test methodology is more and more important for deep submicron designs
- Two key components of BIST
  - Test pattern generator
    - E.g., LFSR
  - Response evaluator
    - E.g., BILBO