Chapter 8 Design for Testability

測試導向設計技術

Outline

- Introduction
- Ad-Hoc Approaches
- Full Scan
- Partial Scan

Design For Testability

- Definition
  - Design For Testability (DFT) refers to those design techniques that make test generation and testing cost-effective
- DFT deals with ways of improving
  - Controllability
  - Observability
- DFT Methods
  - Ad-hoc methods
  - Scan, full and partial
  - Built-In Self-Test (BIST)
  - Boundary scan, core test architecture, etc.
- Cost of DFT
  - Pin count, area, performance, design-time, test-time

Testability

- A concept that deals with the costs associated with testing.
- When the testability of a circuit is increased, some test costs are being reduced
    - Test application time
    - Test generation time
    - Fault simulation time
    - Fault localization time
    - Test equipment cost

What limits application of DFT ?

- Short-sighted view of management
- Time-to-market pressure
- Area/functionality/performance myths
- Lack of knowledge by design engineers
- Testing is someone else’s problem
- Lack of tools to support DFT until recently

DFT is Important for Successful Production

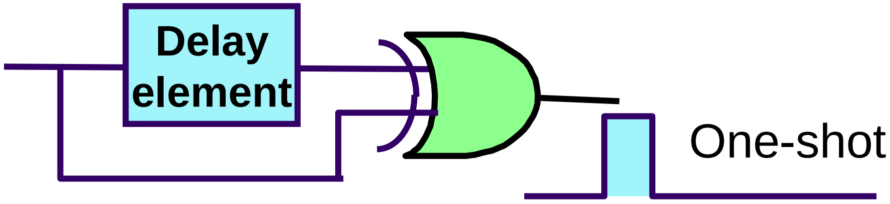
- Certain DFT techniques are widely and successfully used
  - Scan
  - Boundary Scan
  - Test compression
  - BIST

Ad-Hoc Design For Testability

- Design Guidelines
  - Avoid redundancy
  - Avoid asynchronous logic
  - Avoid clock gating (e.g., power control, ripple counter)
  - Avoid large fan-in
- Disadvantages of ad-hoc methods
  - Circuit too large for manual inspection and test generation.
  - Not too many testability experts to consult.
  - High fault coverage not guaranteed

Ad-Hoc DFT Techniques (I)

- Test Points
  - Employ test points to enhance controllability and observability
- Initialization
  - Design circuit to be easily initialized
- Oscillators and clocks
  - Disable internal oscillators and clocks during test
- Monostable multivibrators
  - Disable internal one-shots during test



One-shot signal is hard to predict

Ad-Hoc DFT Techniques (II)

- Partition counters / shift-registers
  - Partition large counters and SR into smaller units
- Partition large circuits
  - Partition large circuits into small sub-circuits to reduce test generation cost
- Logical redundancy
  - Avoid the use of redundant logic
- Break global feedback paths
  - Provide logic to break global feedback paths

Test Point Insertion

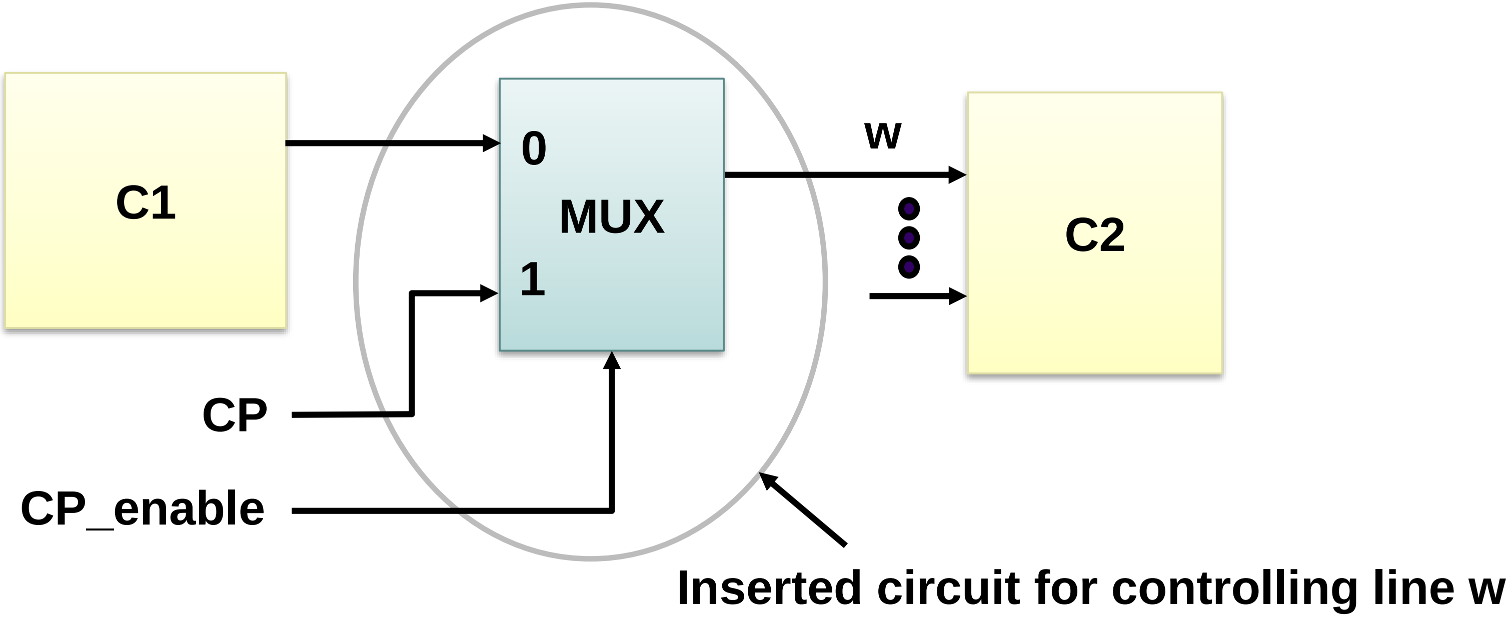
- Employ test points to enhance Controllability and Observability
- Control Points (CP)
  - **Extra PIs used to enhance controllability**
- Observability Points (OP)
  - **Extra POs used to enhance observability**

0/1 Injection Circuitry

- Normal operation

  When CP_enable = 0

- Inject 0
  - Set CP_enable = 1 and CP = 0
- Inject 1
  - Set CP_enable = 1 and CP = 1

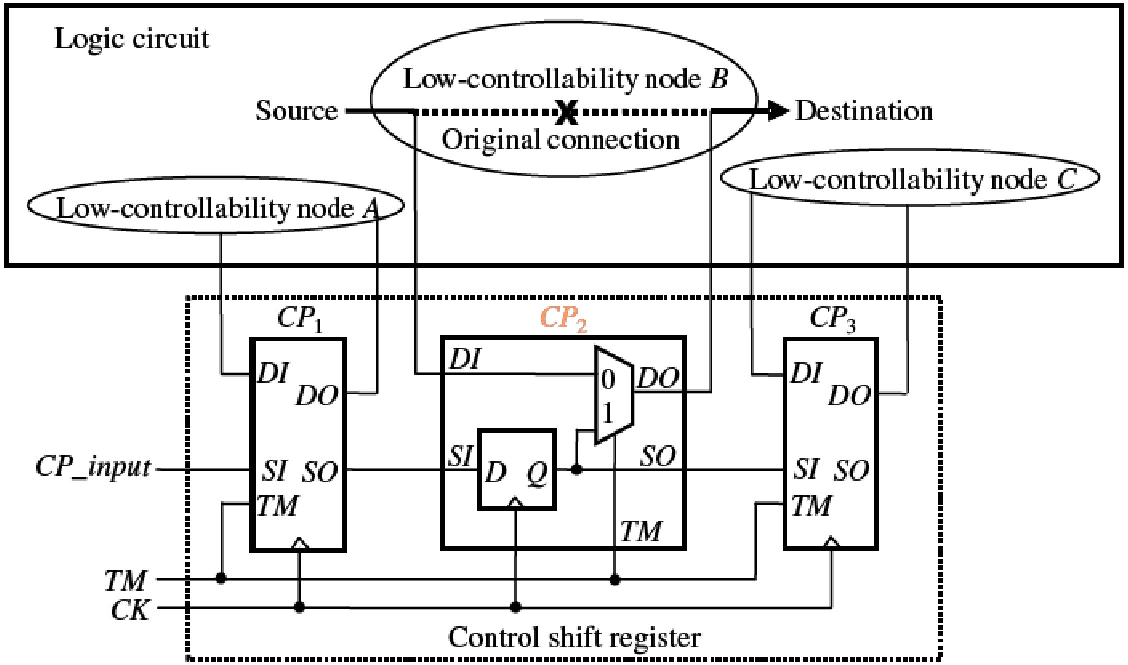


Problems of CP & OP

- Large number of I/O pins
  - **Add MUX’s to reduce the number of I/O pins**
  - **Serially shift CP values by shift-registers**
- Long test time for some CP/OP architecture
- Increase performance and area overheads

Shift Registers for Control Point Insertion

- To avoid large number of PIs.
- During normal operation, TM = 0, DI→DO.
- During test, TM = 1, Q→DO.



Control Point Selection

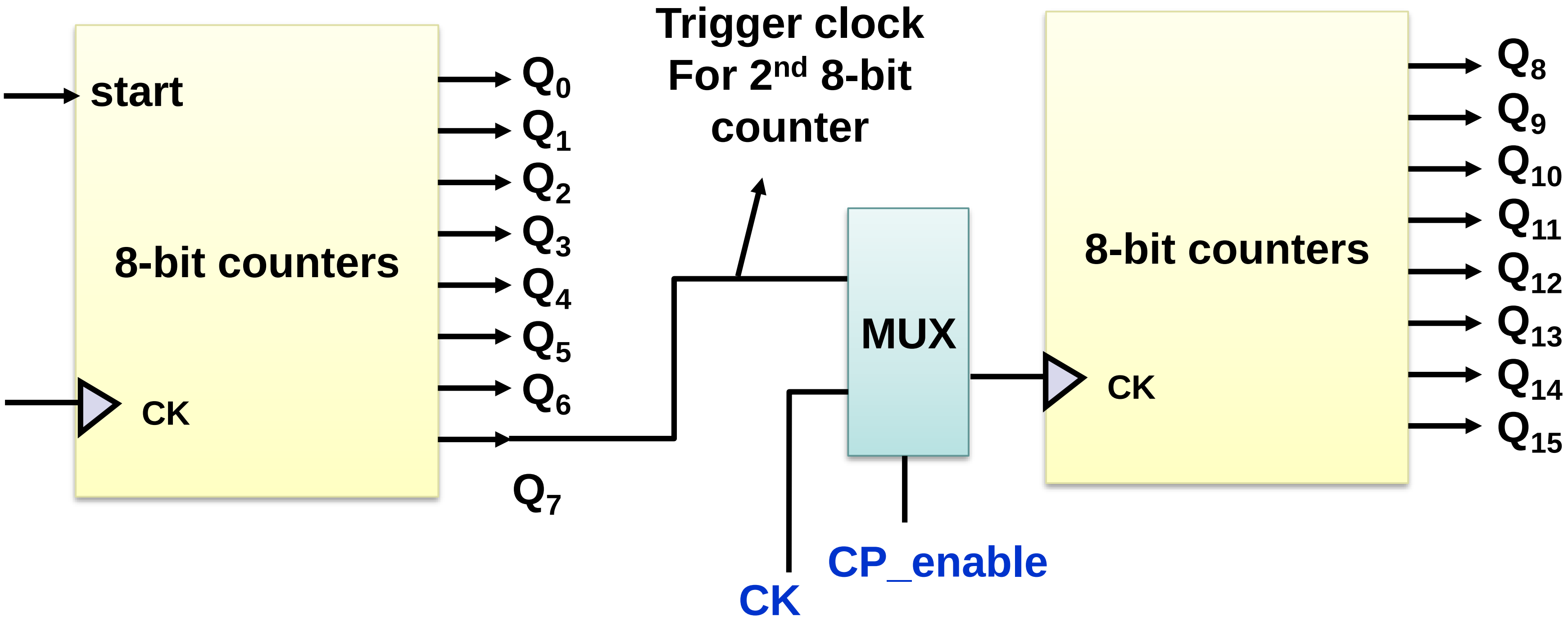
- Impact
  - The controllability of the fanout-cone of the added point is improved
- Possible candidates
  - Control, address, and data buses
  - Enable / Hold inputs
  - Enable and read/write inputs to memory
  - Clock and preset/clear signals of flip-flops
  - Data select inputs to multiplexers and demultiplexers

Observation Point Selection

- Impact
  - The observability of the transitive fanins of the added point is improved
- Common choice
  - Stem lines having high fanout
  - Global feedback paths
  - Redundant signal lines
  - Output of logic devices having many inputs
    - MUX, XOR trees
  - Output from state devices
  - Address, control and data buses

Example: Partitioning Counter

- Consider a 16-bit ripple-counter
  - Could take up to $2^{16} = 65536$ cycles to test
  - After being partitioned into two 8-bit counters below, it can be tested with just $2^{8} = 256$ cycles



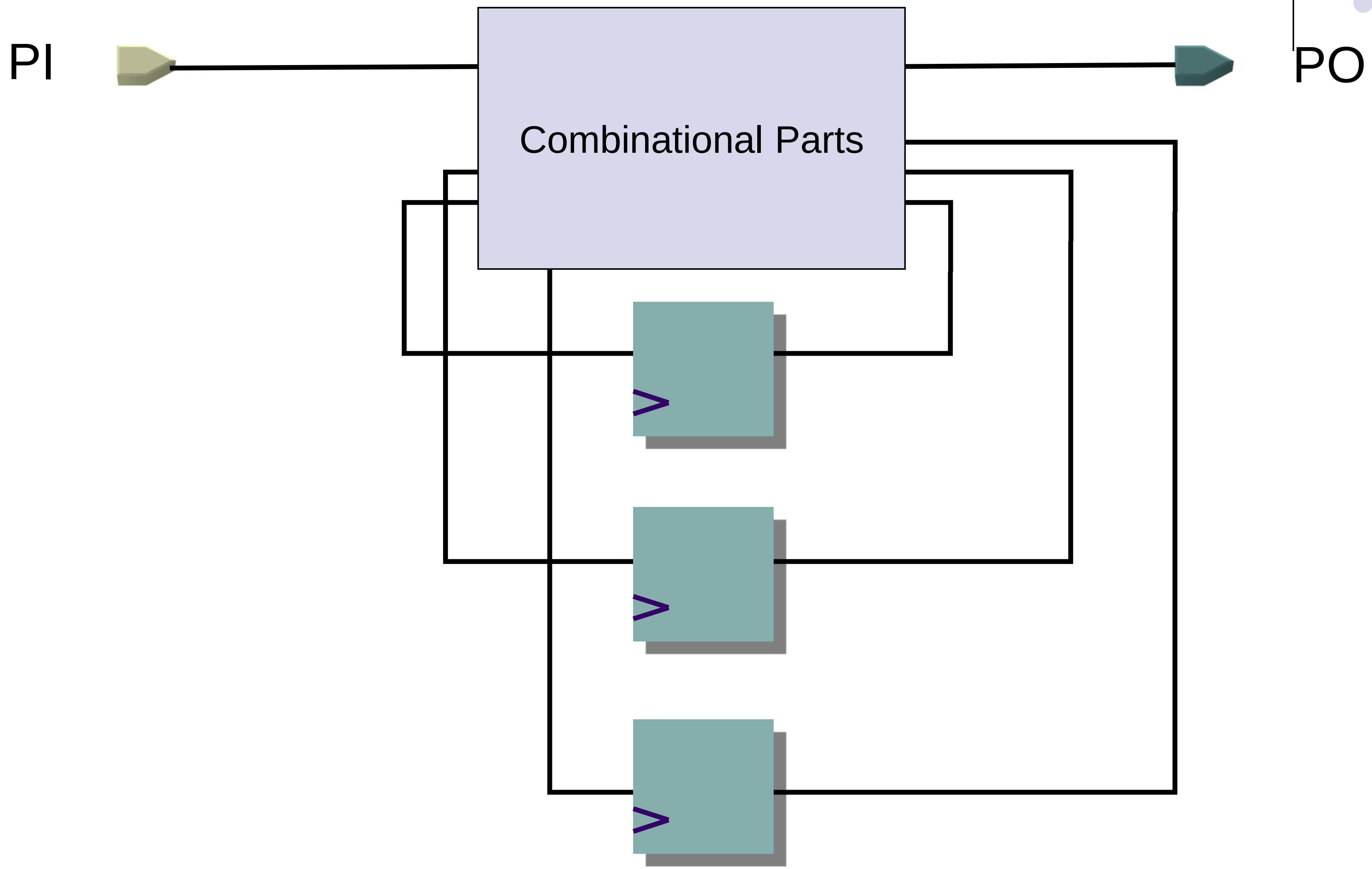
Outline

- Introduction
- Ad-Hoc Approaches
- Full Scan
- Partial Scan

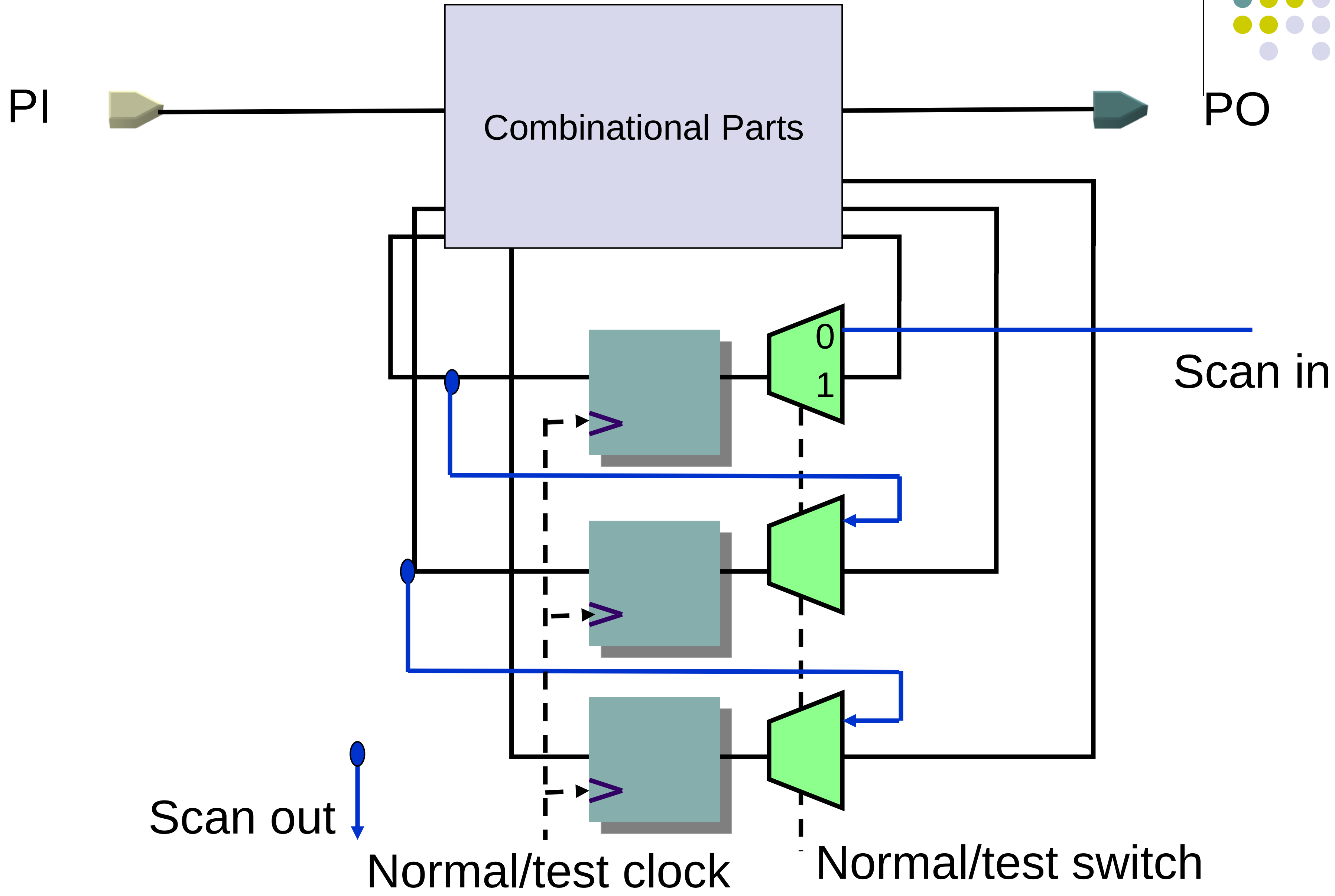
What Is Scan ?

- Objective
  - **To provide controllability and observability at internal state variables for testing**
- Method
  - **Add test mode control signal(s) to circuit**
  - **Connect flip-flops to form shift registers in test mode**
  - **Make inputs/outputs of the flip-flops in the shift register controllable and observable**
- Types
  - **Internal scan**
    - **Full scan, Partial scan, Random access, etc.**
  - **Boundary scan**

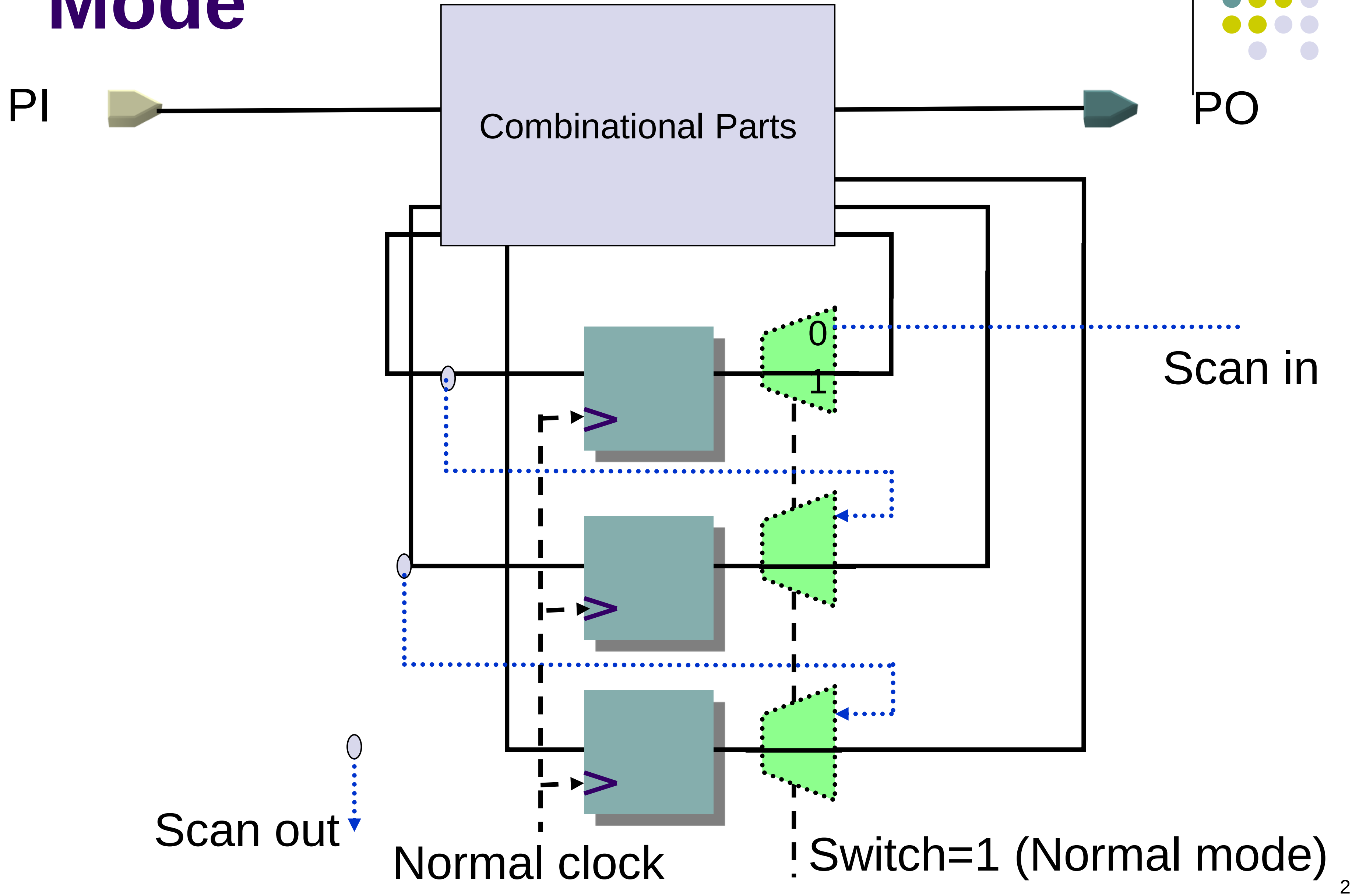
Revisit Sequential Circuit Model



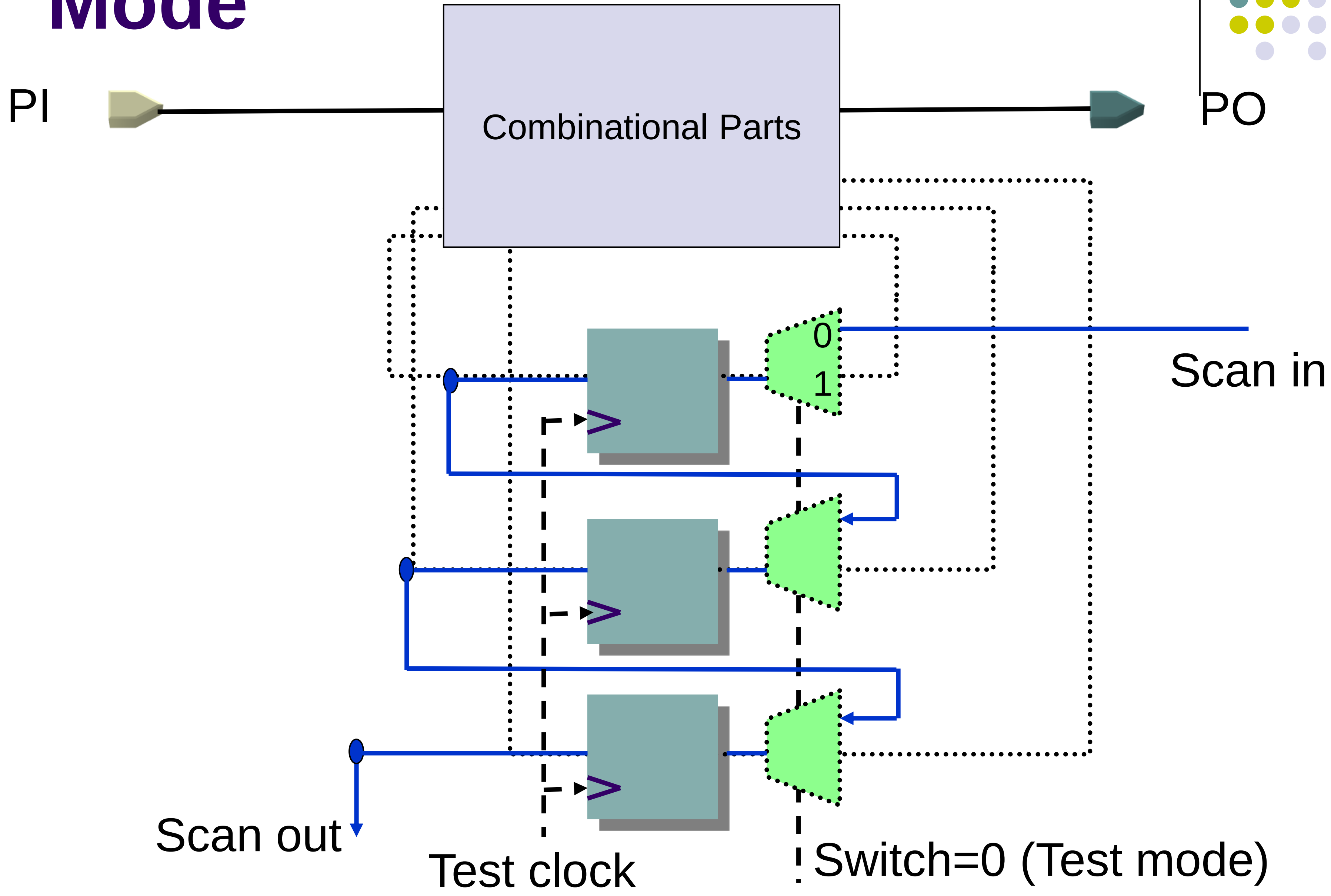
Scan Architecture



Scan Architecture In Normal Mode



Scan Architecture In Scan Mode



Applying Tests for Scan Circuits

- Phase I (test the scan chain):
  - Shift test
  - Targets the scan flip-flops.
- Phase II (applying test patterns for combination circuits):
  - Target the single stuck-at faults in the combinational circuit.
  - Test vectors are generated by a combinational ATPG.

Phase I: Shift test

- A toggle sequence 00110011… of length n_{sff}+4 is scanned in. (n_{sff} is the maximum number of FFs in a scan chain.)
- Each SFF experiences all four transitions: 0→1, 0→0, 1→1, 1→0.
- The shift test covers most single stuck-at faults in the FFs.
- The shift test also verifies the correctness of the shift operation.

Phase II: Combinational Test

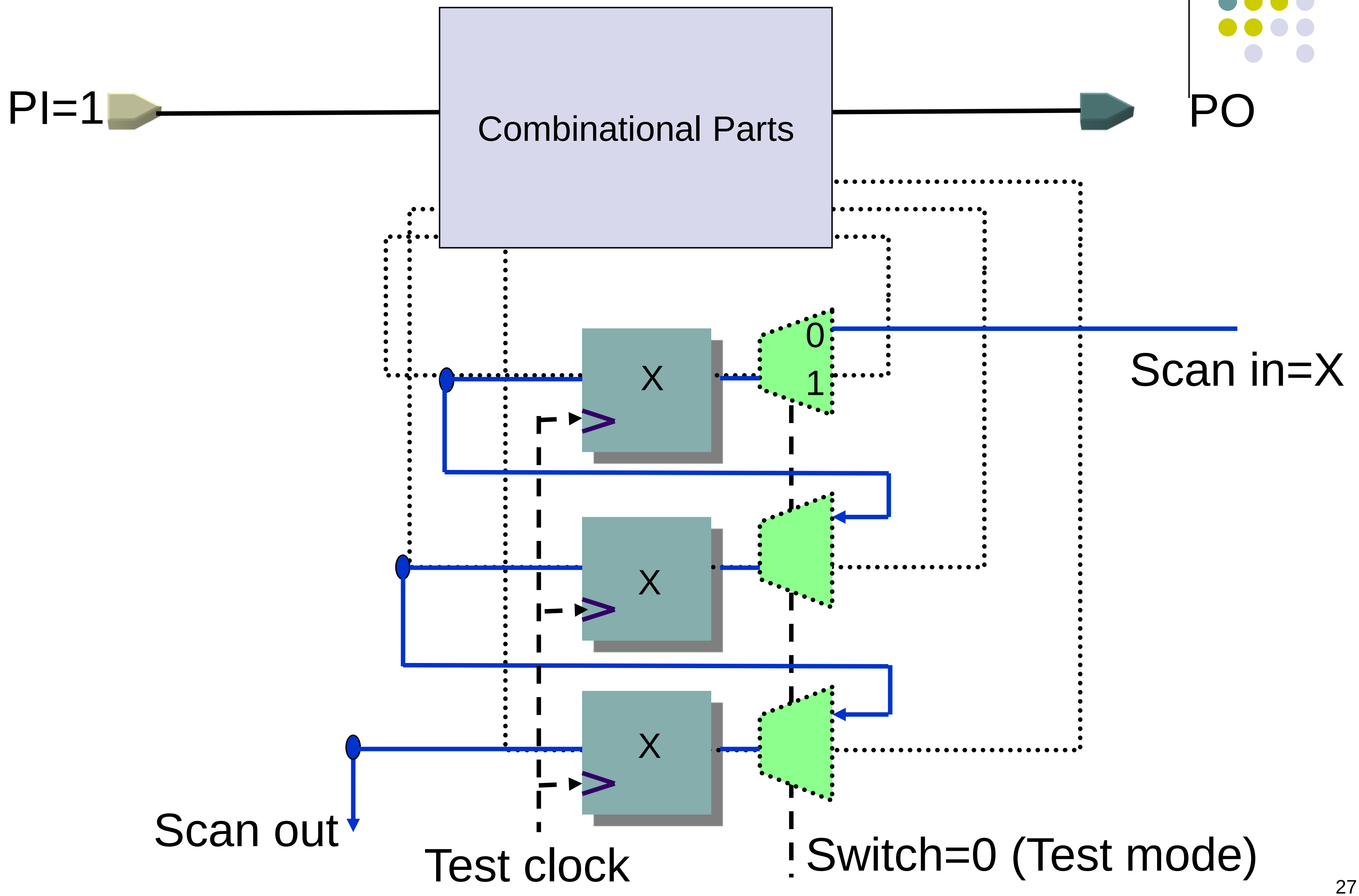
- For each combinational test vector
  1. Assert PI signals
  2. Switch to test mode (scan)
  3. Scan in
     1. Assert scan test patterns
     2. Apply test clock
     3. Repeat until all FFs are set
  4. Switch to normal mode
  5. Apply functional clock
  6. Probe PO signals
  7. Switch to test mode
  8. Scan out

Scan Test Example

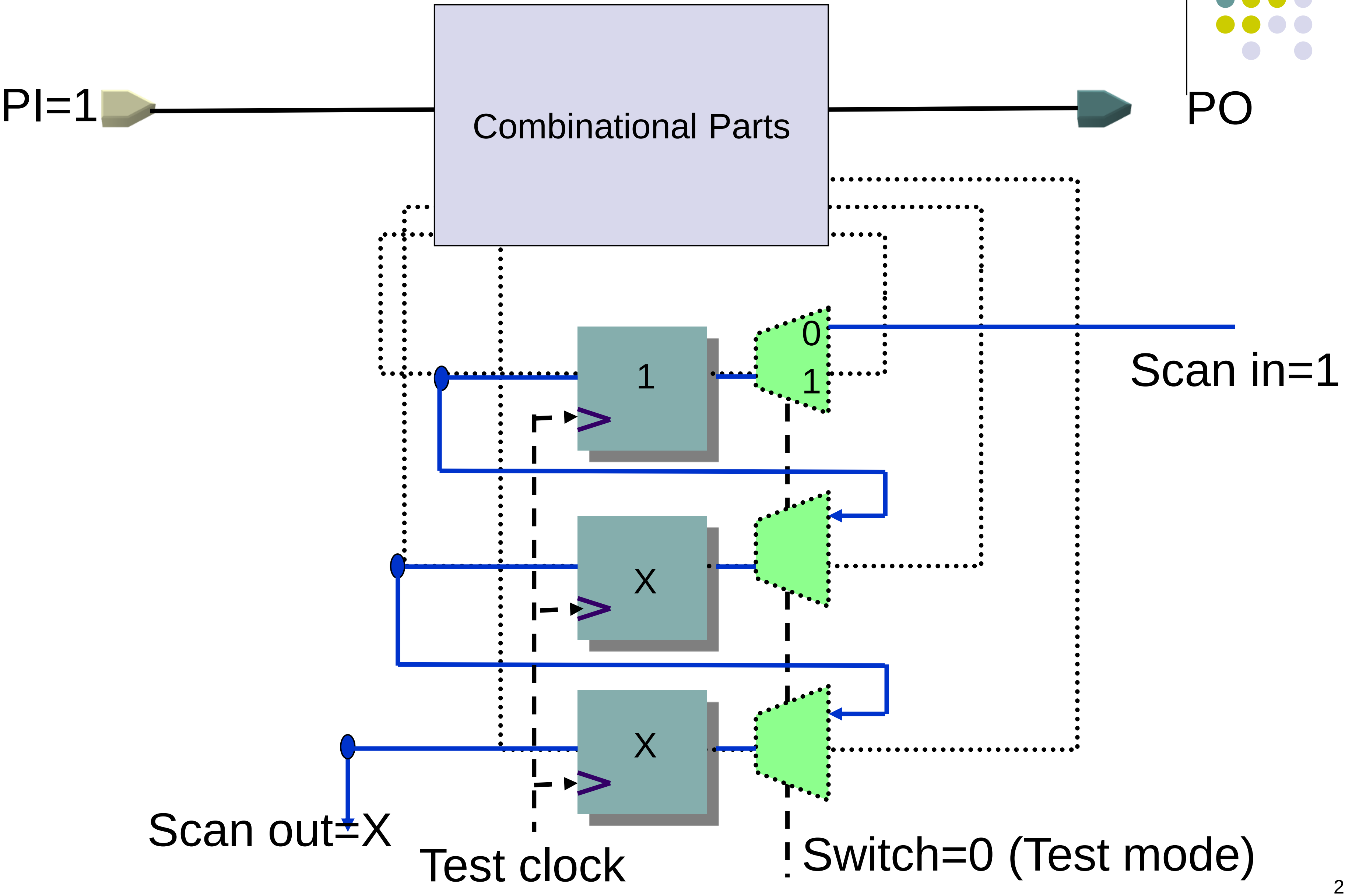
- Assume we have two test vector to be applied in the following formats

PI	PPI	PO	PPO
1	110	0	011
0	101	1	111

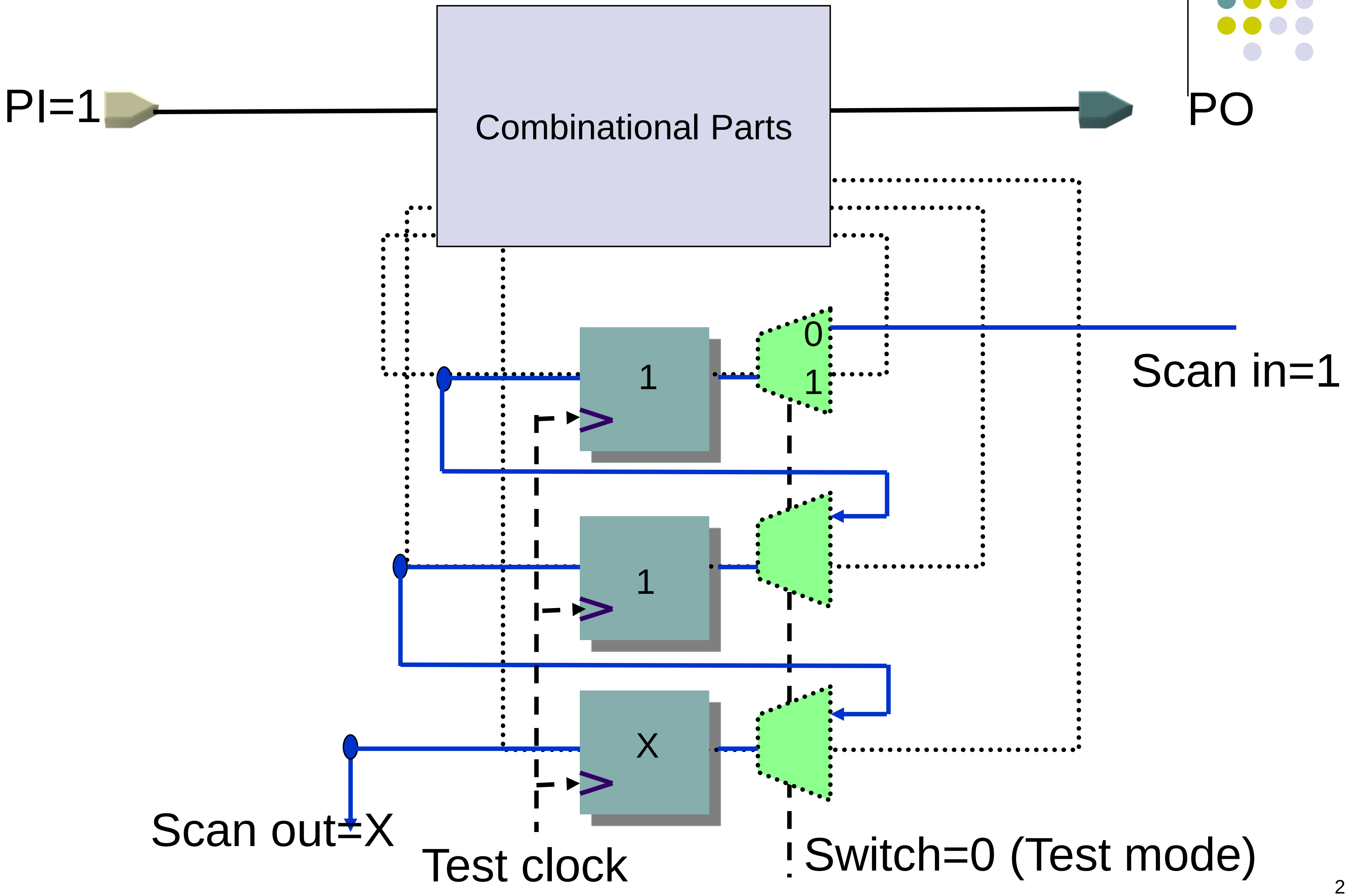
Scan Example (Assert PI)



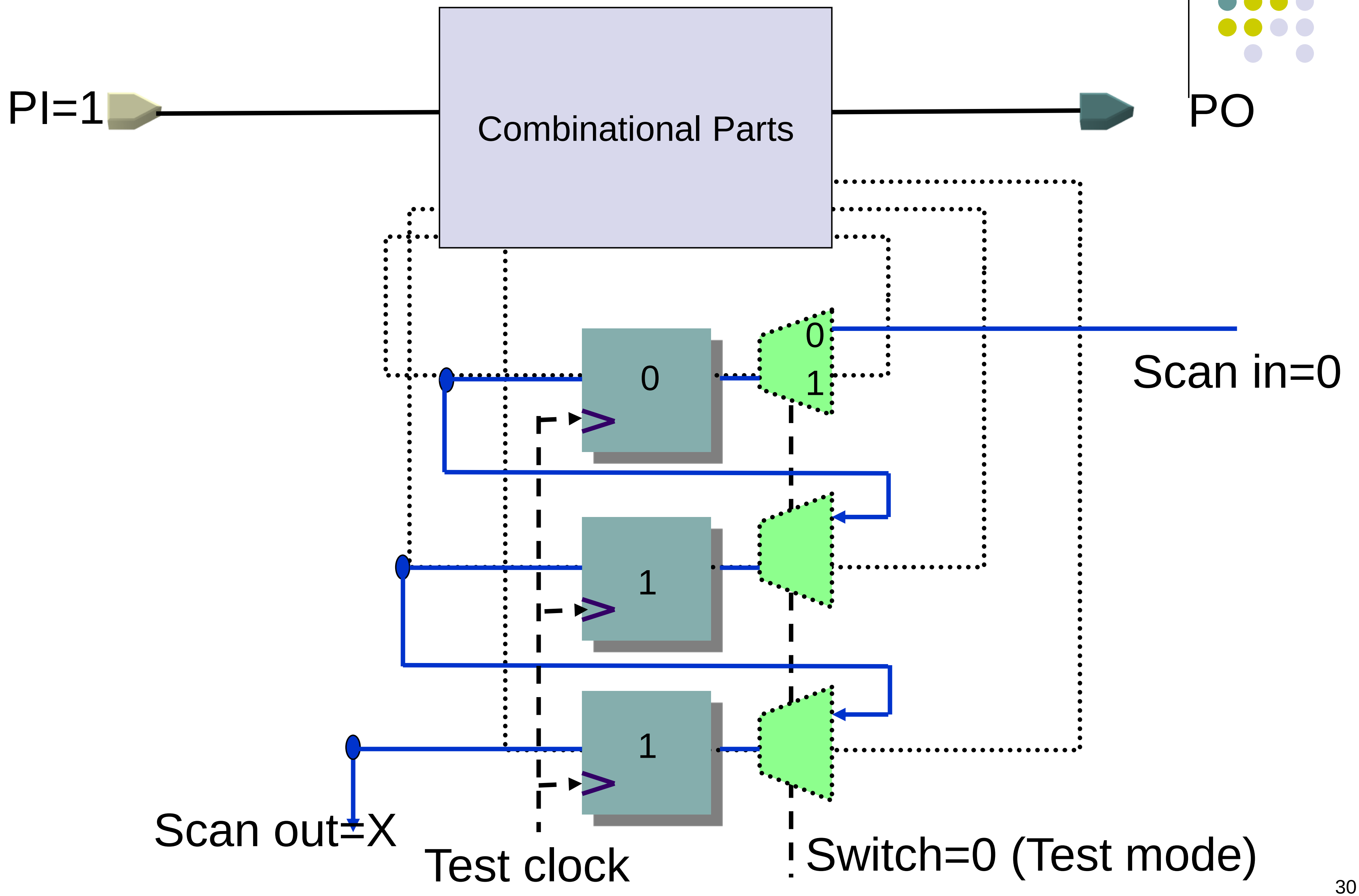
Scan Example (Scan In)



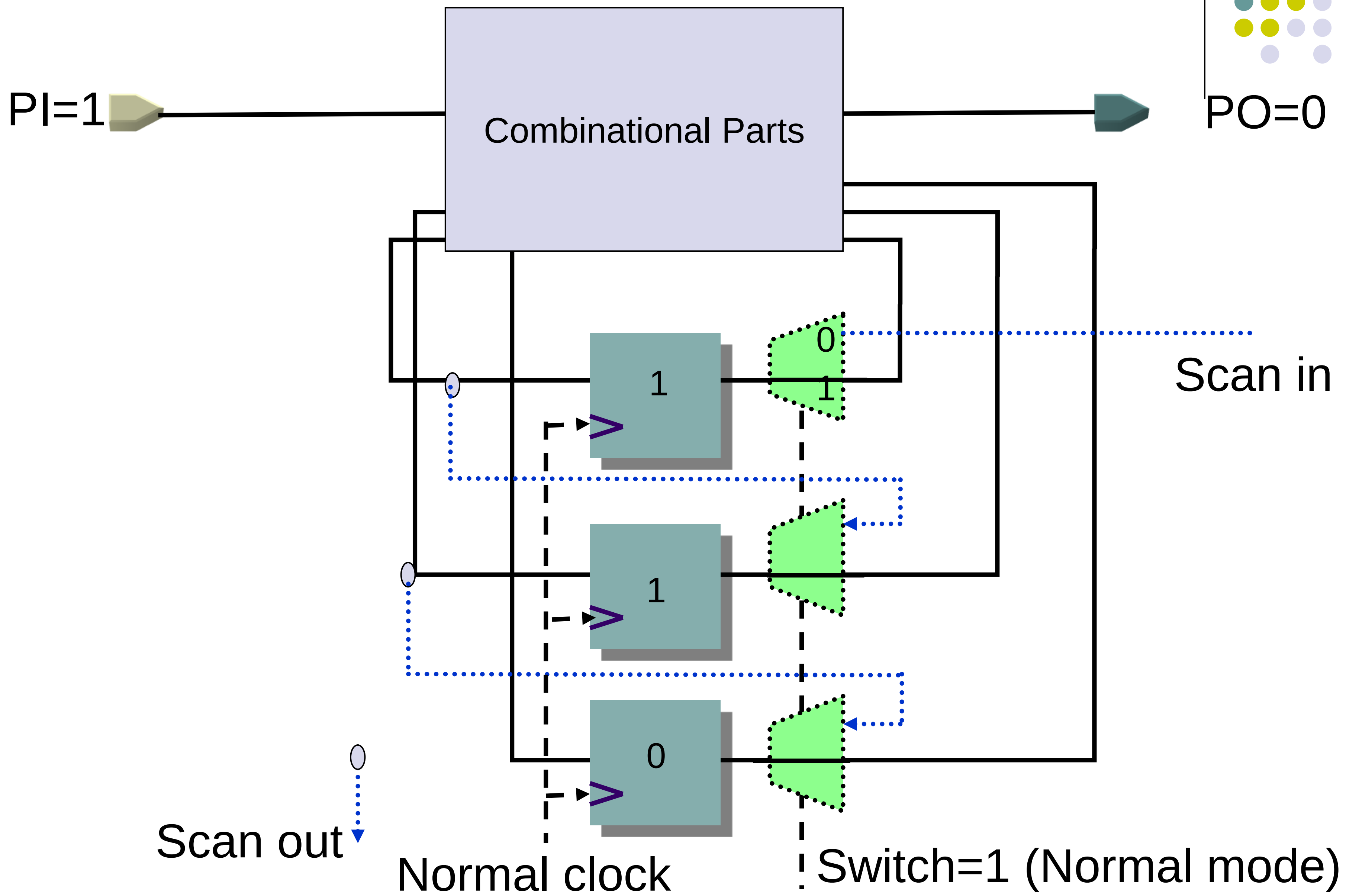
Scan Example (Scan In)



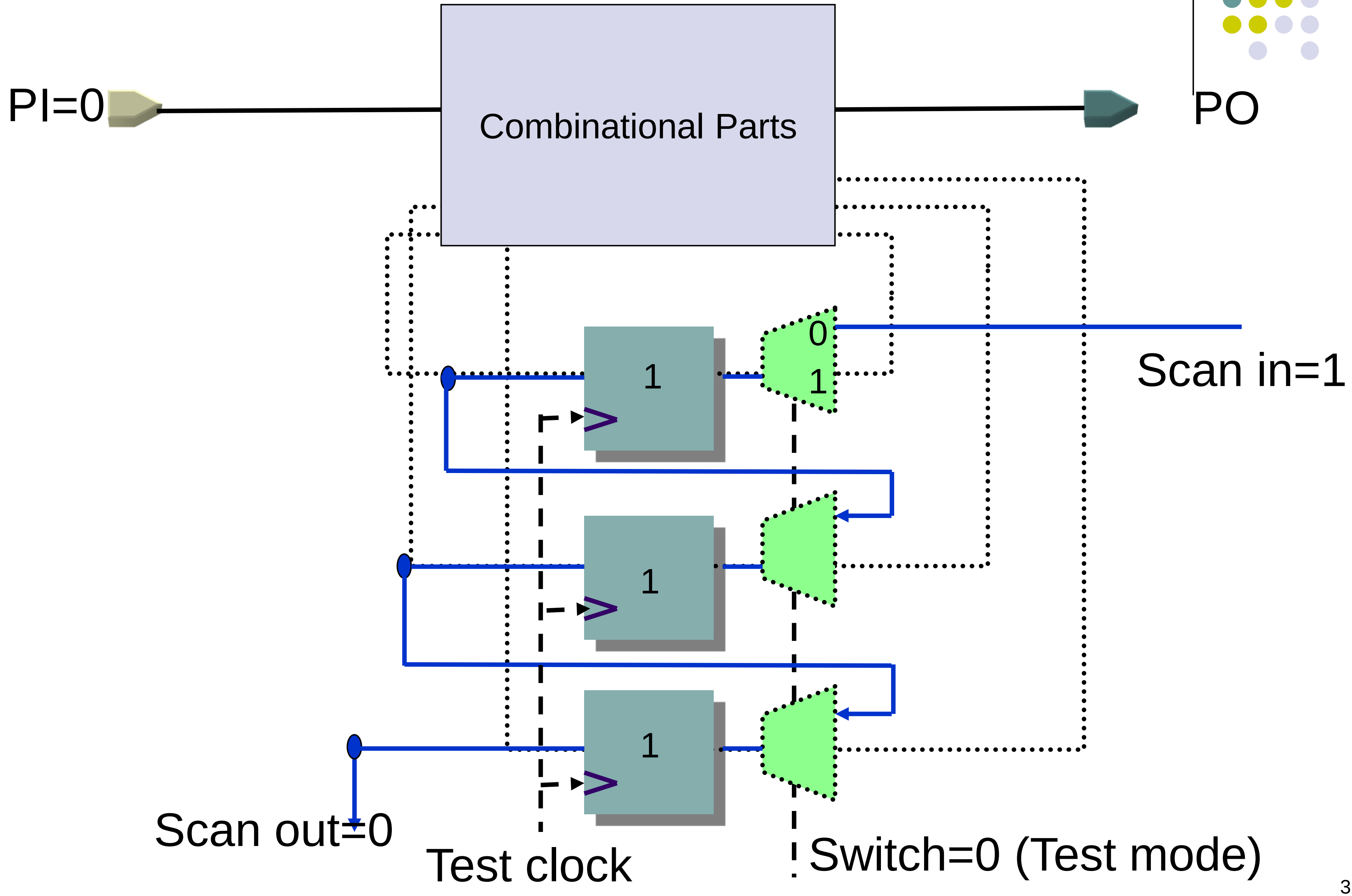
Scan Example (Scan In)



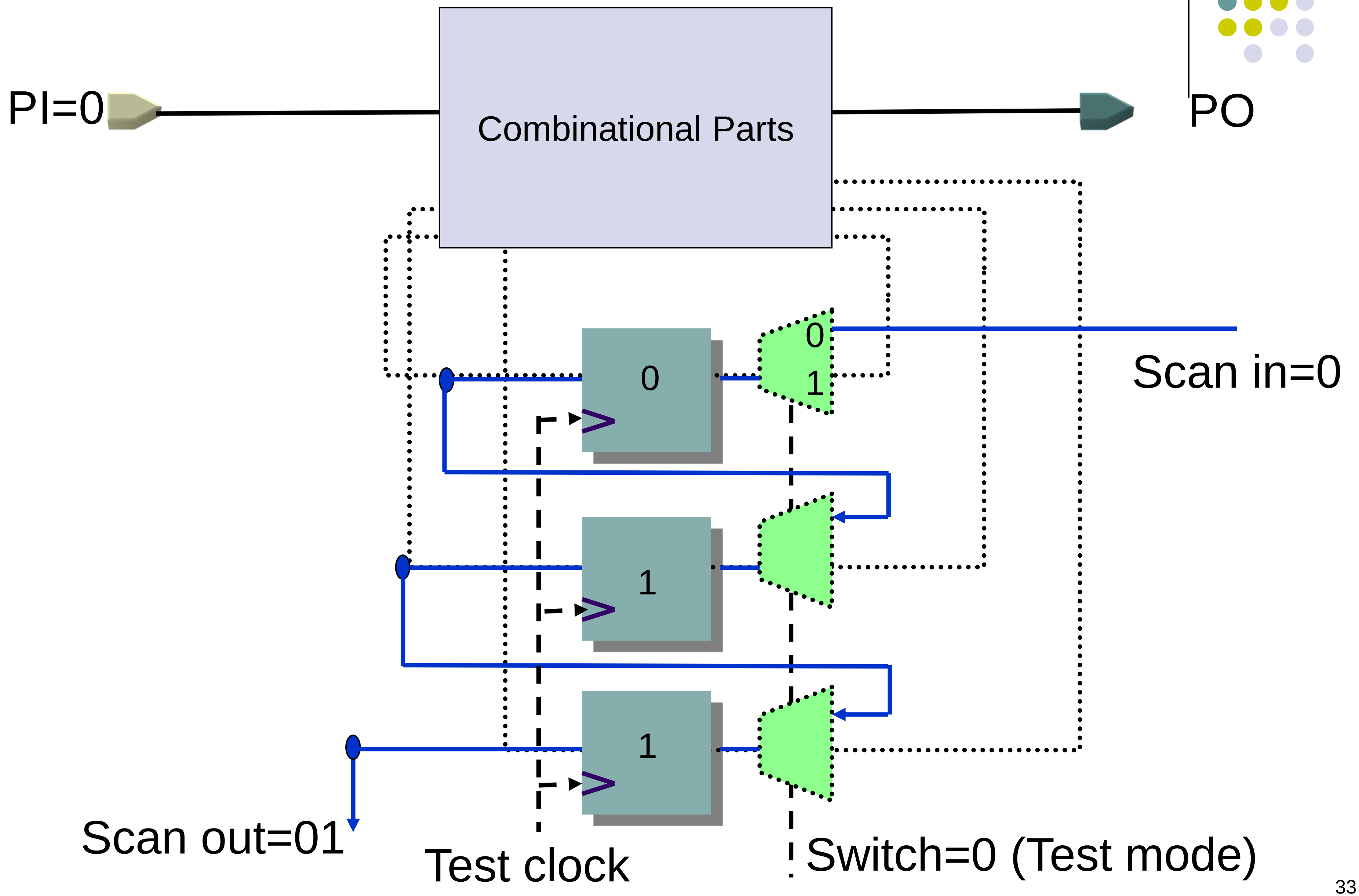
Scan Example (Normal Mode)



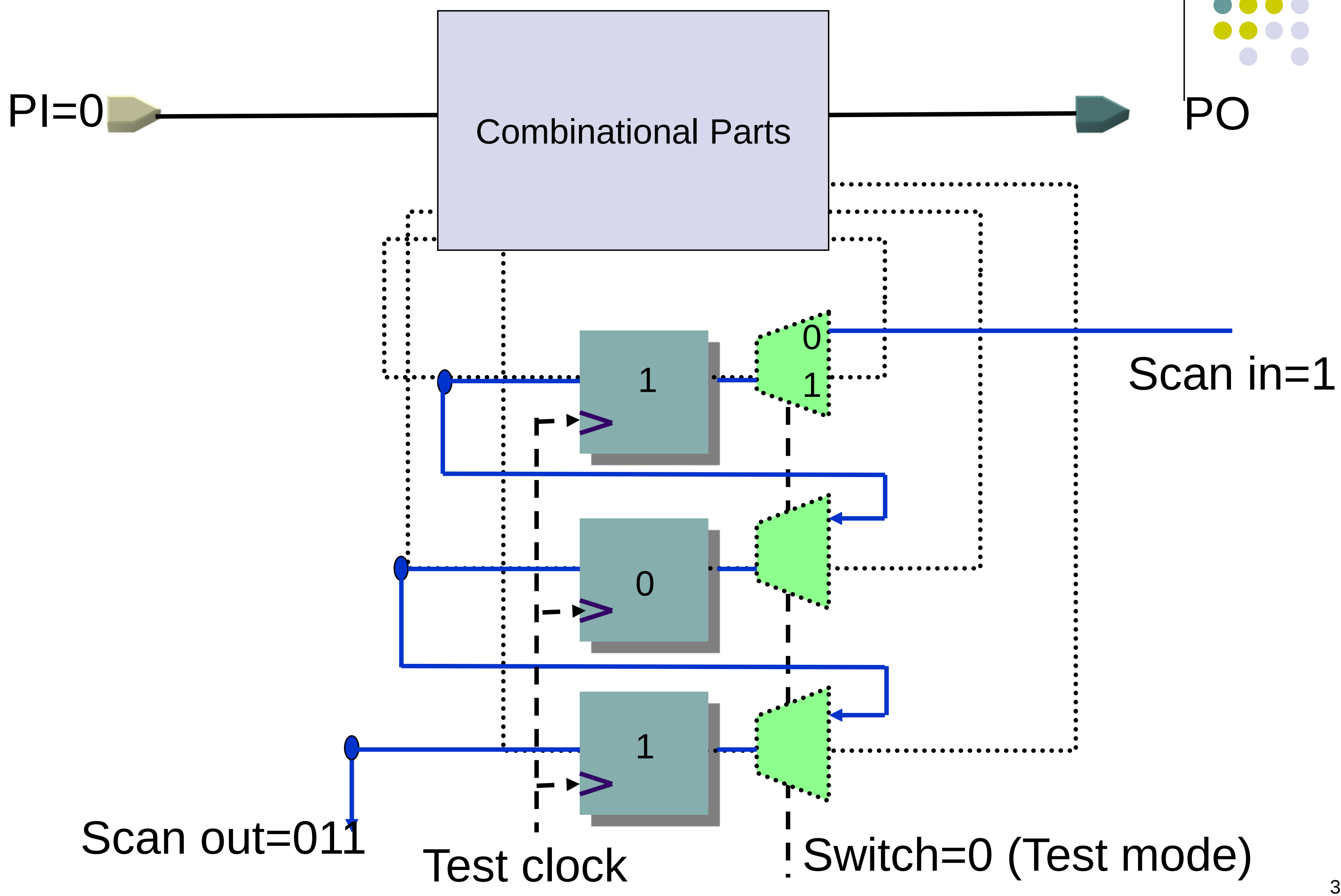
Scan Example (Scan Out)



Scan Example (Scan Out)



Scan Example (Scan Out)



Calculating Scan Test Clocks

- For each test vector, we need to shift in n_{sff} clock cycles (to setup the FFs) and apply one functional clock, and shift out with another n_{sff} clocks.
- The total number of clocks

$$n_{\text{sff}} + 1 + n_{\text{sff}} + 1 + n_{\text{sff}} + 1 + n_{\text{sff}} + \ldots + n_{\text{sff}} + 1 + n_{\text{sff}}$$

1st vector

2nd vector

3rd vector

last vector

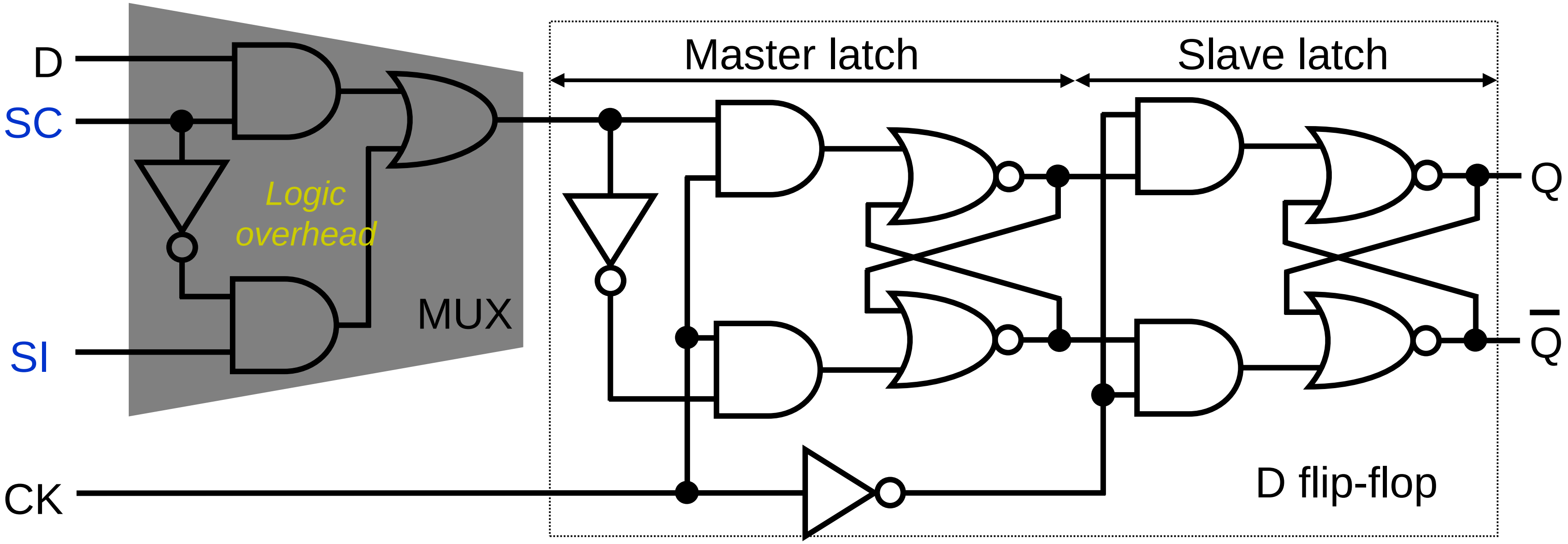
- Scan test length = $n_{\text{comb}}(n_{\text{sff}} + 1) + n_{\text{sff}}$

n_{sff}: number of scan flip-flops;
n_{comb}: number of combinational tests

Scan Cell Designs

MUXed Scan Flip-Flop

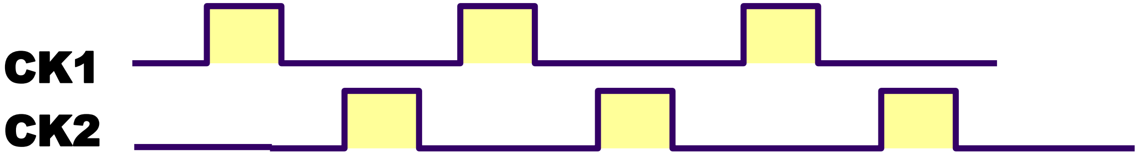
- Only D-type master-slave flip-flops are used
- 2 PIs (SC & SI) and 1 PO (SO) are used for test
- All flip-flop clocks controlled from primary inputs
  - No gated clock allowed
- Clocks must not feed data inputs of flip-flops
- Most popularly supported in standard cell libraries

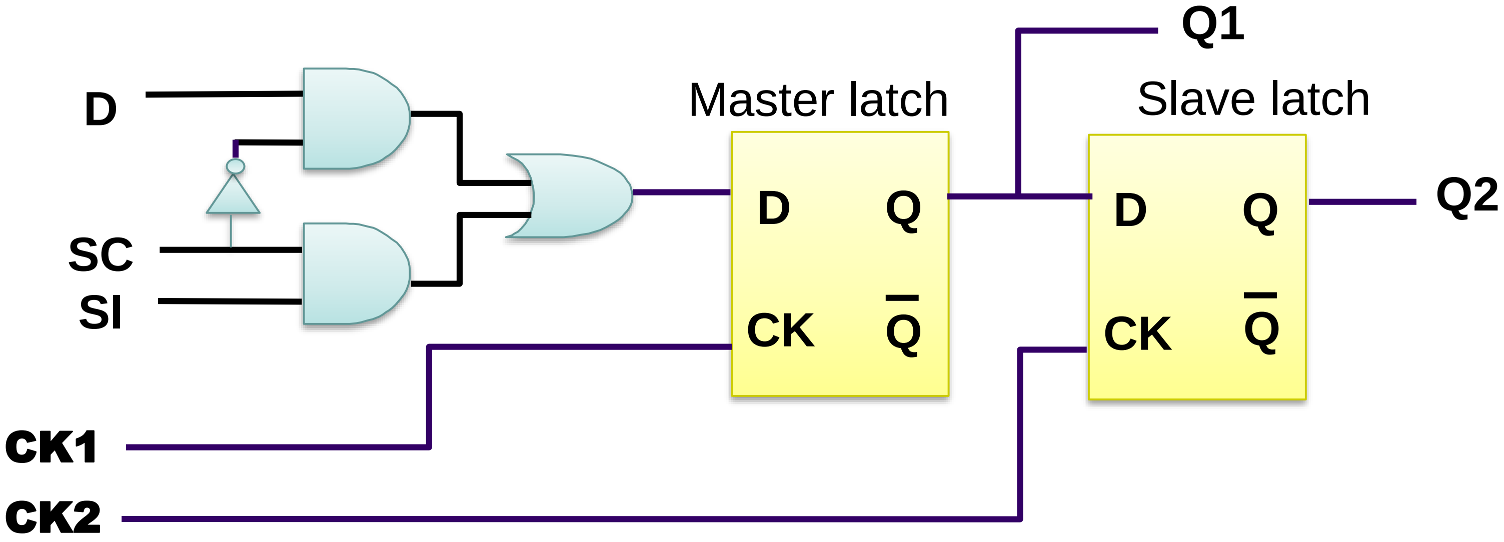


SC: normal / test SI: scan input

Multiplex Data Shift-Register Latch

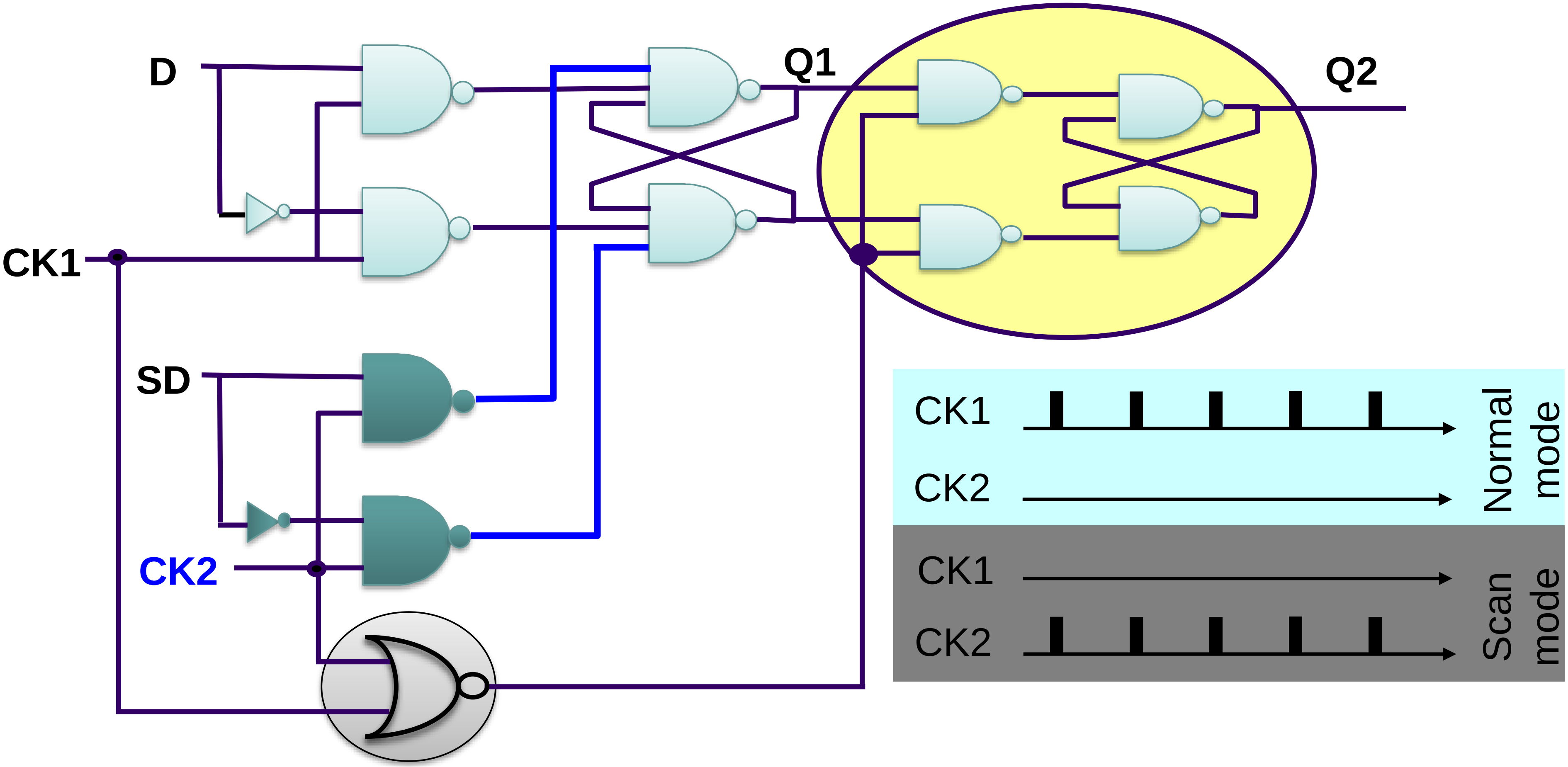
- Use two-phase clocking
  - CK1 and CK2 are two-phase non-overlapping clock which insures race-free operation





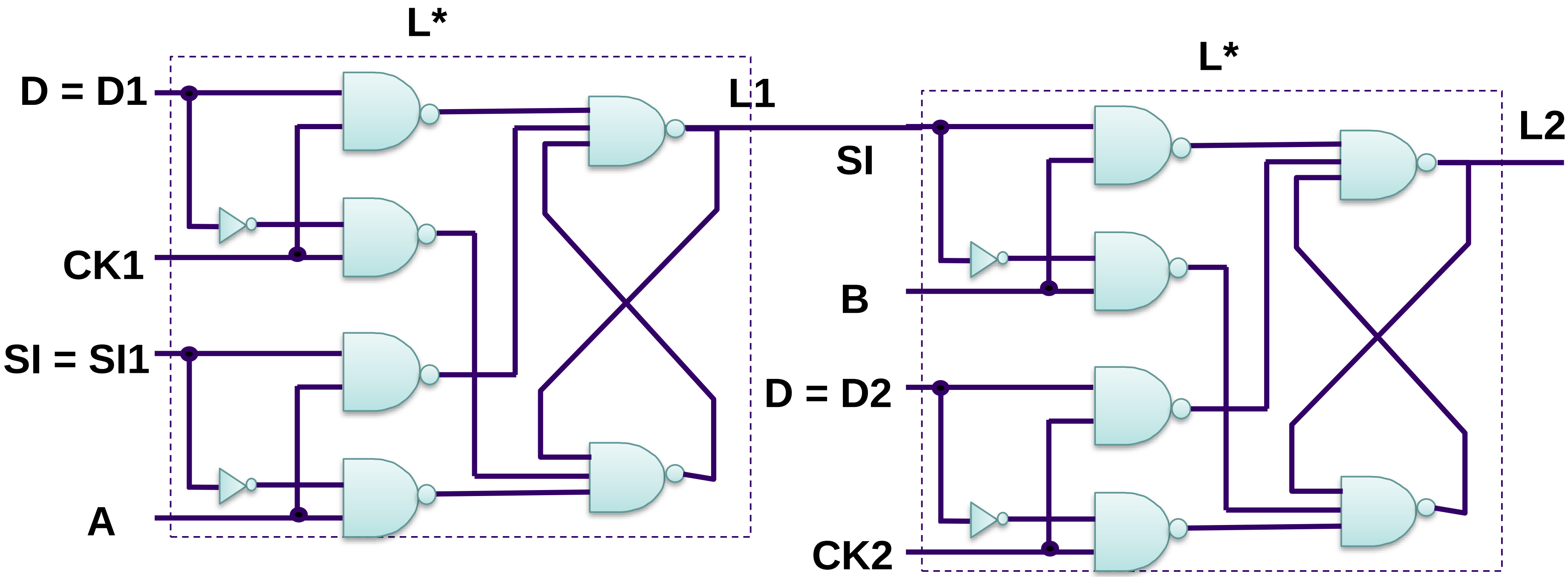
Two-Port Dual-Clock Scan Flip-Flop

- Less performance degradation than MUXed scan FF

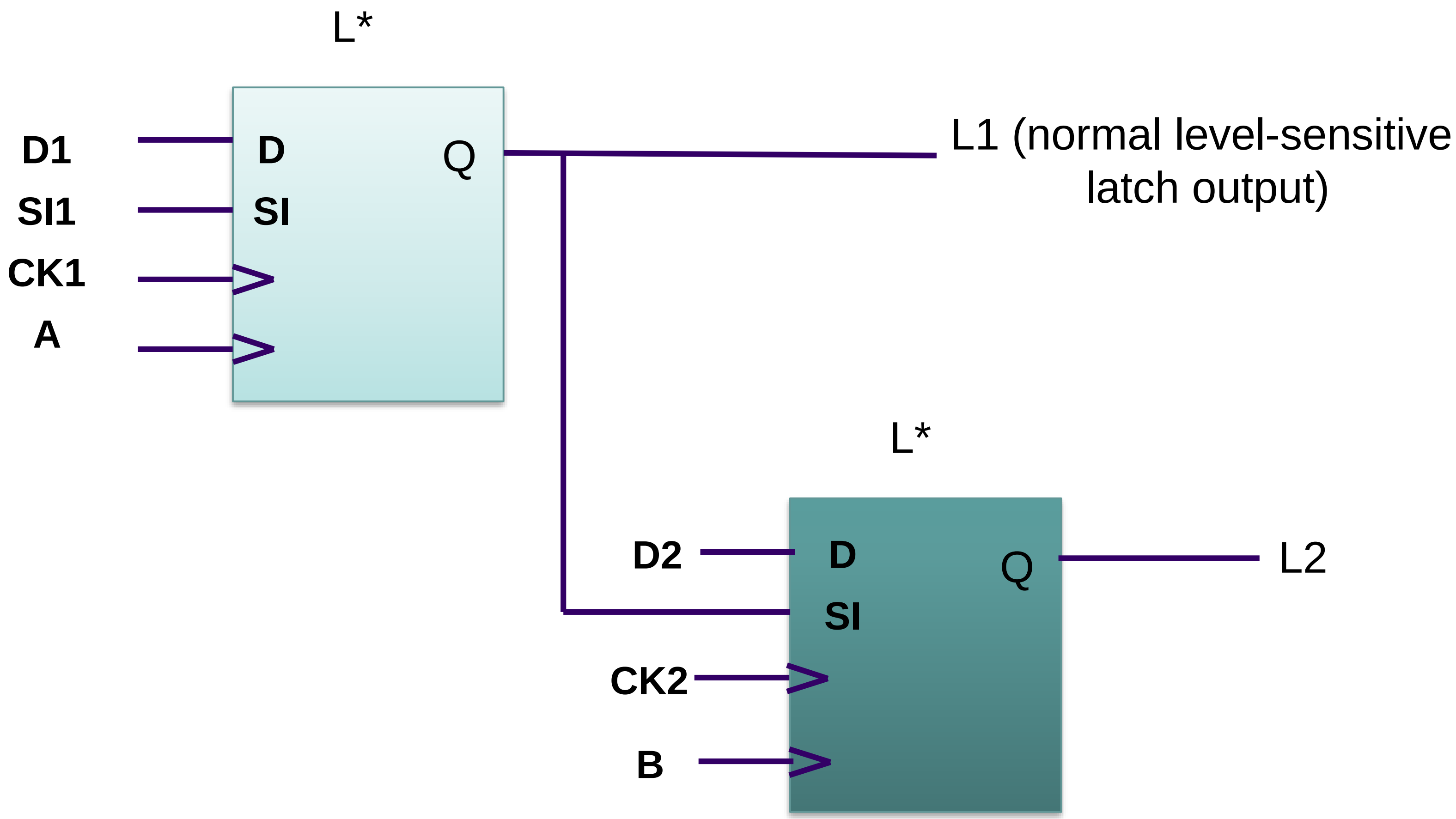


LSSD Single-latch Design (1977 IBM)

- LSSD: level-sensitive scan design
- Can be used for latch designs



Symbol of LSSD Scan FF



Comparing Three Scan Cell Designs

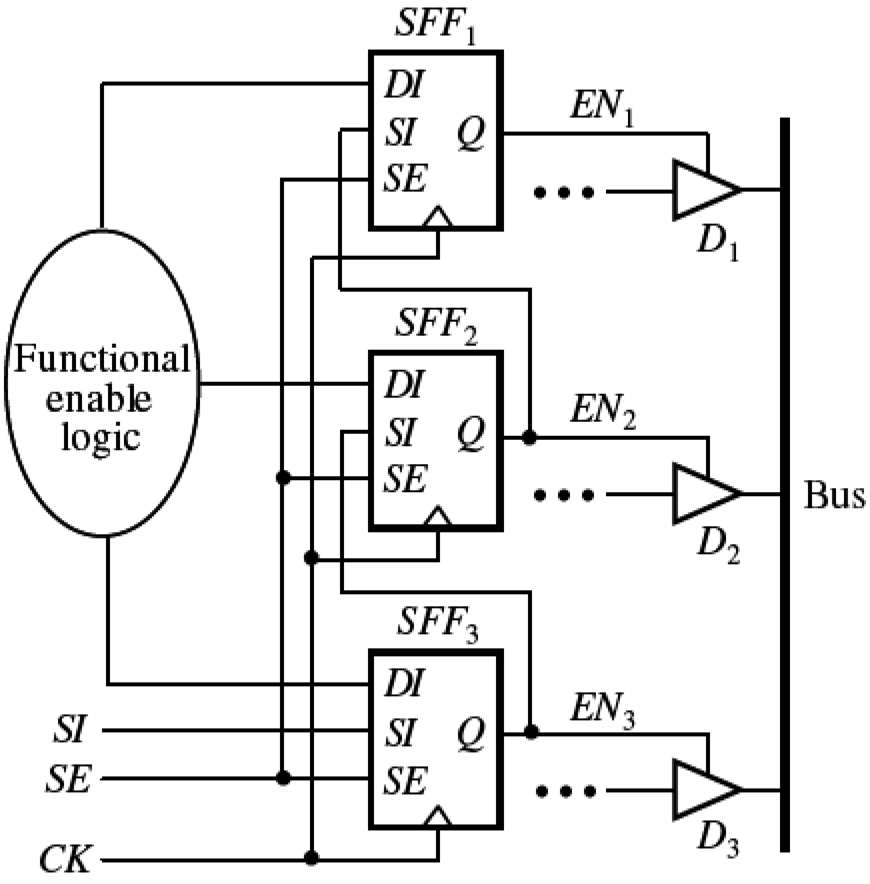
	Disadvantage	Advantage
Muxed-D Scan Cell	Compatibility to modern designs Comprehensive support provided by existing design automation tools	Add a multiplexer delay
Two-Port Dual-Clock Scan Cell	No performance degradation	Require additional shift clock routing
LSSD Scan Cell	Insert scan into a latch-based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

Scan 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/output mode
Gated clocks	Avoid during shift	Enable clocks during shift
Derived clocks	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	Not recommended	Use standard cells
Non-scan storage elements	Not recommended	Initialize to known states; bypass; or make transparent

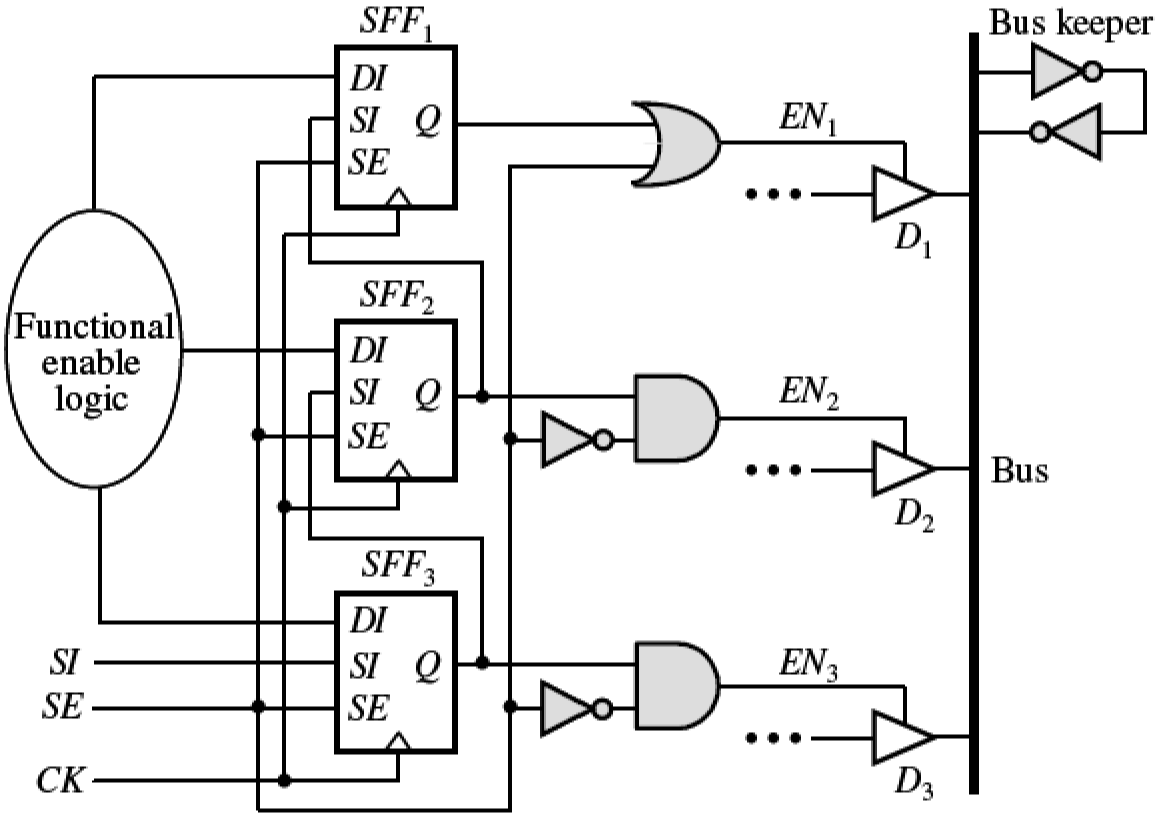
Tri-State Buses

- Bus contention occurs when two bus drivers force opposite logic values onto a tri-state bus.
- During the shift operation, contention can happen with continuous 1's as in the example.



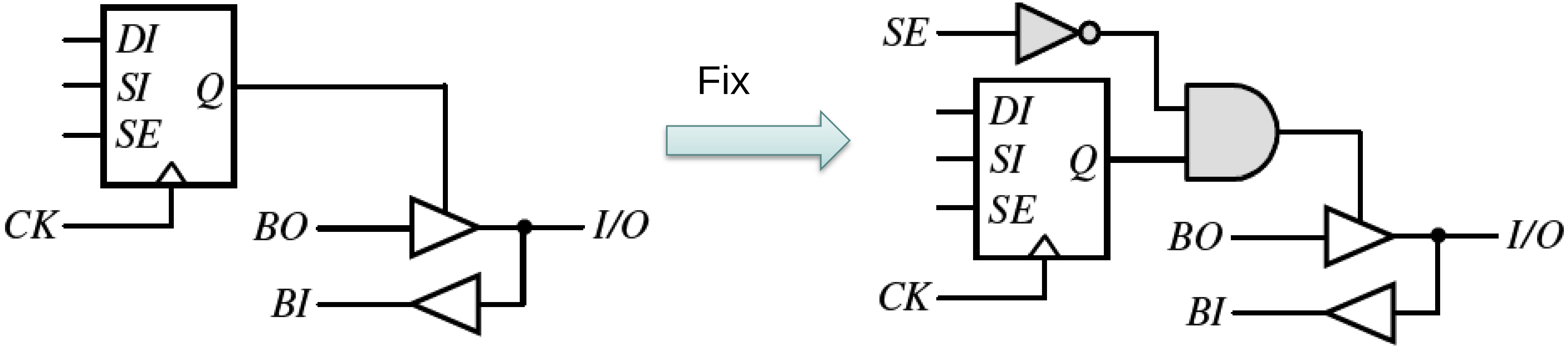
Tri-State Buses Fixes

- *when SE = 1*
  - *EN1=1, EN2=0 and EN3*=0 (only D1 enabled).
- The bus keeper is added to avoid uninitialized Z.



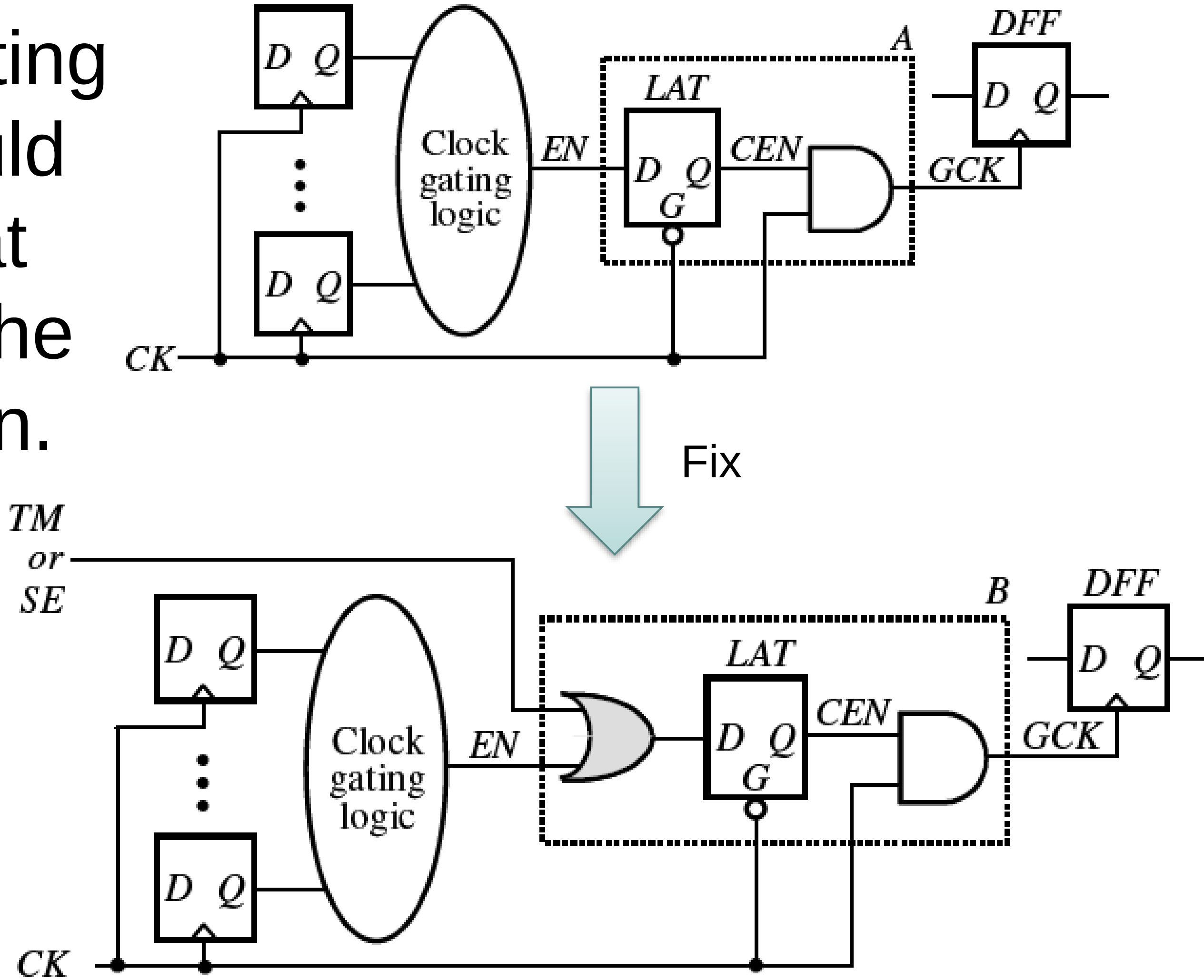
Bi-Directional I/O Ports

- During the shift operation, the input/output tristate buffer may become active, resulting in a conflict if BO and the I/O port driven by the tester have opposite logic values.



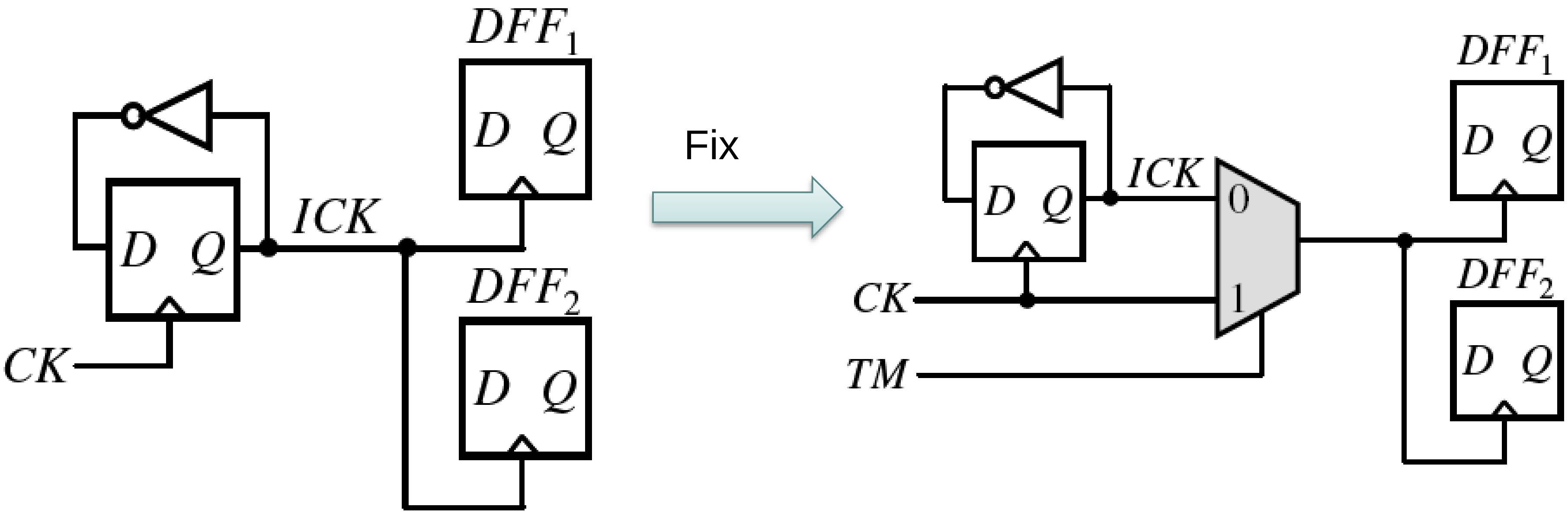
Gated Clocks

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



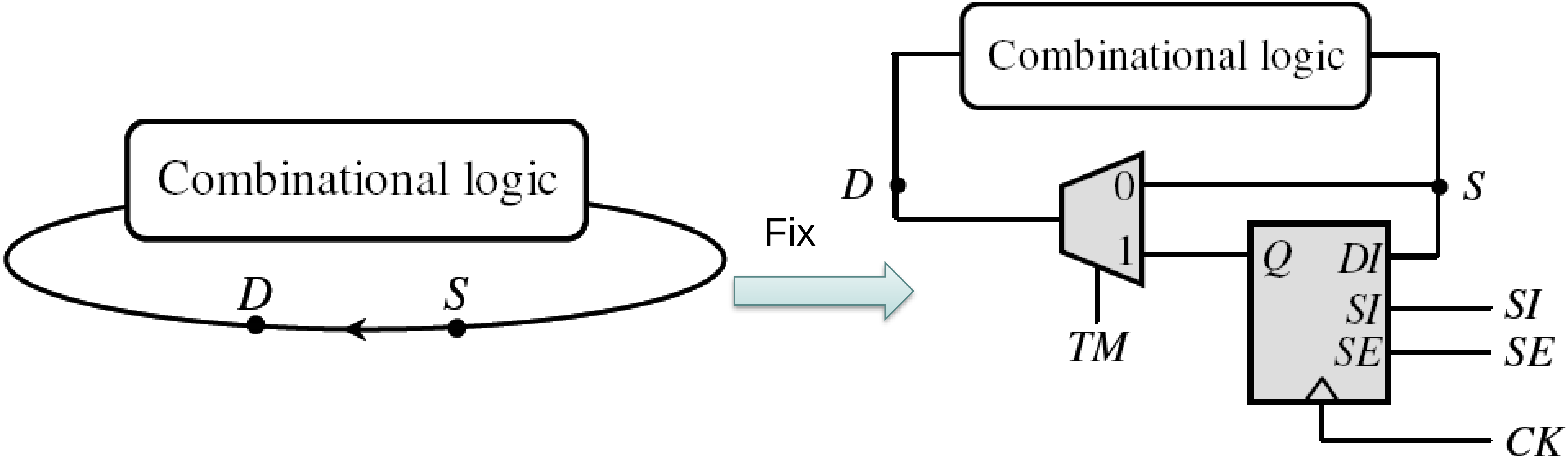
Derived Clocks

- A multiplexer selects *CK,* which is a clock directly controllable from a primary input, to drive *DFF1 and DFF2, during* the entire test operation, when *TM = 1.*



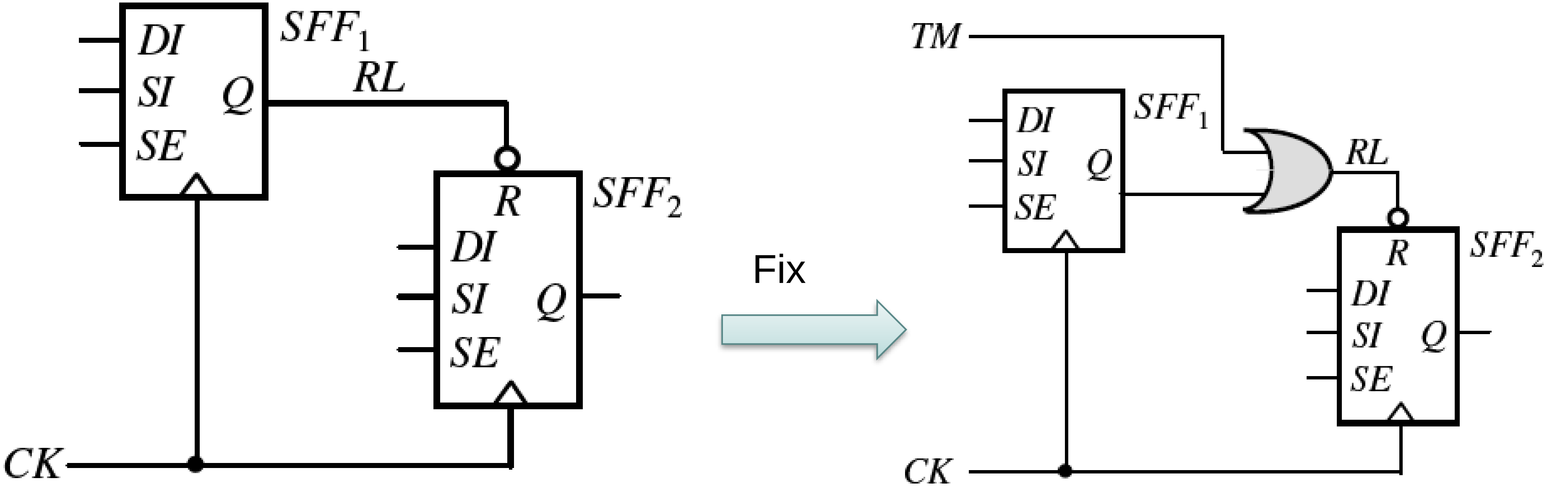
Combinational Feedback Loops

- 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.
- The best way is to rewrite the RTL code.

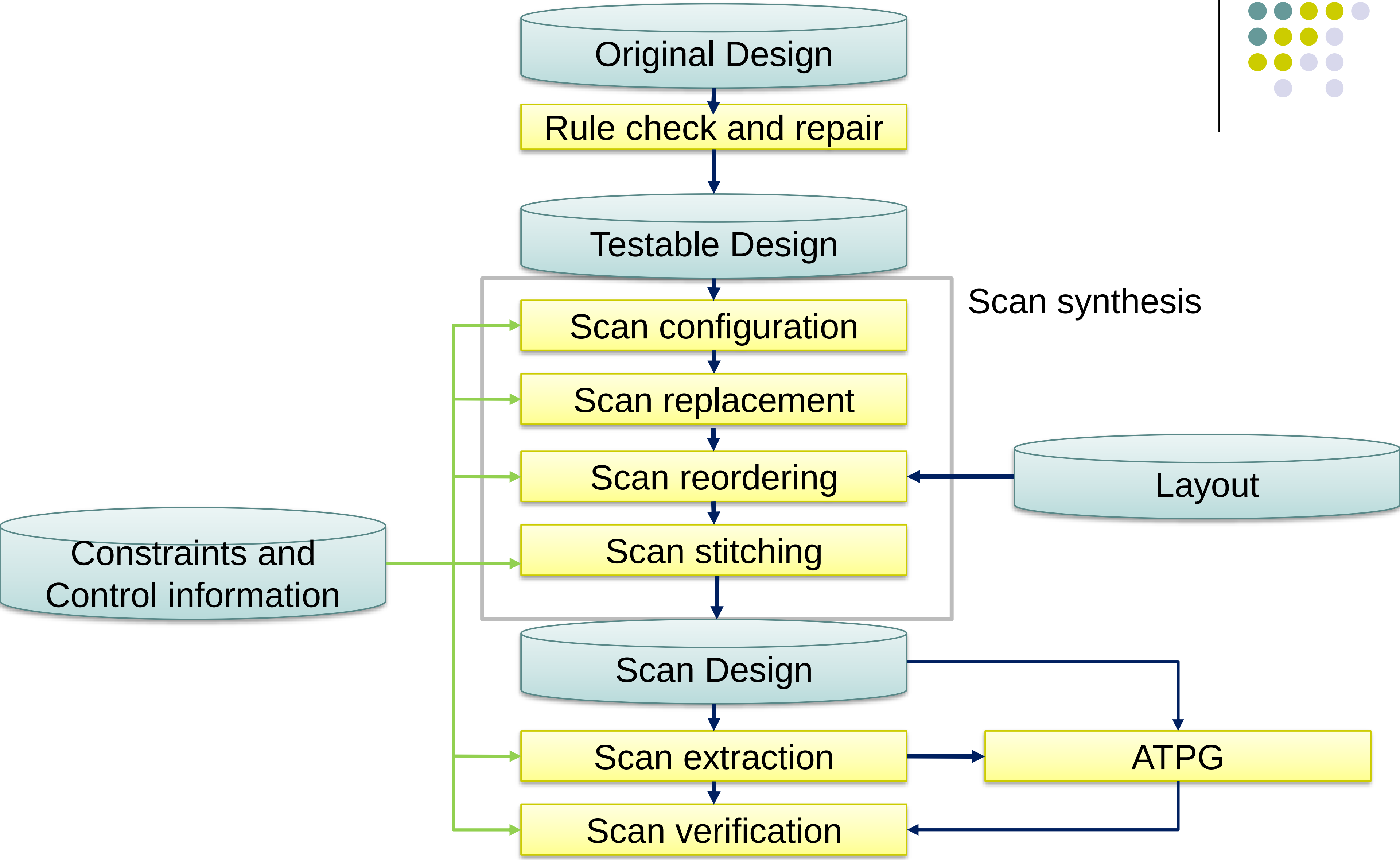


Asynchronous Set/Reset Signals

- Asynchronous set/reset signals of scan cells that are not directly controlled from primary inputs can prevent scan chains from shifting data properly.
  - To avoid this problem, these asynchronous set/reset signals are forced to an inactive state during the shift operation.



Scan Design Flow



Scan Design Steps (I)

- 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 Steps (II)

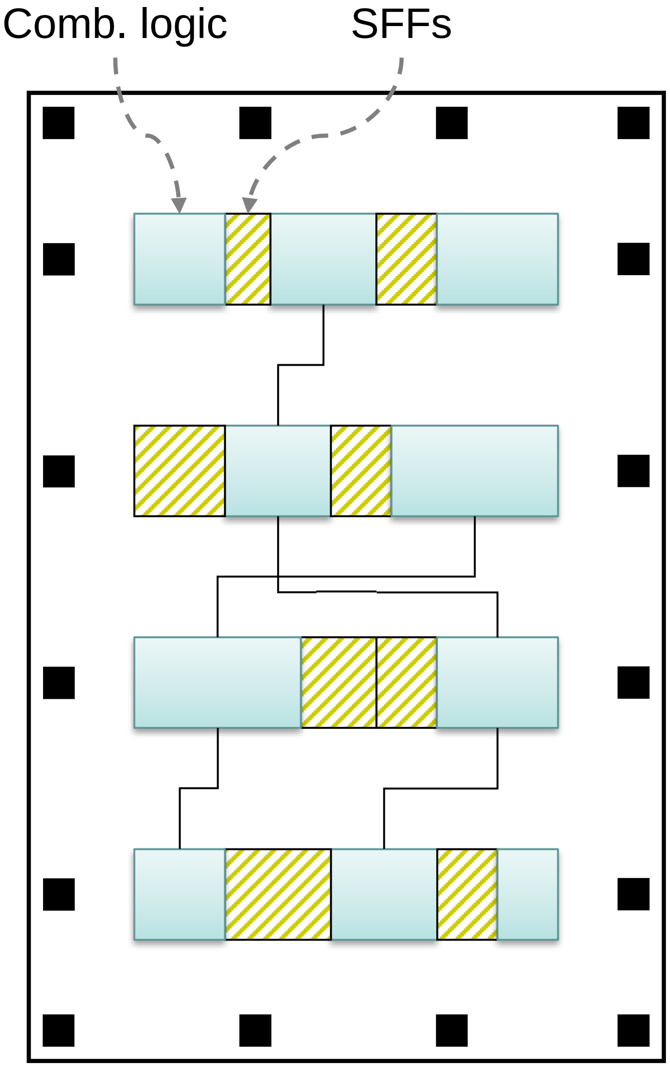
- Scan Extraction
  - Is the process used for extracting all scan cell instances from all scan chains specified in the scan design
- Scan Verification
  - A timing file in *standard delay format (SDF) which resembles the timing* behavior of the manufactured device is used to
    - Verifying the scan shift operation
    - Verifying the scan capture operation

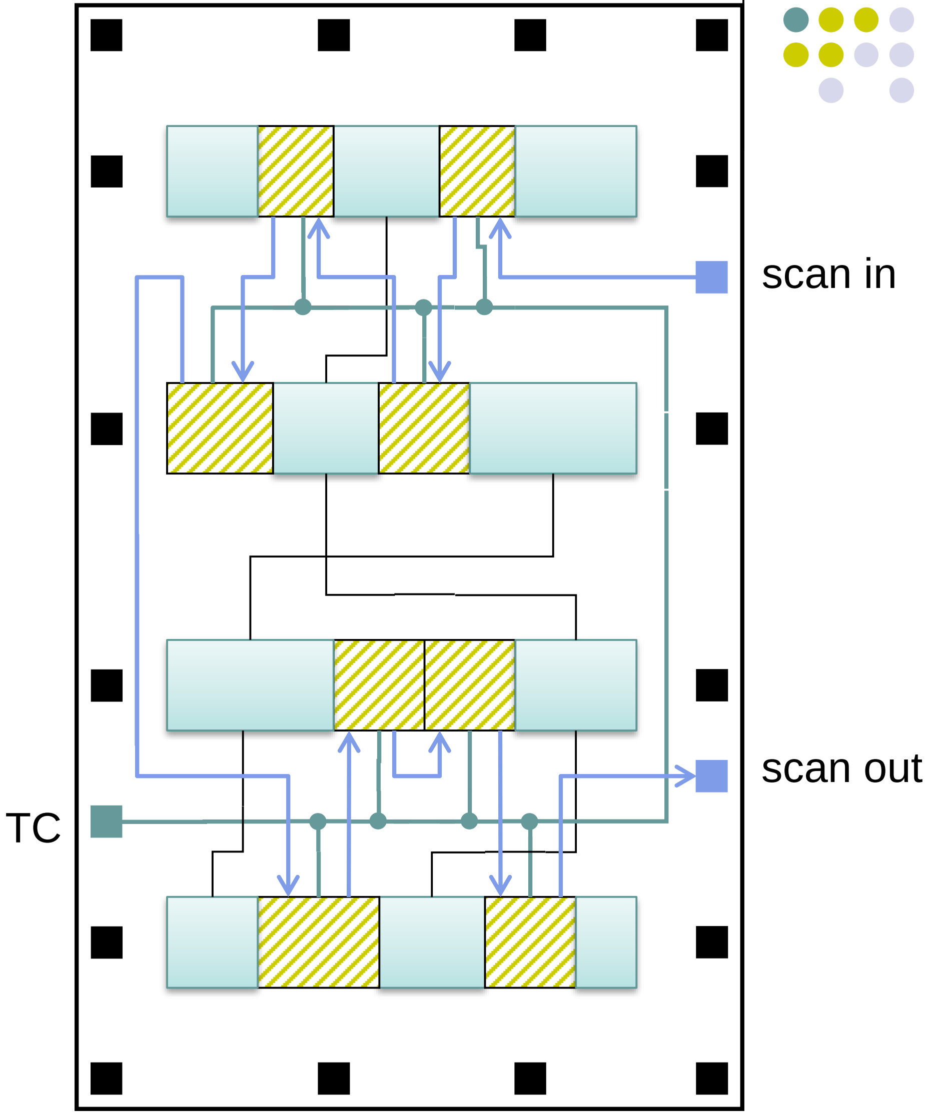
Four Processes for Scan Synthesis

- 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

Physical Design of Scan with Standard Cells

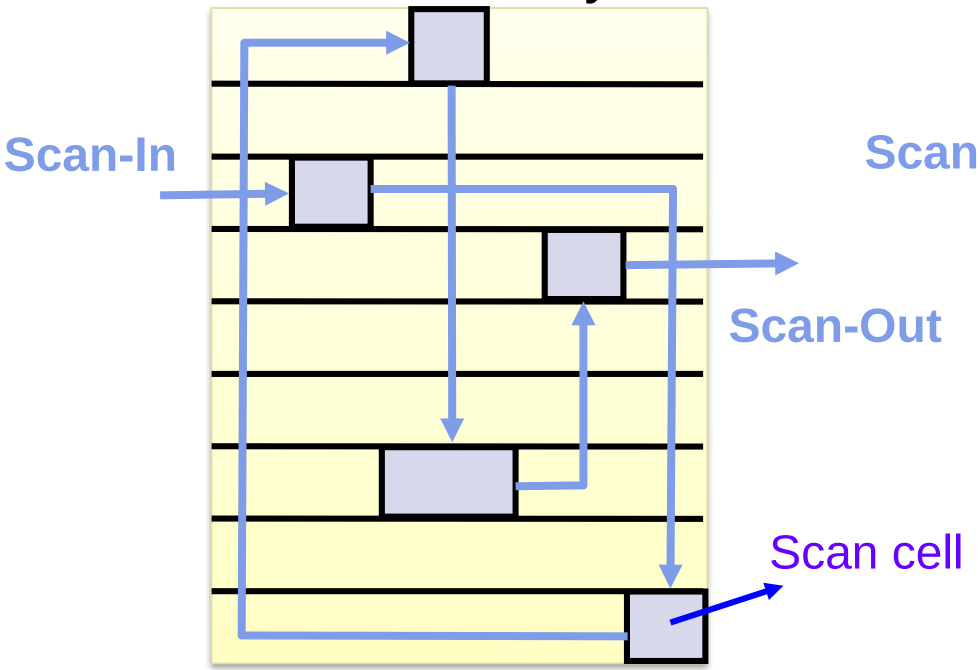
- First, placing the cells without scan wiring.
  - To avoid adversely affect the functional interconnects.
- Replace FFs with SFFs.
  - Wider than original.
- Add *TC* control line.
  - At most one track in every alternate routing channel.
- Scan path routing.
  - One track in every alternate routing channel is possible.


Comb. logic
SFFs

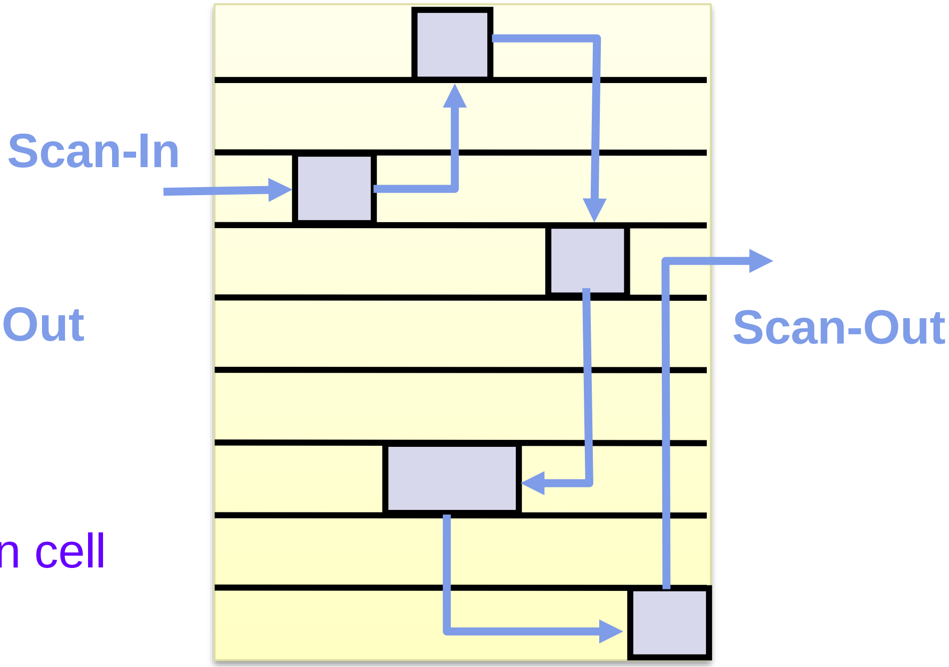

scan in
scan out
TC

Scan-Chain Reordering

- Scan-chain order is often decided at gate-level without knowing the cell placement
- Scan-chain consumes a lot of routing resources, and could be minimized by re-ordering the flip-flops in the chain after layout is done

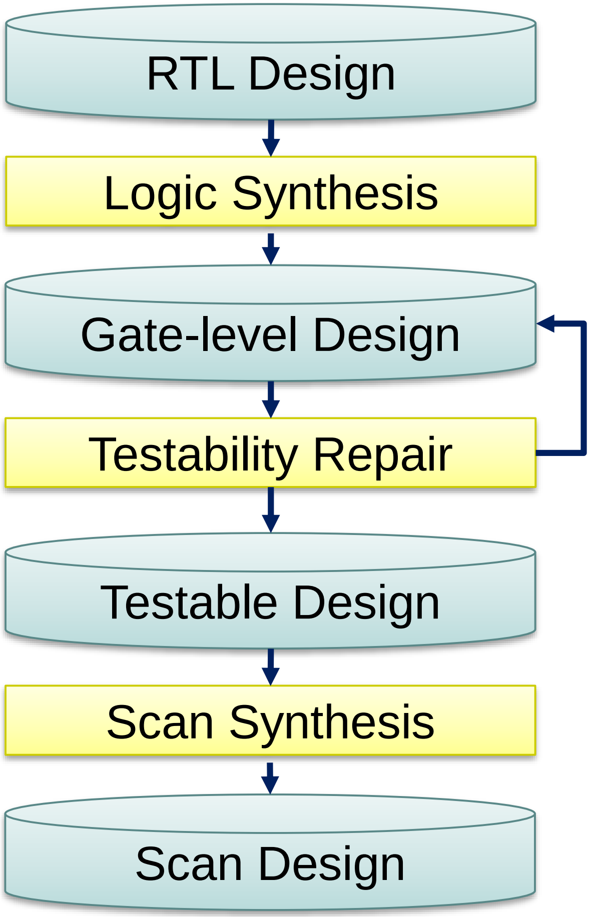


Layout of a cell-based design



A better scan-chain order

RTL Design for Testability



RTL Design

Logic Synthesis

Gate-level Design

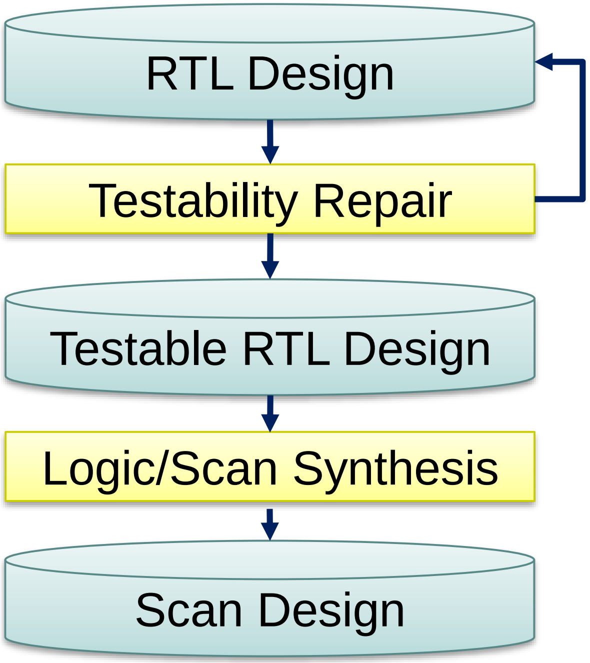
Testability Repair

Testable Design

Scan Synthesis

Scan Design

Gate-level testability repair flow



RTL Design

Testability Repair

Testable RTL Design

Logic/Scan Synthesis

Scan Design

RTL testability repair flow

RTL Scan Design Rule Checking

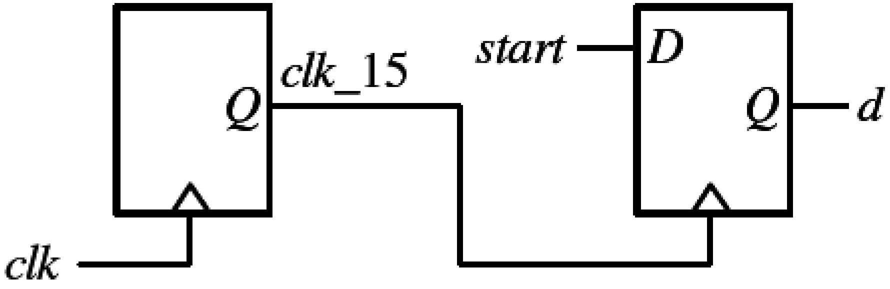
- Identify testability problems
  - Static solutions (without simulation)
  - Dynamic solutions (with simulation)
  - Not typically used in current design flow
- DFT rule violations are solved in RTL code by designers themselves, instead of in gate level by EDA tools

RTL Scan Design Repair – An Example

- Original design

```
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)



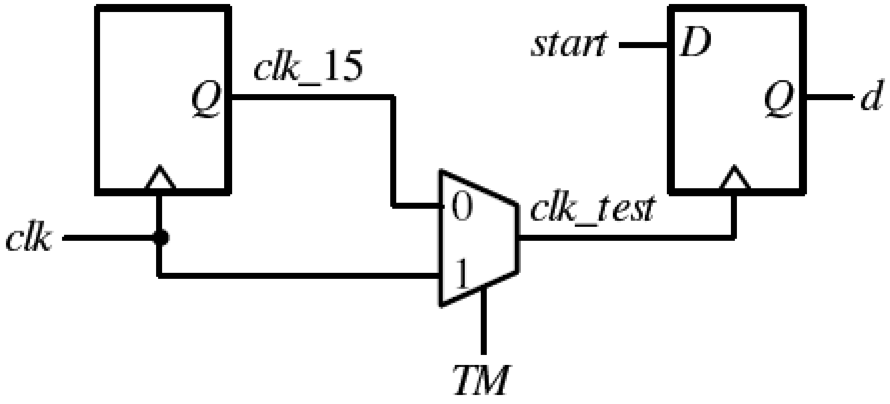
(b) Generated clock (Schematic)

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;
```

(a) Generated clock (RTL code)



(b) Generated clock repair (Schematic)

Problems with Scan Design

- Area overhead
  - Increased gate count
  - Increased routing area
- Performance degradation
  - Extra gate delay due to the multiplexer
  - Extra delay due to the capacitive loading of the scan-wiring at each flip-flop’s output
- Long test application time.
- Not applicable to all designs.
  - Must follow the scan design rules.
- High power dissipation during testing.

Long Test Times for Scans

- Test data volume ≈ scan cells * scan patterns
- Test time ≈ $\dfrac{\text{scan cells} * \text{scan patterns}}{\text{scan chains} * \text{frequency}}$
- An example circuit
  - 10M gates, 16 scan chains, one scan cell per 20 gates.
  - The test time to apply 10000 scan patterns at 20MHz scan-shift frequency=16 seconds!

$$Time = \frac{(\frac{10M}{20})*10000}{16*20M} = 15.625 \approx 16$$

Multiple Scan Chains

- To reduce test time.
  - However, each scan register has its own scan-in and scan-out.
- The scan chains may differ in length.
  - Test time determined by the longest one.

Scan Chain Debug

- Given more scanned FFs in a circuit, the probability of having failed cells is increasing.
- Failure modes can be either functional or timing errors.
  - Stuck-at faults.
  - Hold-time violations for scan-in.

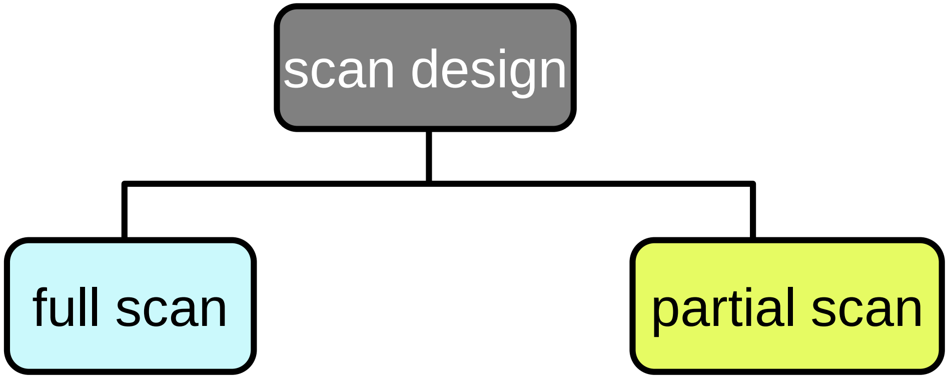
Outline

- Introduction
- Ad-Hoc Approaches
- Full Scan
- Partial Scan
  - Cycle Breaking Techniques
  - BALLAST approach

Partial Scan

- Basic idea
  - Select a subset of flip-flops for scan
  - Lower overhead (area and speed)
  - Relaxed design rules
- Storage elements on the data path are left out of the scan cell replacement process
- Cycle-breaking technique
  - Cheng & Agrawal, IEEE Trans. On Computers, April 1990
  - Select scan flip-flops to simplify sequential ATPG
  - Overhead is about 25% off than full scan
- Timing-driven partial scan
  - Jou & Cheng, ICCAD, Nov. 1991
  - Allow optimization of area, timing, and testability simultaneously

Full Scan vs. Partial Scan



Every flip-flop is a scaned.

NOT every flip-flop is scaned.

test time	longer	shorter
hardware overhead	more	less
fault coverage	~100%	unpredictable
ease-of-use	easier	harder

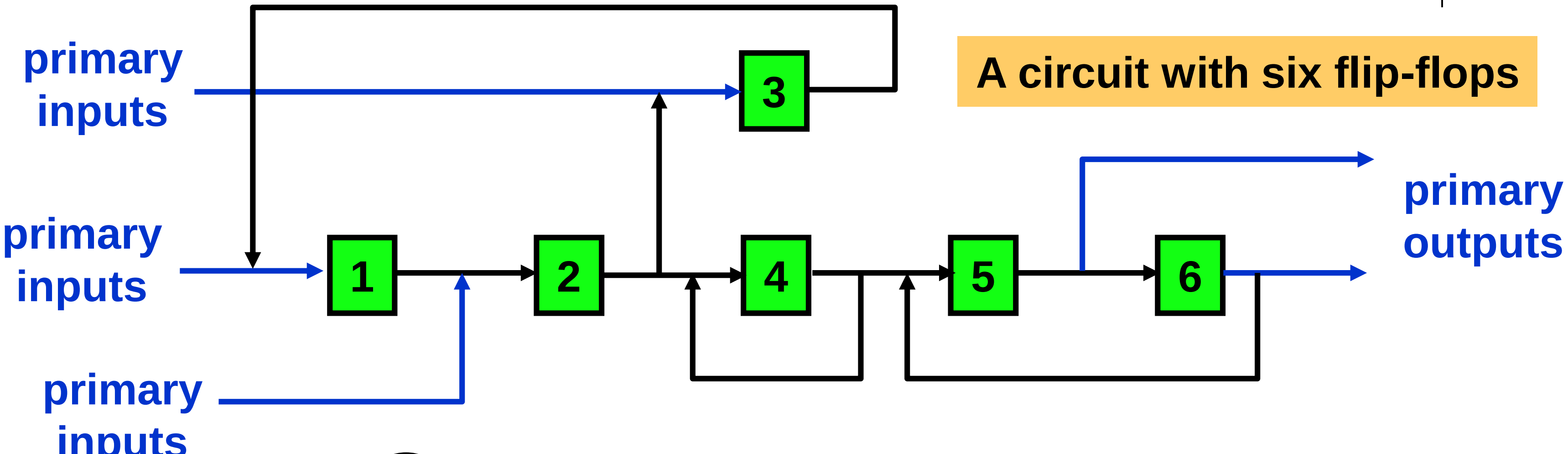
What Makes Test Generation Difficult ?

- Poor initializability
- Poor controllability and observability of memory elements
- Structure-dependence

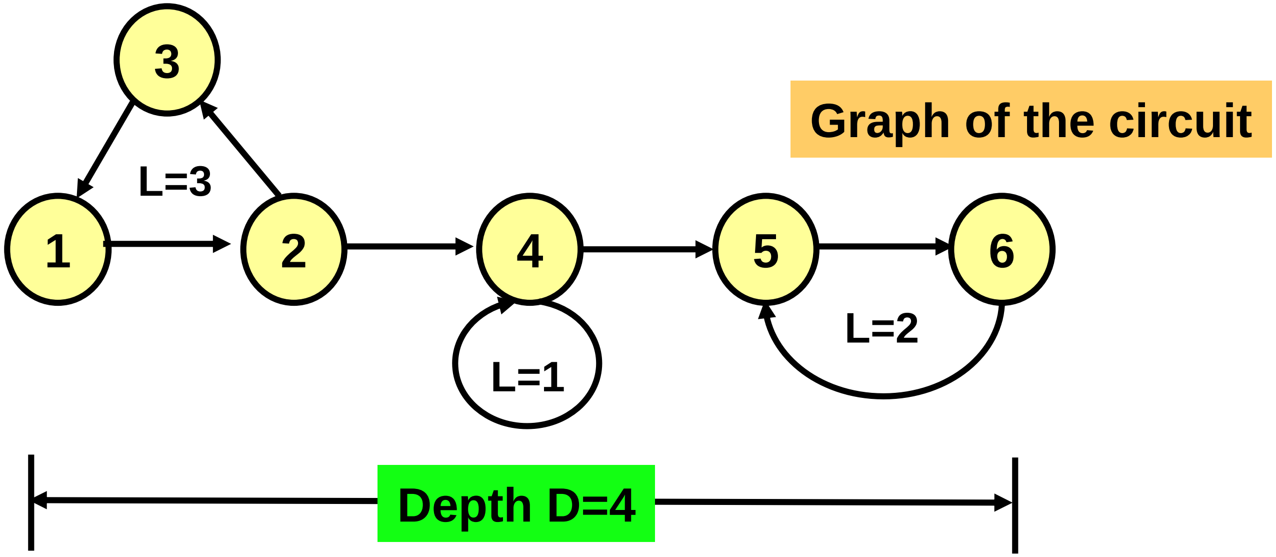
Circuit	No. of Gates	No. of Flip-flops	Sequential Depth	Test Gen. CPU sec.	Fault Coverage
TLC	355	21	14	1247	89.01%
CHIP-A	1112	39	14	269	98.80%

- **Gate count, memory element count, and sequential depth do not explain the results**
- **Cycles in the circuit are mainly responsible for the test generation complexity**

Directed Graph Of A Synchronous Sequential Circuit



A circuit with six flip-flops



Graph of the circuit

Depth D=4

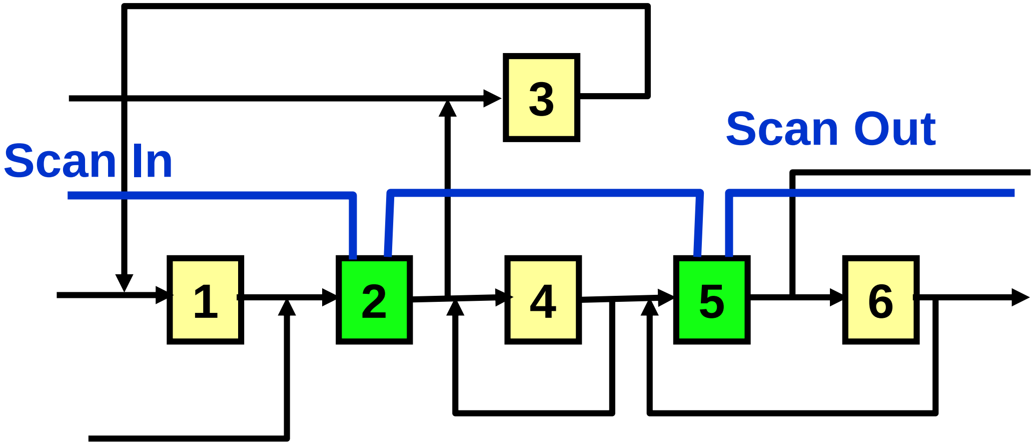
Test Length In A Sequential Circuit

- Notations:
  - D: sequential depth (The distance along the longest path in its graph)
  - L: maximum length of any cycle
- Test Generation Complexity
  - For a cycle-free circuit (e.g., pipeline structure), the complexity is similar to that of a combinational circuit
  - In a circuit with depth D, any single fault can be tested by at most D vectors
  - The length of a test sequence ~ $D \cdot 2^L$

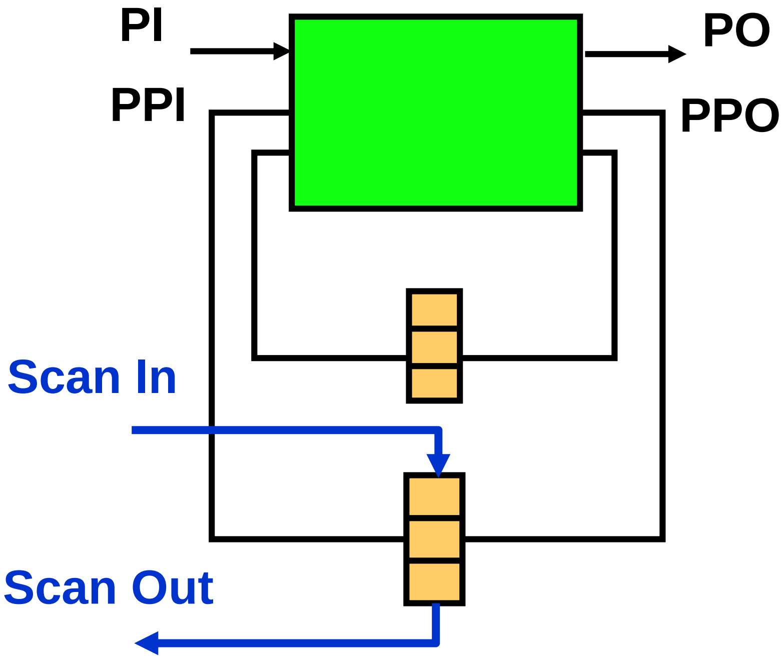
Partial Scan For Cycle-Free Structure

- Select minimal set of flip-flops
  - To eliminate some or all cycles
- Self-loops of unit length
  - Are not broken to reduce scan overhead
  - The number of self-loops in real design can be quite large
- Limit the length of sequential depth
  - Long sequential depth in large circuits may pose problems to sequential ATPG

Partial Scan Design

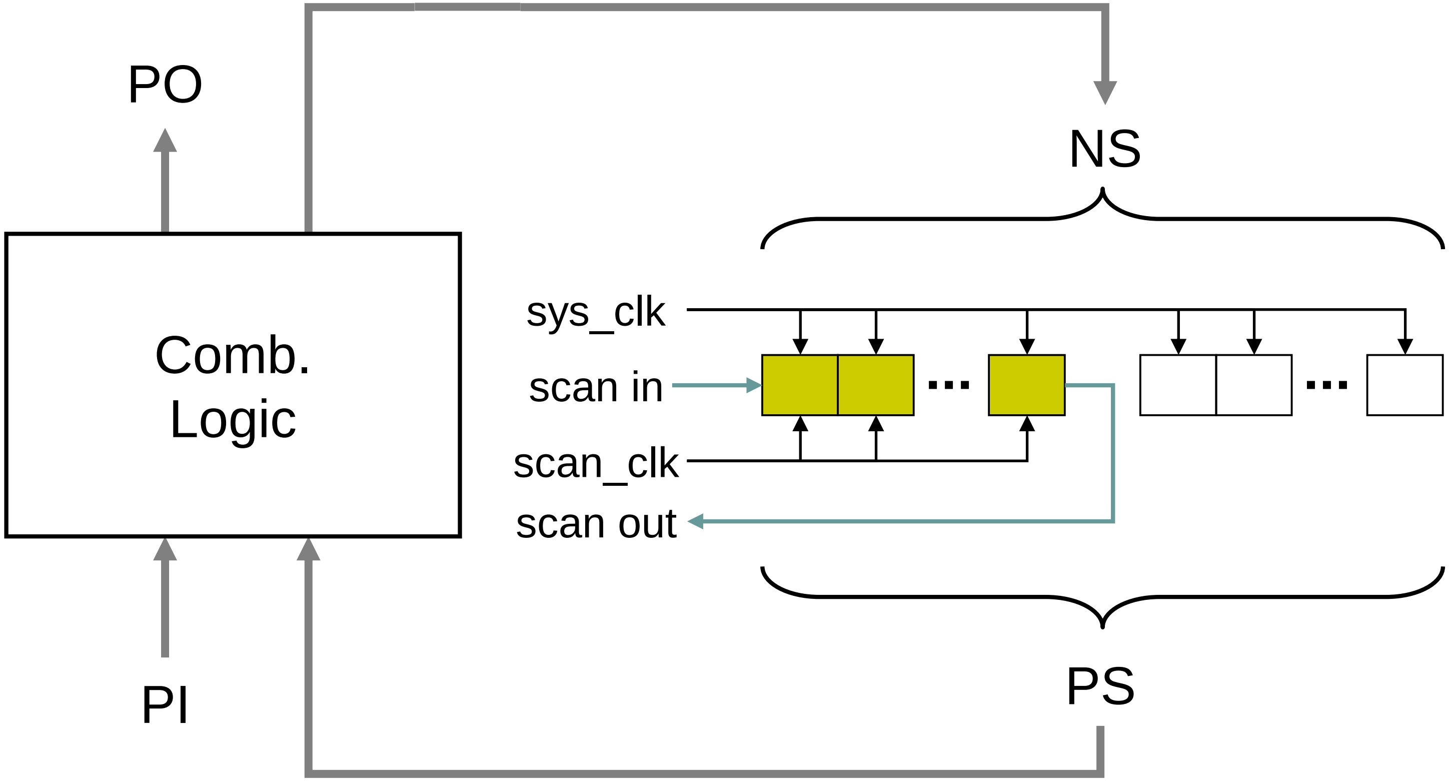


Scan Flip-Flops: {2, 5}
Non-Scan FFs: {1, 3, 4, 6}

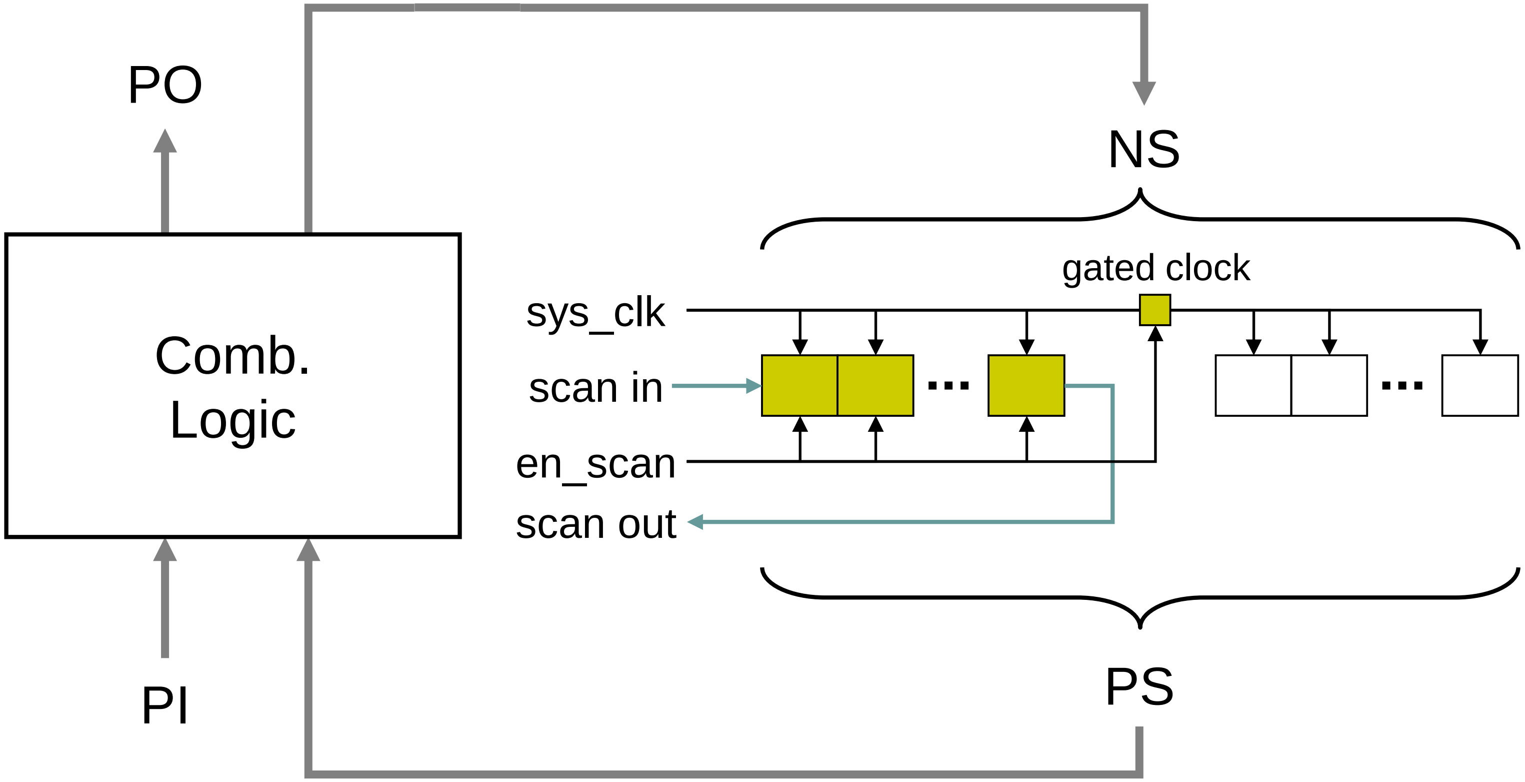


Clocking Schemes for Partial Scan Circuits

- Scheme I:
  - Use a separate scan clock



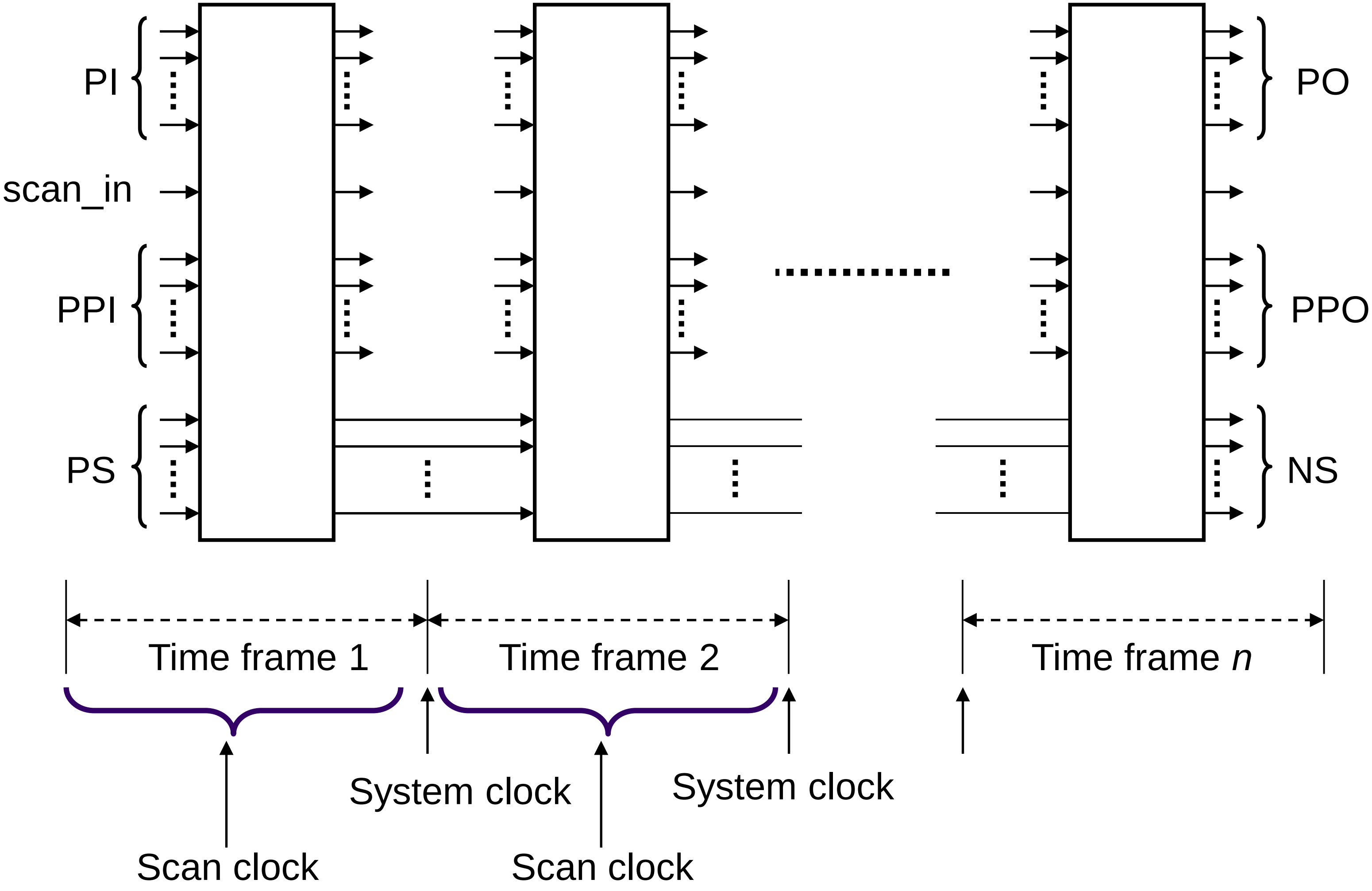
- Scheme II:
  - Gate the system clock



Partial Scan w/ a Separate Scan Clock or Gated Clock

- Require multiple clock trees
  - Extra clock signal routing efforts
- Test generation is easier
  - Scan FFs are fully controllable and observable.
- Test generation procedure:
  - Scan FFs are removed and their I/O's are added to the PO/PI lists.
  - A sequential ATPG is used for test generation.
  - The vector sequences are then converted into scan sequences
    - Each vector is preceded by a scan-in sequence to set the states of the SFFs.
    - A scan-out sequence is added to each vector sequence.

Test Generation Model – A Separate Scan Clock or Gated Clock

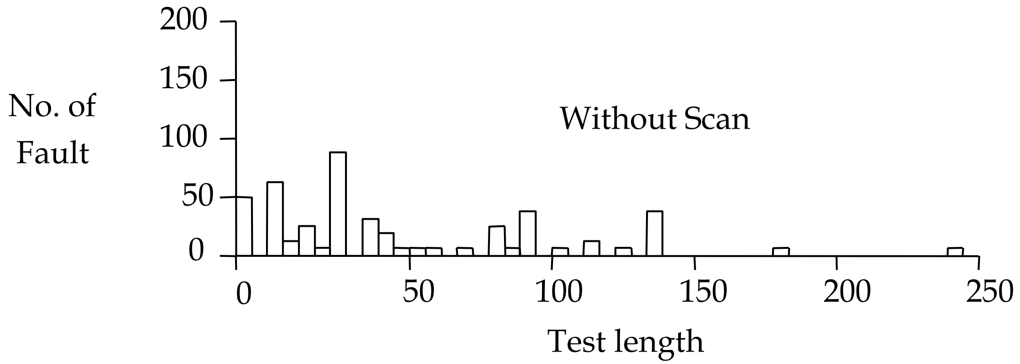


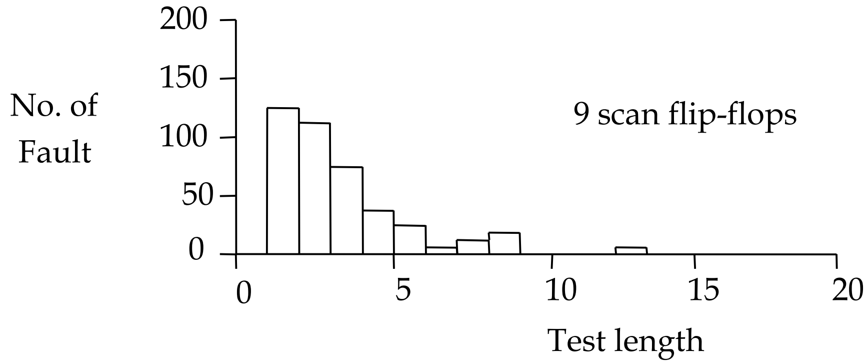
Experimental Results

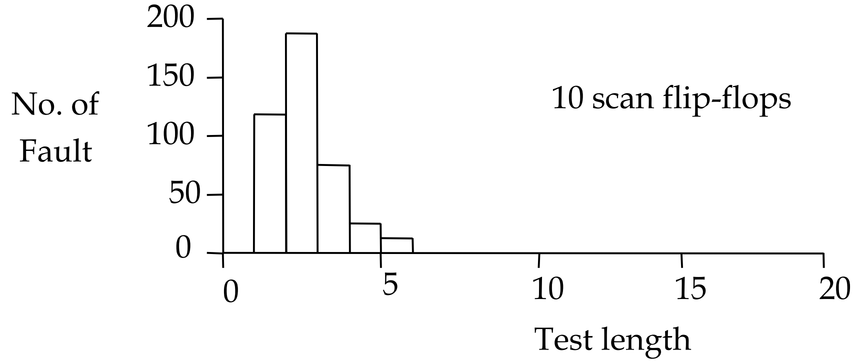
- Test case: TLC circuit
  - Gate count 355
  - Flip-flop count 21

No. of Scan Flip-Flops	Max. Cycle Length	Depth	Test Gen. CPU Sec.	Fault Sim. CPU Sec.	Fault Coverage	No. Of Test	Total Vectors
0	4	14	1247	61	89.01%	805	805
4	2	10	157	11	95.90%	247	988
9	1	5	32	4	99.20%	136	1224
10	1	3	13	4	100%	112	1120
21	0	0	2	2	100%	52	1092

Test Length Statistics For The TLC Circuit

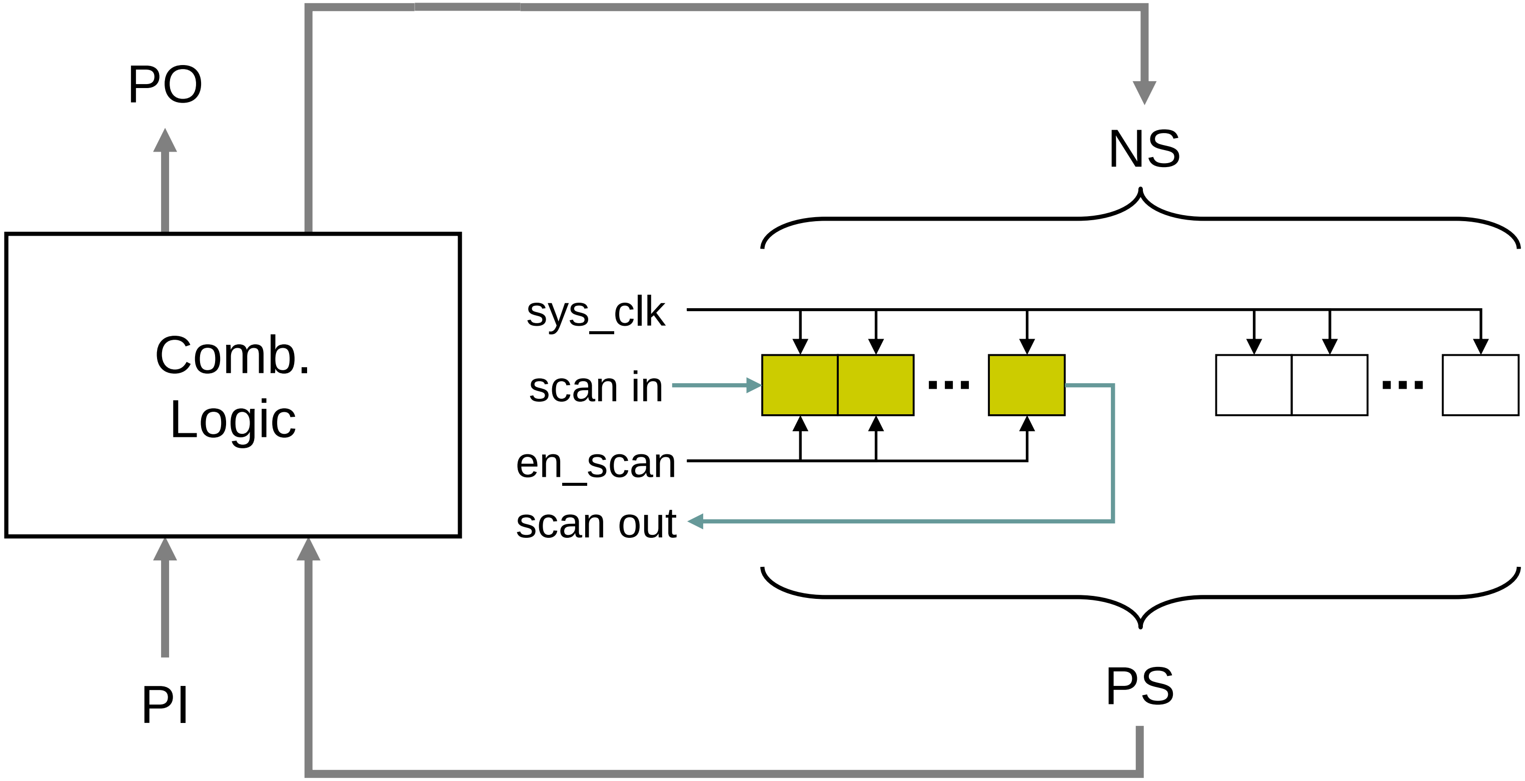






Clocking Schemes for Partial Scan Circuits

- Scheme III:
  - Using the system clock as a scan clock but without gating the the clock



Using System Clock for Scan Operation

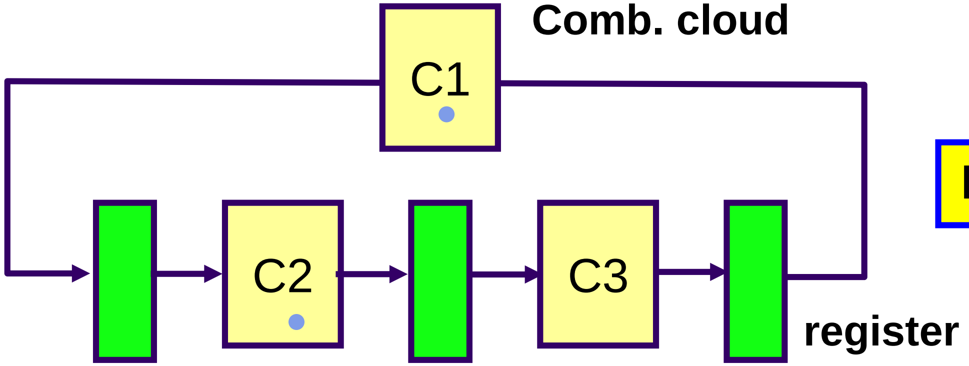
- The contents of the non-scan FFs may change during the scan operations.
  - Test generation process is more complicated.
  - The fault coverage may be slightly lower than that of the two-clock partial scan designs.
- The total test length (including scan sequences) is usually shorter.

Timing-Driven Partial Scan

- Aim at reducing both area overhead and performance degradation caused by test logic.
- Timing analysis data can be used to guide SFF selection.
  - Avoid selecting FFs on critical paths.
- Can be incorporated into existing logic synthesis systems to satisfy or trade-off design constraints in terms of area, performance, and testability.

BALLAST – A Structured Partial Scan Design

- BALLAST (Gupta et al. 1989)
  - Stands for "Balanced Structure Scan Test"
- B-Structure
  - Definition: A synchronous sequential circuit S is said to be balanced, denoted as B-structure, if for any two combinational clouds C1 and C2 in S, all signal paths (if any) between C1 and C2 go through the same number of registers
  - The above definition implies acyclic structure



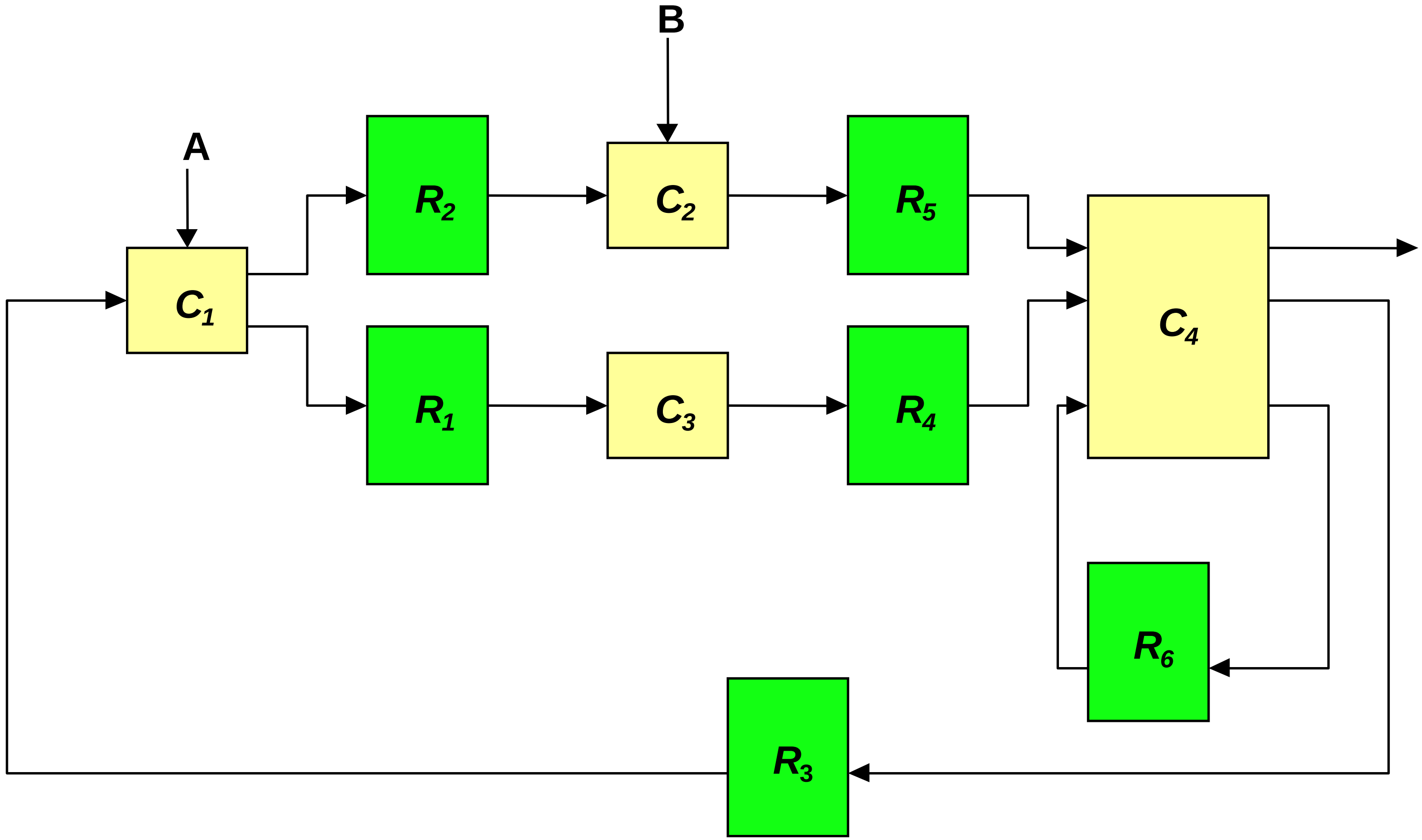
Not B-structure !

Example: A Sequential Circuit

Combinational clouds: C1, C2, C3, C4

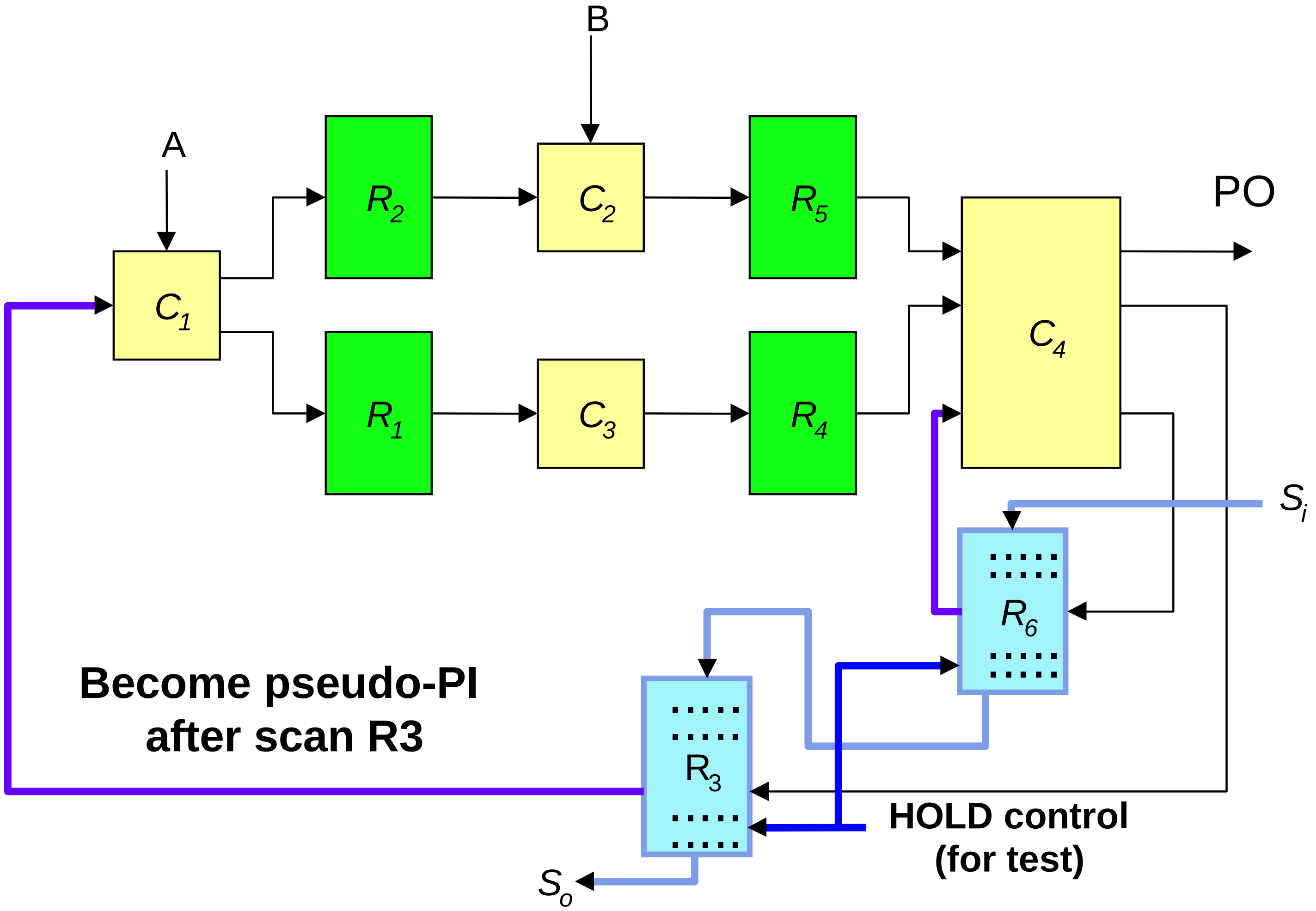
Registers: R1, R2, R3, R4, R5, R6

This example is not balanced !



BALLAST-based Partial Scan

This circuit becomes balanced after scanning registers R3 and R6



Become pseudo-PI
after scan R3

HOLD control
(for test)

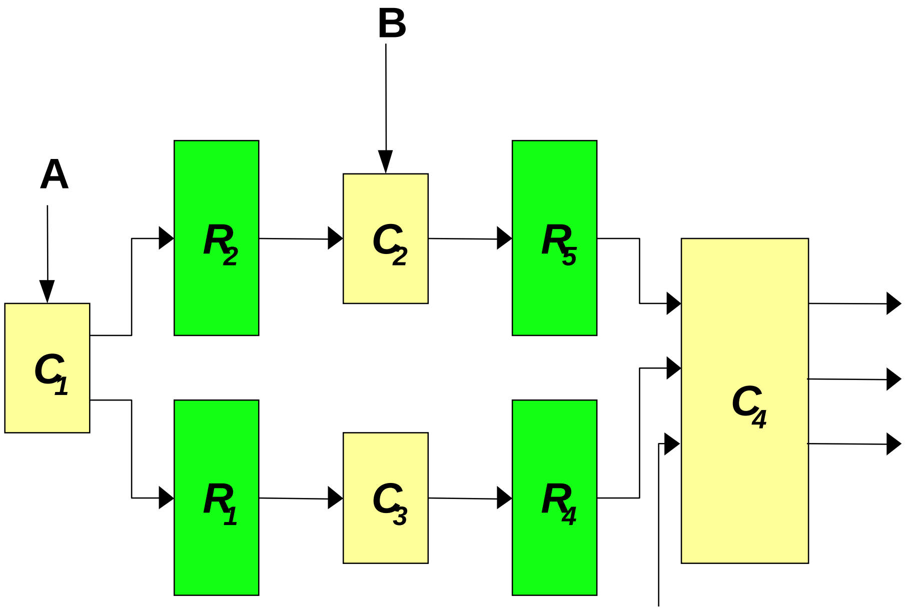
Test Procedure for B-Structure

- Depth of a B-structure
  - The largest number of registers on any path between any two combinational clouds
- Test Procedure
  - Step 1: Scan in the test pattern for scan flip-flops
  - Step 2: Apply primary input pattern
  - Step 3: Clock the registers *d* times (where *d* is the depth), while holding patterns at PI and scan flip-flops
  - Step 4: Place the scan flip-flops in normal mode for one clock (capture results into scaned FFs)
  - Step 5: Observe the primary output response
  - Step 6: Simultaneously scan out the results in the scan paths and scan in next scan pattern

Advantage of BALLAST

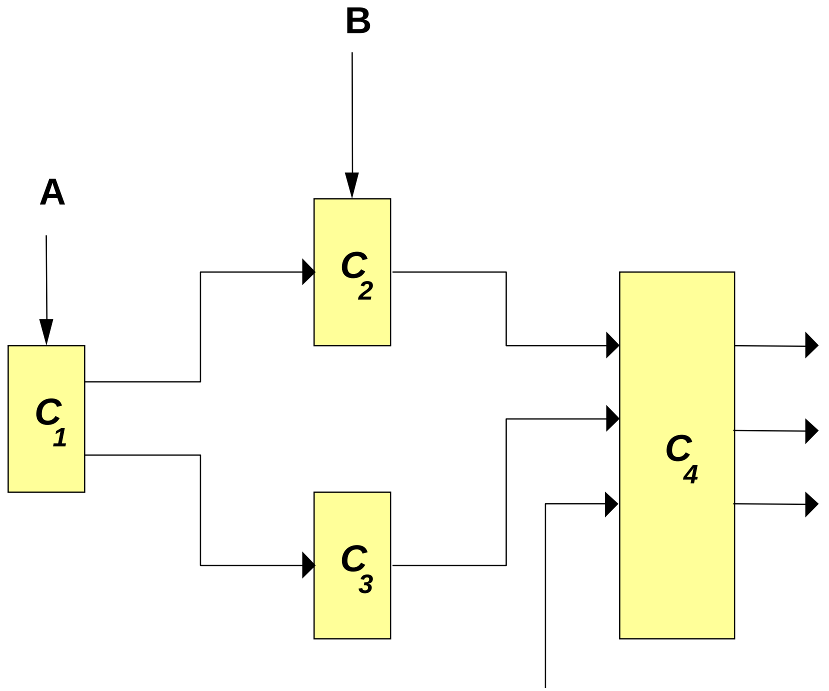
- The ATPG complexity for balanced circuits
  - Is reduced to a combinational one

A balanced circuit

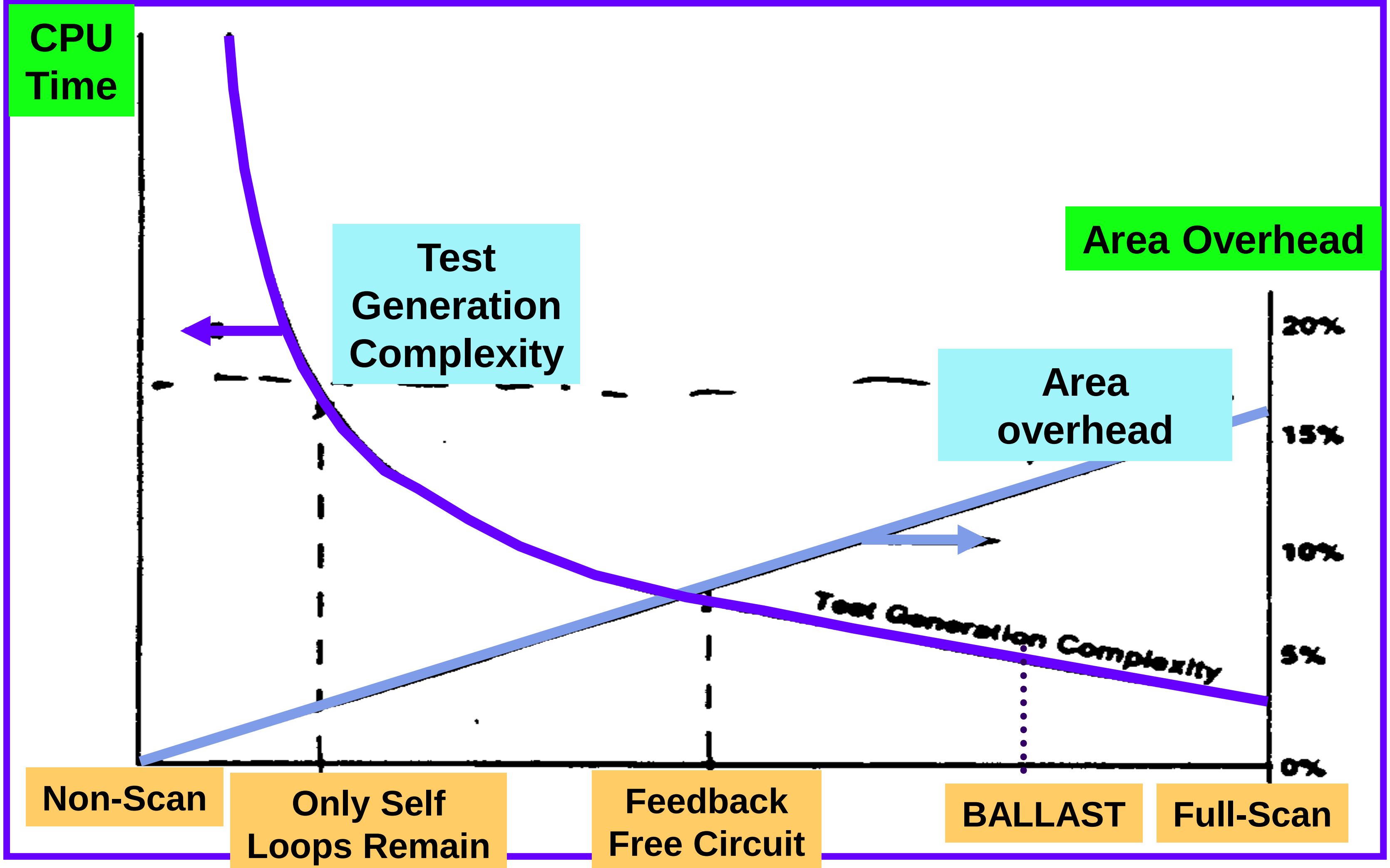


Depth = 2

Combinational equivalent for ATPG after replacing registers with wires



Trade-Off of Area Overhead v.s. Test Generation Effort



Summary

- Partial Scan
  - Allows the trade-off between test generation effort and hardware overhead to be automatically explored
- Breaking Cycles
  - Dramatically simplifies the sequential ATPG
- Limiting The Length of Self-Loop Paths
  - Is crucial in reducing test generation effort for large circuits
- Performance Degradation
  - Can be minimized by using timing analysis data for flip-flop selection