Chapter 2

Design for Testability

Design For Testability - contents

Introduction

- History
  - During early years, design and test were separate
    - The final quality of the test was determined by keeping track of the number of defective parts shipped to the customer
    - Defective parts per million (PPM) shipped was a final test score.
    - This approach worked well for small-scale integrated circuit
  - During 1980s, fault simulation was used
    - Failed to improve the circuit's fault coverage beyond 80%
  - Increased test cost and decreased test quality lead to DFT engineering

Introduction

- History
  - Various testability measures & *ad hoc* testability enhancement methods
    - To improve the testability of a design
    - To ease sequential ATPG (automatic test pattern generation)
    - Still quite difficult to reach more than 90% fault coverage
  - Structured DFT
    - To conquer the difficulties in controlling and observing the internal states of sequential circuits
    - Scan design is the most popular structured DFT approach
  - Design for testability (DFT) has migration recently
    - From gate level to register-transfer level (RTL)

Testability Analysis

- Testability:
  - A relative measure of the effort or cost of testing a logic circuit
- Testability Analysis:
  - The process of assessing the testability of a logic circuit
- Testability Analysis Techniques:
  - Topology-based Testability Analysis
    - SCOAP - *Sandia Controllability/Observability Analysis Program*
    - *Probability-based testability analysis*
  - Simulation-based Testability Analysis

***Testability Analysis* – SCOAP**

- *Controllability*
  - Reflects the difficulty of setting a signal line to a required logic value from primary inputs
- *Observability*
  - Reflects the difficulty of propagating the logic value of the signal line to primary outputs

Testability Analysis – SCOAP

- calculates six numerical values for each signal *s* in a logic circuit
  - CC0(*s*): combinational 0-controllability of *s*
  - CC1(*s*): combinational 1-controllability of *s*
  - CO(*s*): combinational observability of *s*
  - SC0(*s*): sequential 0-controllability of *s*
  - SC1(*s*): sequential 1-controllability of *s*
  - SO(*s*): sequential observability of *s*

Testability Analysis – SCOAP

- The value of controllability measures range between 1 to infinite
- The value of observability measures range between 0 to infinite
  - The CC0 and CC1 values of a primary input are set to 1
  - The SC0 and SC1 values of a primary input are set to 0
  - The CO and SO values of a primary output are set to 0

Testability Analysis - SCOAP

Combinational Controllability Calculation Rules

	0-controllability (Primary input, output, branch)	**1-controllability** (Primary input, output, branch)
Primary Input	1	1
AND	*min* {input 0-controllabilities} + 1	Σ (input 1-controllabilities) + 1
OR	Σ (input 0-controllabilities) + 1	*min* {input 1-controllabilities} + 1
NOT	Input 1-controllability + 1	Input 0-controllability + 1
NAND	Σ (input 1-controllabilities) + 1	*min* {input 0-controllabilities} + 1
NOR	*min* {input 1-controllabilities} + 1	Σ (input 0-controllabilities) + 1
BUFFER	Input 0-controllability + 1	Input 1-controllability + 1
XOR	*min* {CC1(*a*)+CC1(*b*), CC0(*a*)+CC0(*b*)} + 1	*min* {CC1(*a*)+CC0(*b*), CC0(*a*)+CC1(*b*)} + 1
XNOR	*min* {CC1(*a*)+CC0(*b*), CC0(*a*)+CC1(*b*)} + 1	*min* {CC1(*a*)+CC1(*b*), CC0(*a*)+CC0(*b*)} + 1
Branch	Stem 0-controllability	Stem 1-controllability

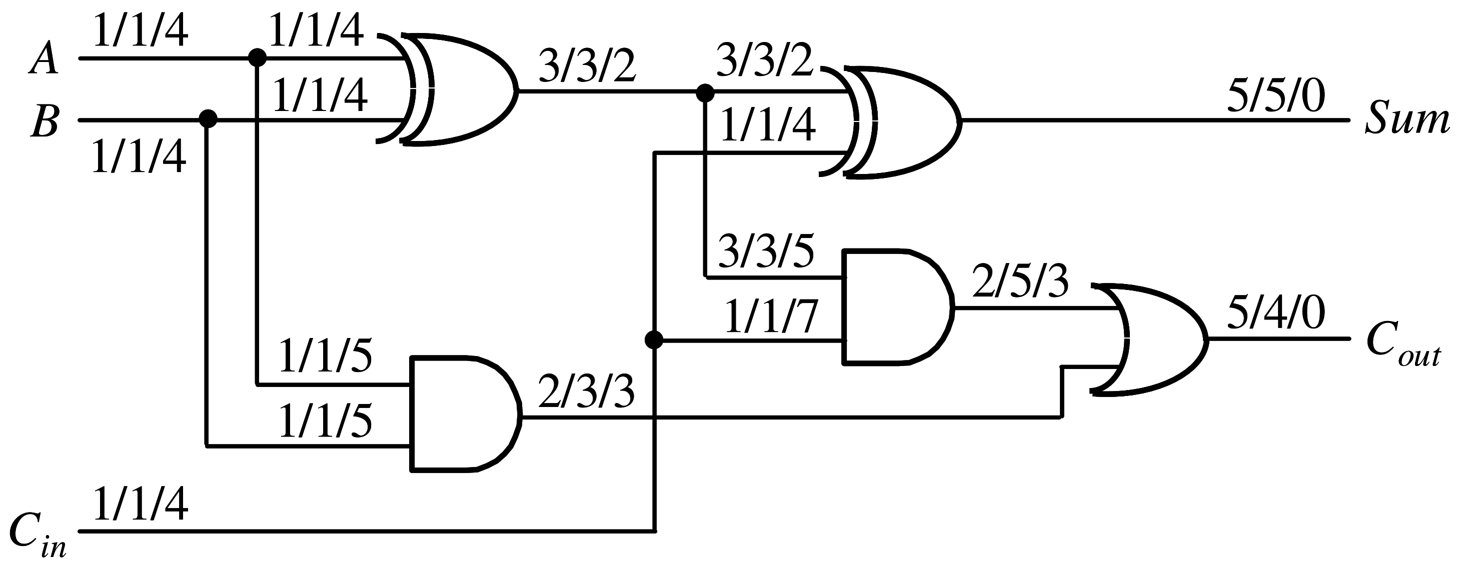
Testability Analysis - SCOAP

Combinational Controllability Observability Rules

	Observability (Primary output, input, stem)
Primary Output	0
AND / NAND	Σ (output observability, 1-controllabilities of other inputs) + 1
OR / NOR	Σ (output observability, 0-controllabilities of other inputs) + 1
NOT / BUFFER	Output observability + 1
XOR / XNOR	*a*: Σ (output observability, *min* {CC0(*b*), CC1(*b*)}) + 1 *b*: Σ (output observability, *min* {CC0(*a*), CC1(*a*)}) + 1
Stem	*min* {branch observabilities}

a, b: inputs of an XOR or XNOR gate

Testability Analysis – SCOAP

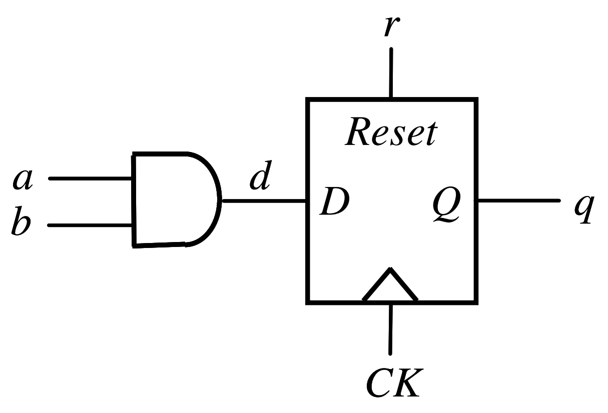


Example of Combinational SCOPA measures

*v*1/*v*2/*v*3 represents the signal's 0-controllability (*v*1), 1-controllability (*v*2), and observability (*v*3)

Testability Analysis - SCOAP

Sequential Controllability and Observability Calculation



SCOAP sequential circuit example

The combinational and sequential controllability measures of signal *d*:

$$\mathrm{CC0(d) = min\,\{CC0(a), CC0(b)\} + 1}$$
$$\mathrm{SC0(d) = min\,\{SC0(a), SC0(b)\}}$$
$$\mathrm{CC1(d) = CC1(a) + CC1(b) + 1}$$
$$\mathrm{SC1(d) = SC1(a) + SC1(b)}$$

Testability Analysis - SCOAP

The combinational and sequential controllability and observability measures of *q:*

$$\mathrm{CC0(q) = min\,\{CC0(d) + CC0(CK) + CC1(CK) + CC0(r), CC1(r) + CC0(CK)\}}$$

$$\mathrm{SC0(q) = min\,\{SC0(d) + SC0(CK) + SC1(CK) + SC0(r) + 1, SC1(r) + SC0(CK)\}}$$

$$\mathrm{CC1(q) = CC1(d) + CC0(CK) + CC1(CK) + CC0(r)}$$

$$\mathrm{SC1(q) = SC1(d) + SC0(CK) + SC1(CK) + SC0(r) + 1}$$

Testability Analysis - SCOAP

The data input *d* can be observed at *q* by holding the reset signal *r* at 0 and applying a rising clock edge to *CK:*

$$\mathrm{CO(d) = CO(q) + CC0(CK) + CC1(CK) + CC0(r)}$$
$$\mathrm{SO(d) = SO(q) + SC0(CK) + SC1(CK) + SC0(r) + 1}$$

Signal *r* can be observed by first setting *q* to 1, and then holding *CK* at the inactive state 0:

$$\mathrm{CO(r) = CO(q) + CC1(q) + CC0(CK)}$$
$$\mathrm{SO(r) = SO(q) + SC1(q) + SC0(CK)}$$

Testability Analysis - SCOAP

- Two ways to indirectly observe the clock signal *CK* at *q*:
  - set *q* to 1, *r* to 0, *d* to 0, and apply a rising clock edge at *CK*
  - set both *q* and *r* to 0, *d* to 1, and apply a rising clock edge at *CK*

$$\mathrm{CO(CK)} = \mathrm{CO(q)} + \mathrm{CC0(CK)} + \mathrm{CC1(CK)} + \mathrm{CC0(r)} + \min\{\mathrm{CC0(d)} + \mathrm{CC1(q)}, \mathrm{CC1(d)} + \mathrm{CC0(q)}\}$$

$$\mathrm{SO(CK)} = \mathrm{SO(q)} + \mathrm{SC0(CK)} + \mathrm{SC1(CK)} + \mathrm{SC0(r)} + \min\{\mathrm{SC0(d)} + \mathrm{SC1(q)}, \mathrm{SC1(d)} + \mathrm{SC0(q)}\} + 1$$

Testability Analysis - SCOAP

The combinational and sequential bservability measures for both inputs *a* and *b:*

$$\mathrm{CO(a) = CO(d) + CC1(b) + 1}$$
$$\mathrm{SO(a) = SO(d) + SC1(b)}$$
$$\mathrm{CO(b) = CO(d) + CC1(a) + 1}$$
$$\mathrm{SO(b) = SO(d) + SC1(a)}$$

Probability-Based *Testability Analysis*

- Used to analyze the **random testability** of the circuit
  - C0(s): probability-based 0-controllability of s
  - C1(s): probability-based 1-controllability of s
  - O(s): probability-based observability of s
- Range between 0 and 1
- C0(s) + C1(s) = 1

Probability-based controllability calculation rules

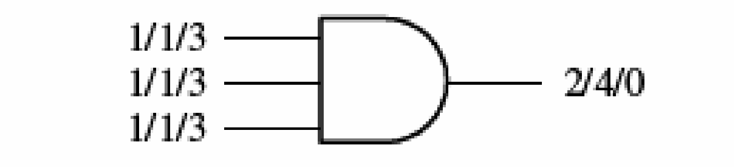
	0-controllability (Primary input, output, branch)	**1-controllability** (Primary input, output, branch)
Primary Input	p_0	$p_1 = 1 - p_0$
AND	1 – (output 1-controllability)	Π (input 1-controllabilities)
OR	Π (input 0-controllabilities)	1 – (output 0-controllability)
NOT	Input 1-controllability	Input 0-controllability
NAND	Π (input 1-controllabilities)	1 – (output 0-controllability)
NOR	1 – (output 1-controllability)	Π (input 0-controllabilities)
BUFFER	Input 0-controllability	Input 1-controllability
XOR	1 – 1-controllabilty	$\Sigma(\mathrm{C1}(a) \times \mathrm{C0}(b), \mathrm{C0}(a) \times \mathrm{C1}(b))$
XNOR	1 – 1-controllability	$\Sigma(\mathrm{C0}(a) \times \mathrm{C0}(b), \mathrm{C1}(a) \times \mathrm{C1}(b))$
Branch	Stem 0-controllability	Stem 1-controllability

Probability-based observability calculation rules

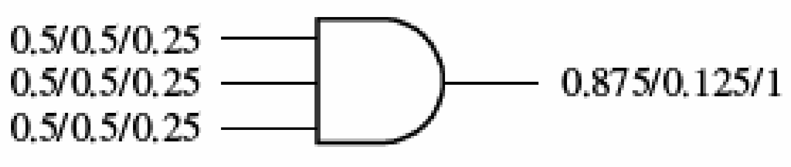
	Observability (Primary output, input, stem)
Primary Output	1
AND / NAND	Π (output observability, 1-controllabilities of other inputs)
OR / NOR	Π (output observability, 0-controllabilities of other inputs)
NOT / BUFFER	Output observability
XOR / XNOR	*a*: Π (output observability, *max* {0-controllability of *b*, 1-controllability of *b*}) *b*: Π (output observability, *max* {0-controllability of *a*, 1-controllability of *a*})
Stem	*max* {branch observabilities}

a, b: inputs of an XOR or XNOR gate

Difference between SCOAP testability measures and probability-based testability measures of a 3-input AND gate



(a) SCOAP combinational measures



(b) Probability-based measures

*v*1/*v*2/*v*3 represents the signal's 0-controllability (*v*1), 1-controllability (*v*2), and observability (*v*3)

Simulation-Based ***Testability Analysis***

- ❑ Supplement to static or topology-based testability analysis
- ❑ Performed through statistical sampling
- ❑ Guide testability enhancement in test generation or logic BIST
- ❑ Generate more accurate estimates
- ❑ Require a long simulation time

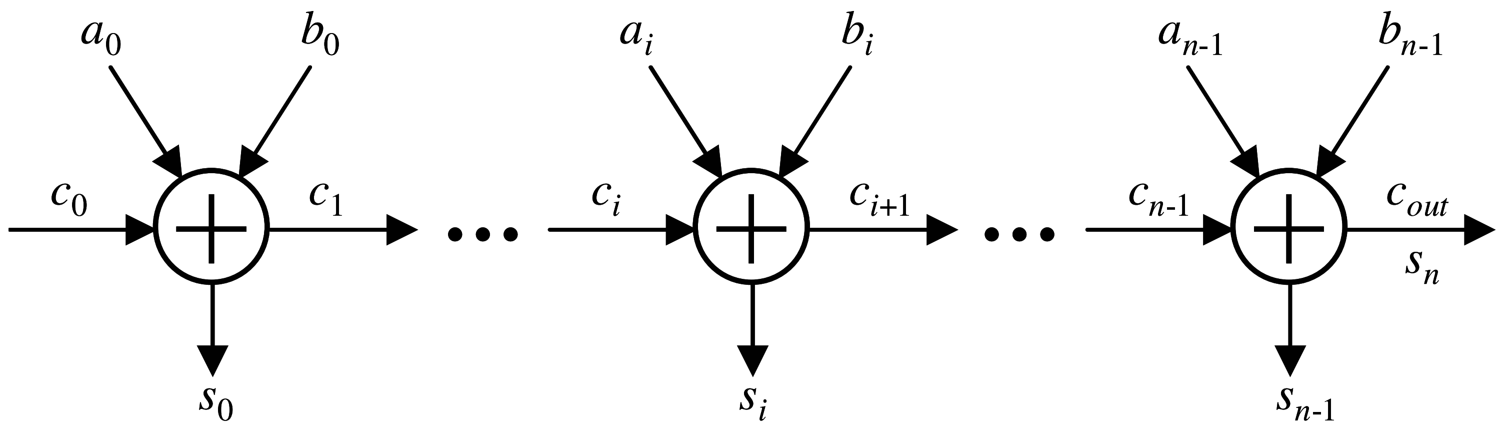
RTL *Testability Analysis*

- Disadvantages of Gate-Level Testability Analysis
  - Costly in term of area overhead
  - Possible performance degradation
  - Require many DFT iterations
  - Long test development time

RTL *Testability Analysis*

- Advantages of RTL Testability Analysis
  - Improve data path testability
  - Improve the random pattern testability of a scan-based logic BIST circuit
  - Lead to more accurate results
    - The number of reconvergent fanouts is much less
  - Become more time efficient
    - Much simpler than an equivalent gate-level model

RTL ***Testability Analysis*** - Example



Ripple-carry adder composed of n full-adders

RTL ***Testability Analysis*** - Example

The probability-based 1-controllability measures of s_i and c_i+1, denoted by C1(s_i) and C1(c_i+1), are calculated as follows:

$$\mathrm{C1(s_i)} = \alpha + \mathrm{C1(c_i)} - 2 \times (\alpha \times \mathrm{C1(c_i)})$$

$$\mathrm{C1(c_i+1)} = \alpha \times \mathrm{C1(c_i)} + \mathrm{C1(a_i)} \times \mathrm{C1(b_i)}$$

$$\alpha = \mathrm{C1(a_i)} + \mathrm{C1(b_i)} - 2 \times \mathrm{C1(a_i)} \times \mathrm{C1(b_i)}$$

α is the probability that $(\mathrm{a_i} \oplus \mathrm{b_i}) = 1$

$\mathrm{C1(s_i)}$ is the probability that $(\mathrm{a_i} \oplus \mathrm{b_i} \oplus \mathrm{c_i}) = 1$

RTL ***Testability Analysis*** - Example

The probability-based 0-controllability of each output *l*, denoted by C0(*l*), in the *n*-bit ripple-carry adder is 1- C1(*l*).

O(*l*, s_i) is defined as the probability that a signal change on *l* will result in a signal change on s_i.

Since $O(a_i, s_i) = O(b_i, s_i) = O(c_i, s_i) = O(s_i)$

$\text{where } i = 0, 1, \ldots, n-1$

This calculation is left as a problem at the end of this chapter.

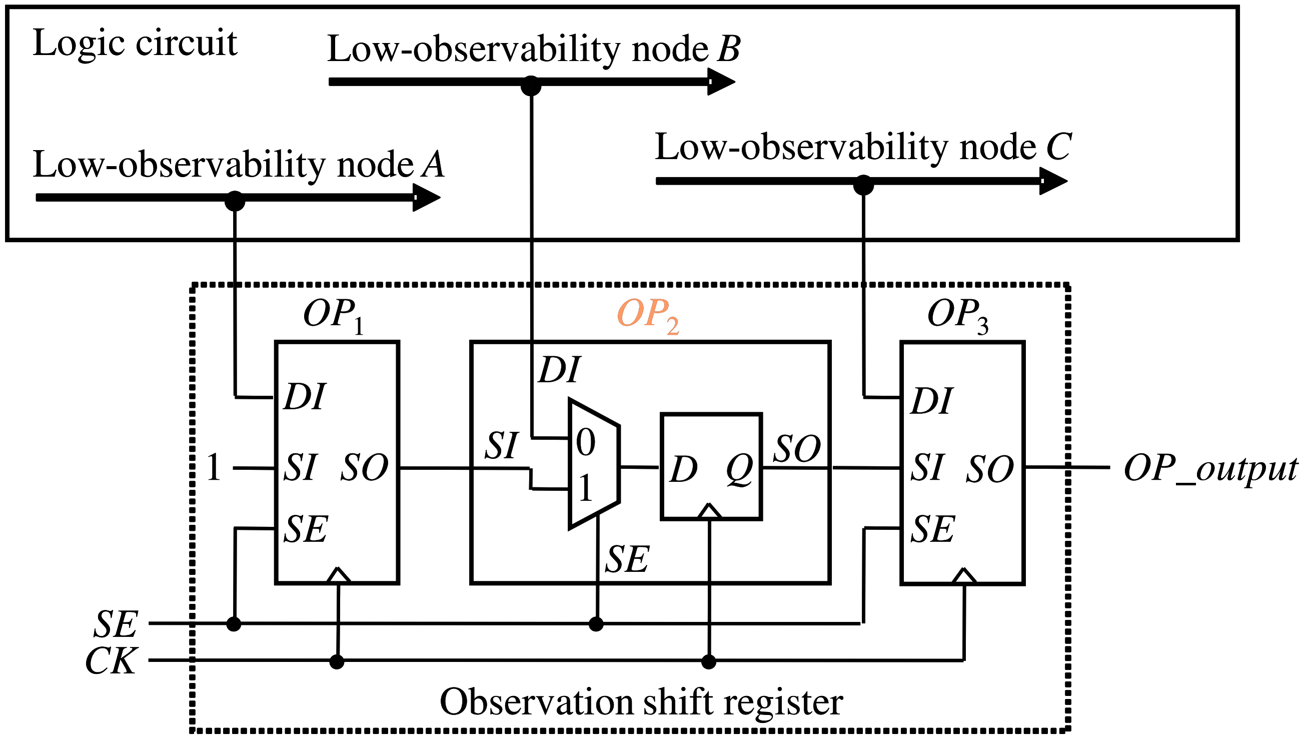
Design for Testability Basics

- *Ad hoc* DFT
  - Effects are local and not systematic
  - Not methodical
  - Difficult to predict
- A structured DFT
  - Easily incorporated and budgeted
  - Yield the desired results
  - Easy to automate

Ad Hoc Approach

- Typical *ad hoc* DFT techniques
  - Insert test points
  - Avoid asynchronous set/reset for storage elements
  - Avoid combinational feedback loops
  - Avoid redundant logic
  - Avoid asynchronous logic
  - Partition a large circuit into small blocks

Ad Hoc Approach – Test Point Insertion

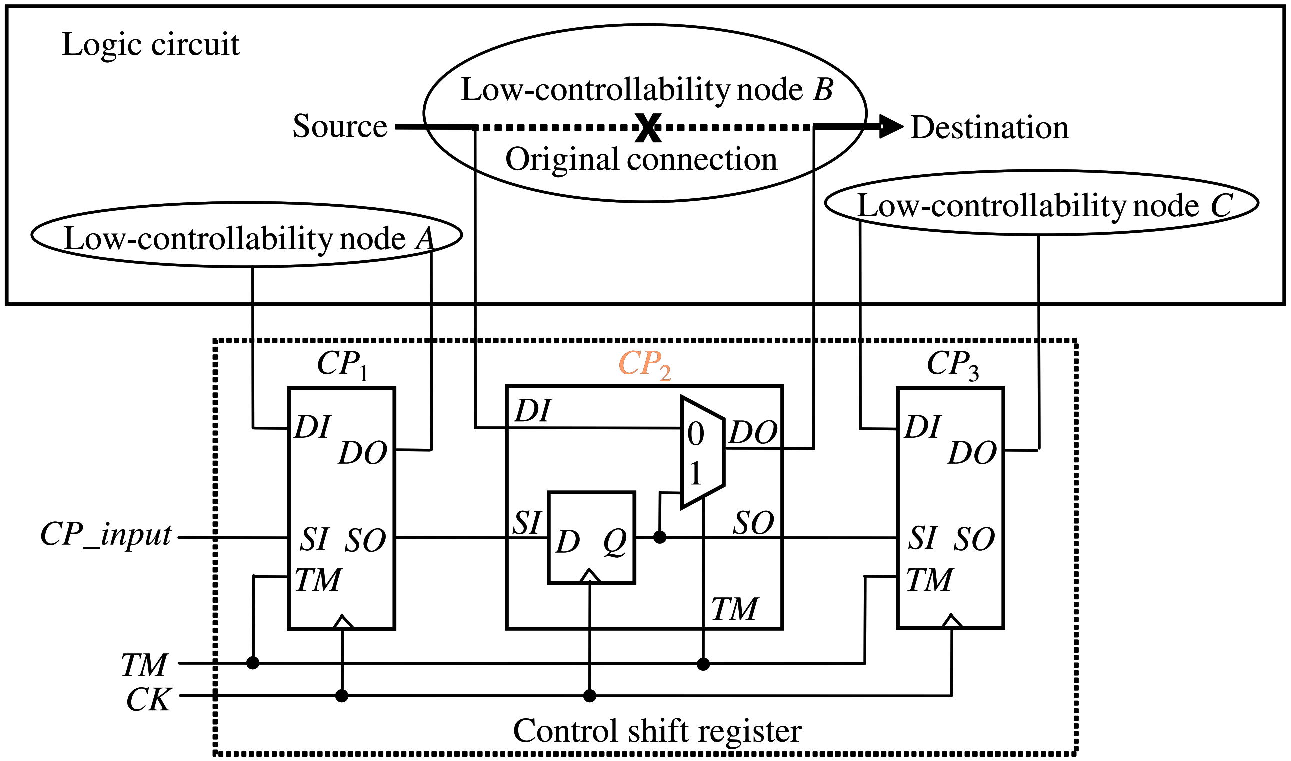


Logic circuit
Low-observability node *B*
Low-observability node *A*
Low-observability node *C*
OP_1
OP_2
OP_3
DI
SI
SO
SE
1
0
D
Q
SE
CK
OP_output
Observation shift register

Observation point insertion

OP2 shows the structure of an observation, which is composed of a multiplexer (MUX) and a D flip-flop.

Ad Hoc Approach – Test Point Insertion



Control point insertion

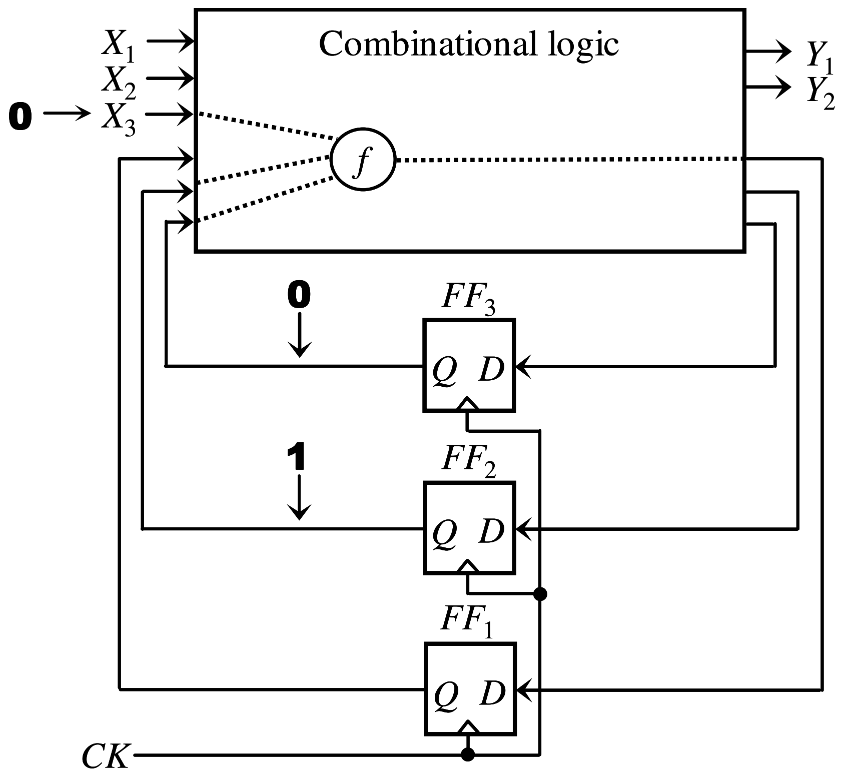
A MUX is inserted between the source and destination ends. During normal operation, $TM = 0$, such that the value from the source end drives the destination end through the 0 port of the MUX.

During test, $TM = 1$ such that the value from the D flip-flop drives the destination end through the 1 port of the MUX.

Structured Approach

- Scan design
  - Convert the sequential design into a scan design
  - Three modes of operation
    - Normal mode
      - All test signals are turned off
      - The scan design operates in the original functional configuration
    - Shift mode
    - Capture mode
      - In both shift and capture modes, a test mode signal *TM* is often used to turn on all test-related fixes

Structured Approach - Scan Design

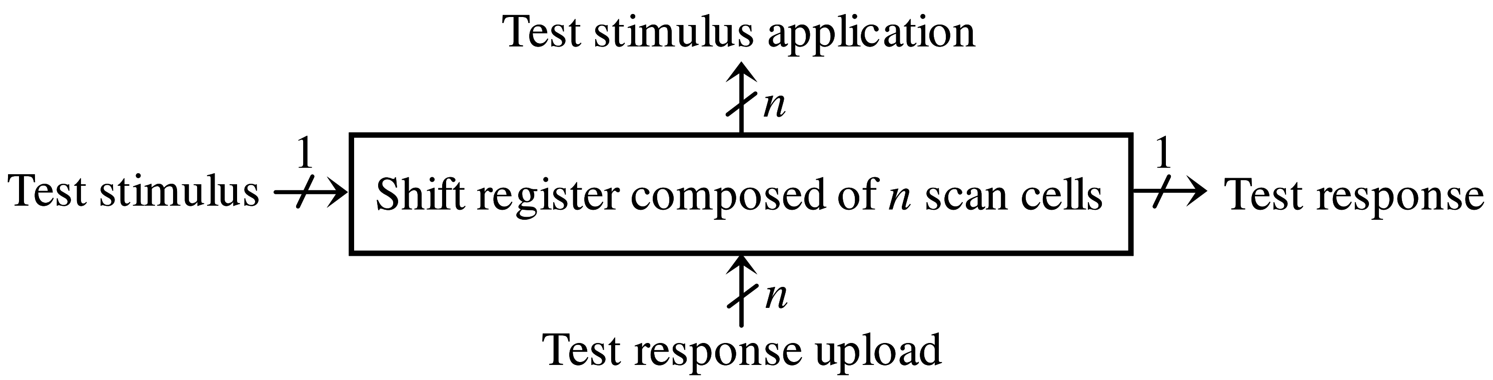


Assume that a stuck-at fault *f* in the combinational logic requires the primary input *X*3, flip-flop FF_2, and flip-flop FF_3, to be set to 0, 1, and 0.

The main difficulty in testing a sequential circuit stems from the fact that it is difficult to control and observe the internal state of the circuit.

Difficulty in testing a sequential circuit

Structured Approach - Scan Design


Test stimulus application
n
1
Test stimulus
Shift register composed of n scan cells
1
Test response
n
Test response upload

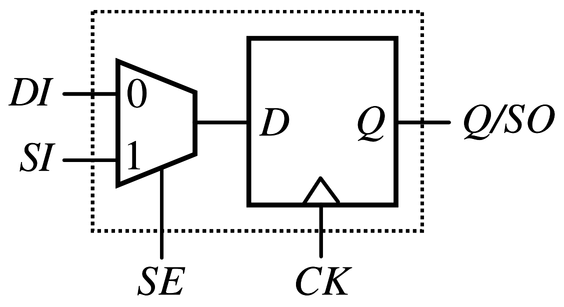
1. Converting selected storage elements in the design into **scan cells**.
1. Stitching them together to form **scan chains**.

How to detect stuck-at fault *f* :
(1) switching to shift mode and shifting in the desired test stimulus, 1 and 0, to FF_2 and FF_3, respectively
(2) driving a 0 onto primary input X_3
(3) switching to capture mode and applying one clock pulse to capture the fault effect into FF_1
(4) switching back to shift mode and shifting out the test response stored in FF_1, FF_2, and FF_3 for comparison with the expected response.

Scan Cell Design

- A scan cell has two inputs: data input and scan input
  - In normal/capture mode, data input is selected to update the output
  - In shift mode, scan input is selected to update the output
- Three widely used scan cell designs
  - Muxed-D Scan Cell
  - Clocked-Scan Cell
  - LSSD Scan Cell

Muxed-D Scan Cell

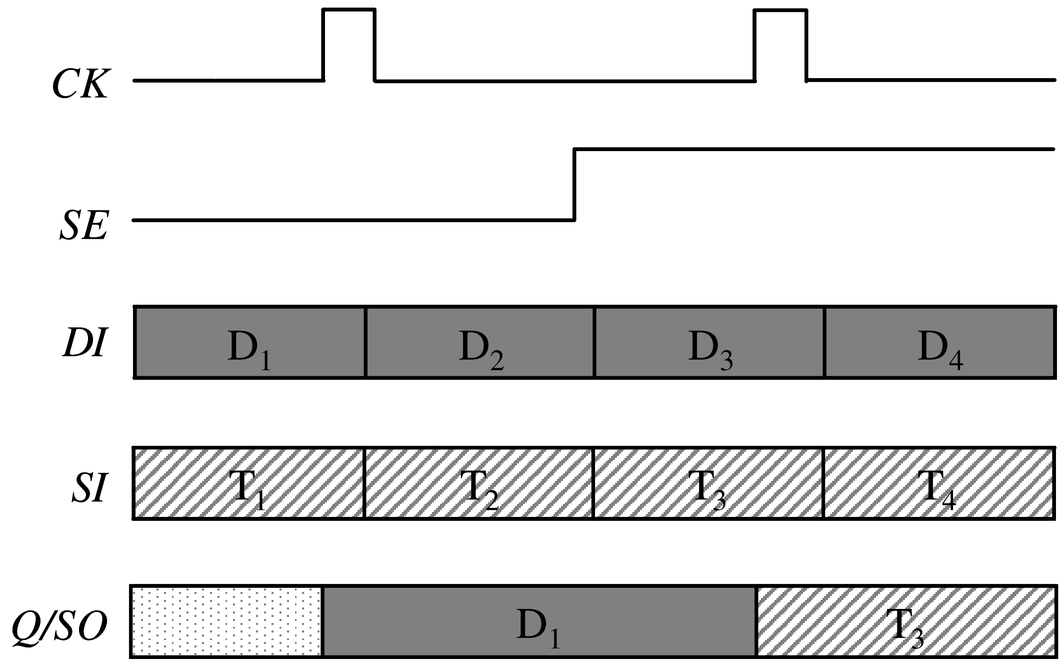


Edge-triggered muxed-D scan cell

This scan cell is composed of a D flip-flop and a multiplexer.

The multiplexer uses an additional scan enable input *SE* to select between the data input *DI* and the scan input *SI*.

Muxed-D Scan Cell

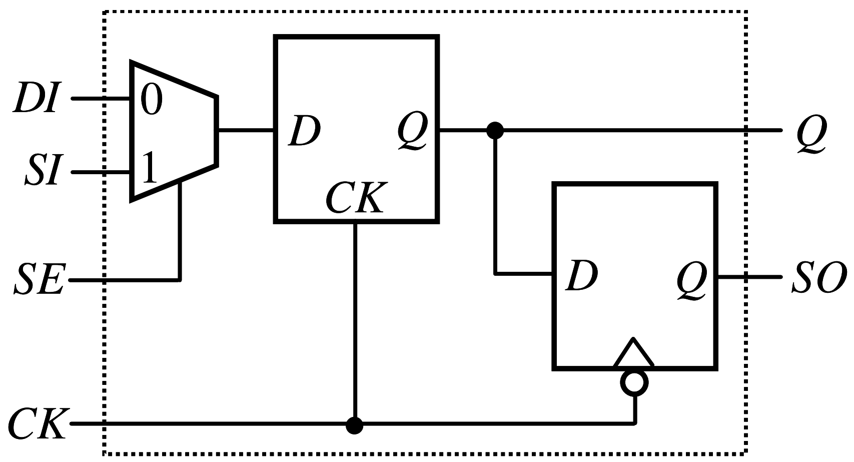


Edge-triggered muxed-D scan cell design and operation

In normal/capture mode, *SE* is set to 0. The value present at the data input *DI* is captured into the internal D flip-flop when a rising clock edge is applied.

In shift mode, *SE* is set to 1. The scan input *SI* is used to shift in new data to the D flip-flop, while the content of the D flip-flop is being shifted out.

Muxed-D Scan Cell

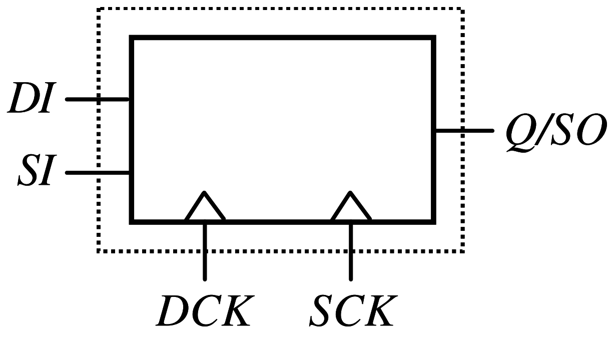


Level-sensitive/edge-triggered muxed-D scan cell design

This scan cell is composed of a multiplexer, a D latch, and a D flip-flop.

In this case, shift operation is conducted in an edge-triggered manner, while normal operation and capture operation is conducted in a level-sensitive manner.

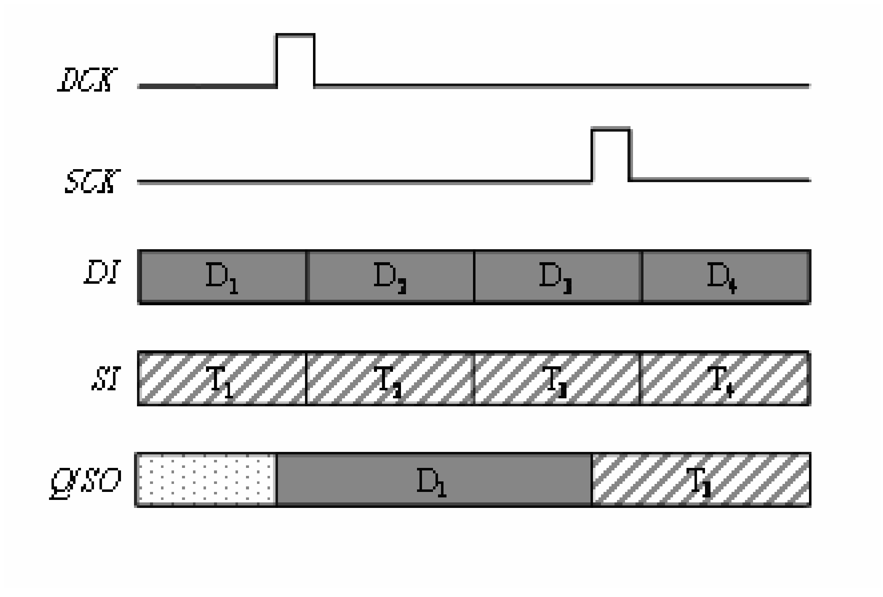
Clocked-Scan Cell



Clocked-scan cell

In the clocked-scan cell, input selection is conducted using two independent clocks, *DCK* and *SCK*.

Clocked-Scan Cell

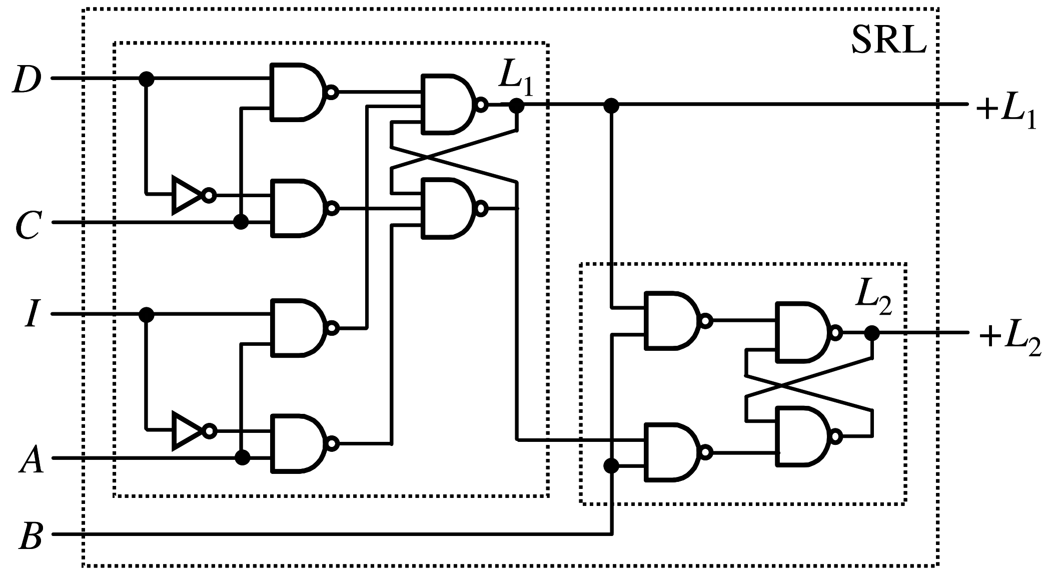


Clocked-scan cell design and operation

In normal/capture mode, the data clock *DCK* is used to capture the contents present at the data input *DI* into the clocked-scan cell.

In shift mode, the shift clock *SCK* is used to shift in new data from the scan input *SI* into the clocked - scan cell, while the content of the clocked-scan cell is being shifted out.

LSSD Scan Cell

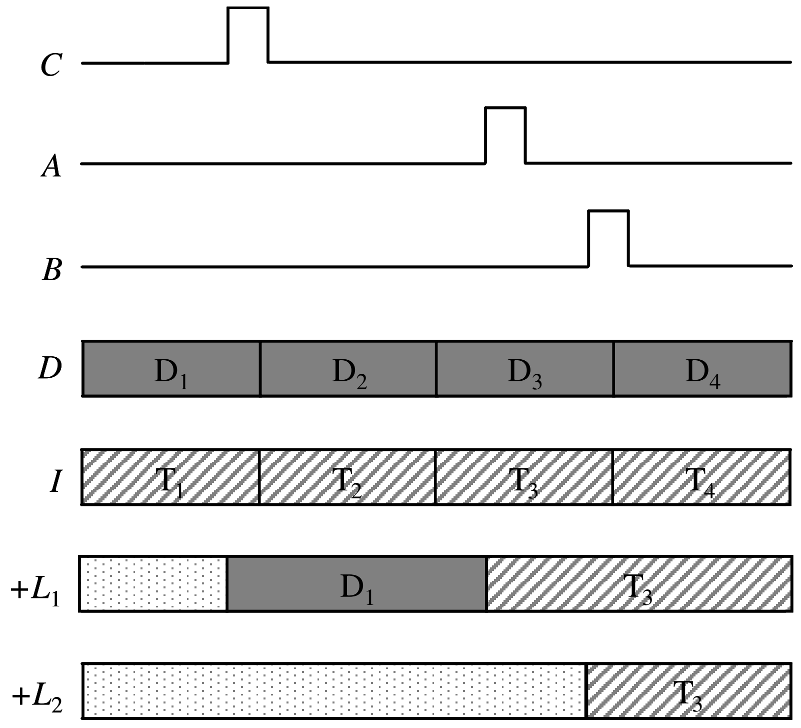


Polarity-hold SRL
(shift register latch)

An LSSD scan cell is used for level-sensitive latch base designs.

This scan cell contains two latches, a master 2-port D latch L_1 and a slave D latch L_2. Clocks C, A and B are used to select between the data input D and the scan input I to drive $+L_1$ and $+L_2$. In an LSSD design, either $+L_1$ or $+L_2$ can be used to drive the combinational logic of the design.

LSSD Scan Cell



Polarity-hold SRL design and operation

In order to guarantee race-free operation, clocks A, B, and C are applied in a non-overlapping manner.

The master latch L_1 uses the system clock C to latch system data from the data input D and to output this data onto $+L_1$. Clock B is used after clock A to latch the system data from latch L_1 and to output this data onto $+L_2$.

Comparing three scan cell designs

	Advantages	Disadvantages
Muxed-D Scan Cell	Compatibility to modern designs Comprehensive support provided by existing design automation tools	Add a multiplexer delay
Clocked-Scan Cell	No performance degradation	Require additional shift clock routing
LSSD Scan Cell	Insert scan into a latch-based design Guarantee to be race-free	Increase routing complexity

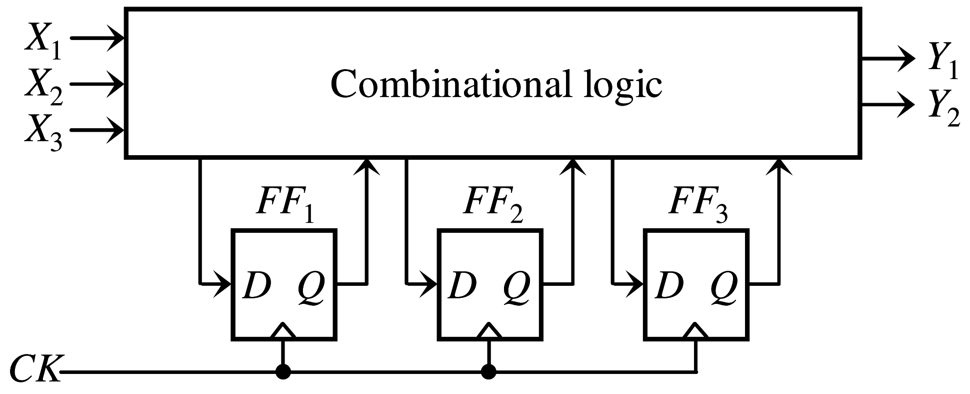
Scan Architectures

- Full-Scan Design
  - All or almost all storage element are converted into scan cells and combinational ATPG is used for test generation
- Partial-Scan Design
  - A subset of storage elements are converted into scan cells and sequential ATPG is typically used for test generation
- Random-Access Scan Design
  - A random addressing mechanism, instead of serial scan chains, is used to provide direct access to read or write any scan cell

Full-Scan Design

- All storage elements are replaced with scan cells
  - All inputs can be controlled
  - All outputs can be observed
- Advantage:
  - Converts sequential ATPG into combinational ATPG
- Almost full-scan design
  - A small percentage of storage elements are not replaced with scan cells
    - For performance reasons
      - Storage elements that lie on critical paths
    - For functional reasons
      - Storage elements driven by a small clock domain that are deemed too insignificant to be worth the additional scan insertion effort

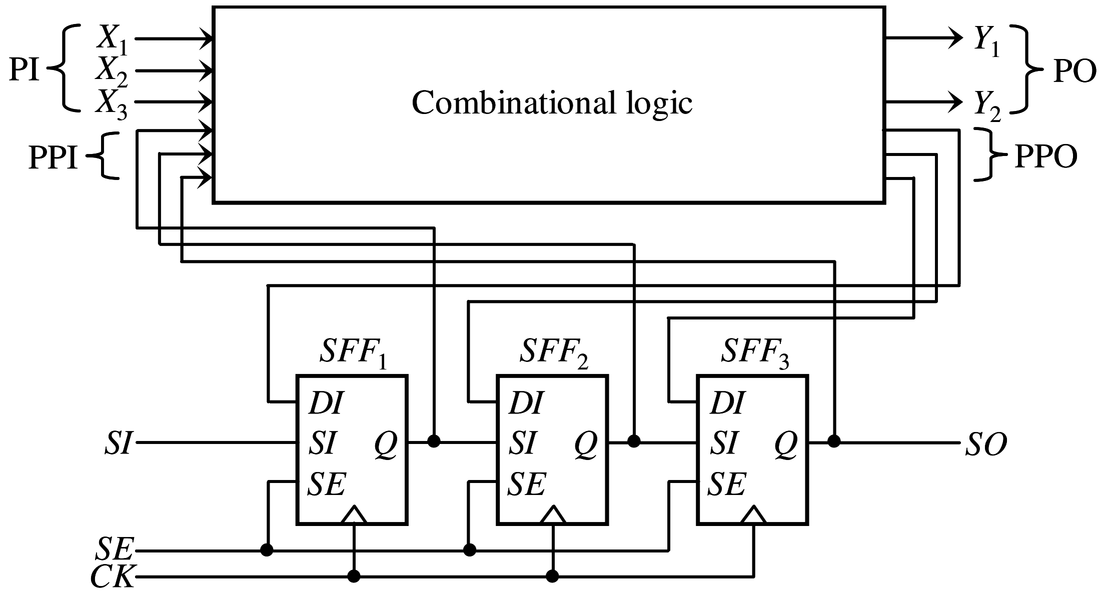
Muxed-D ***Full-Scan Design***



Sequential circuit example

The three D flip-flops, FF_1, FF_2 and FF_3, are replaced with three muxed-D scan cells, SFF_1, SFF_2 and SFF_3, respectively.

Muxed-D ***Full-Scan Design***



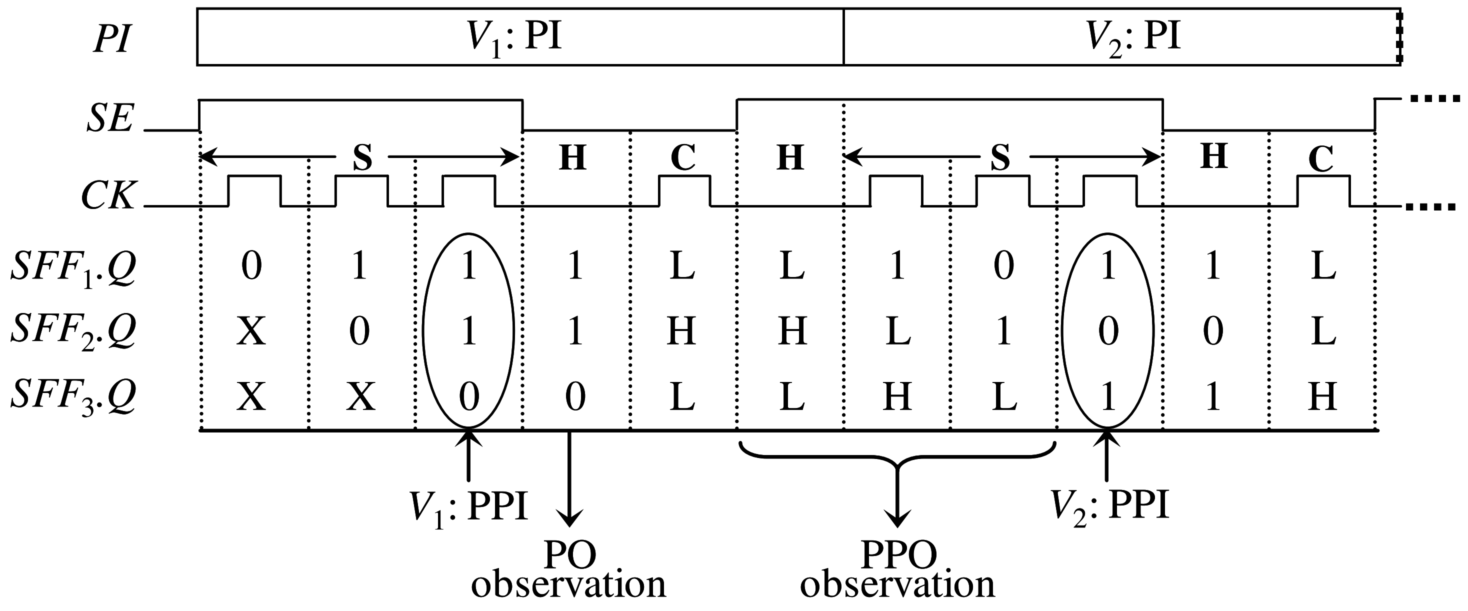
(a) Muxed-D full-scan circuit

To form a scan chain, the scan input *SI* of SFF_2 and SFF_3 are connected to the output *Q* of the previous scan cell, SFF_1 and SFF_2, respectively. In addition, the scan input *SI* of the first scan cell SFF_1 is connected to the primary input *SI*, and the output *Q* of the last scan cell SFF_3 is connected to the primary output *SO*.

Muxed-D ***Full-Scan Design***

- *Primary inputs* (PIs)
  - the external inputs to the circuit
  - can be set to any required logic values
  - set directly in parallel from the external inputs
- *Pseudo primary inputs* (PPIs)
  - the scan cell outputs
  - can be set to any required logic values
  - are set serially through scan chain inputs
- *Primary outputs* (POs)
  - the external outputs of the circuit
  - can be observed
  - are observed directly in
  - parallel from the external outputs
- *Pseudo primary outputs* (PPOs)
  - the scan cell inputs
  - can be observed
  - are observed serially through scan chain outputs

Muxed-D ***Full-Scan Design***



S: shift operation / **C**: capture operation / **H**: hold cycle

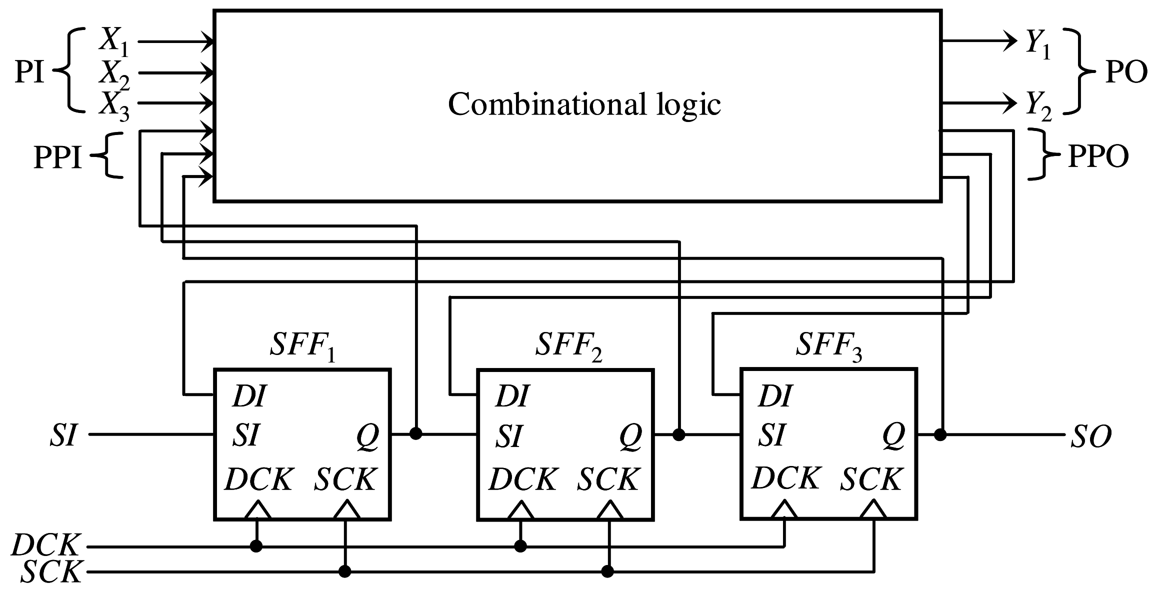
(b) Test operations

Muxed-D ***Full-Scan Design***

Circuit Operation type	Scan cell mode	*TM*	SE
Normal	Normal	0	0
Shift Operation	Shift	1	1
Capture Operation	Capture	1	0

Circuit operation type and scan cell mode

Clocked ***Full-Scan Design***



Clocked full-scan circuit

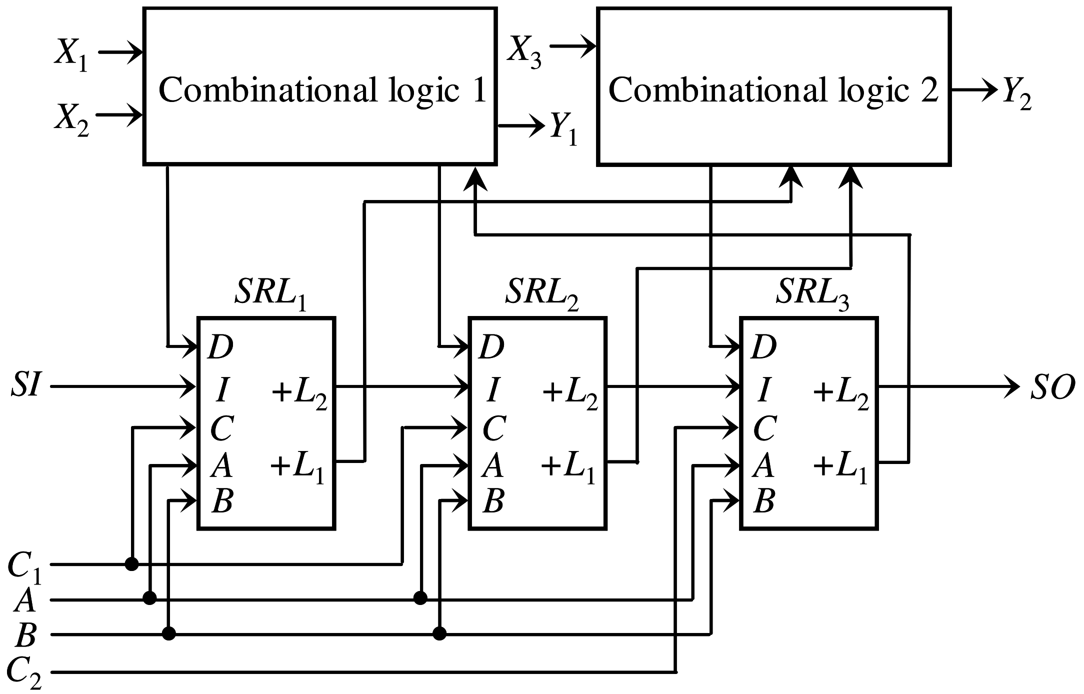
In a muxed-D full-scan circuit, a scan enable signal *SE* is used.

In a clocked full-scan design, two operations are distinguished by properly applying the two independent clocks *SCK* and *DCK* during shift mode and capture mode.

LSSD *Full-Scan Design*

- Single-latch design
- Double-latch design

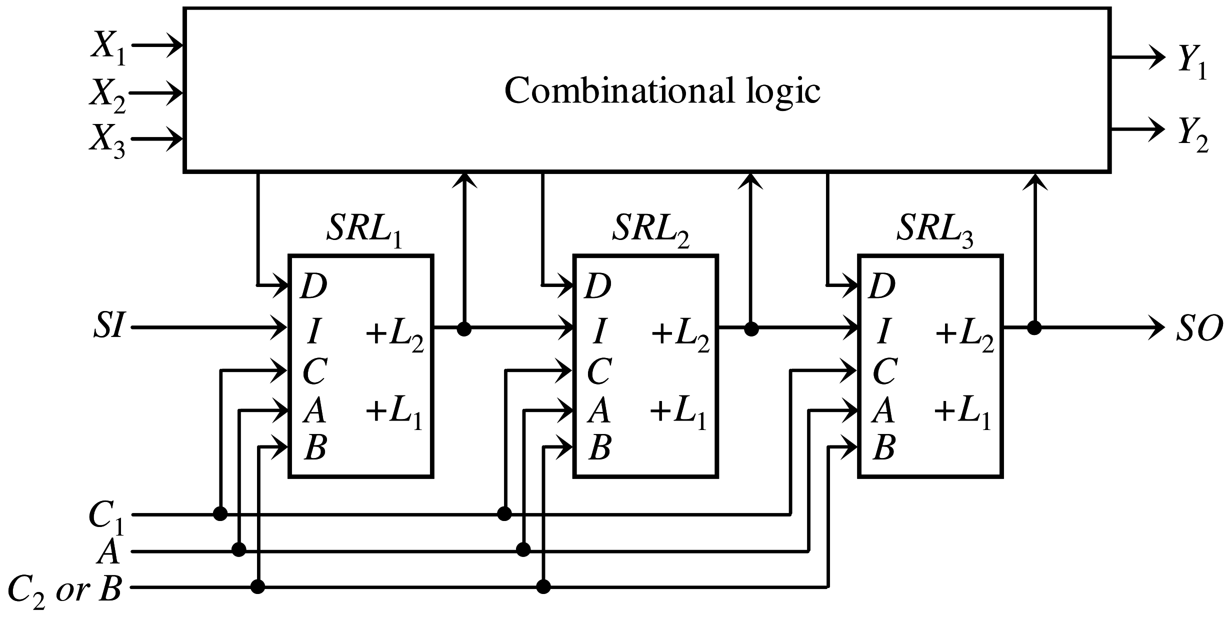
LSSD *Full-Scan Design*



Single-latch design

The output port $+L_1$ of the master latch L_1 is used to drive the combinational logic of the design. In this case, the slave latch L_2 is only used for scan testing.

LSSD *Full-Scan Design*



Double-latch design

In normal mode, the C_1 and C_2 clocks are used in a non-overlapping Manner.

During the shift operation, clocks *A* and *B* are applied in a non-overlapping manner, the scan cells SRL_1 ~ SRL_3 form a single scan chain from *SI* to *SO*.

During the capture operation, clocks C_1 and C_2 are applied to load the test response from the combinational logic into the scan cells.

LSSD Design Rules

- All storage elements must be polarity-hold latches.
- The latches are controlled by two or more non-overlapping clocks.
- A set of clock primary inputs must follow three conditions:
  - All clock inputs to SRLs must be inactive when clock PIs are inactive
  - The clock input to any SRL must be controlled from one or more clock primary inputs
  - No clock can be ANDed with another clock or its complement

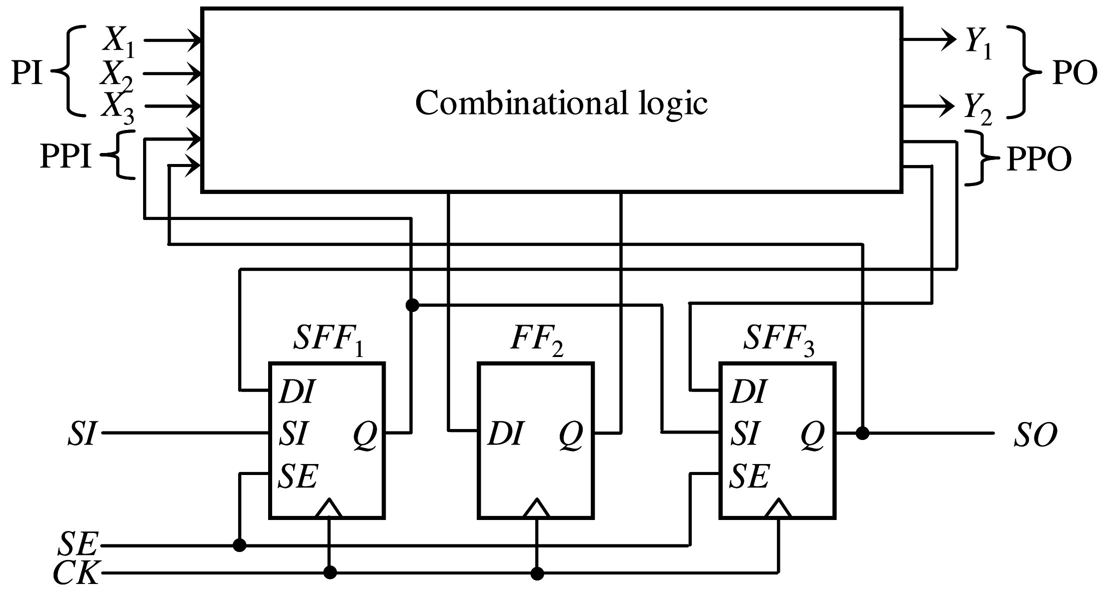
LSSD Design Rules

- Clock primary inputs must not feed the data inputs to SRLs either directly or through combinational logic.
- Each system latch must be part of an SRL, and each SRL must be part of a scan chain.
- A scan state exists under certain conditions:
  - Each SRL or scan out *SO* is a function of only the preceding SRL or scan input *SI* in its scan chain during the scan operation
  - All clocks except the shift clocks are disabled at the SRL clock inputs

Partial-Scan Design

- ❑ Was once used in the industry long before full-scan design became the dominant scan architecture.
- ❑ Can also be implemented using muxed-D scan cells, clocked-scan cells, or LSSD scan cells.
- ❑ Either combinational ATPG or sequential ATPG can be used.

Partial-Scan Design



An example of muxed-D partial-scan design

A scan chain is onstructed with two scan cells SFF_1 and SFF_3, while flip-flop FF_2 is left out.

It is possible to reduce the test generation complexity by splitting the single clock into two separate clocks, one for controlling all scan cells, the other for controlling all non-scan storage elements. However, this may result in additional complexity of routing two separate clock trees during physical implementation.

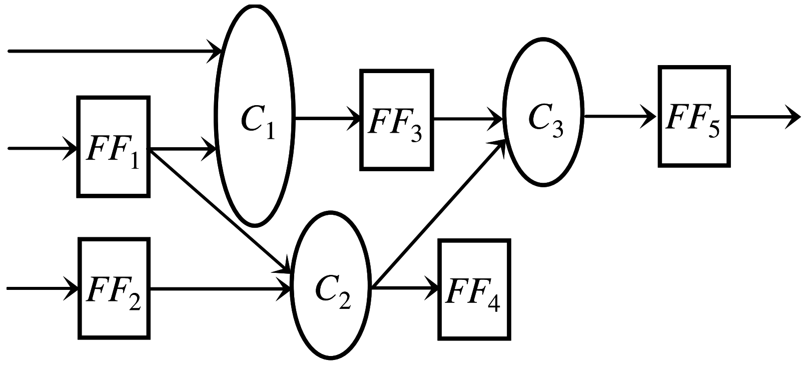
Partial-Scan Design

- Scan cell selection
  - A functional partitioning approach
    - A circuit is composed of a data path portion and a control portion
    - Storage elements on the data path are left out of the scan cell replacement process
    - Storage elements on the control path can be replaced with scan cells
  - A pipelined or feed-forward partial-scan design approach
    - Make the sequential circuit feedback-free by selecting the storage elements to break all sequential feedback loops
    - First construct a structure graph for the sequential circuit
  - A balanced partial-scan design approach
    - Use a target sequential depth to simply the test generation process for the pipelined or feed-forward partial-scan design

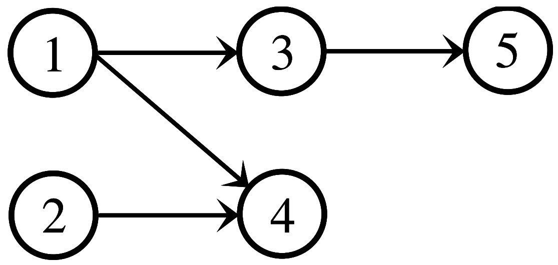
Partial-Scan Design - Structure Graph

- A feedback-free sequential circuit
  - Use a *directed acyclic graph* (DAG)
  - The maximum level in the structure graph is referred to as **sequential depth**
- A sequential circuit containing feedback loops
  - Use a *directed cyclic graph* (DCG)

Sequential circuit and its structure graph



(a) Sequential Circuit



(b) Structure graph
Sequential depth is 3

The sequential depth of a circuit is equal to the maximum number of clock cycles that needs to be applied in order to control and observe values to and from all non-scan storage elements

- The sequential depth of a full-scan circuit is 0

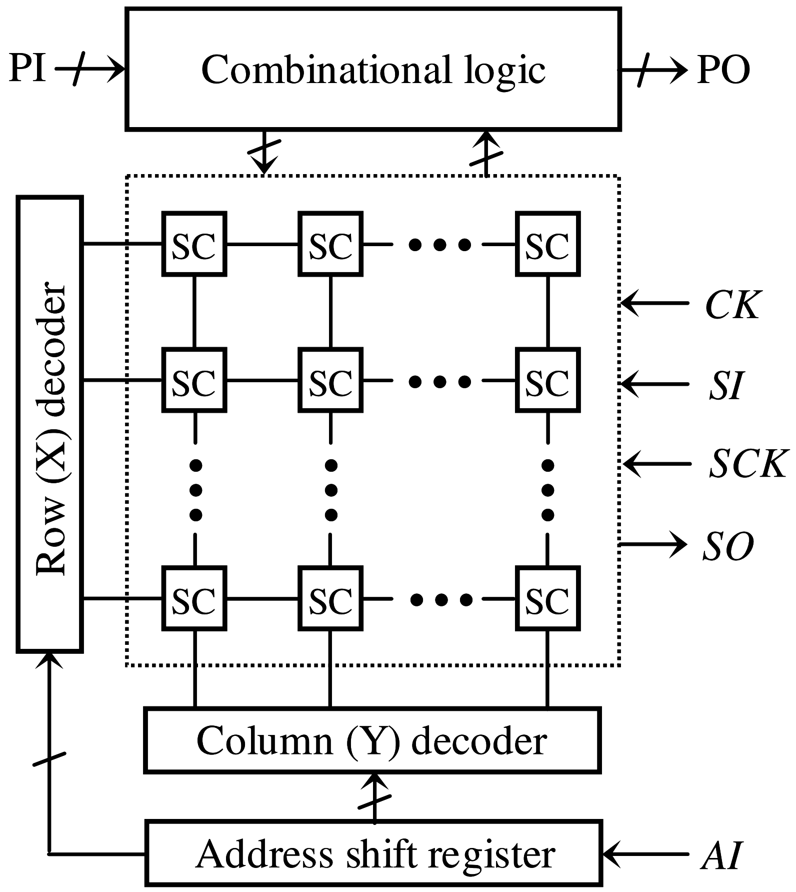
Partial-Scan Design

- Advantage:
  - Reduce silicon area overhead
  - Reduce performance degradation
- Disadvantage:
  - Can result in lower fault coverage
  - Longer test generation time
  - Offers less support for debug, diagnosis and failure analysis

Random-Access Scan Design

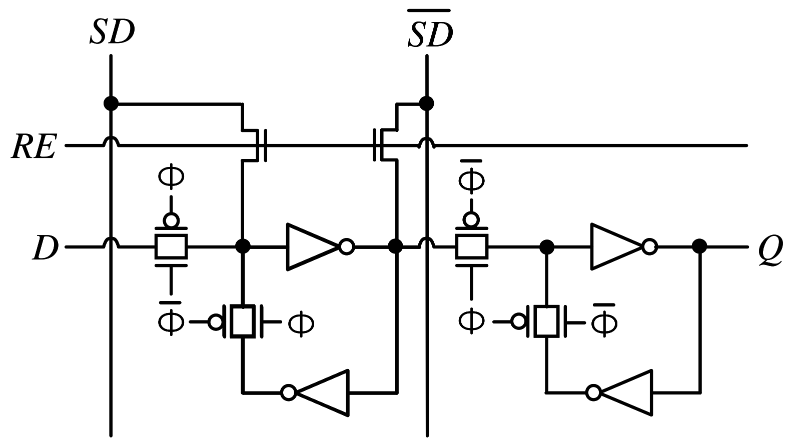
- Advantages of RAS:
  - Can control or observe individual scan cells without affecting others
  - Reduce test power dissipation
  - Simplify the process of performing delay test
- Disadvantages of traditional RAS:
  - High overhead in scan design and routing
  - No guarantee to reduce the test application time
- Progressive Random-Access Scan(PRAS) was proposed to alleviate the disadvantages in traditional RAS

Traditional random-access scan architecture



All scan cells are organized into a two-dimensional array. A $\lceil \log_2^n \rceil$ -bit address shift register, where *n* is the total number of scan cells, is used to specify which scan cell to access.

Progressive Random-Access Scan (PRAS)



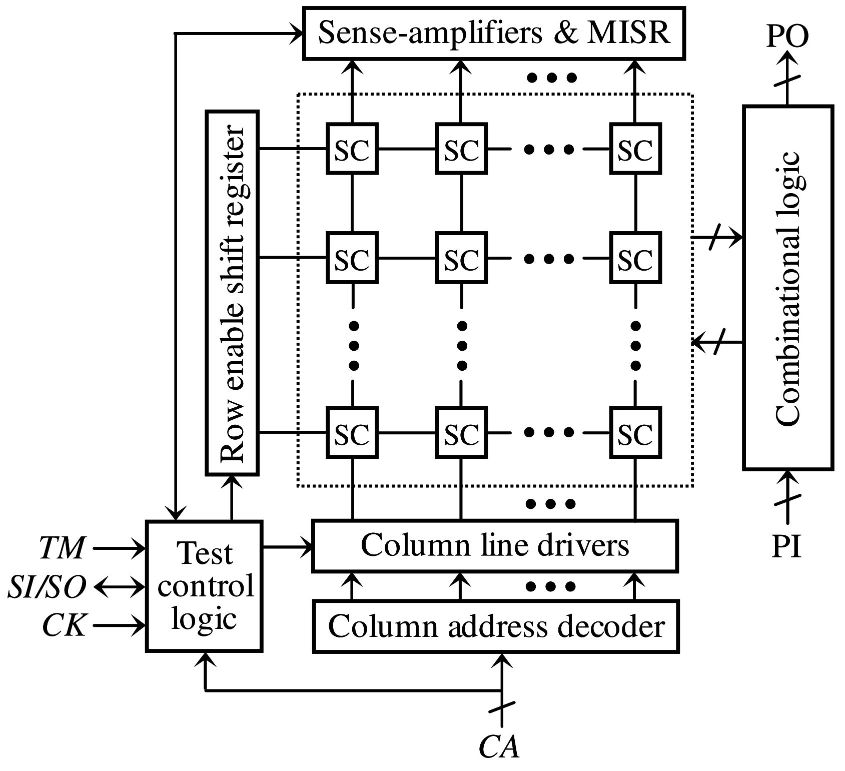
PRAS scan cell design

Structure is similar to that of a *static random access memory* (SRAM) cell or a grid addressable latch.

In normal mode, all horizontal row enable (*RE*) signals are set to 0, forcing each scan cell to act as a normal *D* flip-flop.

In test mode, to capture the test response from *D*, the *RE* signal is set to 0 and a pulse is applied on clock Φ, which causes the value on *D* to be loaded into the scan cell.

Progressive Random-Access Scan (PRAS)



PRAS Architecture

Rows are enabled in a fixed order.

It is only necessary to supply a column address to specify which scan cell in an enabled row to access.

PRAS - test procedure

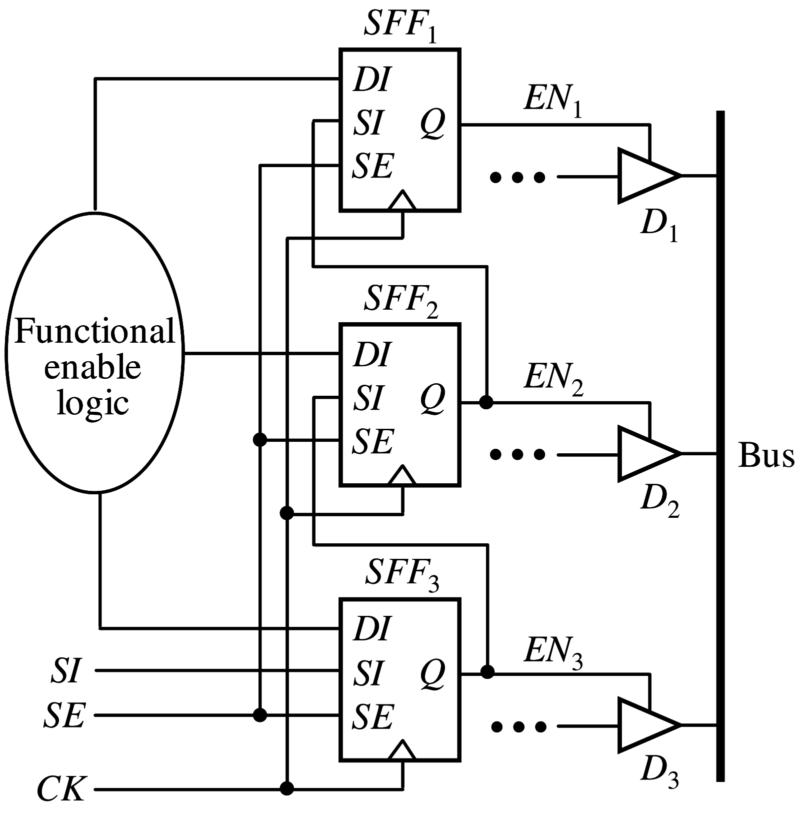
```
for each test vector $v_i$ ($i$ = 1, 2, …, $N$) {
    /* Test stimulus application */
    /* Test response compression */
    enable TM;
    for each row $r_j$ ($j$ = 1, 2, …, $m$) {
        read all scan cells in $r_j$ / update MISR;
        for each scan cell SC in $r_j$
        /* $v$(SC): current value of SC  */
        /* $v_i$(SC): value of SC in $v_i$ */
            if $v(SC) \neq v_i(SC)$
                update SC;
    }
    /* Test response acquisition */
    disable TM;
    apply the normal clock;
}
scan-out MISR as the final test response;
```

For each test vector, the test stimulus application and test response compression are conducted in an interleaving manner when the test mode signal *TM* is enabled.

Scan Design Rules

Design Style	*Scan Design Rule*	*Recommended Solution*
Tri-state buses	Avoid during shift	Fix bus contention during shift
Bi-directional I/O ports	Avoid during shift	Force to input or output mode during shift
Gated clocks (muxed-D full-scan)	Avoid during shift	Enable clocks during shift
Derived clocks (muxed-D full-scan)	Avoid	Bypass clocks
Combinational feedback loops	Avoid	Break the loops
Asynchronous set/reset signals	Avoid	Use external pin(s)
Clocks driving data	Avoid	Block clocks to the data portion
Floating buses	Avoid	Add bus keepers
Floating inputs	Not recommended	Tie to Vcc or ground
Cross-coupled NAND/NOR gates	Not recommended	Use standard cells
Non-scan storage elements	Not recommended for full-scan Design	Initialize to known states, bypass, or make transparent

Tri-State Buses



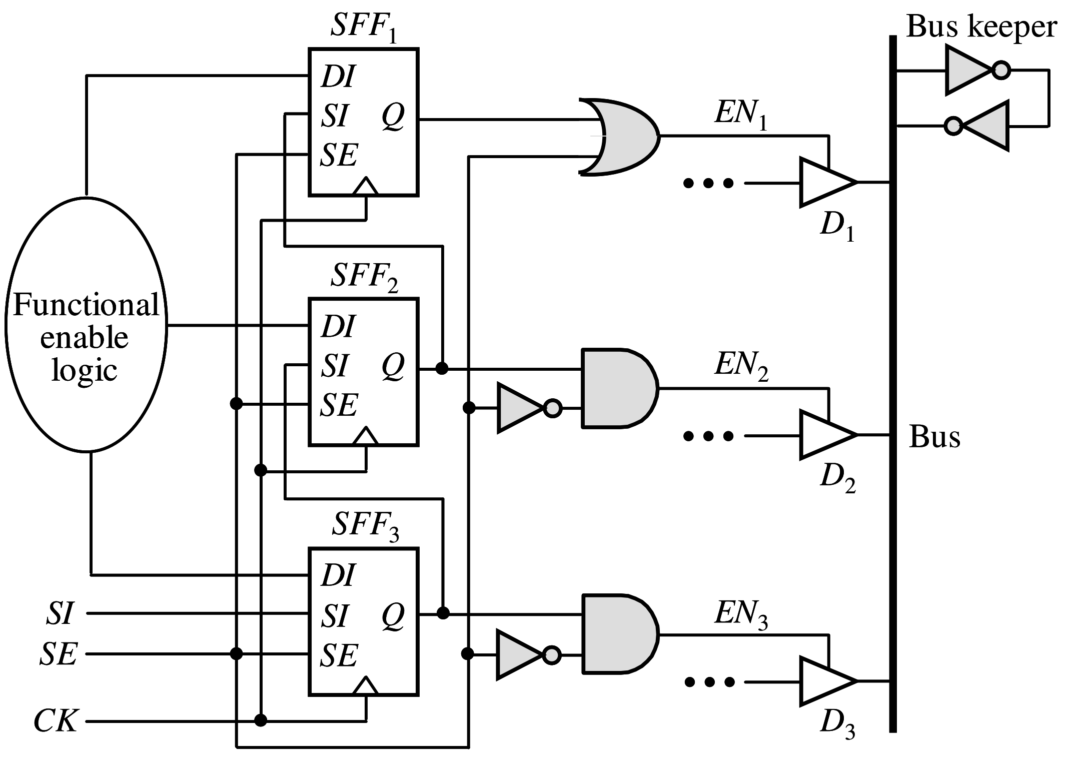
Original Circuit

Bus contention occurs when two bus drivers force opposite logic values onto a tri-state bus.

Bus contention is designed not to happen during the normal operation, and is typically avoided during the capture operation.

However, during the shift operation, no such guarantees can be made.

Tri-State Buses

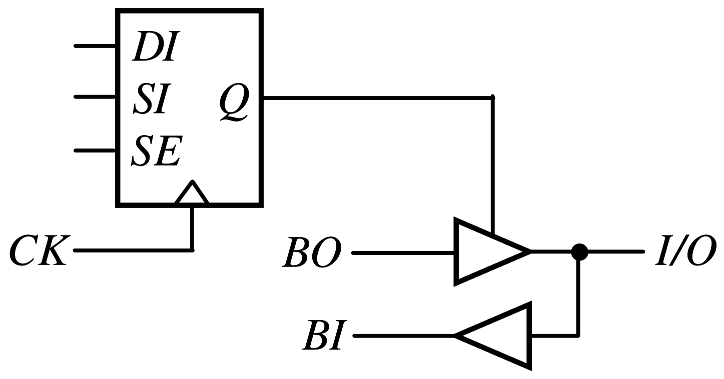


Modified circuit fixing bus contention

EN_1 is forced to 1 to enable the D_1 bus driver, while EN_2 and EN_3 are set to 0 to disable both D_2 and D_3 bus drivers, when $SE = 1$.

A bus without a pull-up, pull-down, or bus keeper may result in fault coverage loss, the bus keeper is added.

Bi-Directional I/O Ports

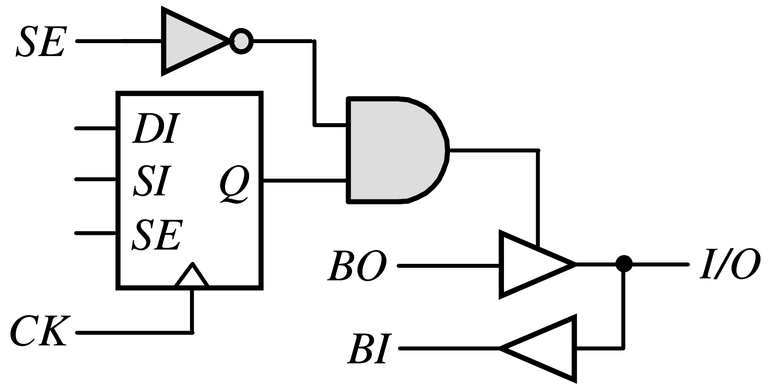


(a) Original circuit

Conflicts may occur at a bidirectional I/O port during the shift operation.

Since the output value of the scan cell can vary during the shift operation, the output tri-state buffer may become active, resulting in a conflict if BO and the I/O port driven by the tester have opposite logic values.

Bi-Directional I/O Ports

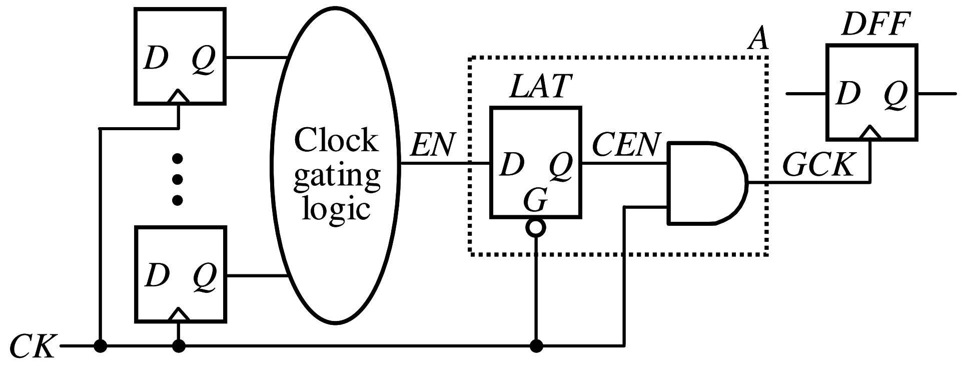


(b) Modified circuit

Fix this problem by forcing the tri-state buffer to be inactive when *SE* = 1, and the tester is used to drive the I/O port during the shift operation.

During the capture operation, the applied test vector determines whether a bi-directional I/O port is used as input or output and controls the tester appropriately.

Gated Clocks

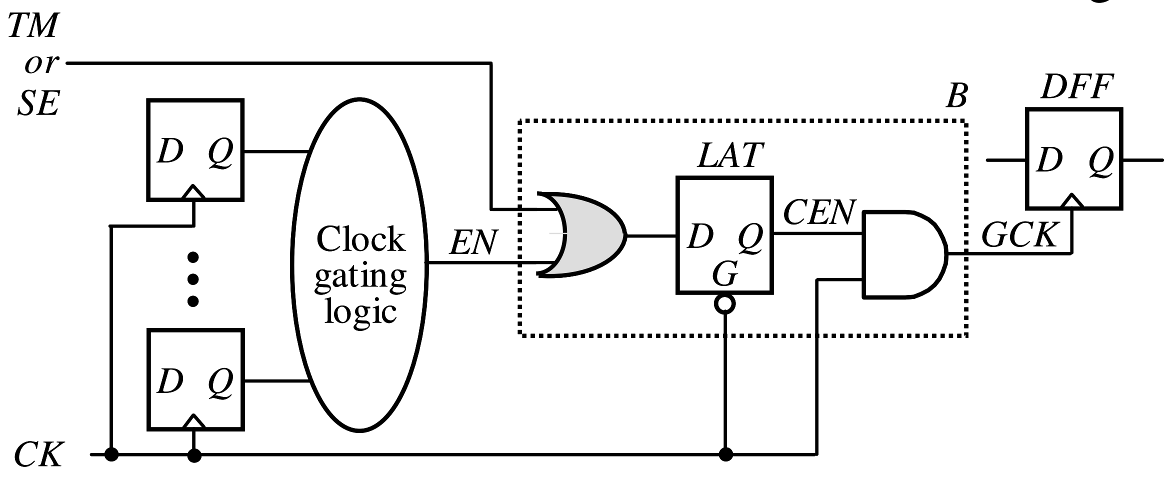


(a) Original circuit

Although clock gating is a good approach for reducing power consumption, it prevents the clock ports of some flip-flops from being directly controlled by primary inputs.

Gated Clocks

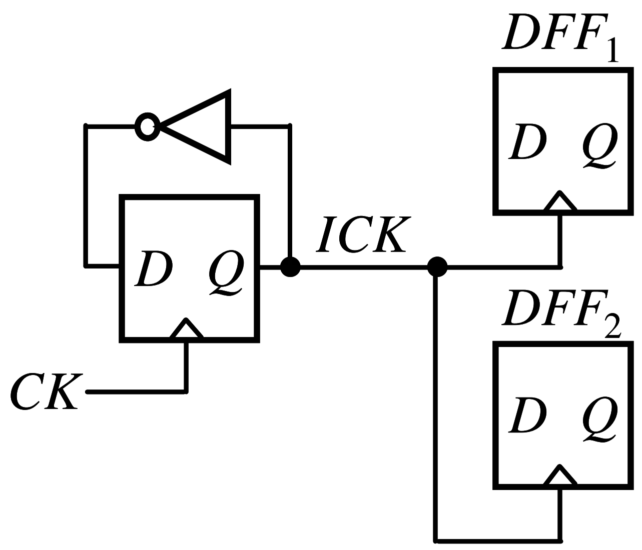
The clock gating function should be disabled at least during the shift operation.



(b) Modified Circuit

An OR gate is used to force *CEN* to 1 using either the test mode signal *TM* or the scan enable signal *SE*.

Derived clocks

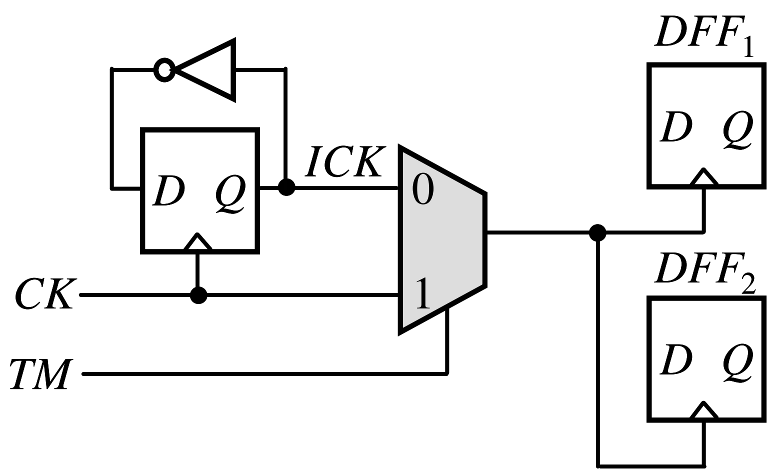


(a) Original circuit

A derived clock is a clock signal generated internally from a storage element or a clock generator.

These clock signals need to be bypassed during the entire test operation.

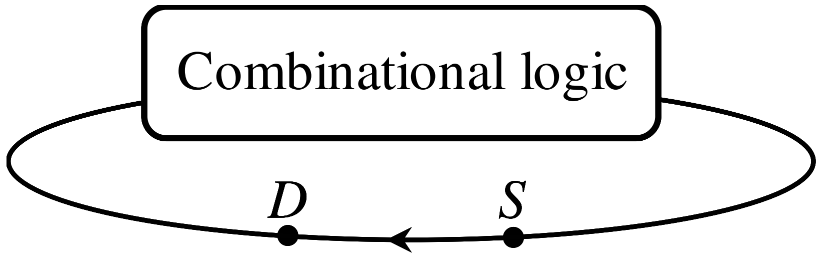
Derived clocks



(b) Modified circuit

A multiplexer selects *CK*, which is a clock directly controllable from a primary input, to drive DFF_1 and DFF_2, during the entire test operation, when *TM* = 1.

Combinational Feedback Loops



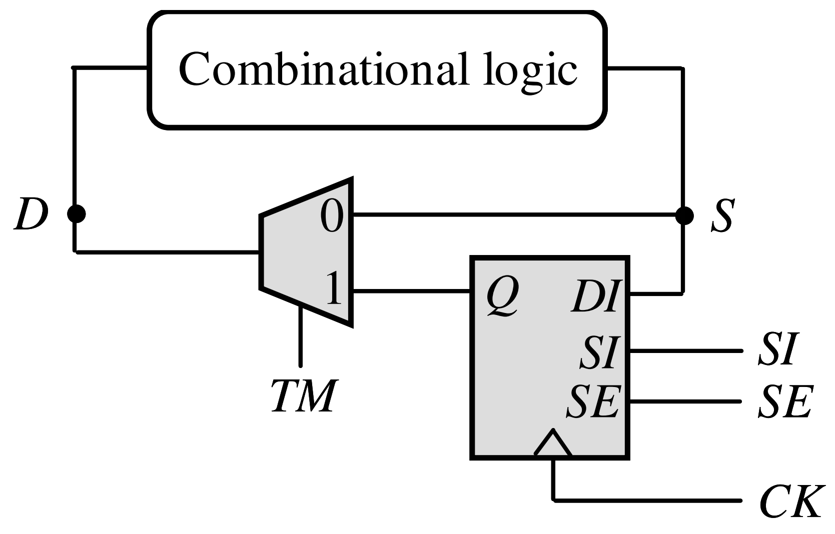
(a) Original circuit

The best way is to rewrite the RTL code.

Depending on whether the number of inversions on a combinational feedback loop is even or odd, it can introduce either sequential behavior or oscillation into a design.

Since the value stored in the loop cannot be controlled or determined during test, this can lead to an increase in test generation complexity or fault coverage loss.

Combinational Feedback Loops

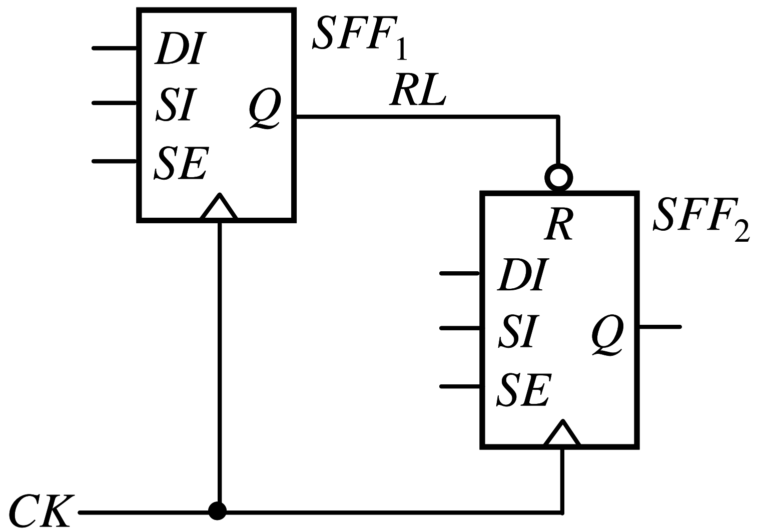


(b) Modified circuit

It can be fixed by using a test mode signal *TM*.

This signal permanently disables the loop throughout the entire shift and capture operations, by inserting a scan point to break the loop.

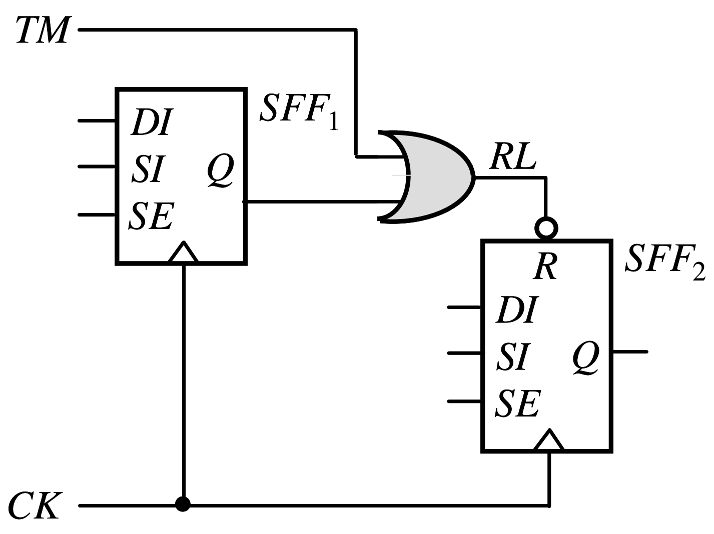
Asynchronous Set/Reset Signals



(a) Original circuit

Asynchronous set/reset signals of scan cells that are not directly controlled from primary inputs can prevent scan chains from shifting data properly.

Asynchronous Set/Reset Signals

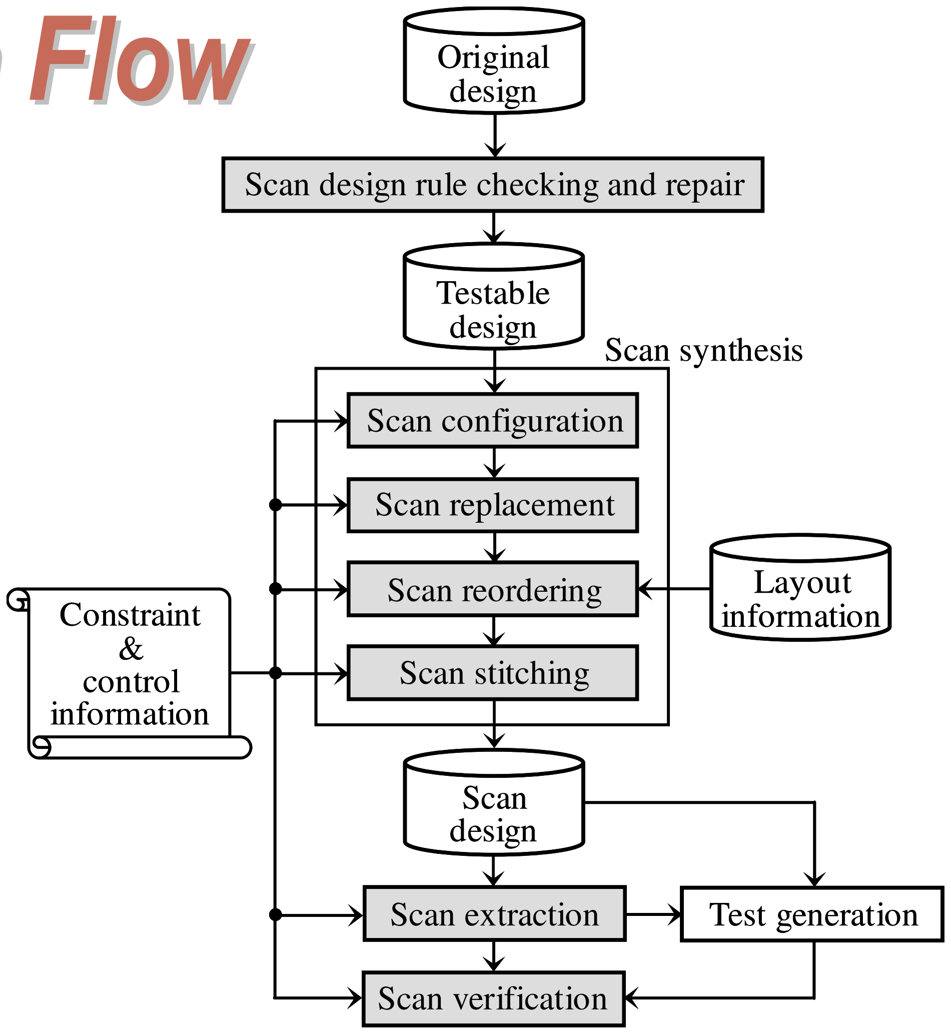


(b) Modified circuit

To avoid this problem, these asynchronous set/reset signals are forced to an inactive state during the shift operation.

Use an OR gate with an input tied to the test mode signal TM .When TM = 1, the asynchronous reset signal RL of scan cell SFF_2 is permanently disabled during the entire test operation.

Scan Design Flow



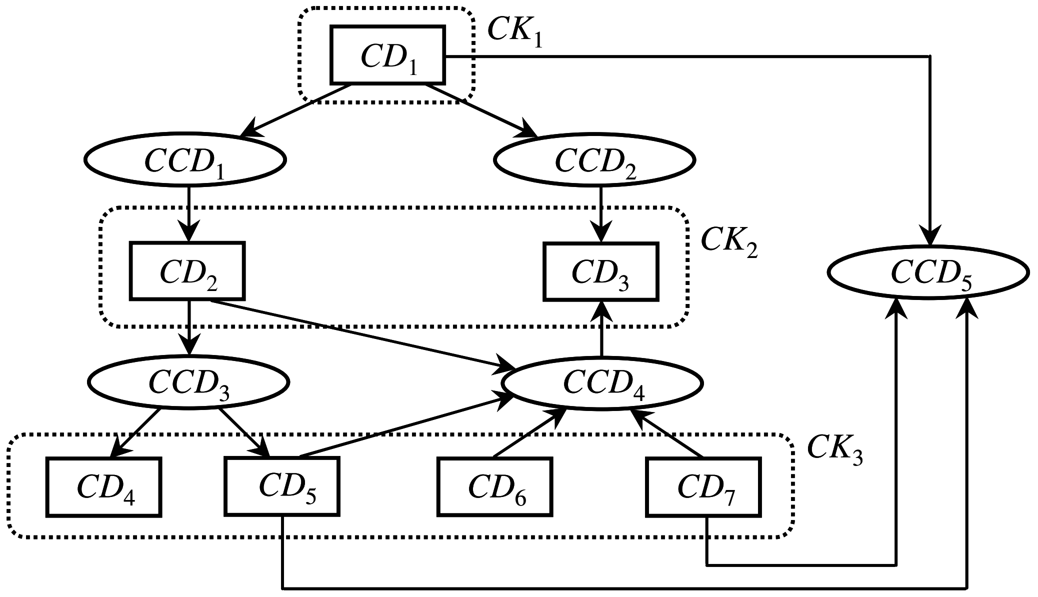
Scan Design Flow

- Scan Design Rule Checking and Repair
  - Identify and repair all scan design rule violations to convert the original design into a testable design
  - Also performed after scan synthesis to confirm that no new violations exist
- Scan Synthesis
  - Converts a testable design into a scan design without affecting the functionality of the original design
    - Scan Configuration
    - Scan Replacement
    - Scan Reordering
    - Scan Stitching

Scan Design Flow

- Scan Extraction
  - Is the process used for extracting all scan cell instances from all scan chains specified in the scan design
- Scan Verification
  - A timing file in *standard delay format* (SDF) which resembles the timing behavior of the manufactured device is used to
    - Verifying the scan shift operation
    - Verifying the scan capture operation
- Scan Design Costs
  - Area overhead cost:
  - I/O pin cost
  - Performance degradation cost
  - Design effort cost

Scan Design Rule Checking and Repair



An arrow means a data transfers from one clock domain to a different clock domain.

7 clock domains, $CD_1 \sim CD_7$

5 crossing-clock-domain data paths, $CCD_1 \sim CCD_5$

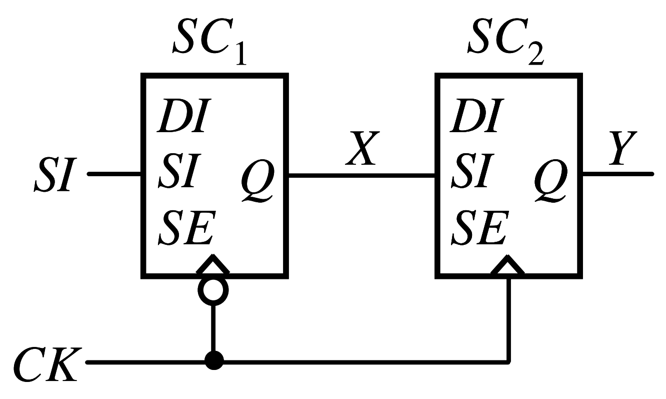
Clock grouping example

Scan Synthesis

- Includes four separate and distinct steps:
  - Scan Configuration
    - The number of scan chains used
    - The types of scan cells used to implement these scan chains
    - Which storage elements to exclude from the process
    - How the scan cells are arranged
  - Scan Replacement
    - Replaces all original storage elements in the testable design with their functionally-equivalent scan cells
  - Scan Reordering
    - The process of reordering the scan chains based on the physical scan cell locations, in order to minimize the amount of interconnect wires used to implement the scan chains
  - Scan Stitching
    - Stitch all scan cells together to form scan chains

Scan Synthesis - Scan Configuration

Mixing negative-edge and positive-edge scan cells in a scan chain

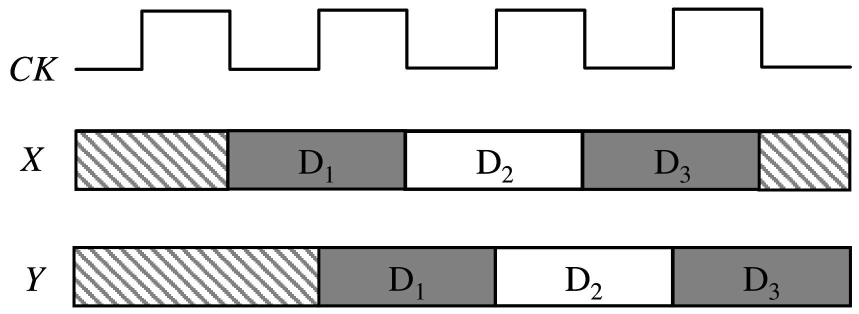


Circuit Structure

This circuit structure comprising a negative-edge scan cell followed by a positive-edge scan cell.

Scan Synthesis - Scan Configuration

Mixing negative-edge and positive-edge scan cells in a scan chain

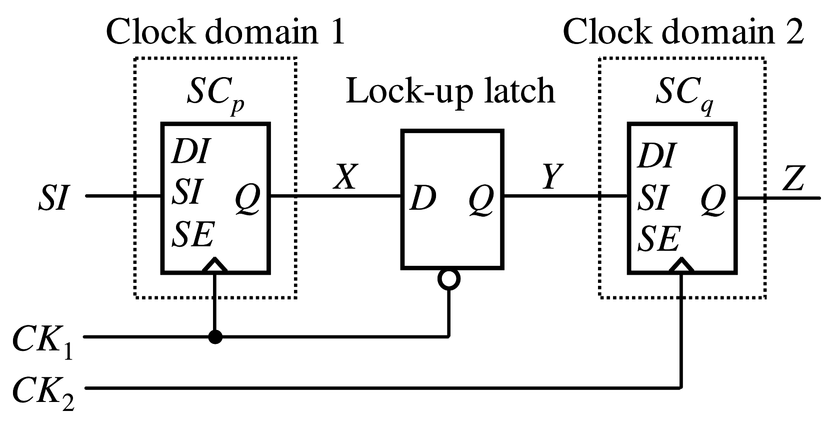


Timing Diagram

Y will first take on the state *X* at the rising *CK* edge, before *X* is loaded with the *SI* value at the falling *CK* edge.

If we accidentally place the positive-edge scan cell before the negative-edge scan cell, both scan cells will always incorrectly contain the same value at the end of each shift clock cycle.

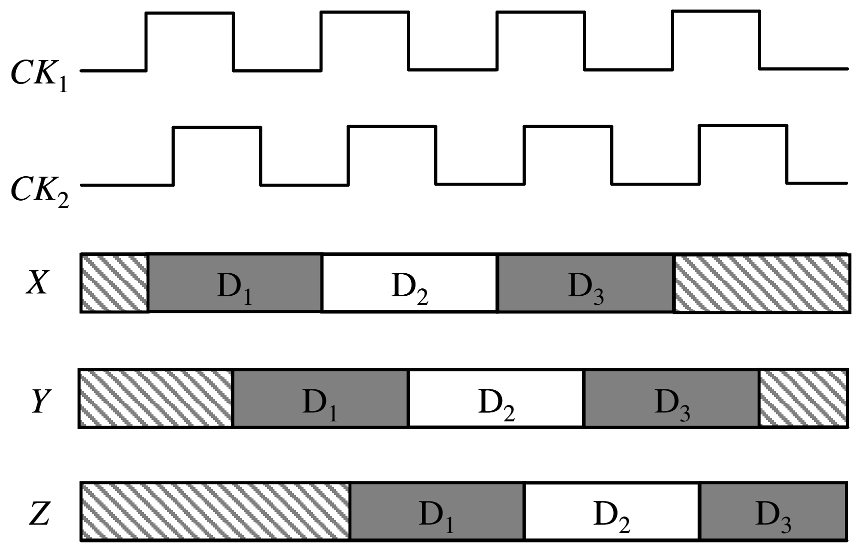
Scan Synthesis - Scan Configuration



Circuit Structure

A lock-up latch is inserted between adjacent cross-clock-domain scan cells, in order to guarantee that any clock skew between the clocks can be tolerated.

Scan Synthesis - Scan Configuration



Timing diagram

During each shift clock cycle, *X* will first take on the *SI* value at the rising CK_1 edge. Then, *Z* will take on the *Y* value at the rising CK_2 edge.

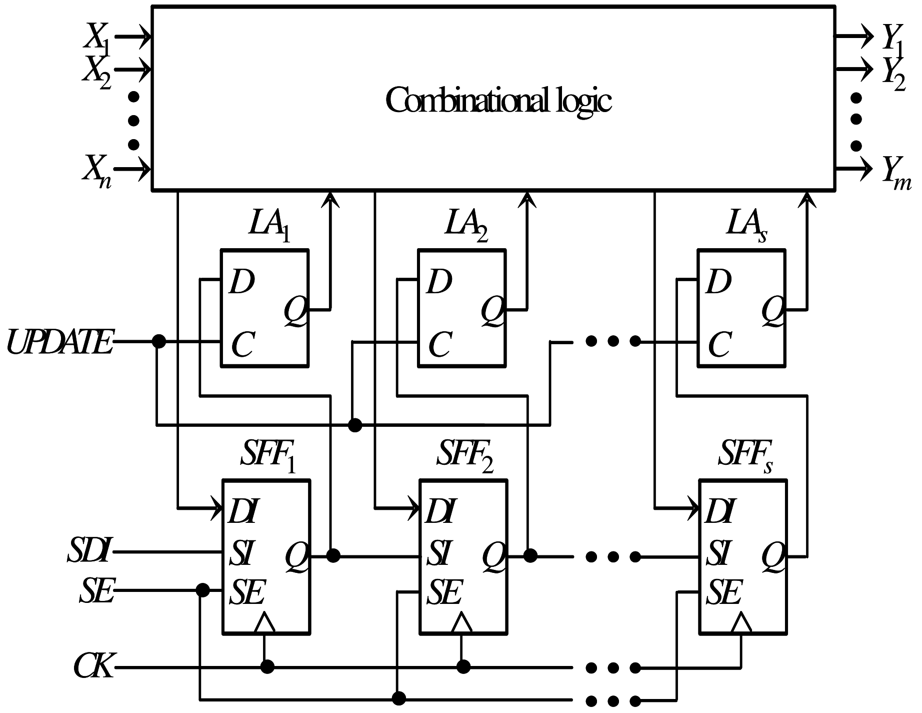
Special-Purpose Scan Designs

- Enhanced scan
- Snapshot scan
- Error-resilient scan

Enhanced Scan

- Why enhanced scan is introduced?
  - Testing for a delay fault requires a pair of test vector at speed
  - Be able to capture the response to the transiton at operating frequency
- What is new?
  - Allow the typical scan cell to store two bits of data
  - Achieved through the addition of a D latch

Enhanced-scan Architecture and Operation



Enhanced-scan architecture

The first test vector V_1 is first shifted into the scan cells (SFF_1 ~ SFF_s) and then stored into the additional latches (LA_1 ~ LA_s) when the *UPDATE* signal is set to 1.

The second test vector V_2 is shifted into the scan cells while the *UPDATE* signal is set to 0.

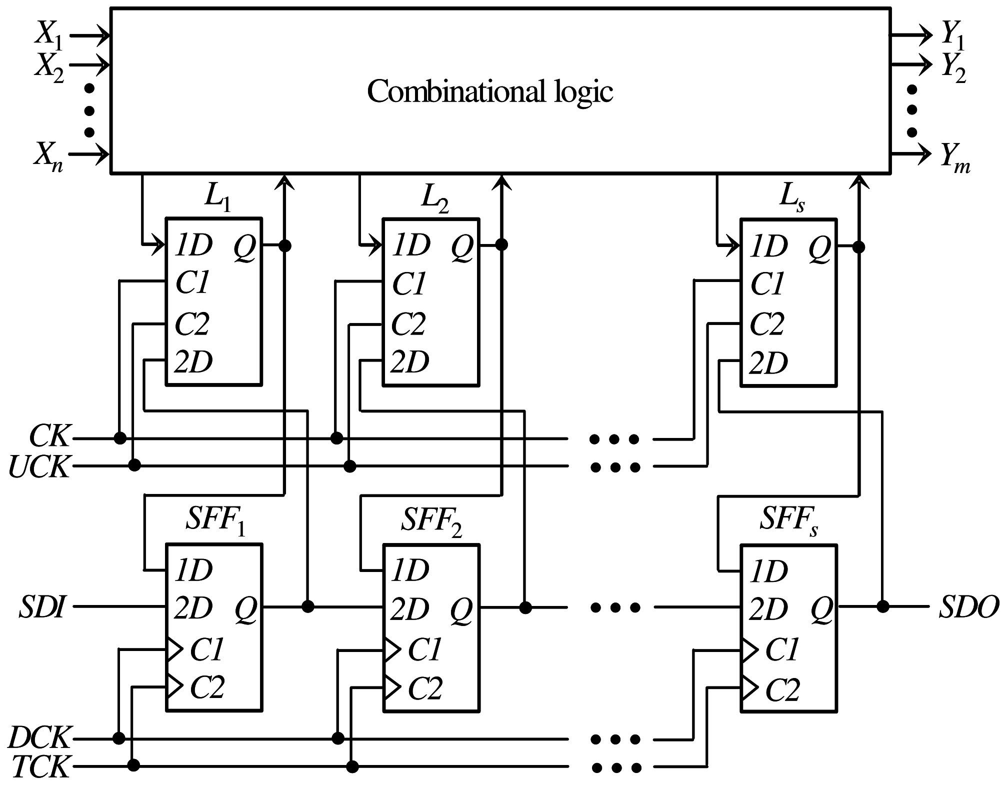
Enhanced Scan - Advantages & Disadvantages

- Advantages
  - High delay fault coverge achieved by applying any arbitrary pair of test vectors
- Disadvantages
  - Requiring an additional scan-hold D latch
  - Difficulty of maintaining the timing relationship between *UPDATE* and *CK*
  - Over-test problem caused by false paths

Snapshot scan

- Why snapshot scan is introduced?
  - Capture a snapshot of the internal states
  - Without disruption of the functional operation
- What is new?
  - Add a scan cell (2-port D latches) to each storage element of interest
  - Implement scan-set architecture

Snapshot scan



Scan-set architecture

(1) Test data can be shifted into and out of the scan cells (SFF_1 ~ SFF_s) from the *SDI* and *SDO* pins using *TCK*.

(2) The test data can be transferred to the system latches (L_1 ~L_s) in parallel through their *2D* inputs using *UCK*.

(3) The system latch contents can be loaded into the scan flip-flops through their *1D* inputs using *DCK*.

(4) The circuit can be operated in normal mode using *CK* to capture the values from the combinational logic into the system latches (L_1 ~ L_s).

Snapshot scan - Advantages & Disadvantage

- Advantages
  - Significantly improve the circuit's diagnostic resolution and silicon debug capability
  - Allow on-chip, on-board and in-system debug and diagnosis
- Disadvantage
  - Increased area overhead

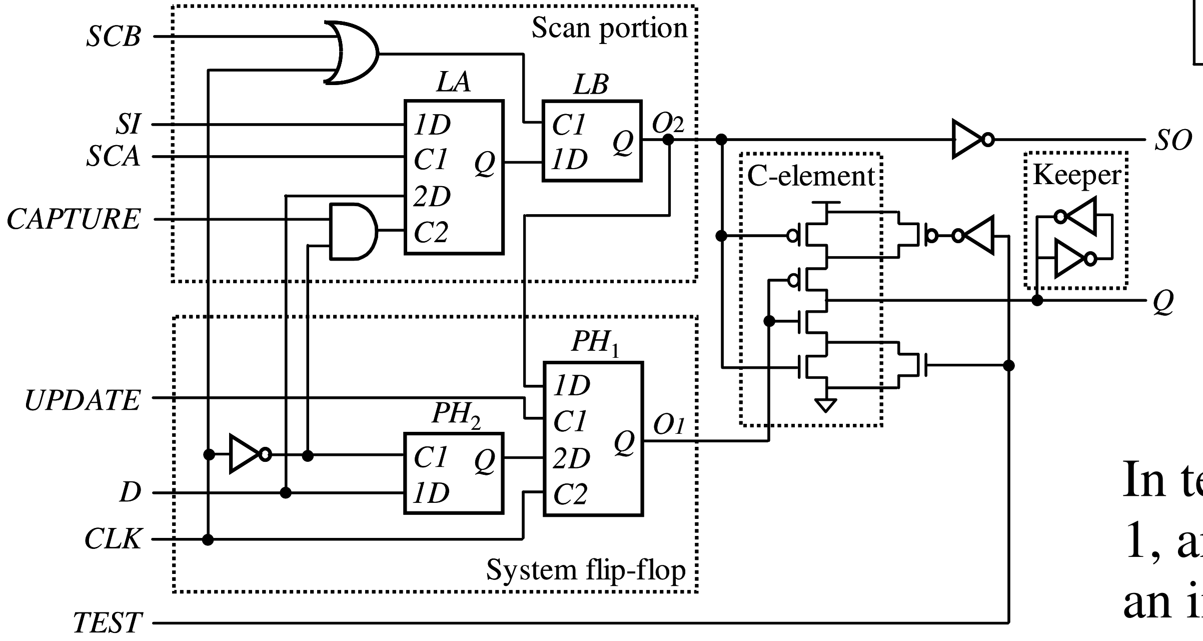
Error-Resilient scan

- Why is error-resilient scan introduced?
  - Soft errors are transient single-event upsets with various causes
  - Soft errors increase as shrinking IC geometry and increasing frequency
  - Reliability concerns are created for protecting a device from soft errors
- What can error-resilient scan do?
  - Observe soft errors occurring in memories and storage elements
  - Observe a transient fault in a combinational gate captured by a memory or storage element

Error-Resilient Scan

C-element truth table

O_1	O_2	Q
0	0	1
1	1	0
0	1	Previous value retained
1	0	Previous value retained



Error-resilient scan cell

In test mode, *TEST* is set to 1, and the C-element acts as an inverter.
In system mode, *TEST* is set to 0, and the C-element acts as a hold-state comparator.

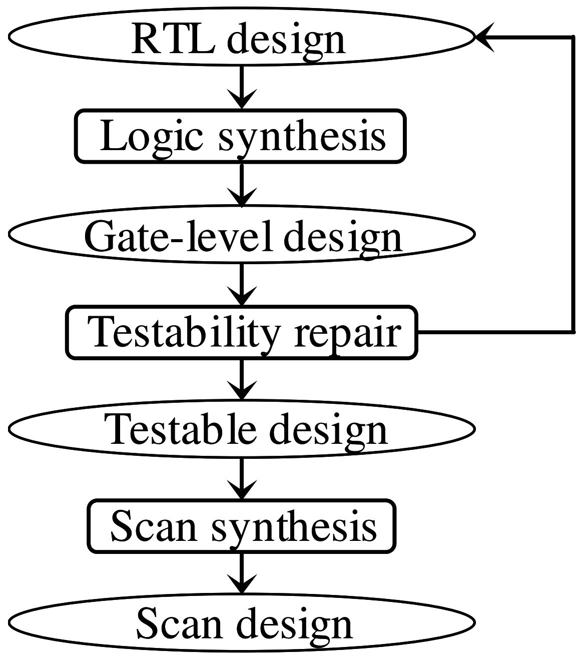
Error-Resilient scan - Advantages & Disadvantages

- Advantages
  - Provide online detection and correction of soft errors
  - Embed with scan testing capability
- Disadvantages
  - Require many test signals and clocks
  - Area overhead

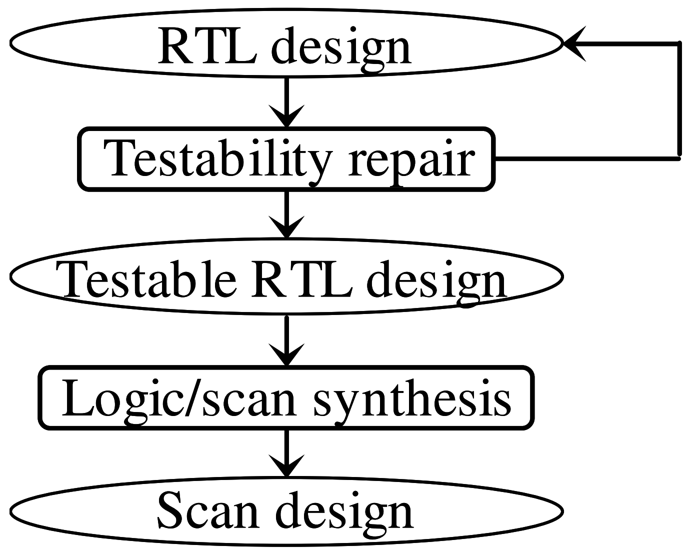
RTL Design for Testability

- Why are RTL designs needed?
  - Growth of device number
  - Tight timing
  - Potential yield loss
  - Low-power issues
  - Increased core reusability
  - Time-to market pressure

Comparison of design flows at RTL and Gate-level



Gate-level testability repair design flow



RTL testability repair design flow

RTL Scan Design Rule Checking

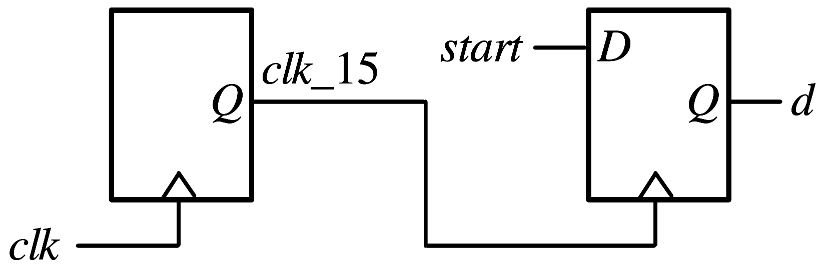
- Fast synthesis
  - Mapped onto combinational primitives and high-level models
- Identify testability problems
  - Static solutions (without simulation)
  - Dynamic solutions (with simulation)

RTL Scan Design Repair – An Example

- original

```
always @(posedge clk)
   if (q == 4'b1111)
      clk_15 <= 1;
   else
      begin
         clk_15 <= 0;
         q <= q + 1;
      end
always @(posedge clk_15)
   d < = start;
```

(a) Generated clock (RTL code)



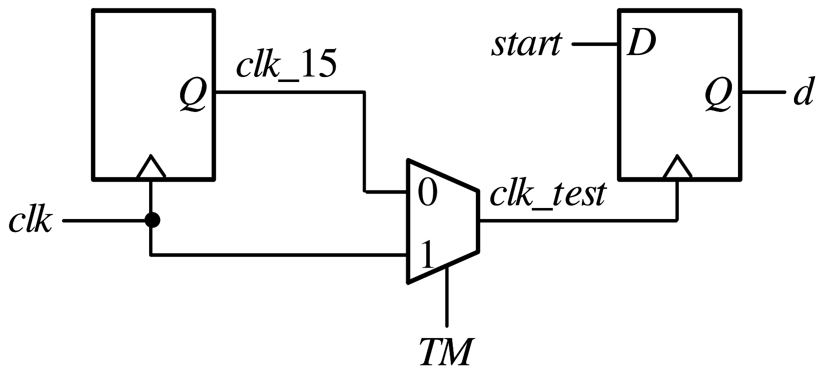
(b) Generated clock (Schematic)

RTL Scan Design Repair – An Example

- Atuomatic repair at the RTL using *TM*

```
always @(posedge clk)
    if (q == 4'b1111)
        clk_15 <= 1;
    else
        begin
            clk_15 <= 0;
            q <= q + 1;
        end
assign clk_test = (TM)? clk : clk_15;
always @(posedge clk_test)
    d < = start;
```

(c) Generated clock (RTL code)



(d) Generated clock repair (Schematic)

RTL Scan Synthesis

- RTL scan synthesis
  - The scan equivalent of each storage element refers to an RTL structure
  - The scan chains are inserted into the RTL design
- Pseudo RTL scan synthesis
  - Specify pseudo primary inputs and pseudo primary outputs
  - Can cope with many other DFT structures
  - Perform one-pass or single-pass synthesis

RTL Scan Extraction and Verification

- Scan extraction
  - Rely on performing fast synthesis on the RTL scan design
  - Generate a software model for tracing the scan connection
- Scan verification
  - Rely on generating a flush testbench to simulate flush tests
  - The flush testbench can be used for both RTL and gate-level designs
  - Apply broadside-load test for verifying the scan capture operation at RTL

Concluding Remarks

- DFT has become vital for ensuring product quality
- Scan design is the most widely used DFT technique
- New design and test challenges
  - Further reduce test power, test data volume and test application time
  - Cope with physical failures of the nanometer design era