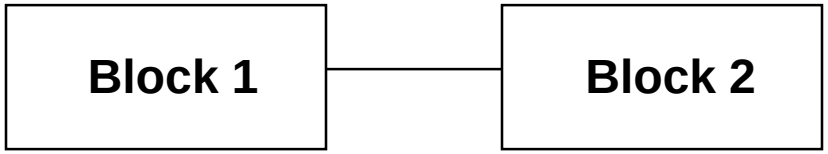
Design for Testability

Outline

- Ad Hoc Design for Testability Techniques
  - Method of test points
  - Multiplexing and demultiplexing of test points
  - Time sharing of I/O for normal working and testing modes
  - Partitioning of registers and large combinational circuits
- Scan-Path Design
  - Scan-path design concept
  - Controllability and observability by means of scan-path
  - Full and partial serial scan-paths
  - Non-serial scan design
  - Classical scan designs

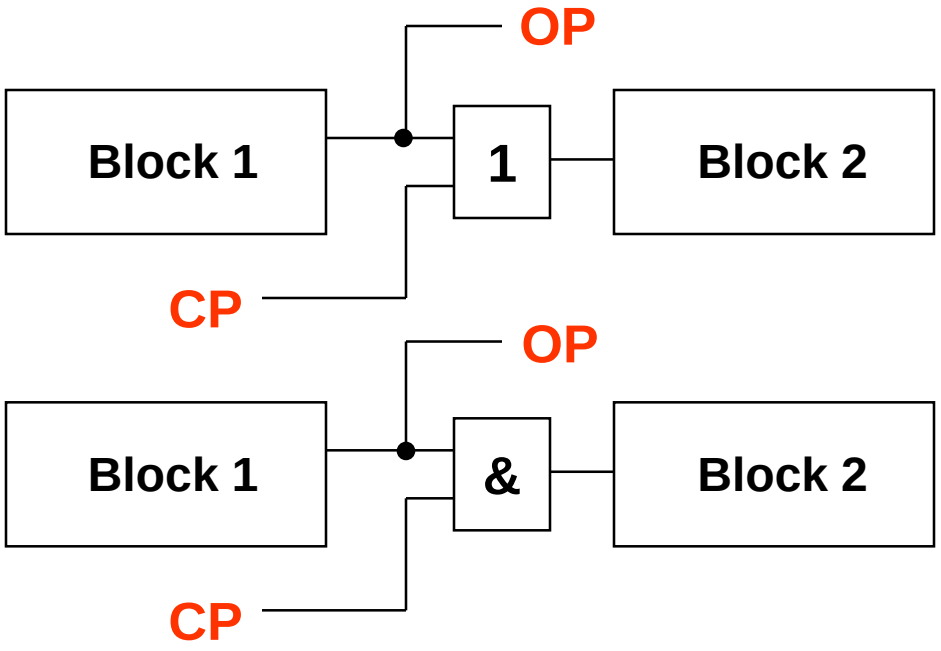
Ad Hoc Design for Testability Techniques

Method of Test Points:



Block 1 is not observable,
Block 2 is not controllable

Improving controllability and observability:

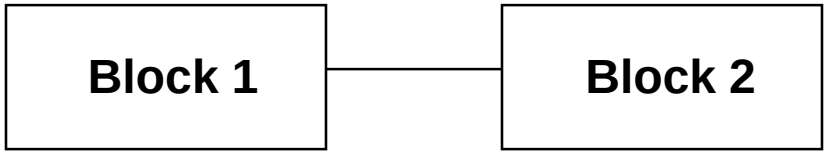


1- controllability:
CP = 0 - normal working mode
CP = 1 - controlling Block 2 with signal 1

0- controllability:
CP = 1 - normal working mode
CP = 0 - controlling Block 2 with signal 0

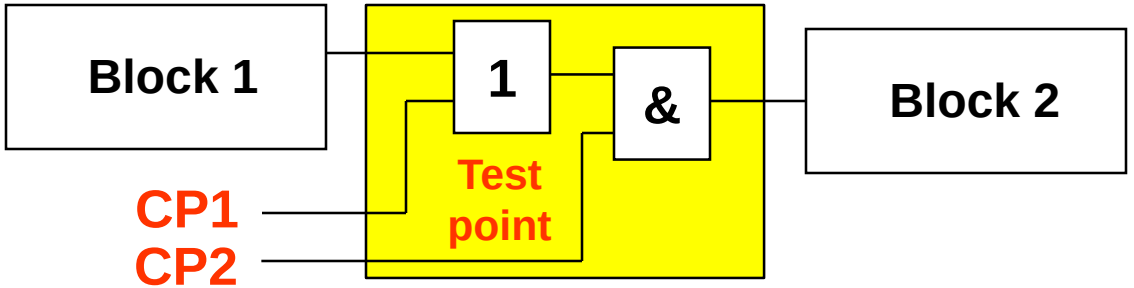
Ad Hoc Design for Testability Techniques

Method of Test Points:



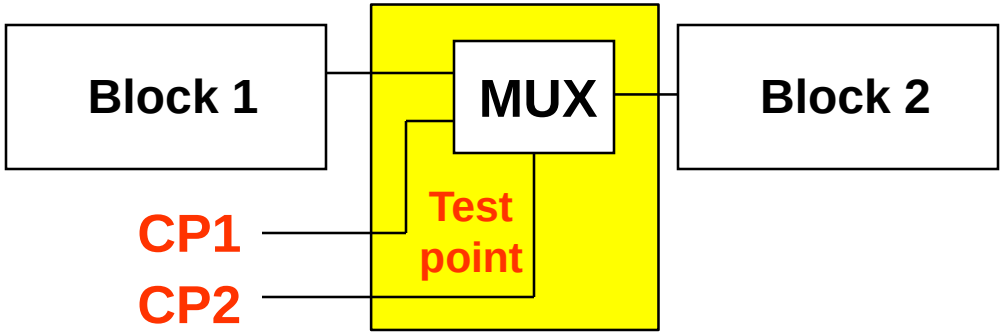
Block 1 is not observable,
Block 2 is not controllable

Improving controllability:



Normal working mode:
CP1 = 0, CP2 = 1

Controlling Block 2 with 1:
CP1 = 1, CP2 = 1
Controlling Block 2 with 0:
CP2 = 0



Normal working mode:
CP2 = 0

Controlling Block 2 with 1:
CP1 = 1, CP2 = 1
Controlling Block 2 with 0:
CP1 = 0, CP2 = 1

Ad Hoc Design for Testability Techniques

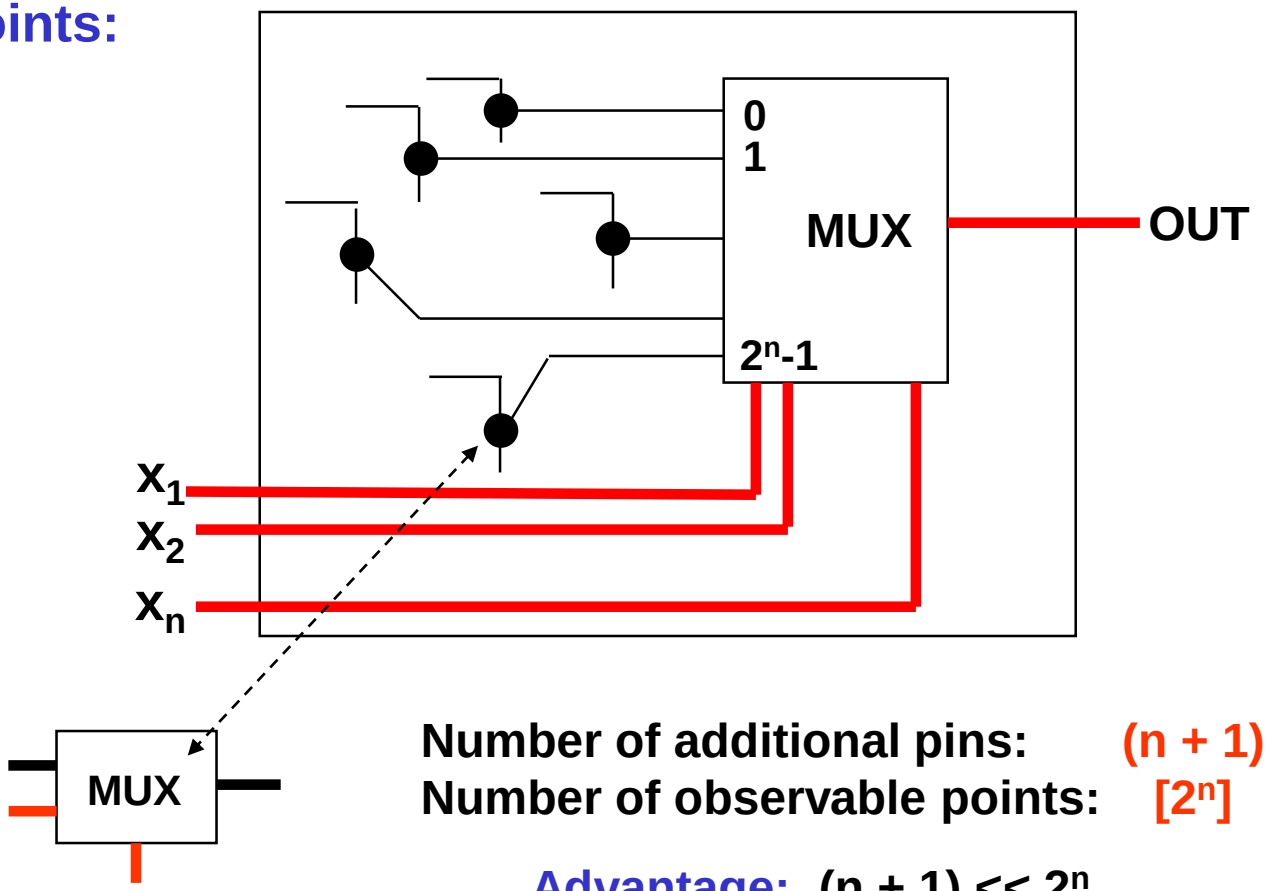
Multiplexing monitor points:

To reduce the number of output pins for observing monitor points, multiplexer can be used:

2^n observation points are replaced by a single output and n inputs to address a selected observation point

Disadvantage:

Only one observation point can be observed at a time



Number of additional pins: (n + 1)
Number of observable points: [2^n]

Advantage: (n + 1) << 2^n

Ad Hoc Design for Testability Techniques

Multiplexing monitor points:

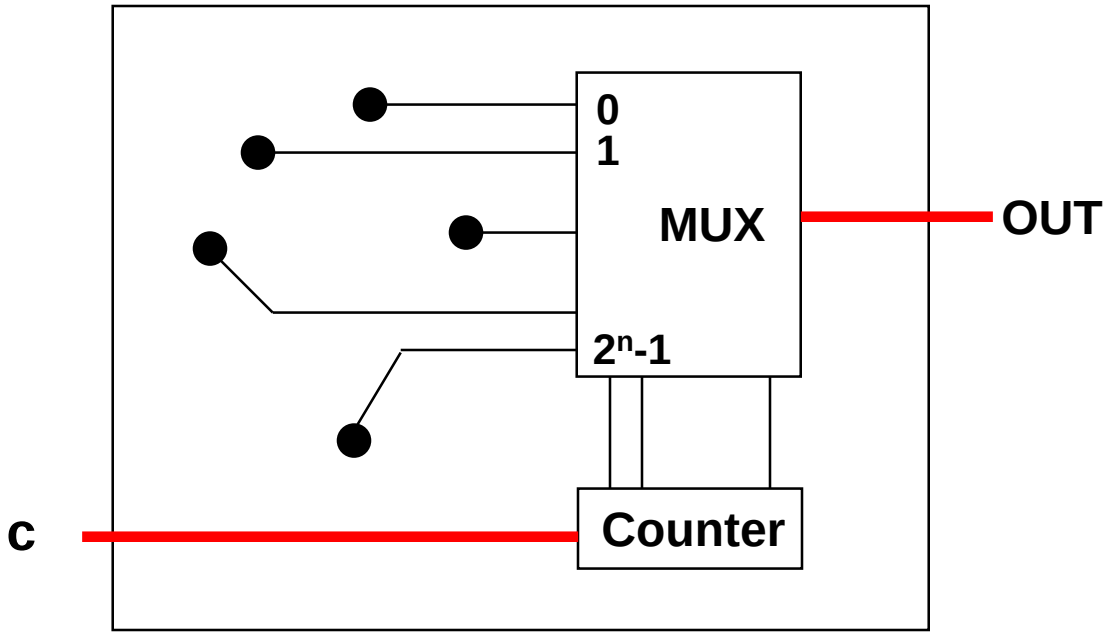
To reduce the number of output pins for observing monitor points, multiplexer can be used:

To reduce the number of inputs, a counter (or a shift register) can be used to drive the address lines of the multiplexer

Disadvantage:

Only one observation point can be observed at a time

Reset for counter**?**

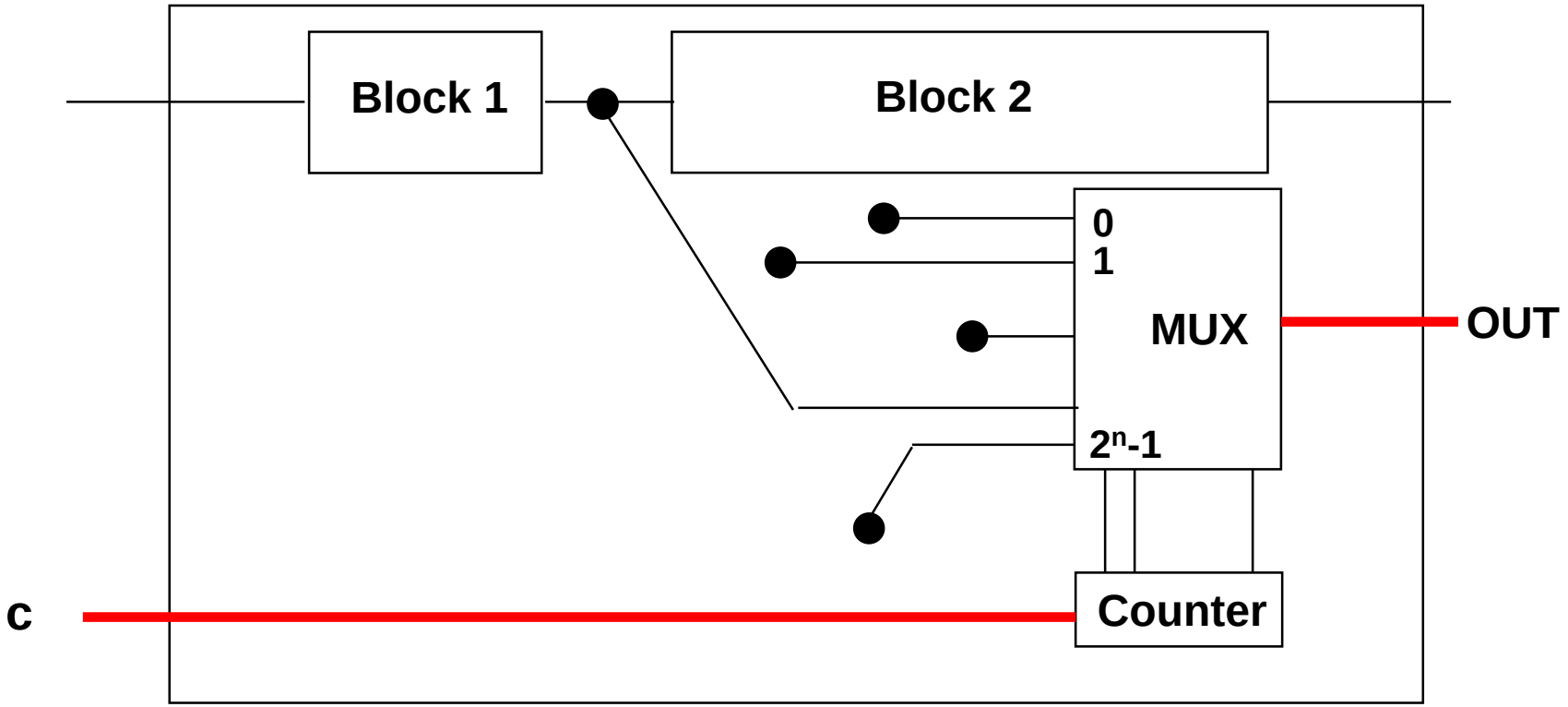


Number of additional pins: 2
Nmber of observable points: [2^n]

Advantage: $2 < n << 2^n$

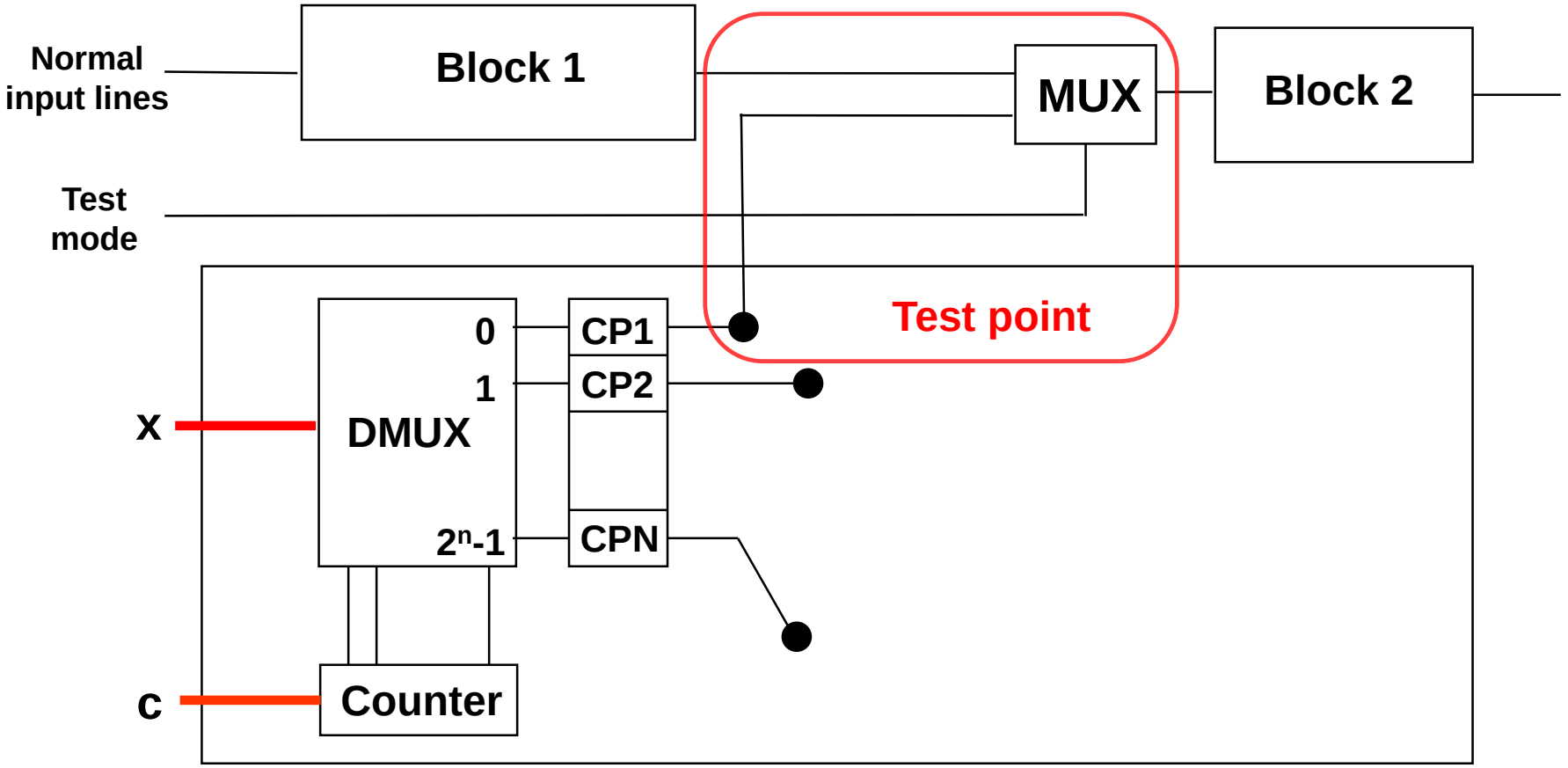
Ad Hoc Design for Testability Techniques

Multiplexing monitor points:



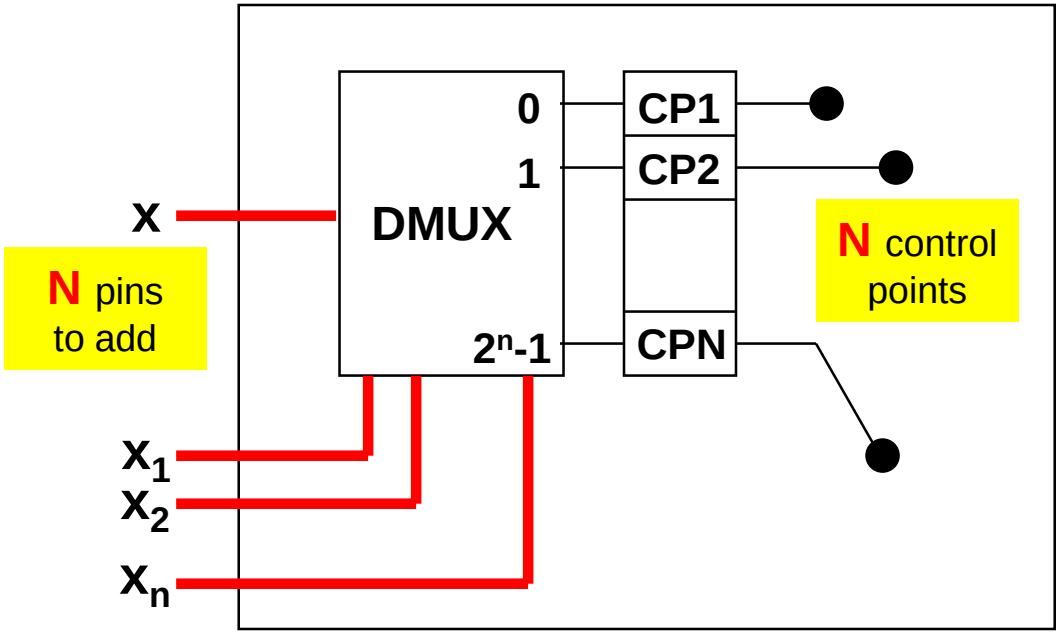
Ad Hoc Design for Testability Techniques

Demultiplexer for implementing control points:



Ad Hoc Design for Testability Techniques

Demultiplexer for implementing control points:



To reduce the number of input pins for controlling testpoints, demultiplexer and latch register are used

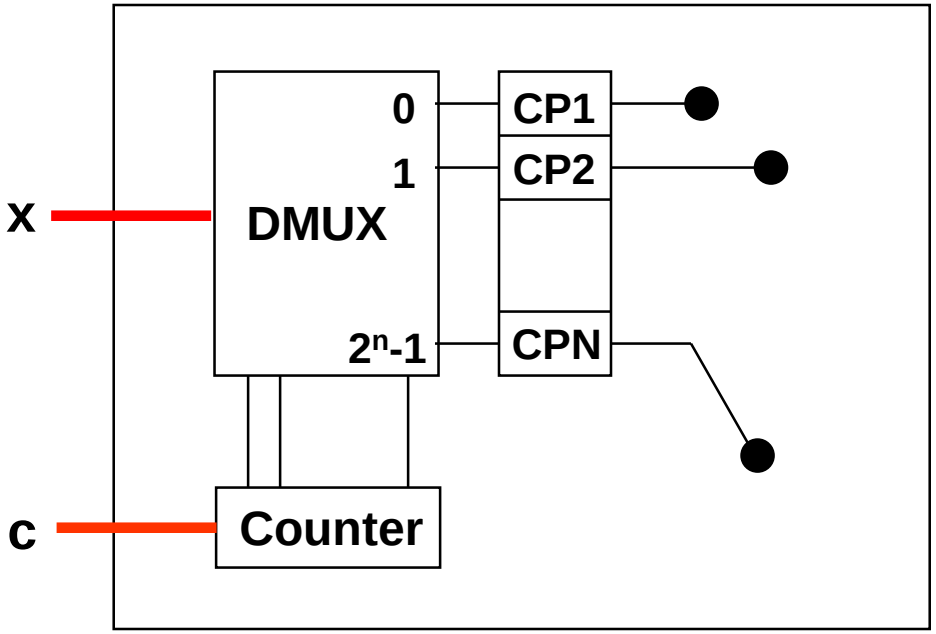
Number of additional pins: $(n + 1)$

Number of control points: $2^{n-1} < N \leq 2^n$

Advantage: $(n + 1) << N$

Ad Hoc Design for Testability Techniques

Demultiplexer for implementing control points:



Number of additional pins: 2
Number of control points: N

Advantage: 2 << N

To reduce the number of inputs for addressing, a counter (or a shift register) can be used to drive the address lines of the demultiplexer

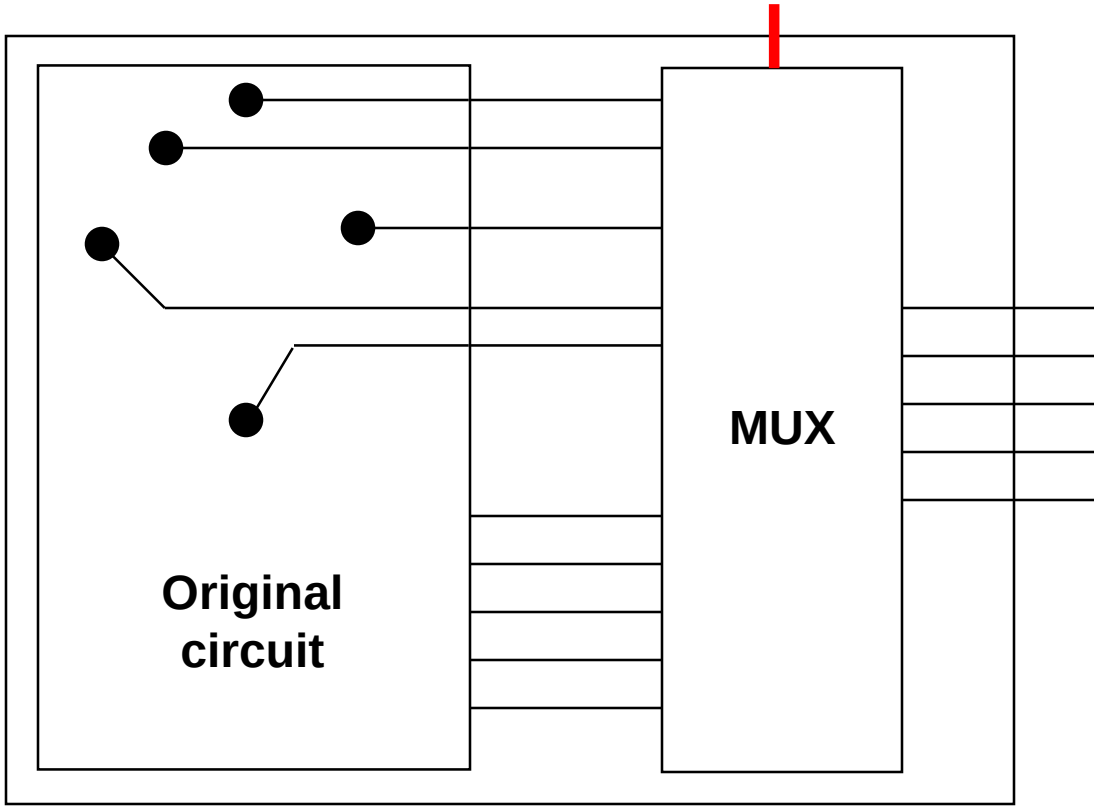
Disadvantage:

N clock times are required between test vectors to set up the proper control values

Time-sharing of outputs for monitoring

To reduce the number of output pins for observing monitor points, time-sharing of working outputs can be introduced: no additional outputs are needed

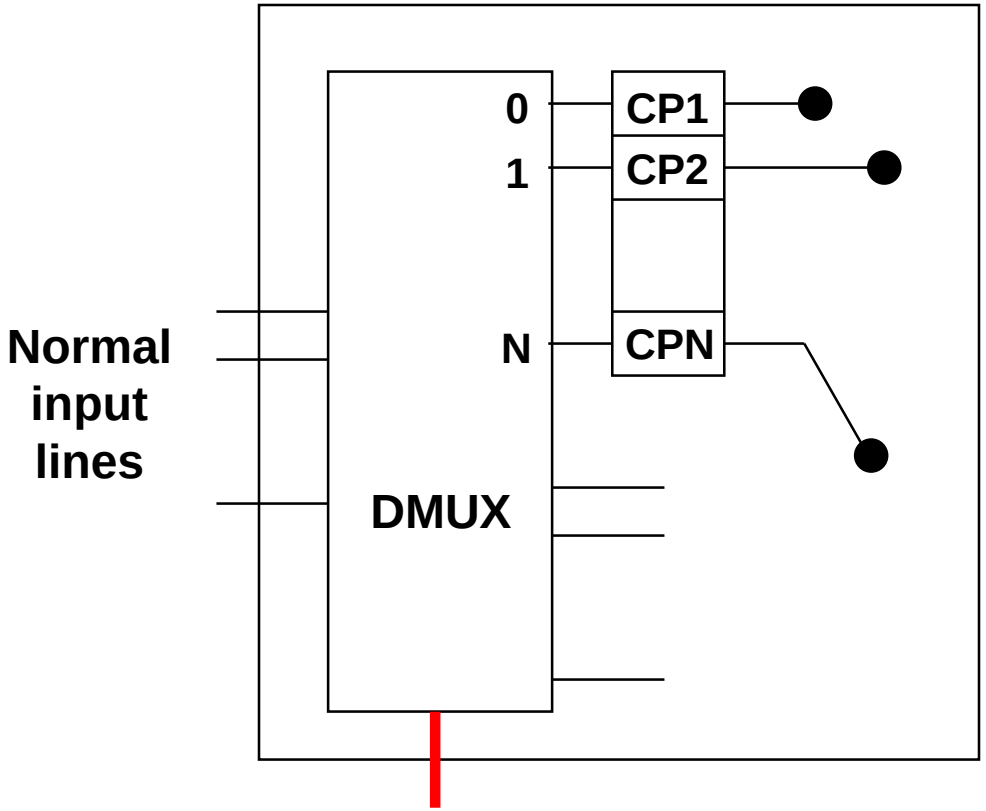
To reduce the number of inputs, again counter or shift register can be used if needed



Number of additional pins: 1
Number of control points: N

Advantage: 1 << N
Test time decreases

Time-sharing of inputs for controlling



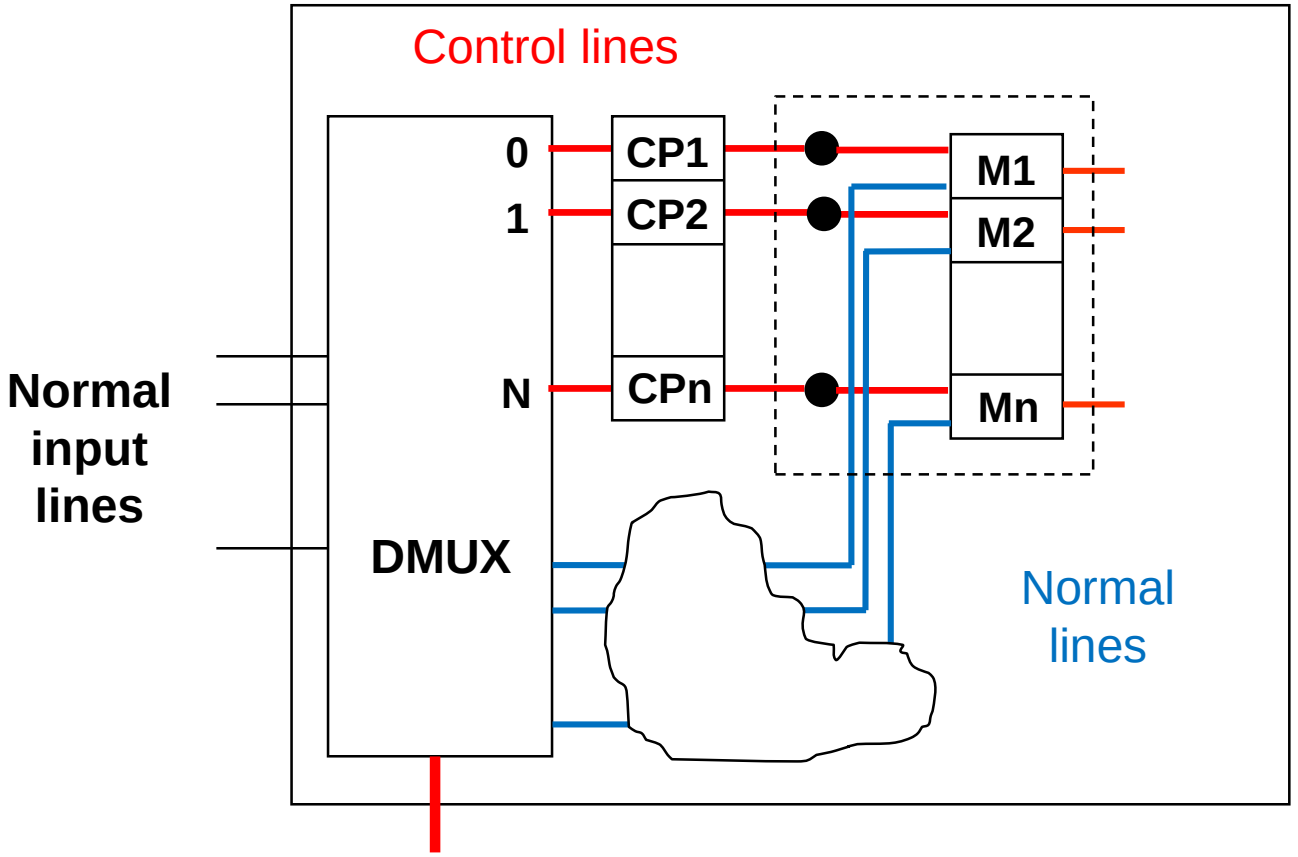
To reduce the number of input pins for controlling test points, time-sharing of working inputs can be introduced.

To reduce the number of inputs for driving the address lines of demultiplexer, counter or shift register can be used if needed

Number of additional pins: 1
Number of control points: N

Advantage: 1 < N
Test time decreases

Time-sharing of inputs for controlling

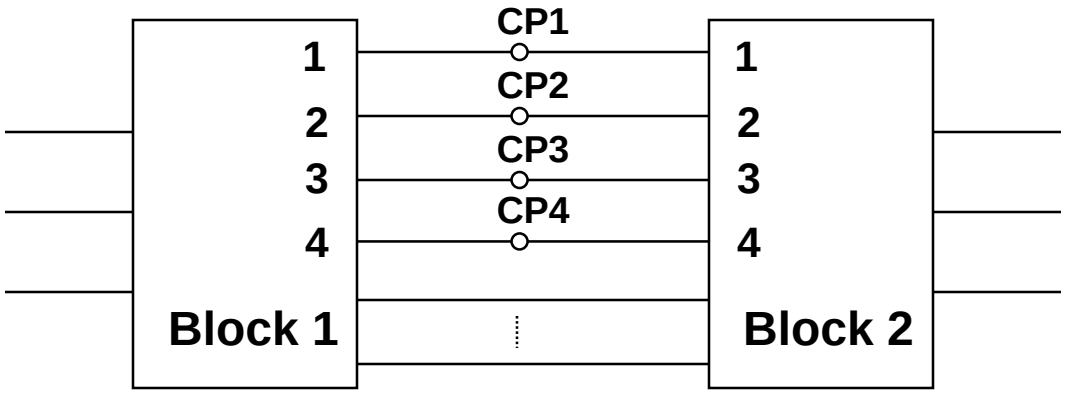


Example: DFT with MUX-s and DMUX-s

Given a circuit:

- CP1 and CP2 are not controllable
- CP3 and CP4 are not observable

DFT task: Improve the testability by using a single control input, no additional inputs/outputs allowed



Example: DFT with MUX-s and DMUX-s

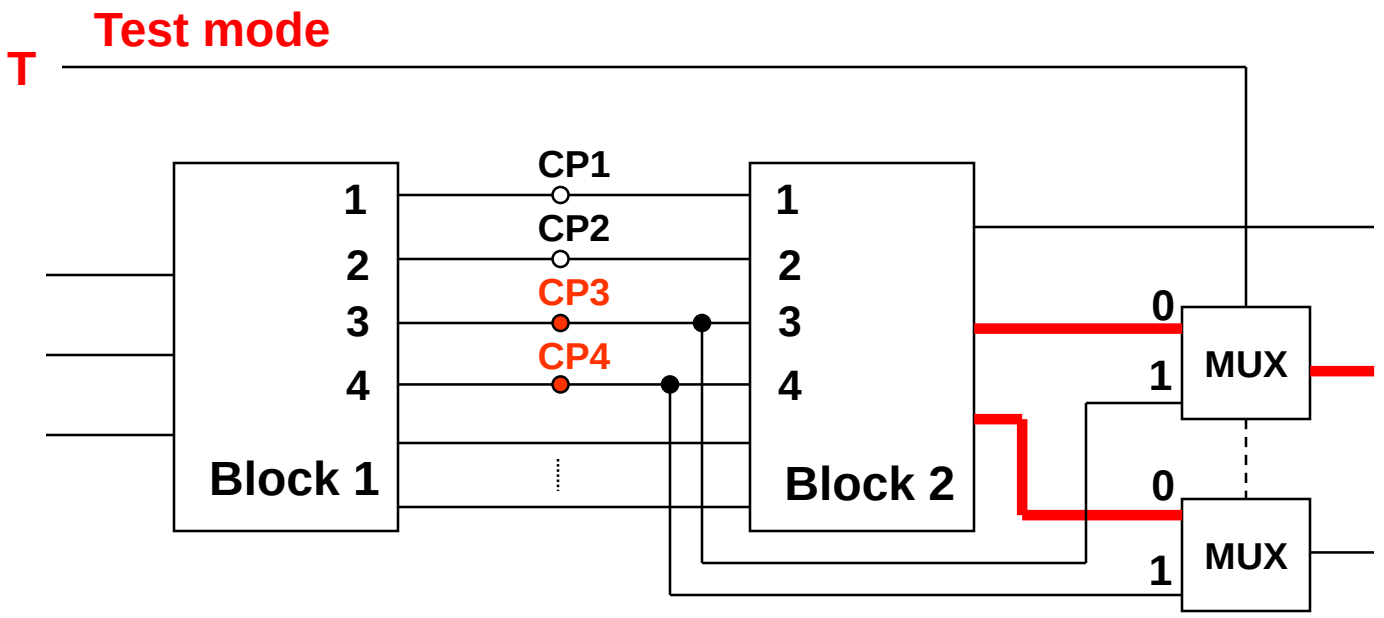
Given a circuit:

CP3 and CP4 are not observable

→ Improving the observability

Coding:

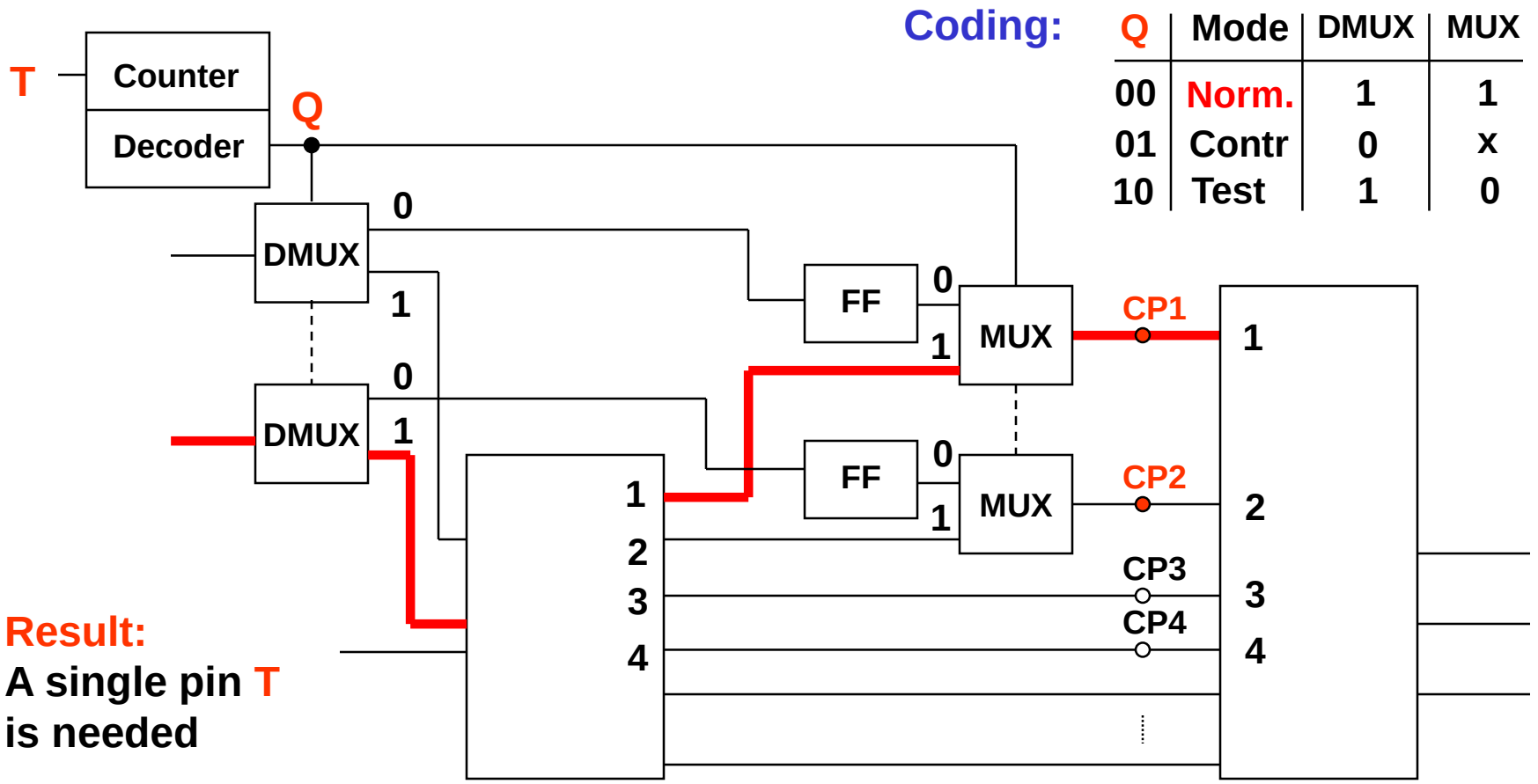
T	Mode	MUX
0	Norm.	0
1	Test	1



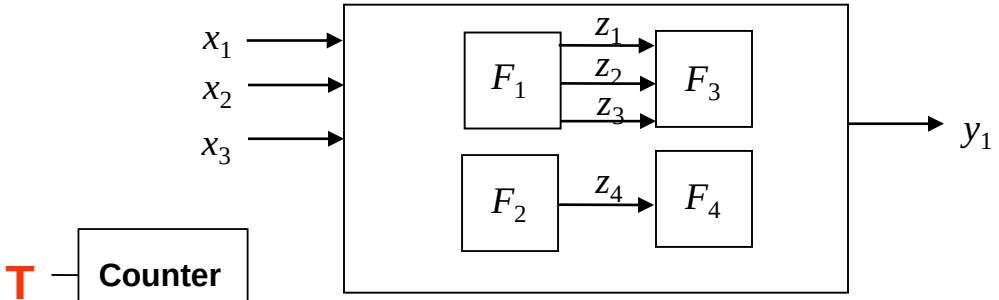
Result: A single pin T (test mode) is needed

Example: DFT with MUX-s and DMUX-s

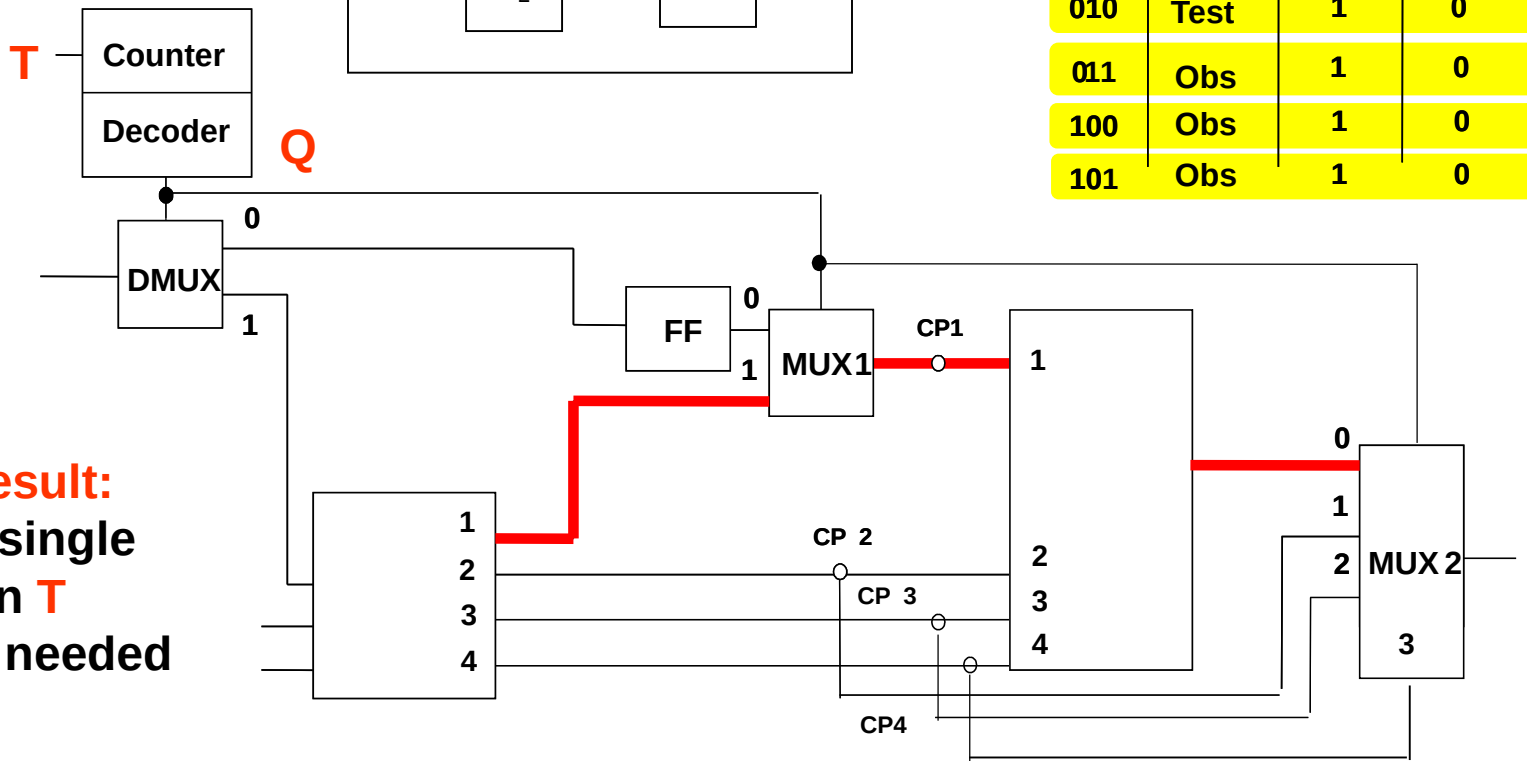
Given a circuit: CP1 and CP2 are not controllable → Improving the controllability



Example: DFT with MUX-s and DMUX-s



Q	Mode	DMUX	MUX 1	MUX 2
000	Norm	1	1	0
001	Contr	0	x	x
010	Test	1	0	0
011	Obs	1	0	1
100	Obs	1	0	2
101	Obs	1	0	3



Result:
A single pin T is needed

Ad Hoc Design for Testability Techniques

Examples of good candidates for control points:

- control, address, and data bus lines on bus-structured designs
- enable/hold inputs of microprocessors
- enable and read/write inputs to memory devices
- clock and preset/clear inputs to memory devices (flip-flops, counters, ...)
- data select inputs to multiplexers and demultiplexers
- control lines on tristate devices

Examples of good candidates for observation points:

- stem lines associated with signals having high fanout
- global feedback paths
- redundant signal lines
- outputs of logic devices having many inputs (multiplexers, parity generators)
- outputs from state devices (flip-flops, counters, shift registers)
- address, control and data busses

Ad Hoc Design for Testability Techniques

Logical redundancy:

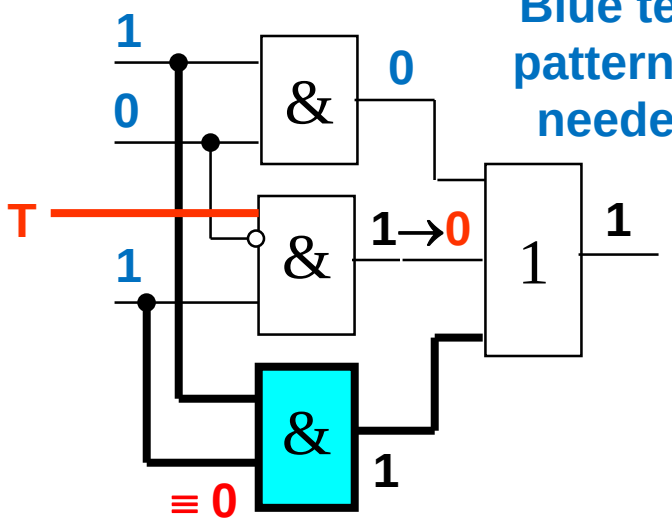
Redundancy should be avoided:

- If a redundant fault occurs, it may invalidate some test for nonredundant faults
- Redundant faults cause difficulty in calculating fault coverage
- Much test generation time can be spent in trying to generate a test for a redundant fault

Redundancy intentionally added:

- To eliminate hazards in combinational circuits
- To achieve high reliability (using error detecting circuits)

Hazard control circuitry:



Blue test pattern is needed

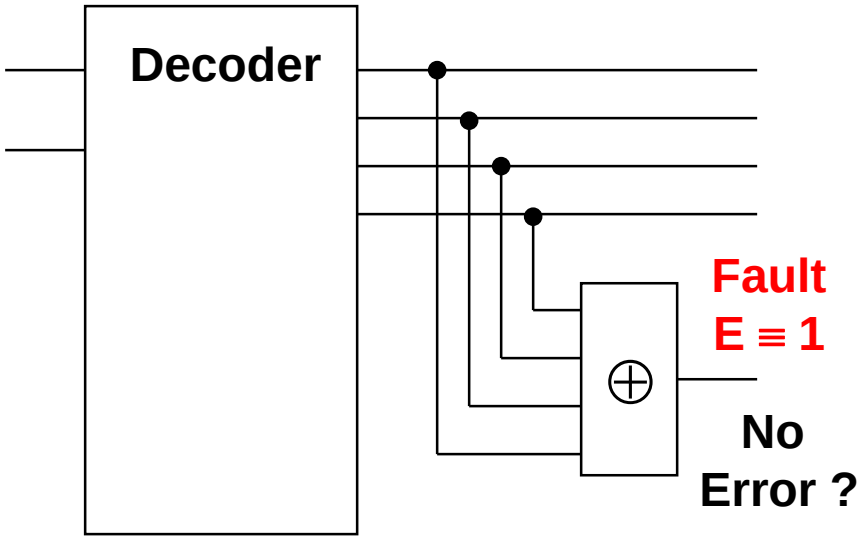
Redundant AND-gate
Fault ≡ 0 not testable

Additional control input added:
T = 1 - normal working mode
T = 0 - testing mode

Ad Hoc Design for Testability Techniques

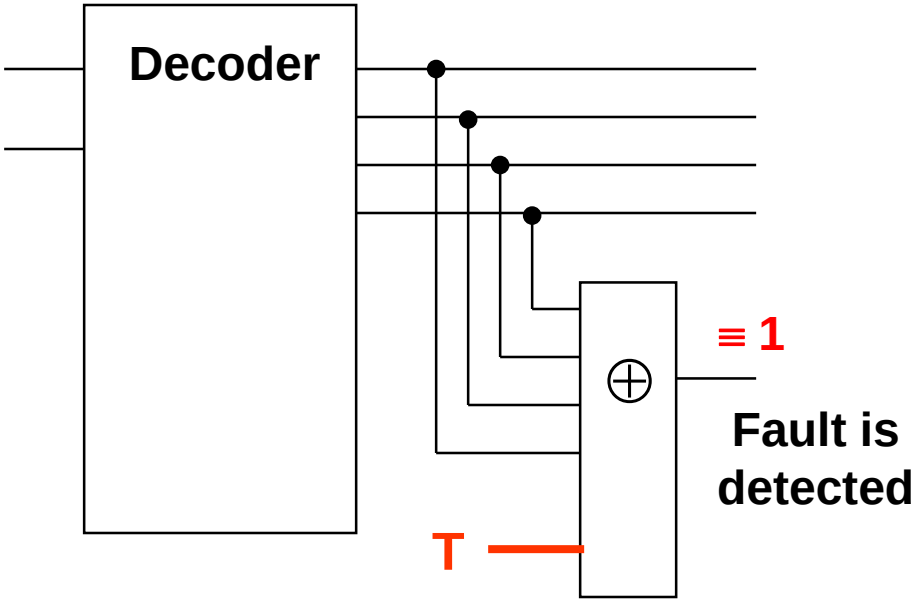
Fault redundancy:

Error control circuitry:



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

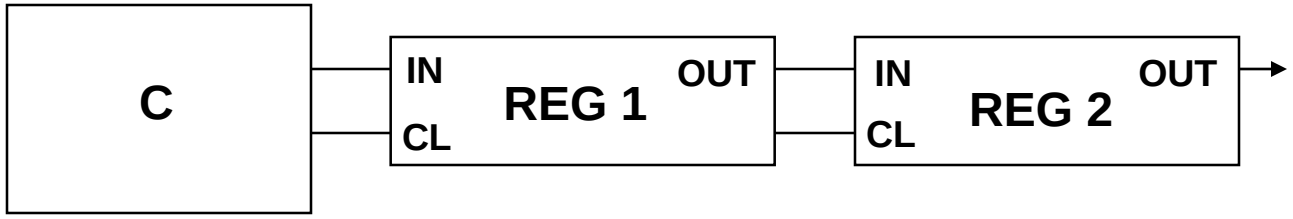
Testable error control circuitry:



Additional control input added:
T ≡ 0 - normal working mode
T = 1 - testing mode

Ad Hoc Design for Testability Techniques

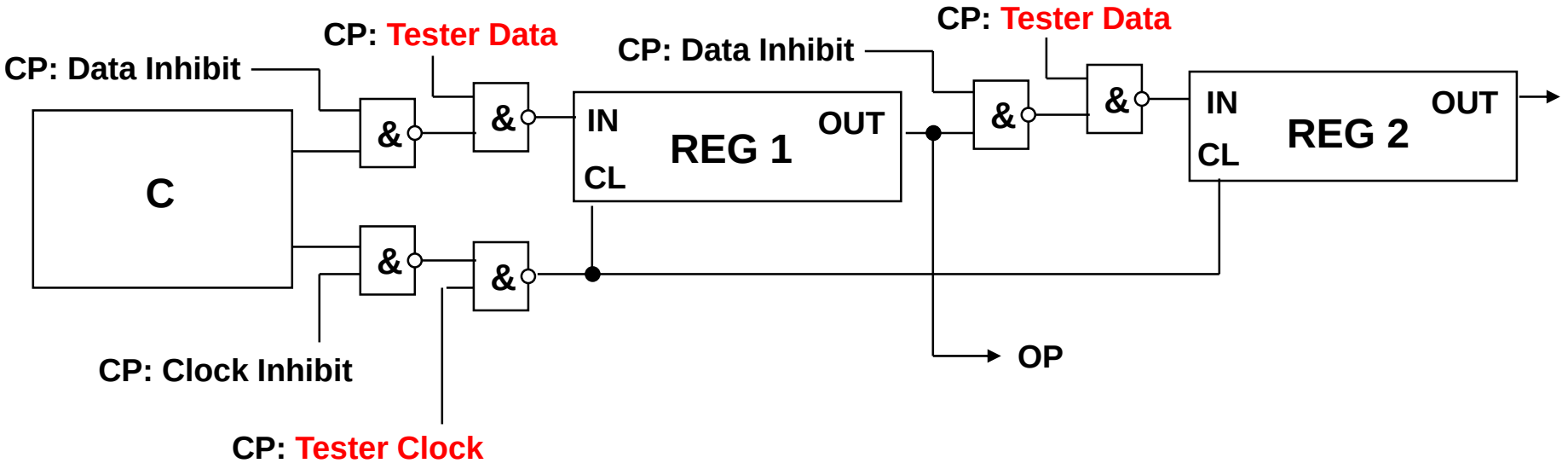
Partitioning of registers (counters):



16 bit counter divided into two 8-bit counters:

Instead of 2^{16} = 65536 clocks, $2x2^{8}$ = 512 clocks needed

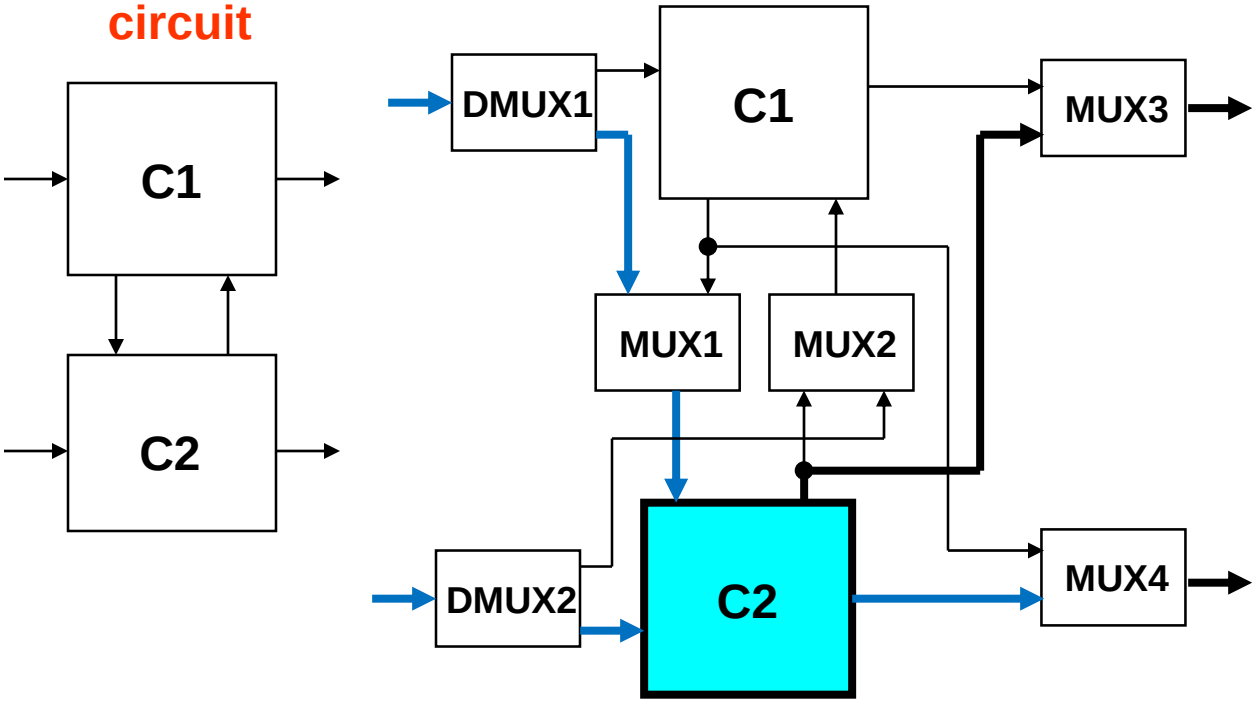
If tested in parallel, only 256 clocks needed



Ad Hoc Design for Testability Techniques

Partitioning of large combinational circuits:

Not testable circuit



The time complexity of test generation and fault simulation grows faster than a linear function of circuit size

Partioning of large circuits reduces the test cost

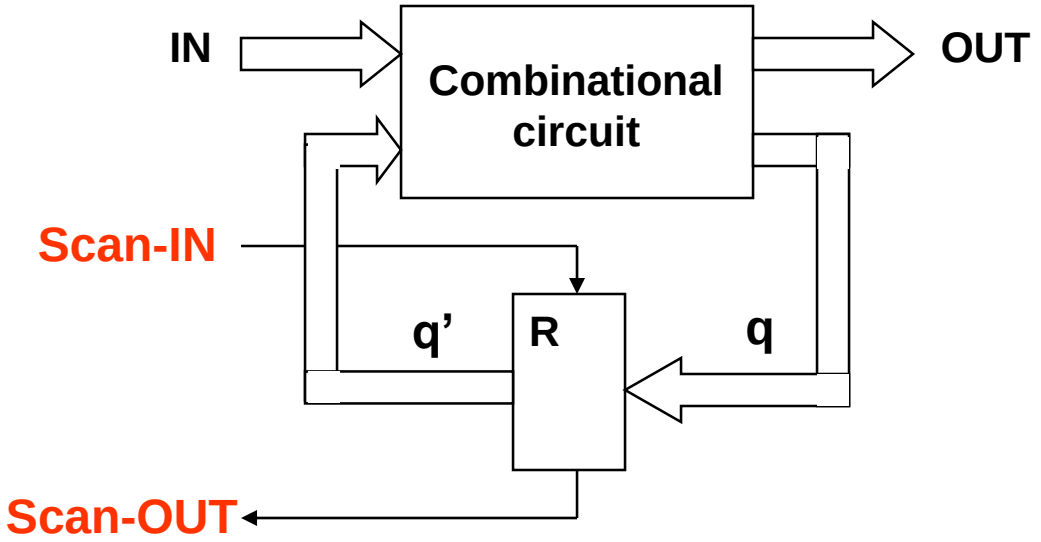
I/O sharing of normal and testing modes is used

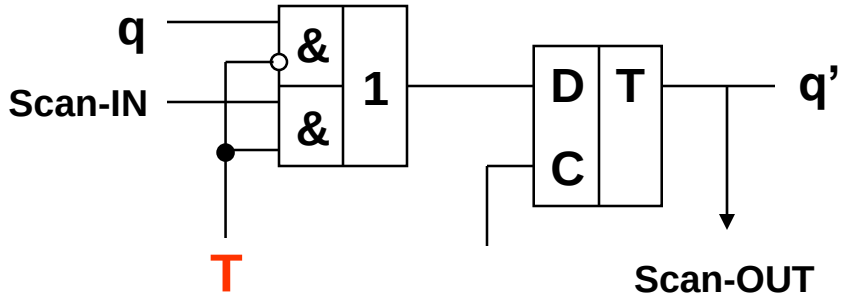
Three modes can be chosen:

- normal mode
- testing C1
- testing C2 (bolded blue lines)

How many additional inputs are needed?

Scan-Path Design





The complexity of testing is a function of the number of feedback loops and their length

The longer a feedback loop, the more clock cycles are needed to initialize and sensitize patterns

Scan-register is a aregister with both shift and parallel-load capability

T = 0 - normal working mode

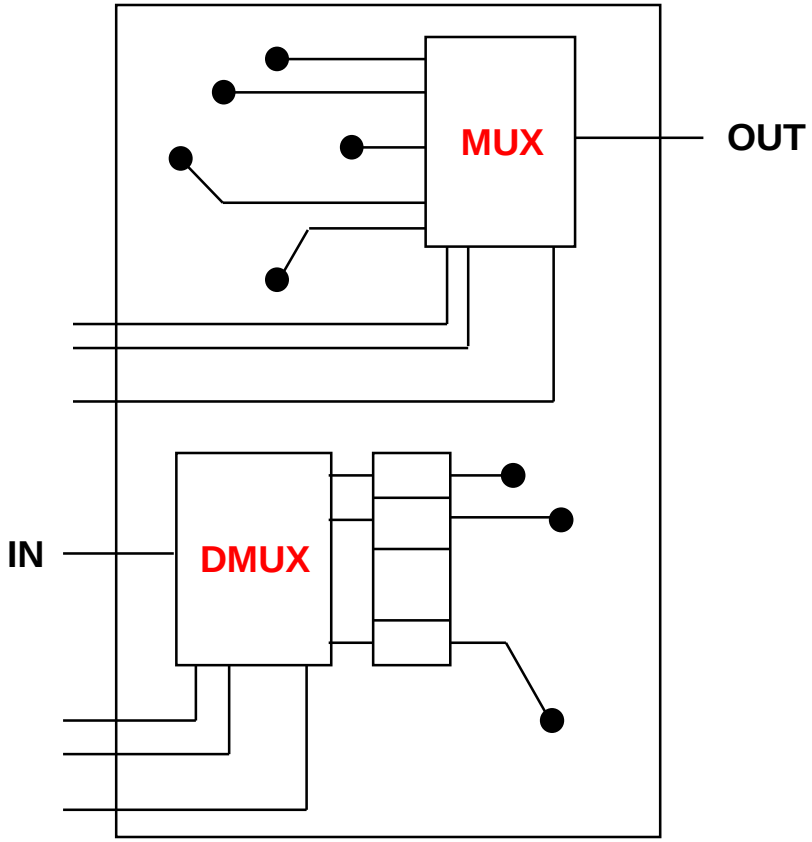
T = 1 - scan mode

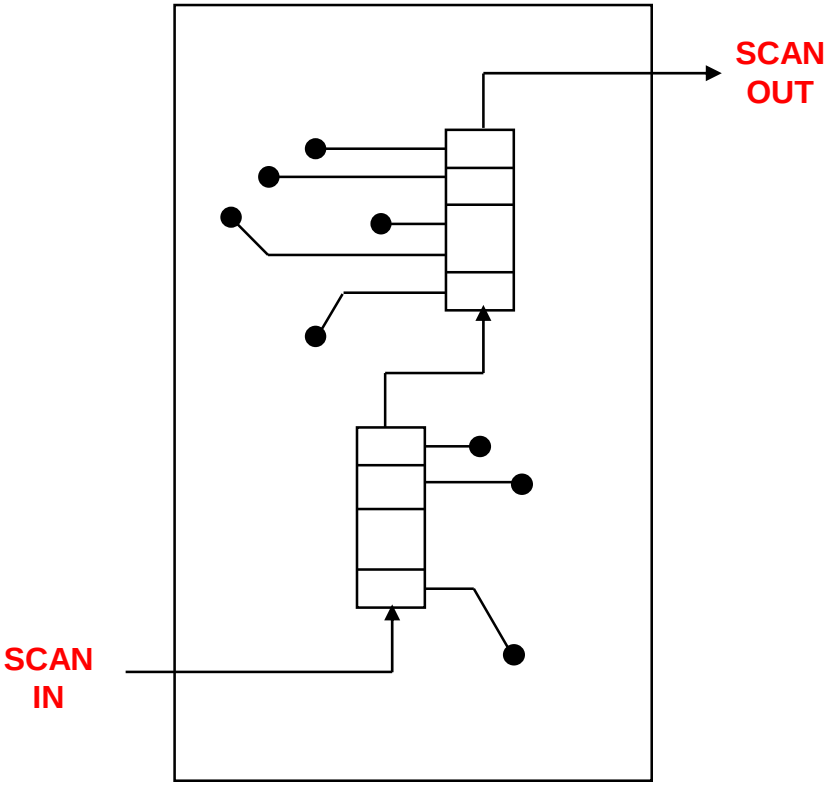
Normal mode : flip-flops are connected to the combinational circuit

Test mode: flip-flops are disconnected from the combinational circuit and connected to each other to form a shift register

Design for Testability & Control Points

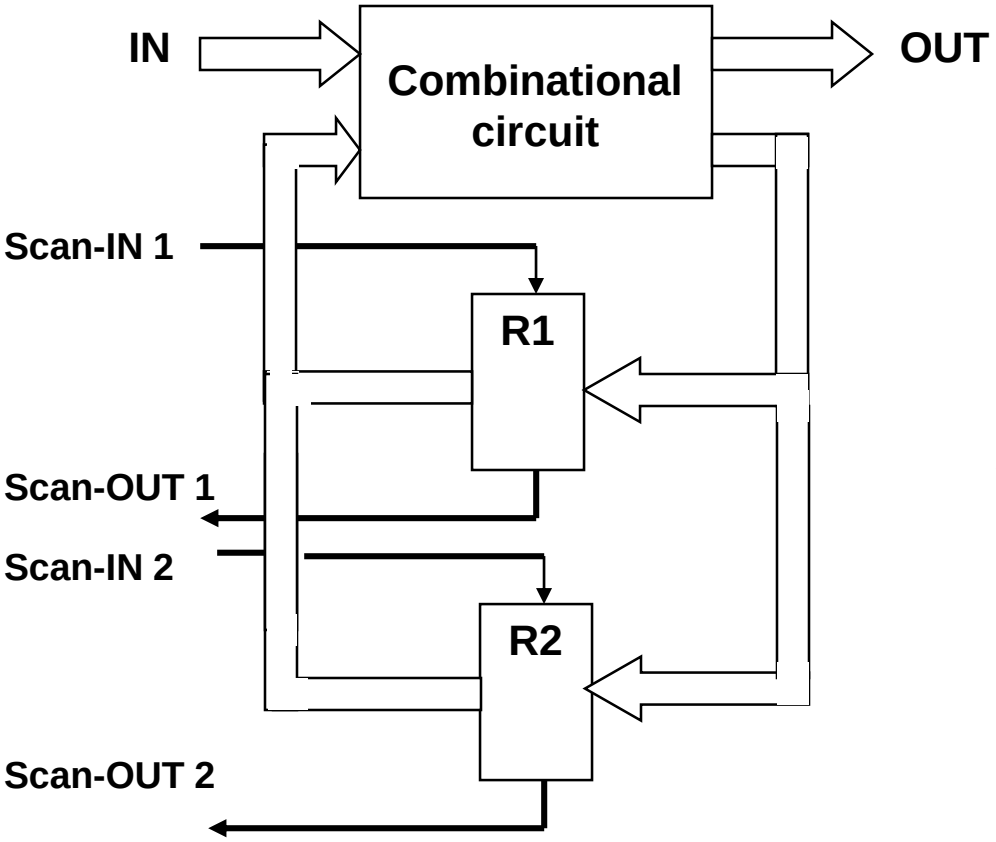
Two possibilities for improving controllability/observability





Two problems with CP-s: access and minimization

Parallel Scan-Path

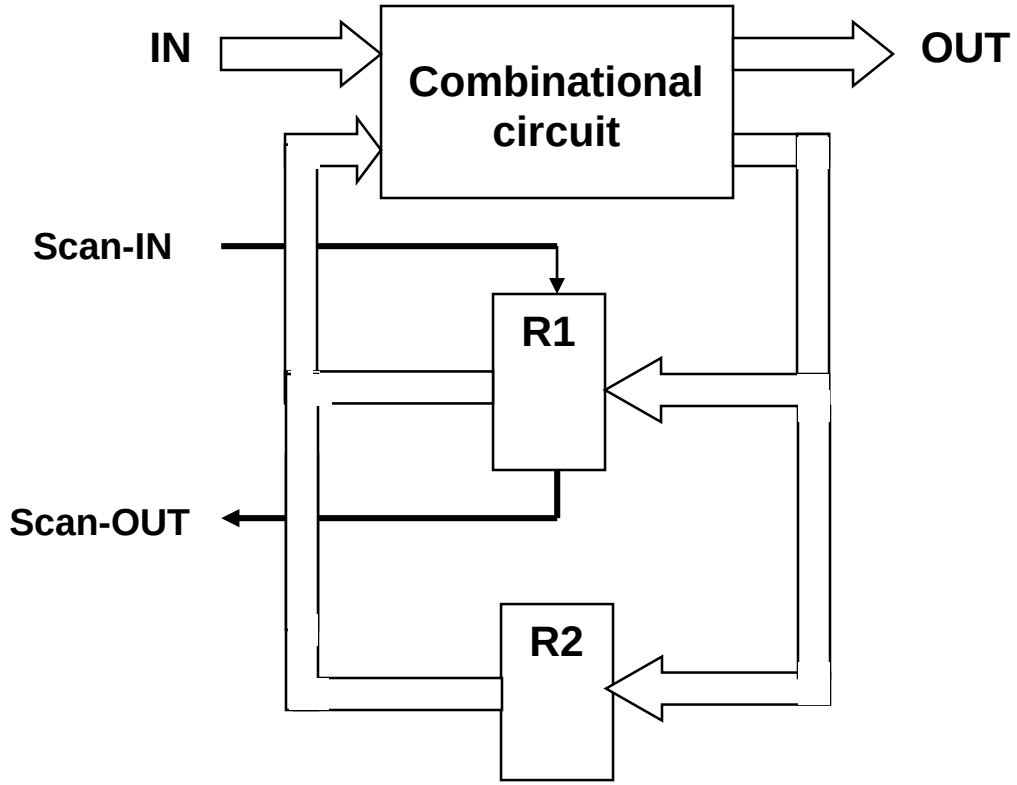


In parallel scan path flip-flops can be organized in more than one scan chain

Advantage: time ↓

Disadvantage: # pins ↑

Partial Scan-Path

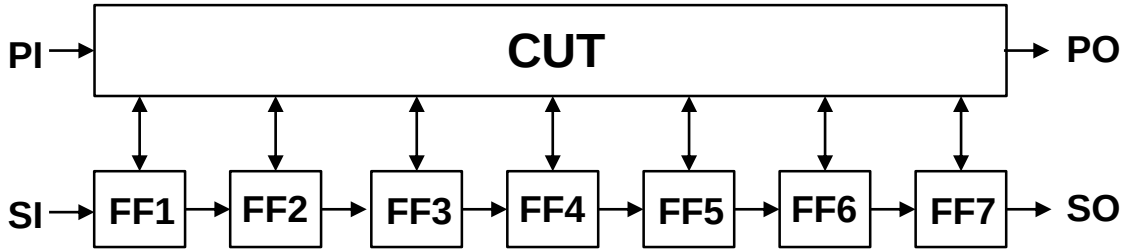


In **partial scan** instead of full–scan, it may be advantageous to scan **only some** of the flip–flops

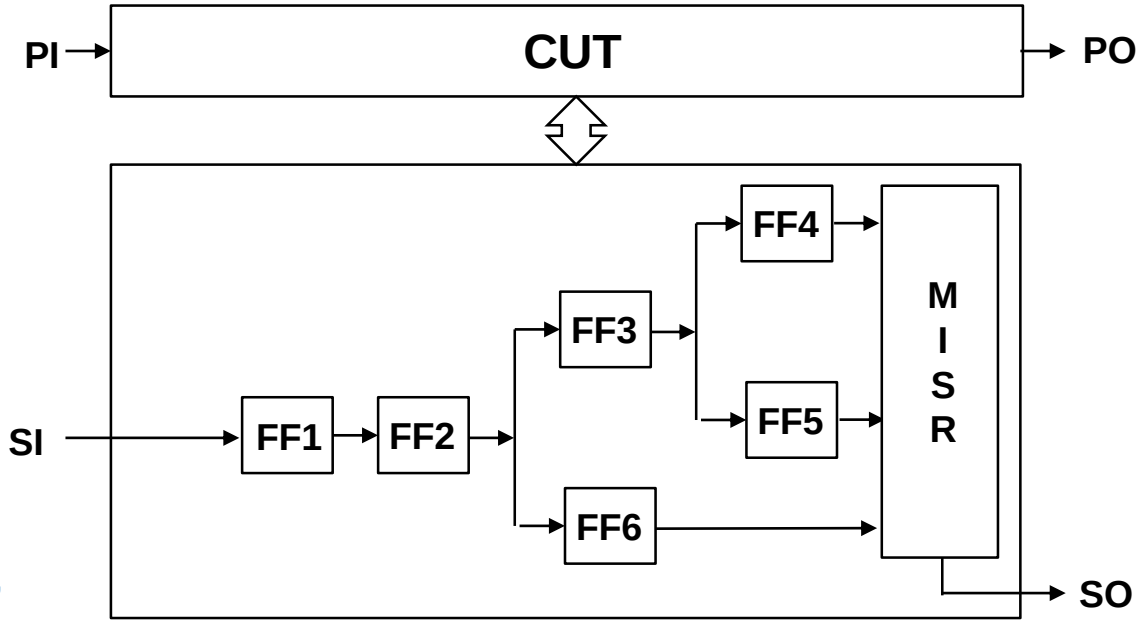
Example: **counter** – even bits joined in the scan–register

Linear Scan-Path vs Tree Architecture

Linear SCAN:



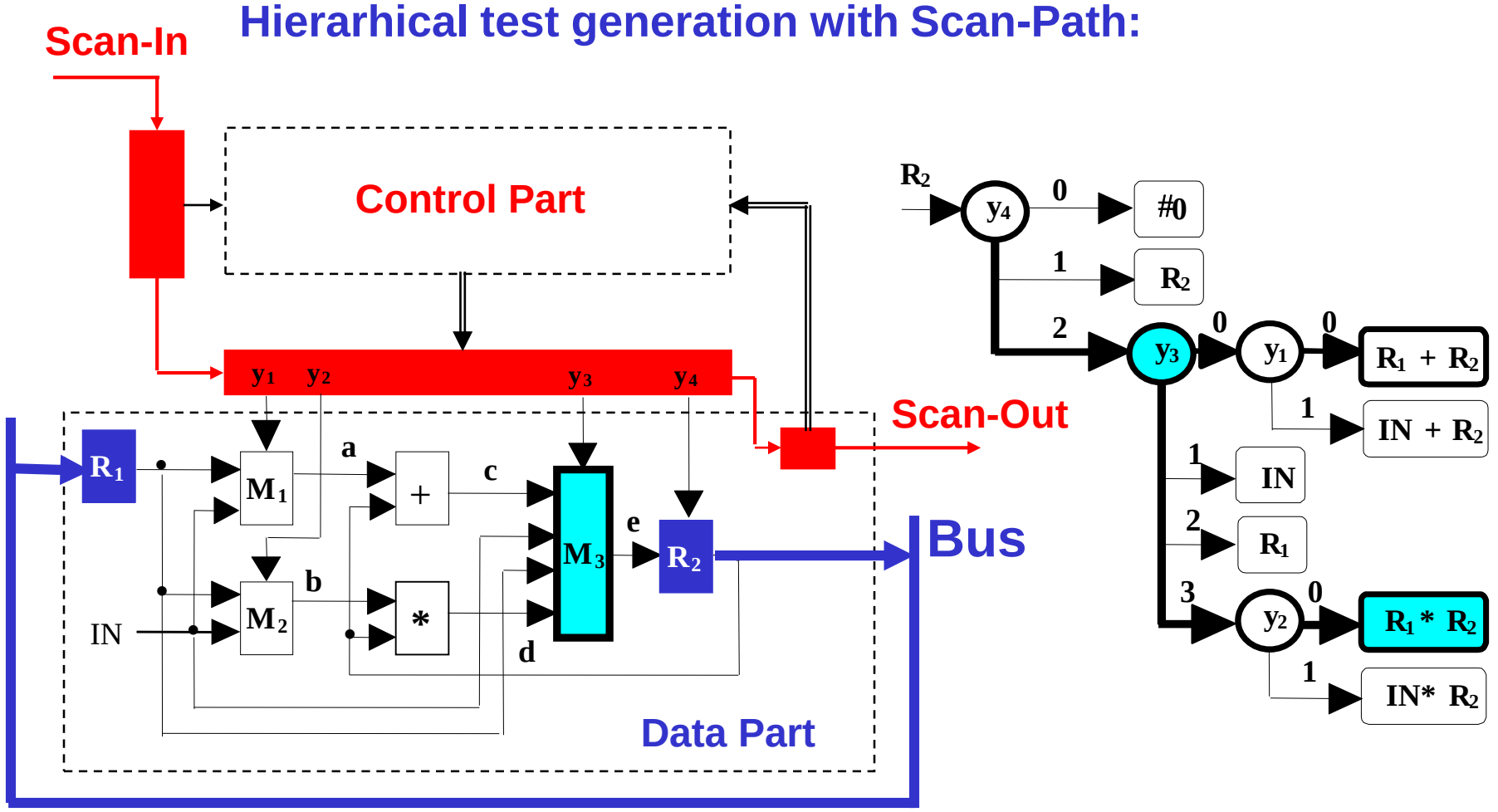
Tree architecture of SCAN:



L. Chen et al. „*Design of optimal scan tree based on compact Test patterns*“, IEEE Trans. on Comp., 2015

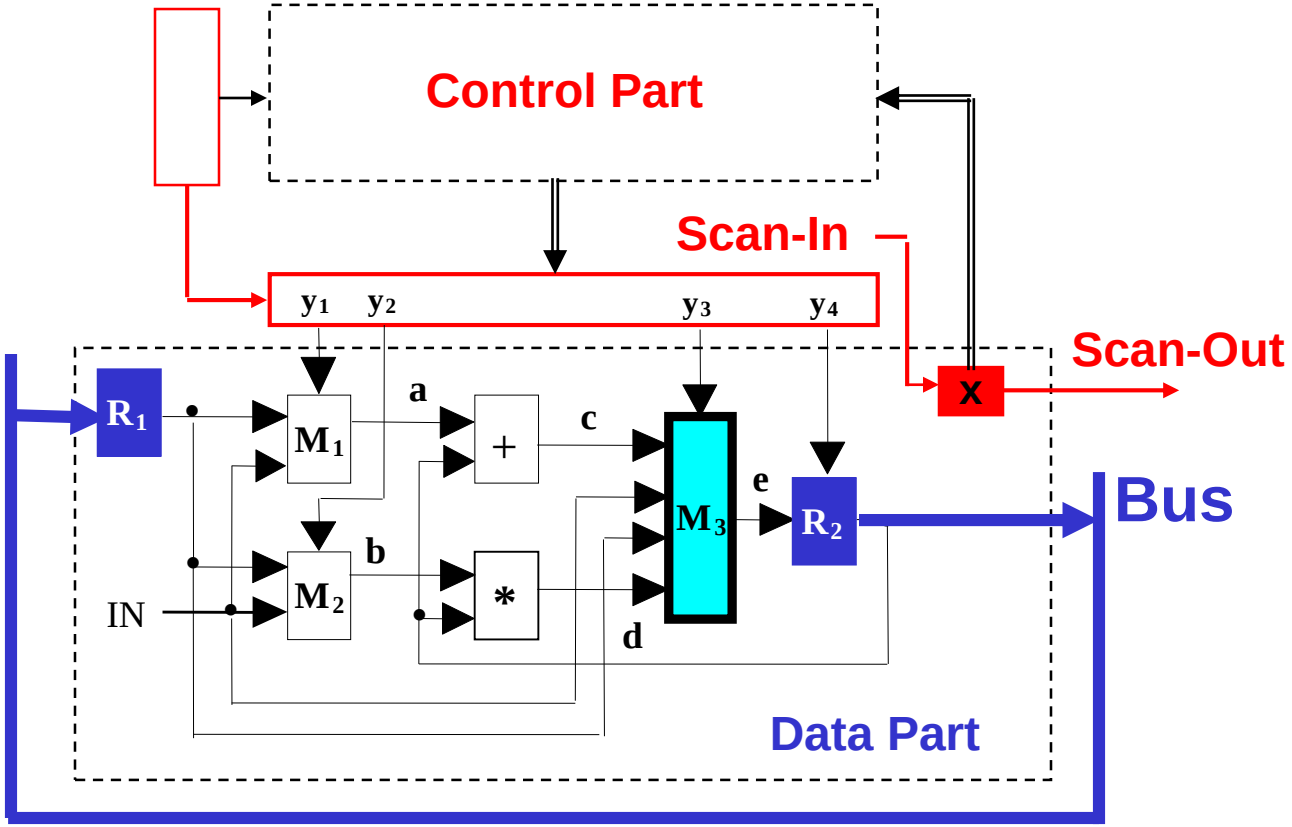
Partial Scan Path

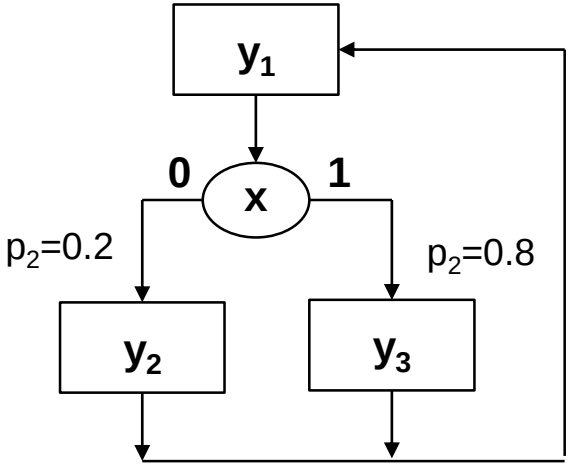
Hierarhical test generation with Scan-Path:



Testing with Minimal DFT

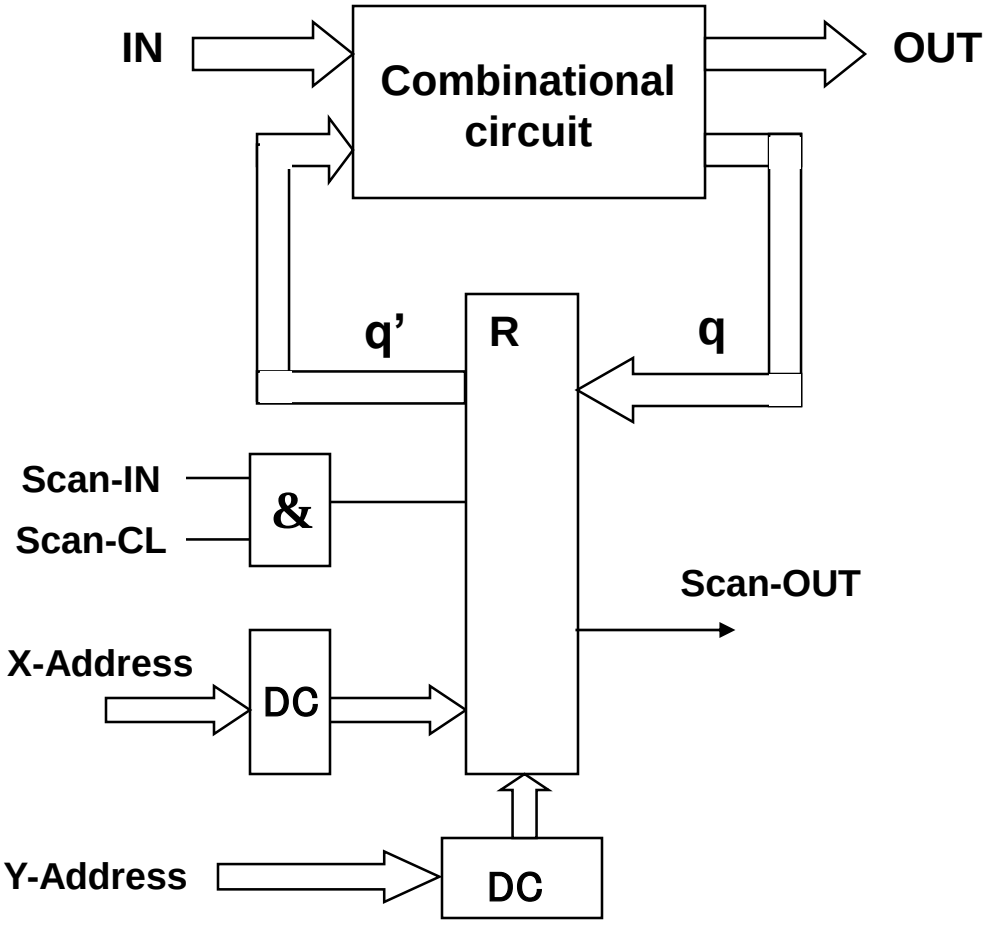
Hierarhical test generation with Scan-Path:





If the control flow sequences are short, only a single or few flip-flops of data-dependent flag-FF-s are included into the scan-path

Random Access Scan

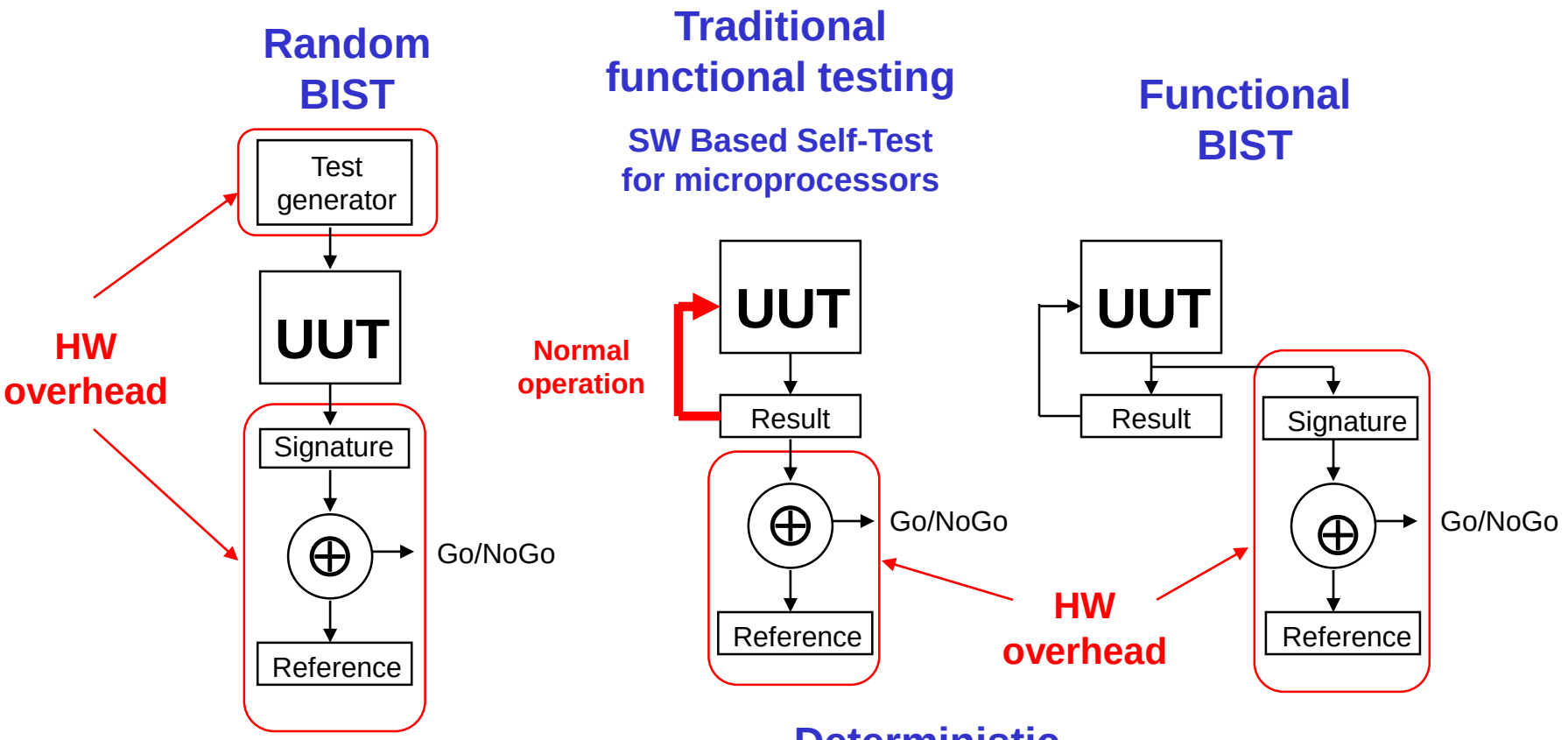


In random access scan each flip–flop in a logic network is selected individually by an address for control and observation of its state

Example:

Delay fault testing

DFT for Random BIST & Functional BIST



Random test ⇨ Universal solutions based on DFT measures

Deterministic functional test set ⇨ Dedicated solutions based on Fault Simulation

Selection of Test Points

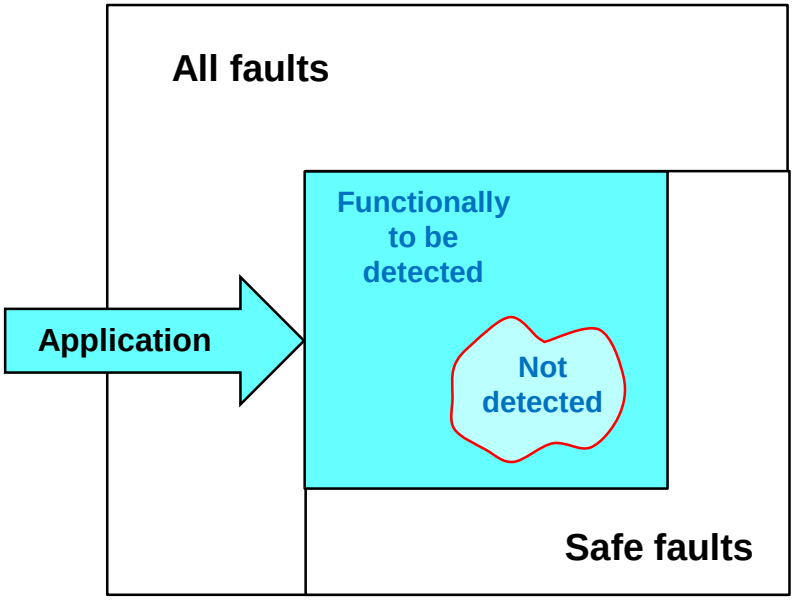
Test point selection approaches

- **Improving testability for any set of pseudo-random patterns (Pseudorandom BIST)**
  - Testability measures are used to characterize the controllability and observability of the circuit (independently of the test applied)
- **Improving testability for a given implementation based sequence of vectors (Functional BIST)**
  - Fault simulation is used for measuring the fault coverage

Methods that are used:

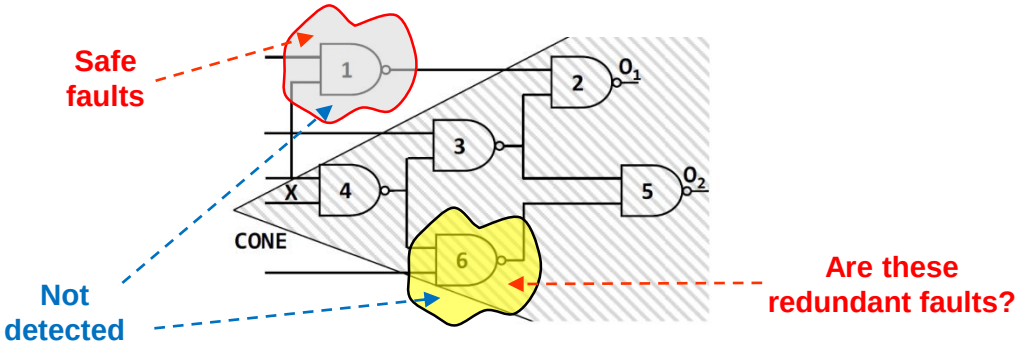
- logic simulation,
- fault simulation,
- evaluation (measuring) of controllability and observability

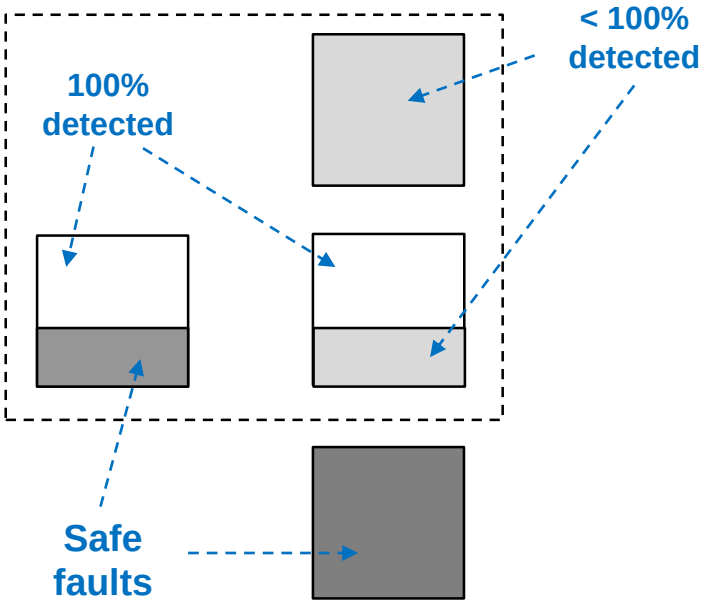
The Problem of Safe/Redundant faults



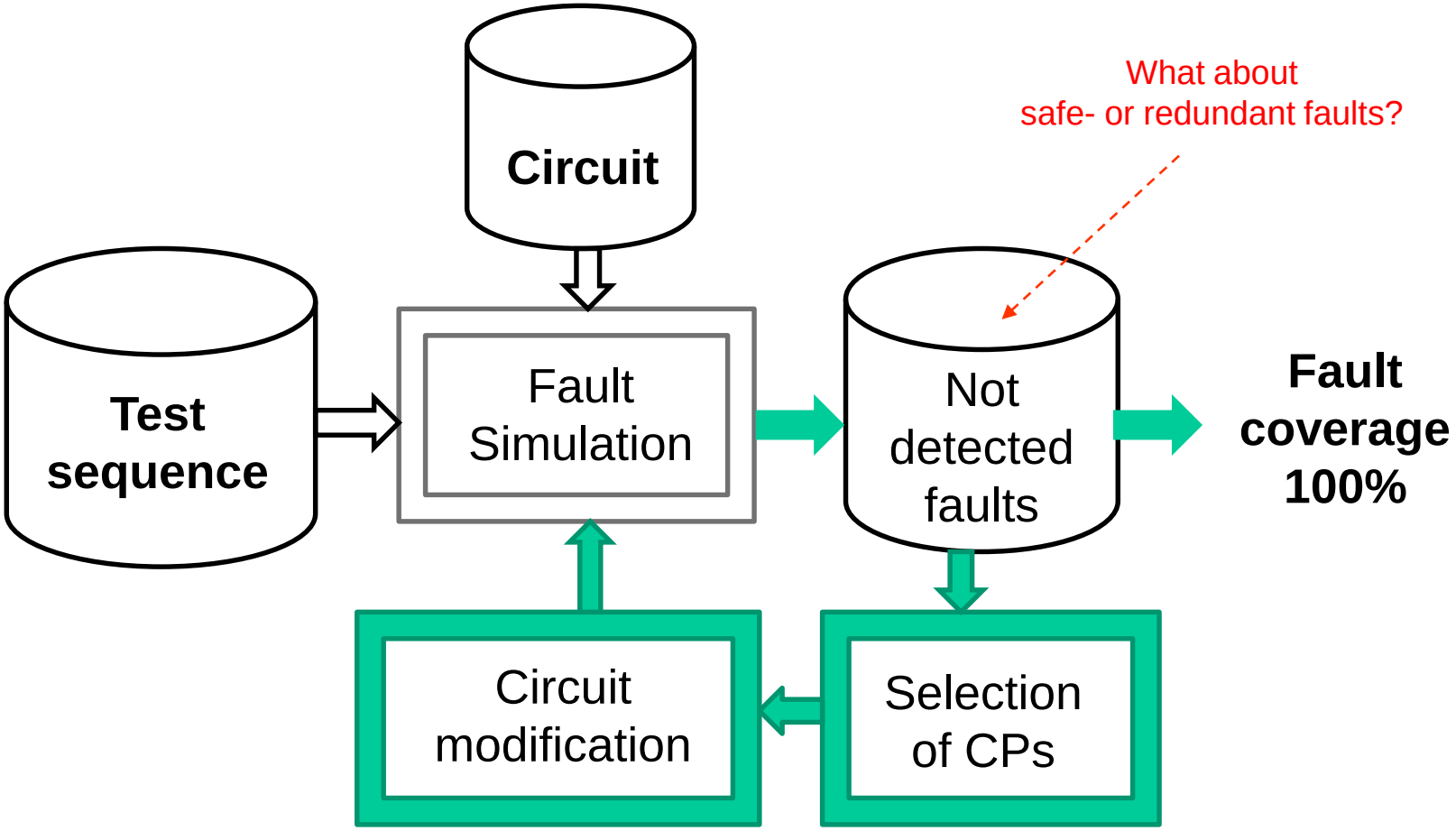
Application software may not use a part of system HW (e.g. Debugging Module, Floating Point Unit), or a part of instruction set (e.g. multiplication)

Safe faults cannot produce any failure due to the **specific (HW or SW) constraints** the system matches during its normal operation



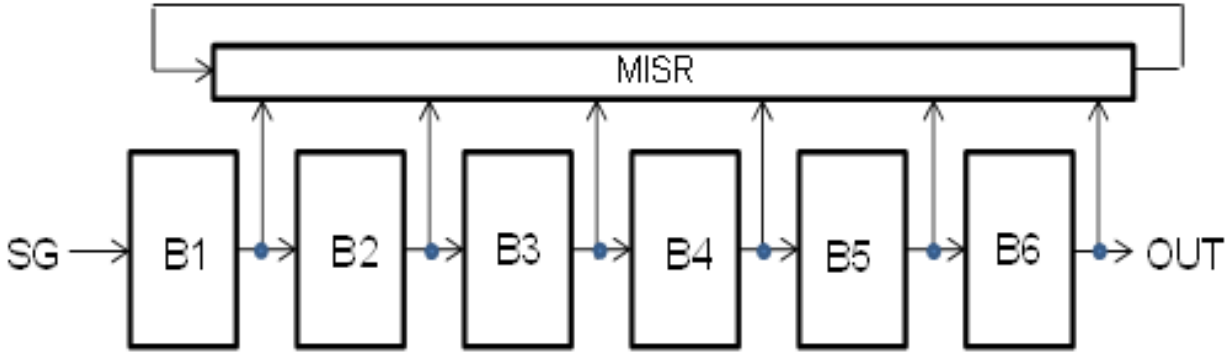


Adhoc Iterative DFT Improvement



High-Level Functional BIST

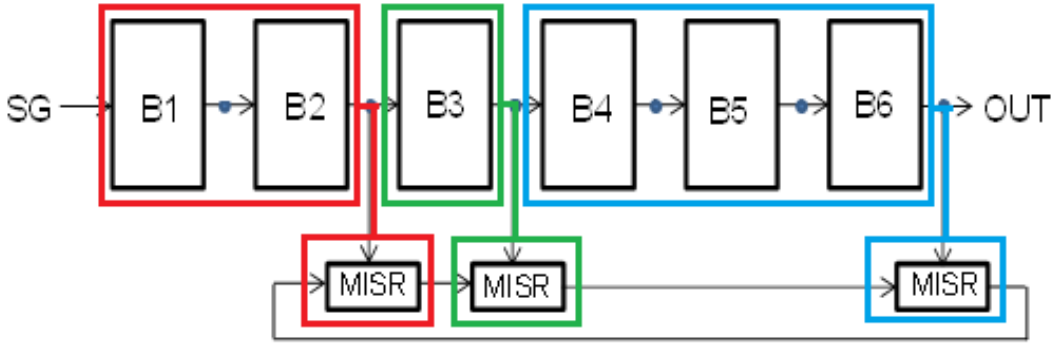
Example: Functional BIST for Pipe-Lined Circuits



Two solutions

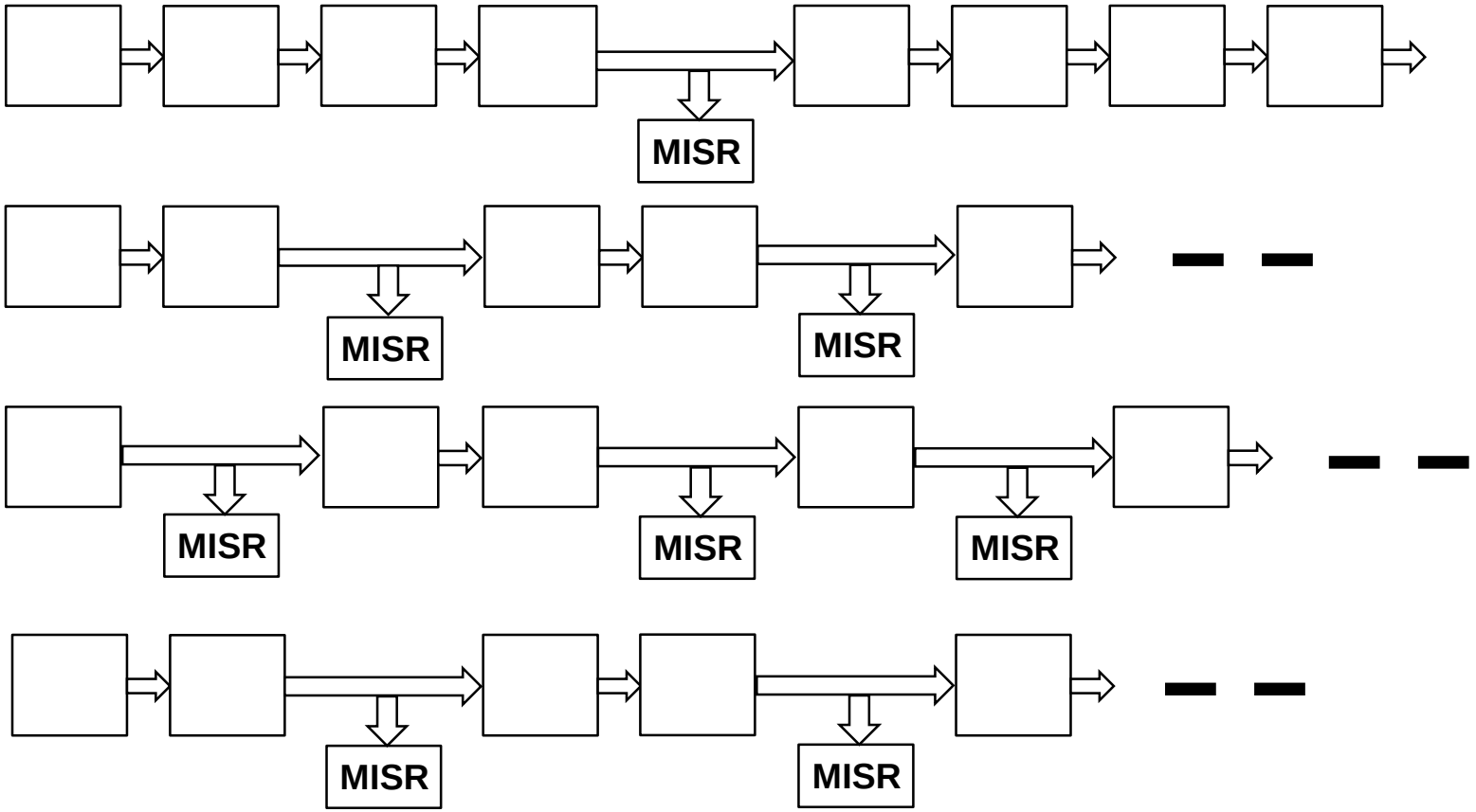
MISR monitors on every register

MISR monitors on part of the register (Combine blocks with good coverage)



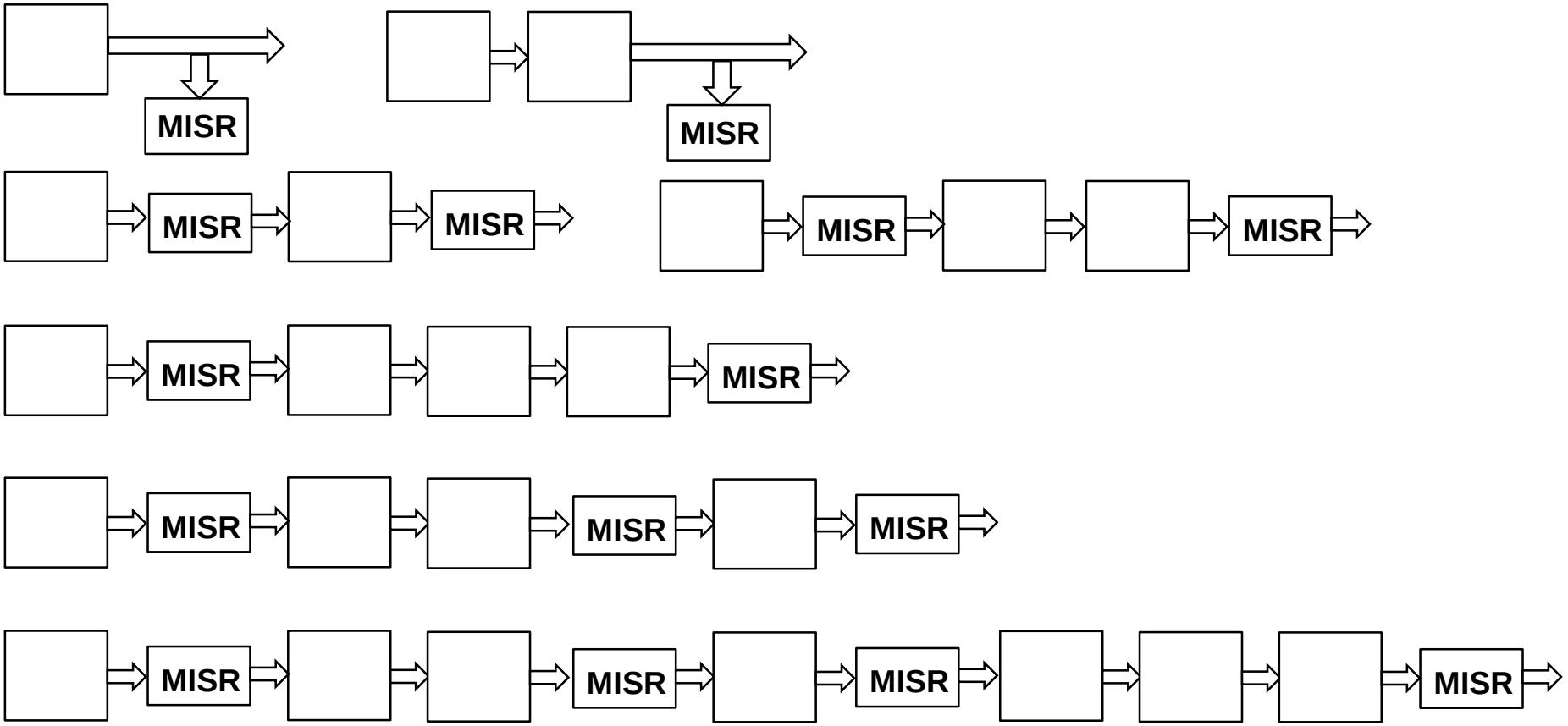
HL-FBIST Synthesis

Start-From-Big method

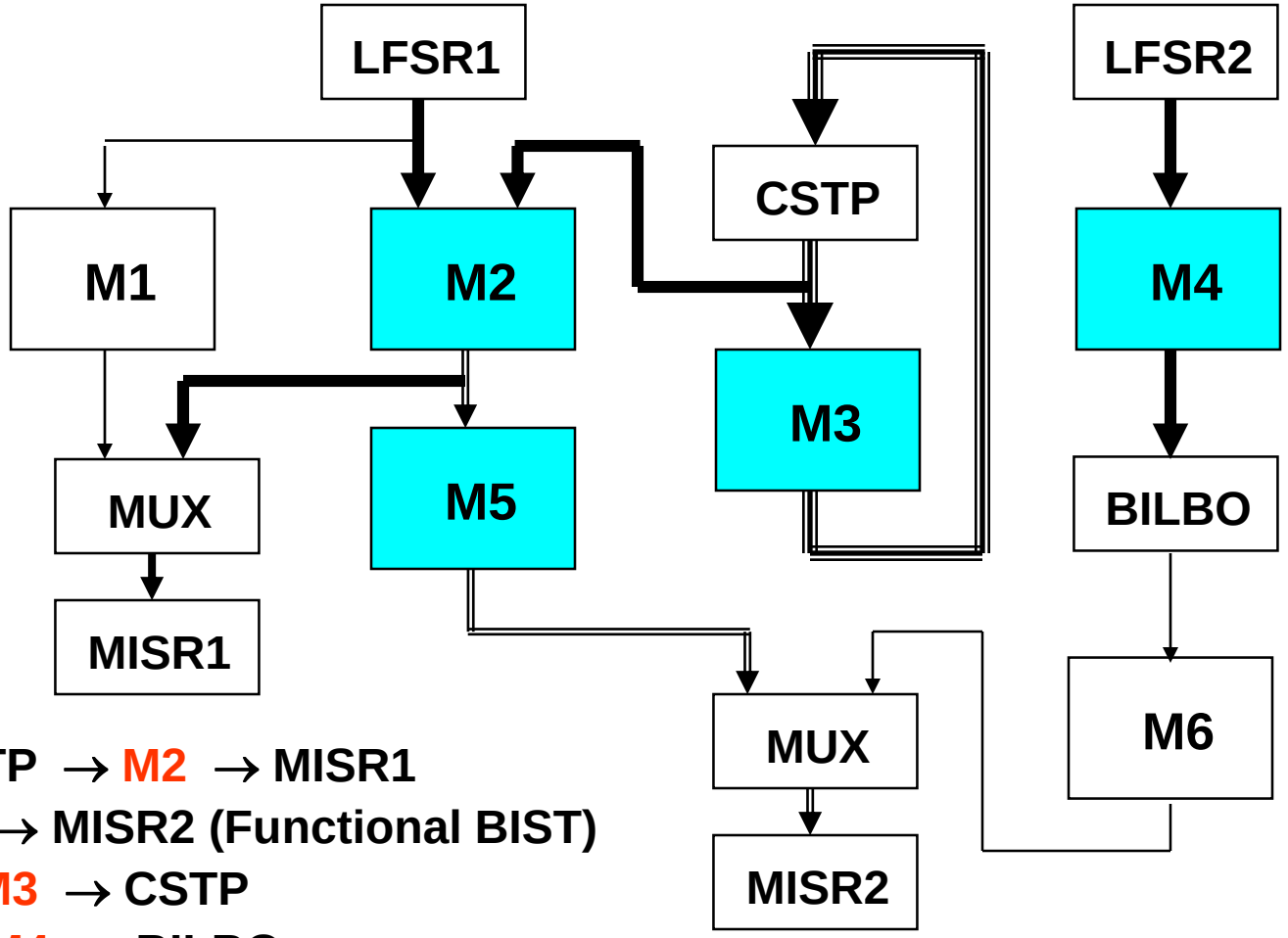


HL-FBIST Synthesis

Start-From-Small method



Distributed BIST Synthesis



Concurrent testing:

LFSR, CSTP → M2 → MISR1

M2 → M5 → MISR2 (Functional BIST)

CSTP → M3 → CSTP

LFSR2 → M4 → BILBO

Selection of Test Points

Method: Simulation of given test patterns

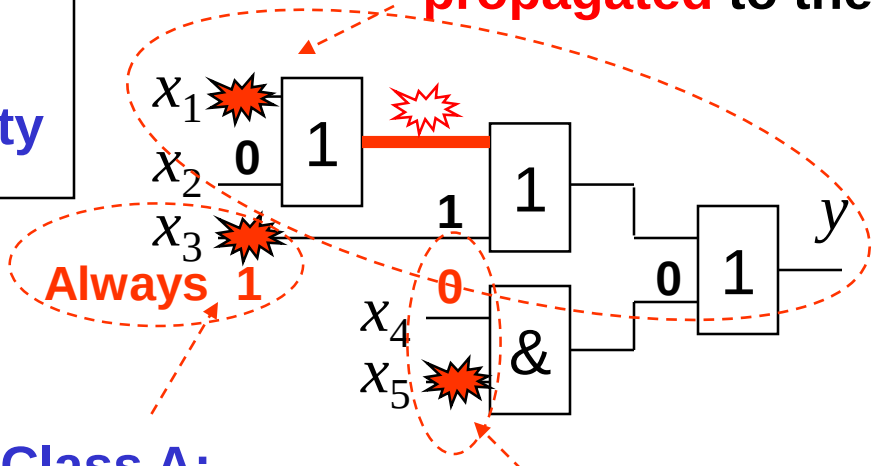
- Identification of the faults that are **detected**
- The remaining faults are classified as
  - **A:** Faults that were **not excited**
  - **B:** Faults at gate inputs that were excited but **not propagated to the gate output**
  - **C:** Faults that were excited but **not propagated to circuit output**
- The faults A and B **require control points** for their detection
- The faults C **may** be detected by either by **observation points** or **by control points**
- Control points selection should be carried out before observation points selection

Classification of Not-Detected Faults

Classes A and B need controllability

Class C:

Faults at x_1 are **not propagated to the output**



Class A:

Fault $x_3 \equiv 1$ is **not activated**

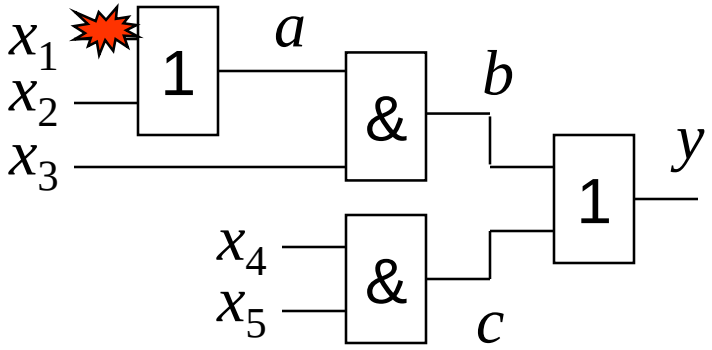
Class B:

Faults at x_5 are **not propagated through the gate**

Class C needs either controllability or observability

Selection of Test Points

Classification of faults



Given test:

No	Test patterns: Inputs					Test patterns: Intern. points			Fault table: Inputs					Fault table: Intern. points		
	1	2	3	4	5	a	b	c	1	2	3	4	5	a	b	c
1	0	0	1	0	1	0	0	0	1	1	-	1	-	1	1	1
2	0	1	0	1	1	1	0	1	-	-	-	0	0	-	-	0
3	0	1	0	1	0	1	0	0	-	-	1	-	1	-	1	1

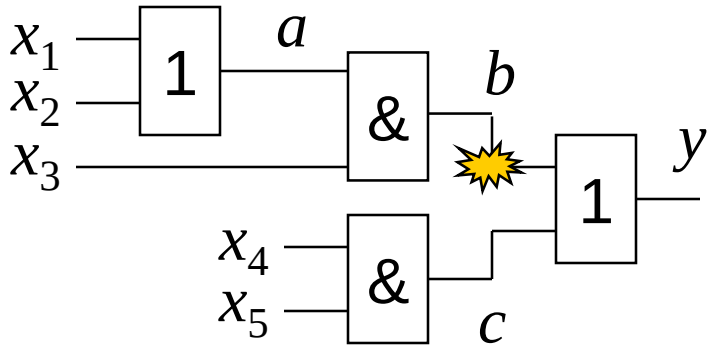
$x_1/0$ $x_2/0$ $x_3/0$ $a/0$ $b/0$

Not detected faults:

Class	Faults	Missing signals	
A	$x_1/0$:	$x_1 = 1$	is missing
A	$b/0$:	$b = 1$	is missing
B	$x_3/0$:	x_3 a = 11	is missing
B	$a/0$:	x_3 a = 11	is missing
C	$x_2/0$:	$x_1x_2 = 01$	OK

Selection of Test Points

Classification of faults



Given test:

No	Test patterns: Inputs					Test patterns: Intern. points			Fault table: Inputs					Fault table: Intern. points		
	1	2	3	4	5	a	b	c	1	2	3	4	5	a	b	c
1	0	0	1	0	1	0	0	0	1	1	-	1	-	1	1	1
2	0	1	0	1	1	1	0	1	-	-	-	0	0	-	-	0
3	0	1	0	1	0	1	0	0	-	-	1	-	1	-	1	1

$x_1/0$ $x_2/0$ $x_3/0$ $a/0$ $b/0$

Not detected faults:

Class	Faults	Missing signals	
A	$x_1/0$:	$x_1 = 1$	is missing
A	$b/0$:	$b = 1$	is missing
B	$x_3/0$:	x_3 a = 11	is missing
B	$a/0$:	x_3 a = 11	is missing
C	$x_2/0$:	$x_1x_2 = 01$	OK

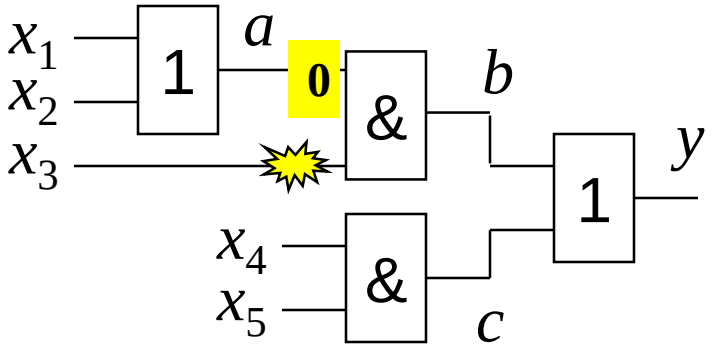
Selection of Test Points

Classification of faults

Given test:

No	Test patterns: Inputs					Test patterns: Intern. points			Fault table: Inputs					Fault table: Intern. points		
	1	2	3	4	5	a	b	c	1	2	3	4	5	a	b	c
1	0	0	1	0	1	0	0	0	1	1	-	1	-	1	1	1
2	0	1	0	1	1	1	0	1	-	-	-	0	0	-	-	0
3	0	1	0	1	0	1	0	0	-	-	1	-	1	-	1	1

$x_1/0$ $x_2/0$ $x_3/0$ $a/0$ $b/0$



Not detected faults:

Class	Faults	Missing signals	
A	$x_1/0$:	$x_1 = 1$	is missing
A	$b/0$:	$b = 1$	is missing
B	$x_3/0$:	x_3 a = 11	is missing
B	$a/0$:	x_3 a = 11	is missing
C	$x_2/0$:	$x_1x_2 = 01$	OK

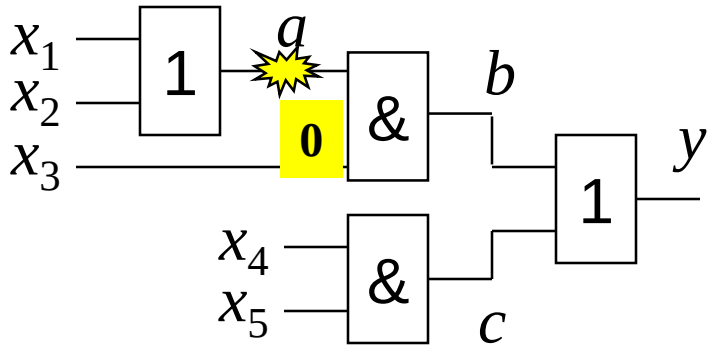
Selection of Test Points

Classification of faults

Given test:

No	Test patterns								Fault table							
	Inputs					Intern. points			Inputs					Intern. points		
	1	2	3	4	5	a	b	c	1	2	3	4	5	a	b	c
1	0	0	1	0	1	0	0	0	1	1	-	1	-	1	1	1
2	0	1	0	1	1	1	0	1	-	-	-	0	0	-	-	0
3	0	1	0	1	0	1	0	0	-	-	1	-	1	-	1	1

$x_1/0$ $x_2/0$ $x_3/0$ $a/0$ $b/0$

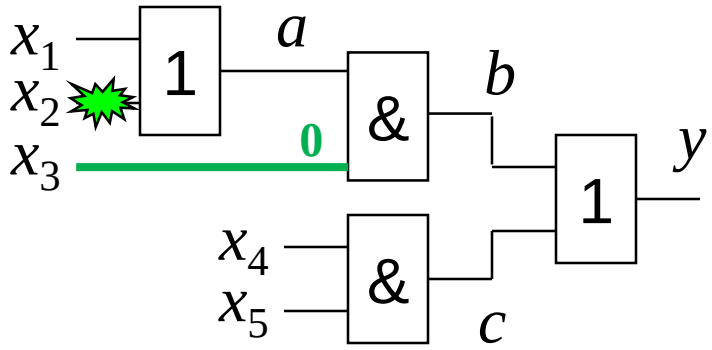


Not detected faults:

Class	Faults		
A	$x_1/0$:	$x_1 = 1$	is missing
A	$b/0$:	$b = 1$	is missing
B	$x_3/0$:	x_3 a = 11	is missing
B	$a/0$:	x_3 a = 11	is missing
C	$x_2/0$:	$x_1x_2 = 01$	OK

Selection of Test Points

Classification of faults



Given test:

No	Test patterns								Fault table							
	Inputs					Intern. points			Inputs					Intern. points		
	1	2	3	4	5	a	b	c	1	2	3	4	5	a	b	c
1	0	0	1	0	1	0	0	0	1	1	-	1	-	1	1	1
2	0	1	0	1	1	1	0	1	-	-	-	0	0	-	-	0
3	0	1	0	1	0	1	0	0	-	-	1	-	1	-	1	1

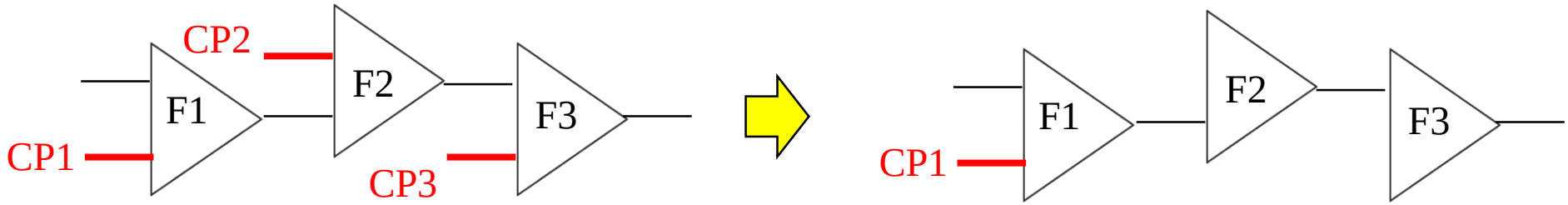
$x_1/0$ $x_2/0$ $x_3/0$ $a/0$ $b/0$

Not detected faults:

Class	Faults	Missing signals
A	$x_1/0$:	$x_1 = 1$ is missing
A	$b/0$:	$b = 1$ is missing
B	$x_3/0$:	x_3 a = 11 is missing
B	$a/0$:	x_3 a = 11 is missing
C	$x_2/0$:	$x_1x_2 = 01$ OK, but path activation x_3 is missing

Selection of Test Points: Procedure

1. Selection of control points:



	F1	F2	F3	F4	F5	F6	F7	F8	F9
CP1	**1**	**1**	**1**			1			
CP2		**1**	**1**	1			1		1
CP3			**1**		1			1	
CP4			1		1	1		1	
CP5	1					1	1		1

Faults, not detected

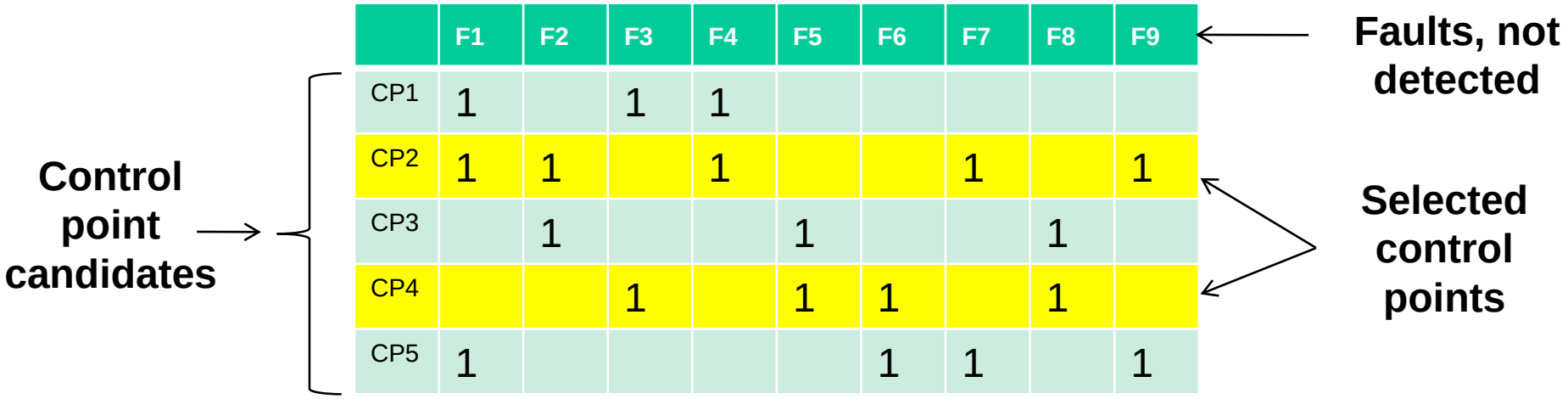
Control point candidates

Selected control points

Selection of Test Points: Procedure

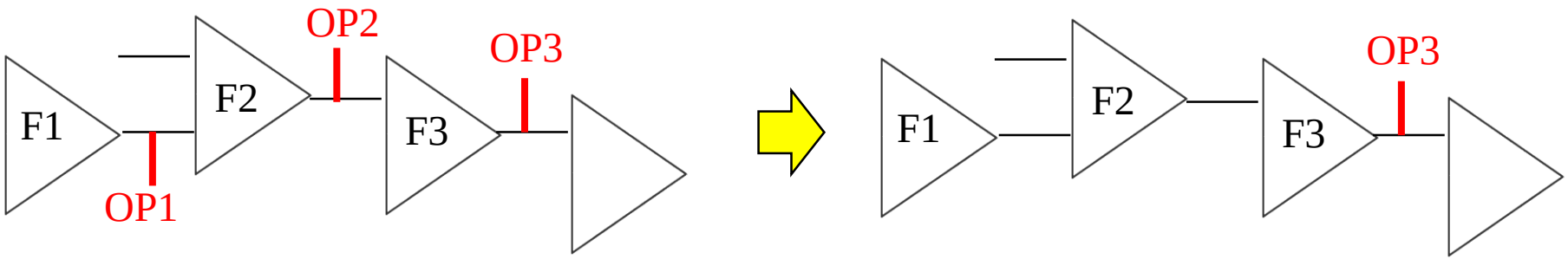
1. Selection of control points:

- Once **control point candidates** are identified for the faults A and B, a **minimum** number of control points (CP) can be identified
- This can be formulated as a **minimum coverage problem** where a minimum CPs are selected such that at least one CP candidate is included for each fault in A and B



Selection of Test Points: Procedure

1. Selection of observation points:



	F1	F2	F3	F4	F5	F6	F7	F8	F9
OP1	**1**					1			
OP2	**1**	**1**		1			1		1
OP3	**1**	**1**	**1**		1			1	
OP4			1		1	1		1	
OP5	1					1	1		1

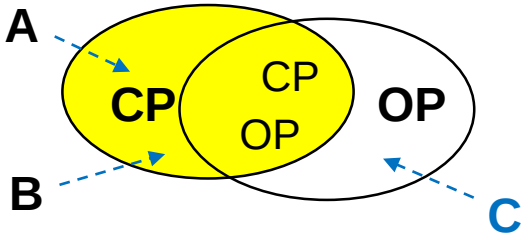
Faults, not detected

Observation point candidates

Selected observation points

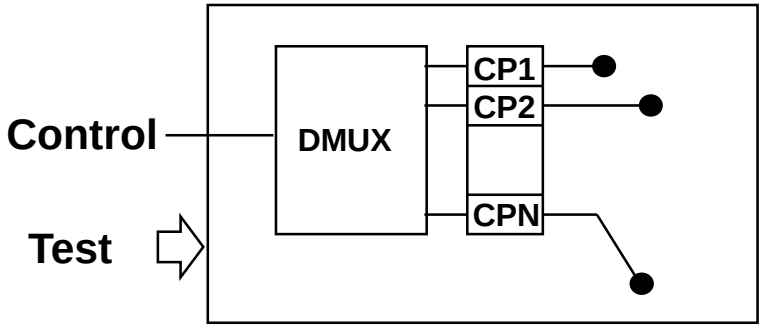
Selection of Test Points: Procedure

2. Selection of observation points



- Once CPs selected, the **test patterns are augmented, fault simulation** is performed
- The fault class **C is updated**
- For each fault, in C the circuit lines to which the effect of the fault propagates, are identified as a potential **observation point candidates**
- A **minimum covering problem** is formulated and solved to find the observation points to be added

New fault simulation



Fault class C updated

Minimization of observation points

	F1	F2	F3	F4	F5	F6	F7	F8	F9
OP1	1		1	1					
OP2	1	1		1			1		1
OP3		1			1			1	
OP4			1		1	1		1	
OP5	1					1	1		1

Selection of Test Points

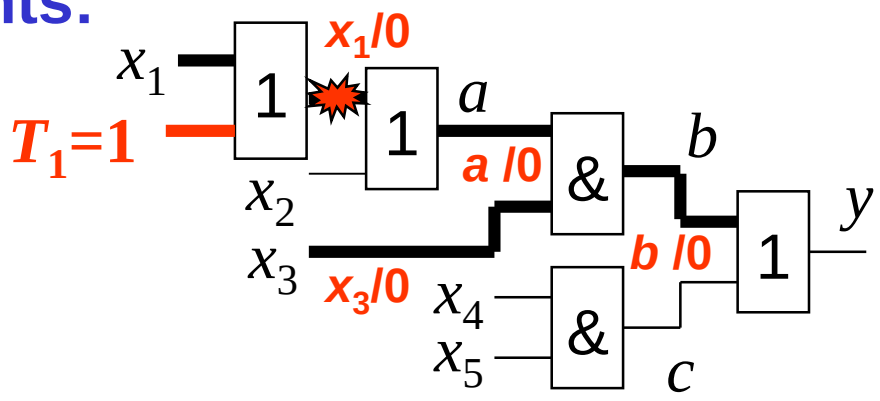
Minimization of control points:

Not detected faults:

Class A: $x_1/0$, $b/0$

Class B: $x_3/0$, $a/0$,

Corrected circuit:



Control point coverage:

To be selected		Not detected faults			
		$x_1/0$	$x_3/0$	$a/0$	$b/0$
Potential control points	$x_1=1$	+	+	+	+
	$x_3=1$		+	+	+
	$a=1$		+	+	+
	$b=1$				+

No	Test patterns							
	Inputs					Intern. points		
	1	2	3	4	5	a	b	c
1	0	0	1	0	1	0	0	0
2	0	1	0	1	1	1	0	1
3	0	1	0	1	0	1	0	0

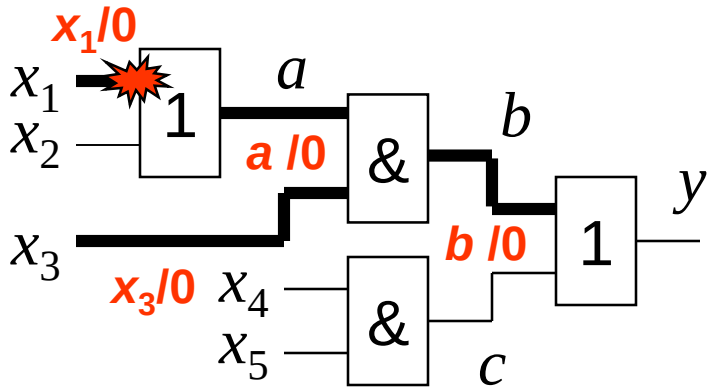
Insertion of Test Points

Test point for $x_1/0$

All faults detected:

Class A: $x_1/0$, $b/0$

Class B: $x_3/0$, $a/0$,

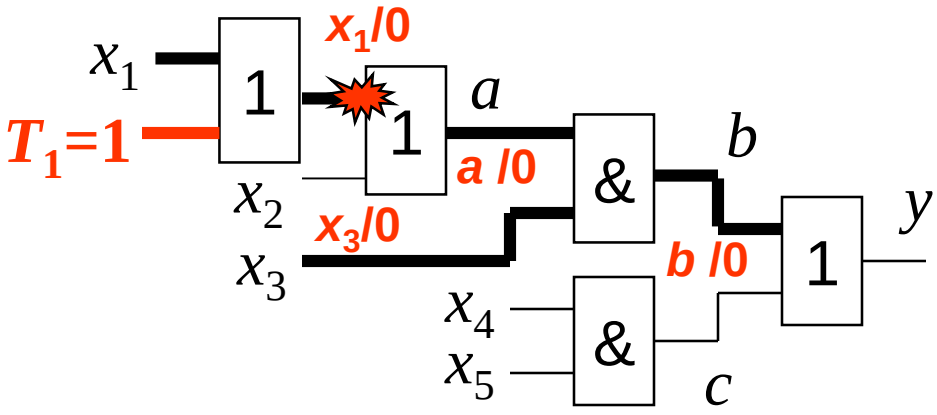


T_1=1

This pattern is to be repeated with T_1=1

No	Test patterns							
	Inputs					Intern. points		
	1	2	3	4	5	a	b	c
1	0	0	1	0	1	0	0	0
2	0	1	0	1	1	1	0	1
3	0	1	0	1	0	1	0	0

Corrected circuit:



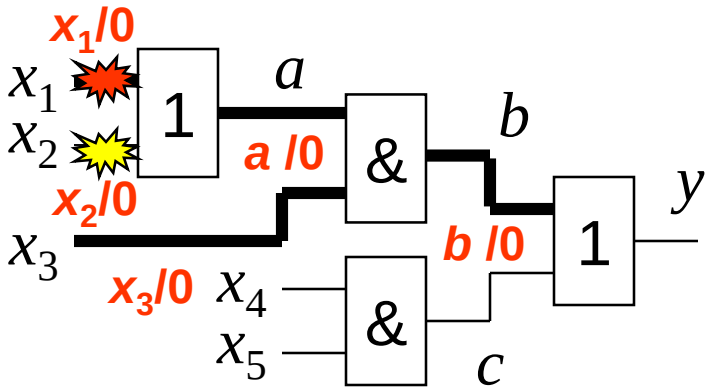
Insertion of Test Points

Two test points:

Selected test points:

Class A: $x_1/0 \rightarrow x_1=1$ **(control point)**

Class C: $x_2/0$ **(observation point)**



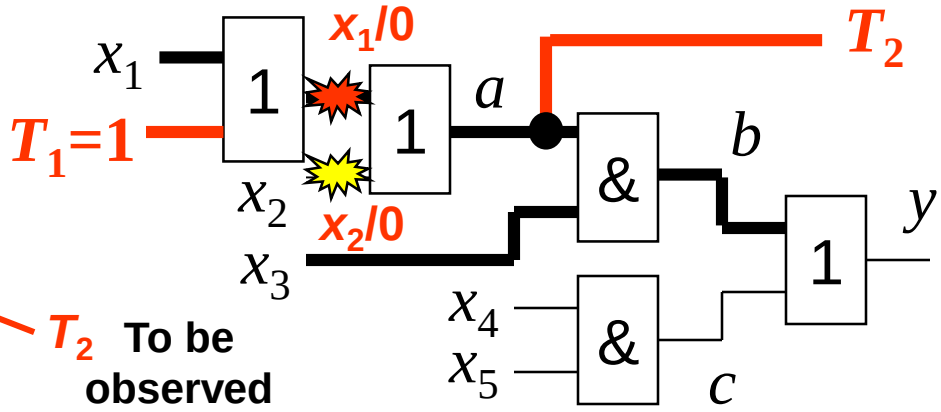
$T_1=1$

This pattern is to be repeated with $T_1=1$

No	Test patterns							
	Inputs					Intern. points		
	1	2	3	4	5	a	b	c
1	0	0	1	0	1	0	0	0
2	0	1	0	1	1	1	0	1
3	0	1	0	1	0	1	0	0

T_2 To be observed

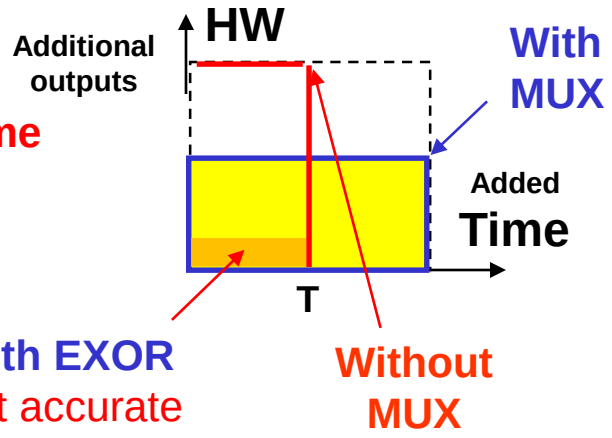
Corrected circuit:

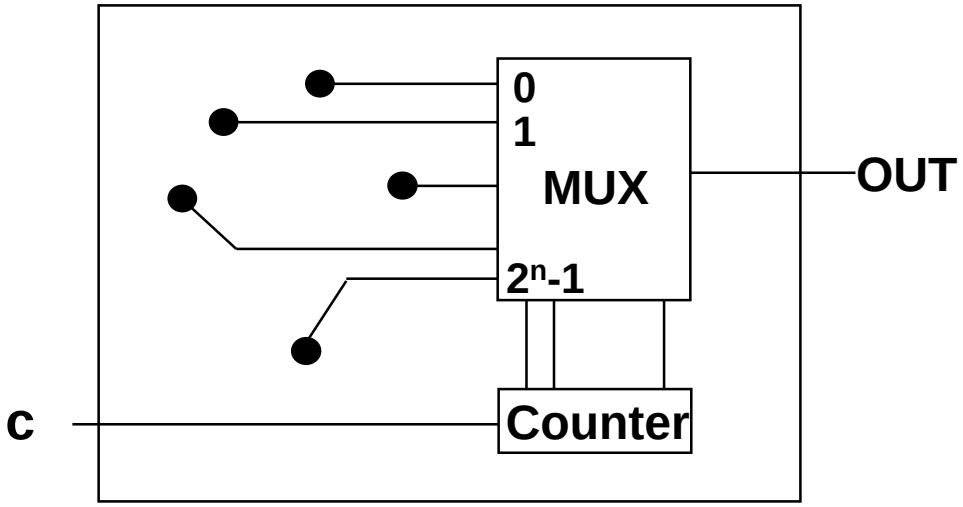


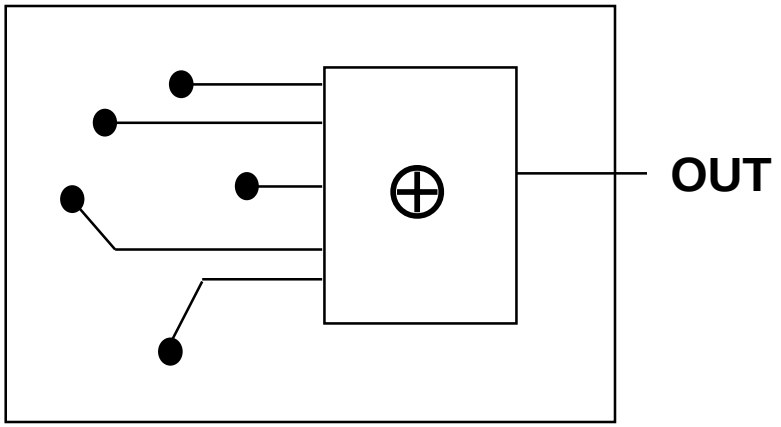
Selection of Test Points – Tradeoff Problem

Minimization of monitoring points:

HW cost and time compaction
T – test time





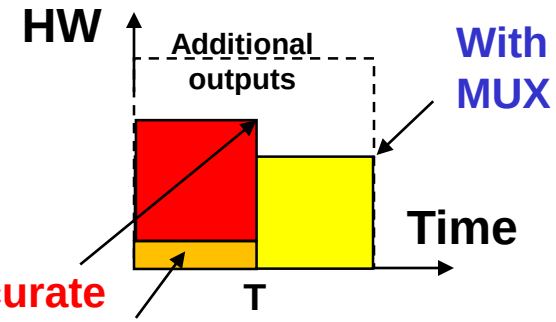


To reduce the number of output pins for observing monitor points, EXOR gates can be used:

Selection of Test Points

Minimization of monitoring points:

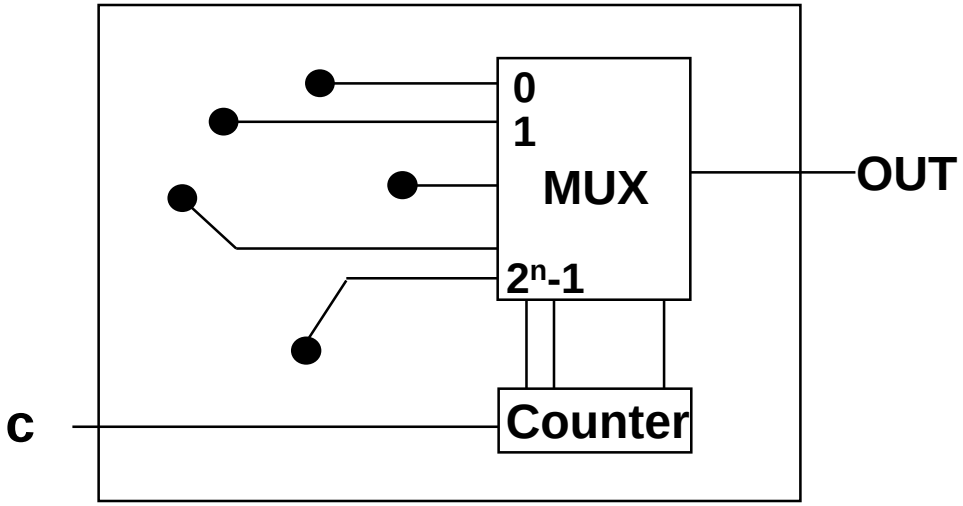
To reduce the number of output pins for observing monitor points, signature analyzers can be used:

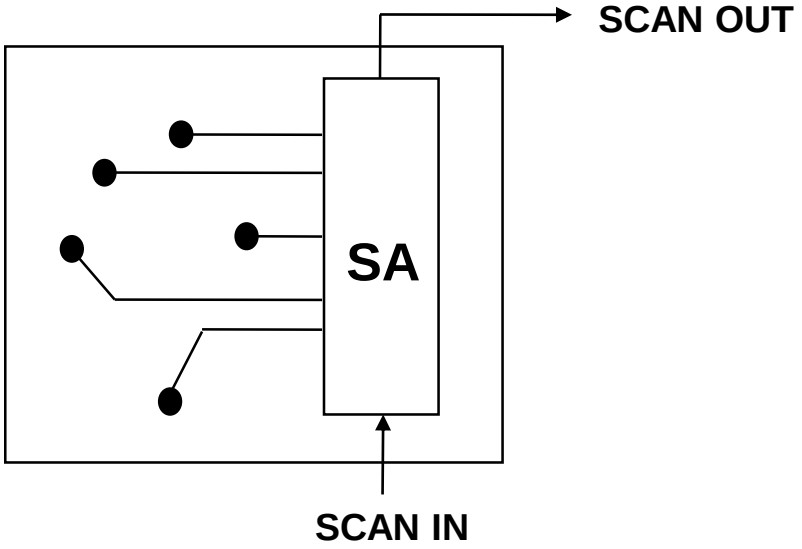


HW cost and time compaction
T – test time

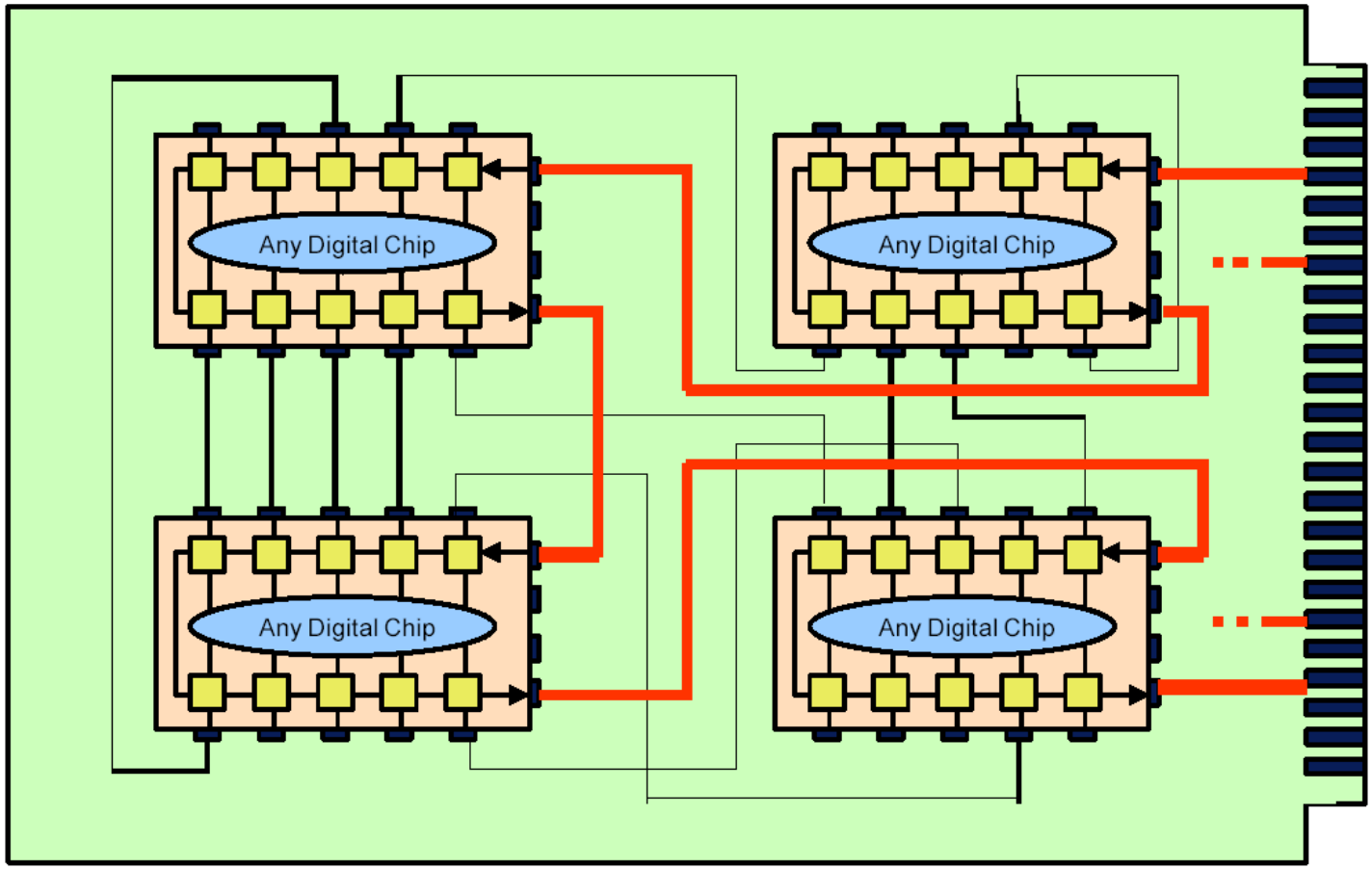
With SA - Accurate

With EXOR – Not accurate

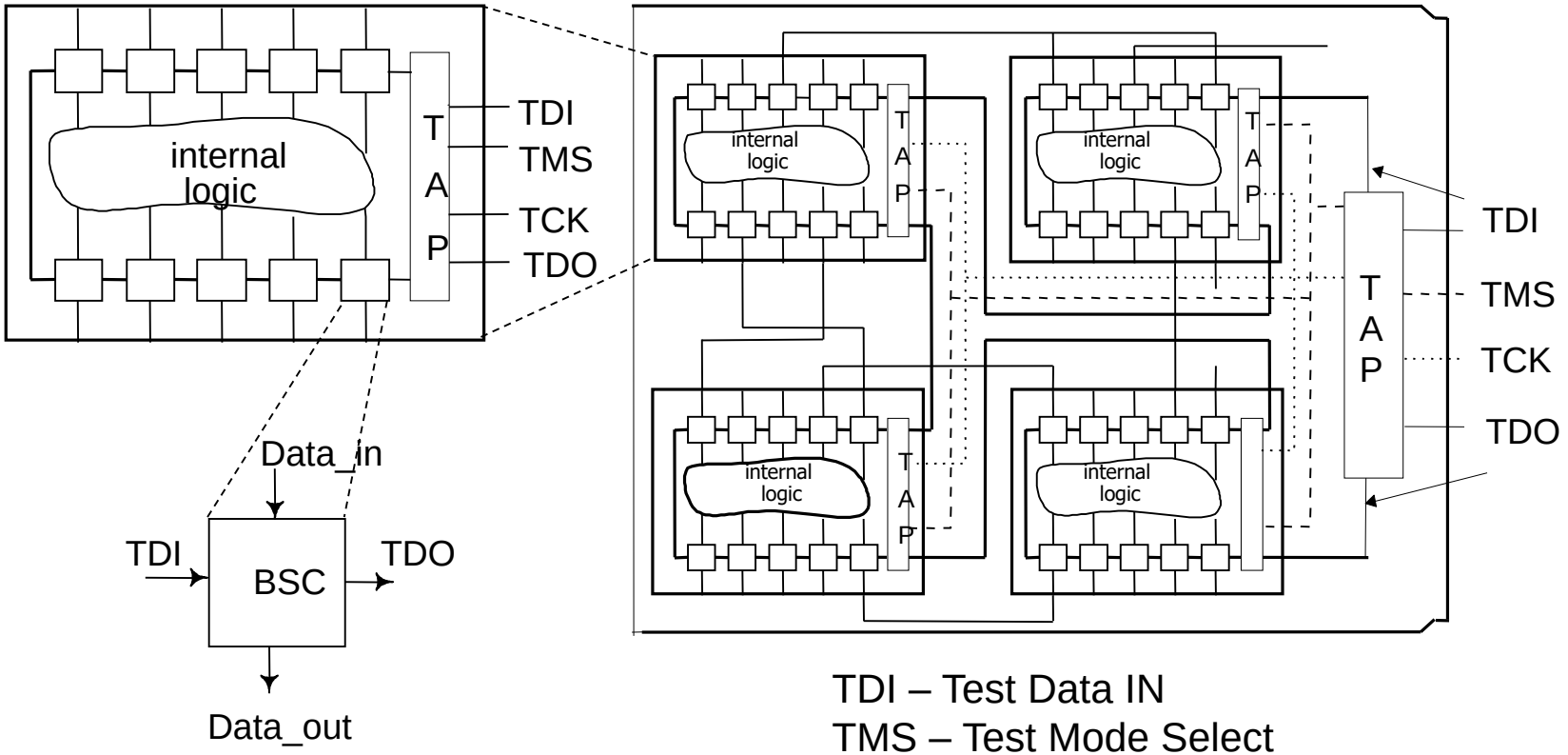




Boundary Scan Standard

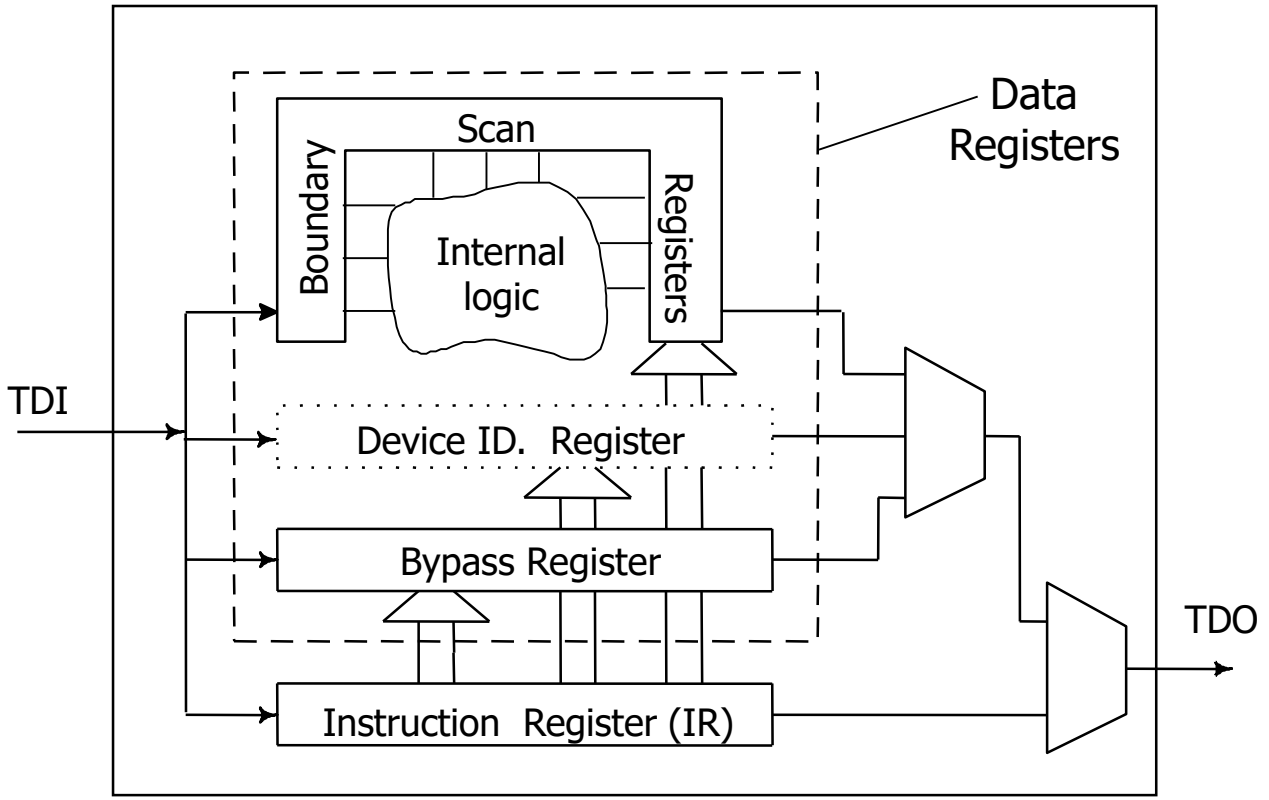


Boundary Scan Architecture

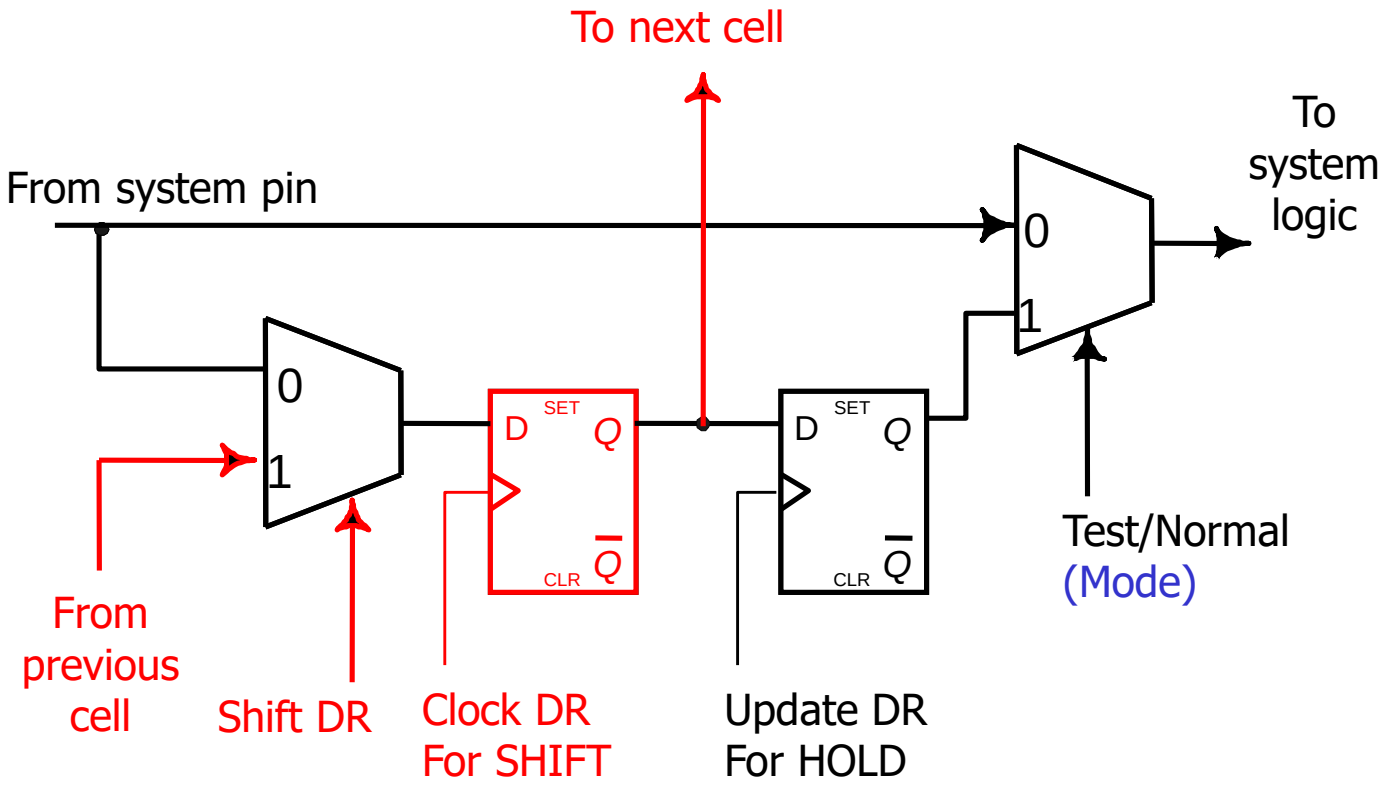


TDI – Test Data IN
TMS – Test Mode Select
TCK – Test Clock
TDO – Test Data OUT
TAP – Test Acess Port

Boundary Scan Architecture



Boundary Scan Cell

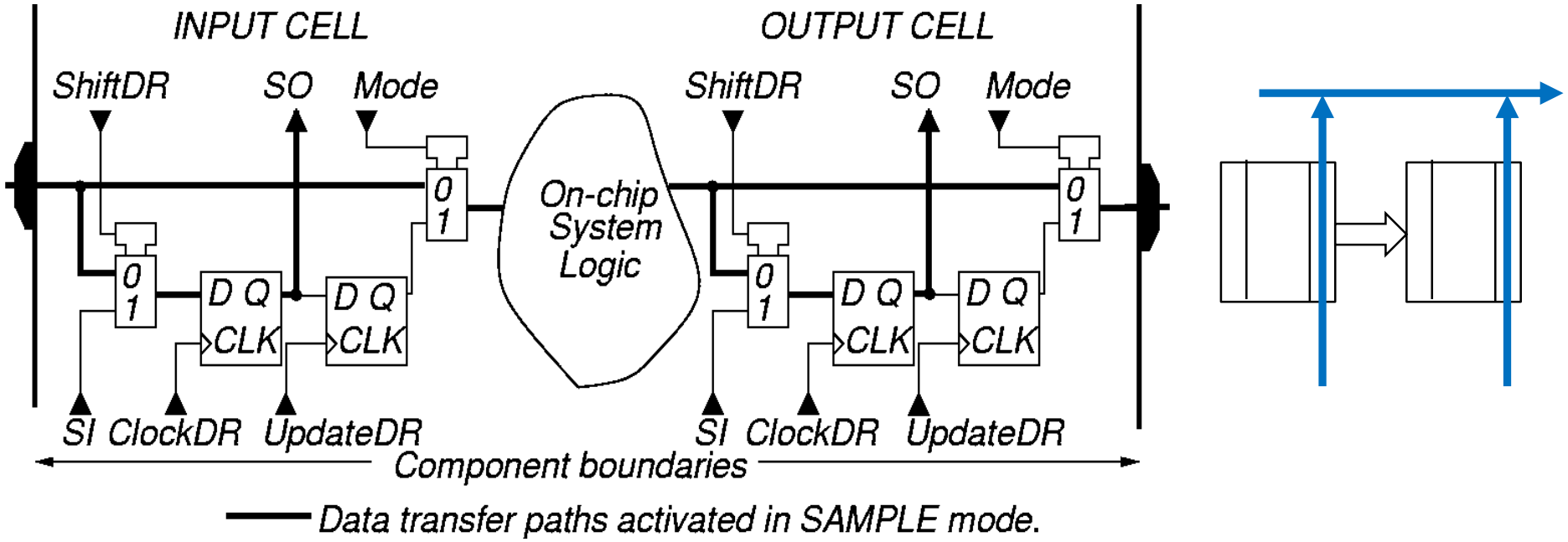


Used at the input or output pins

Boundary Scan Working Modes

SAMPLE mode:

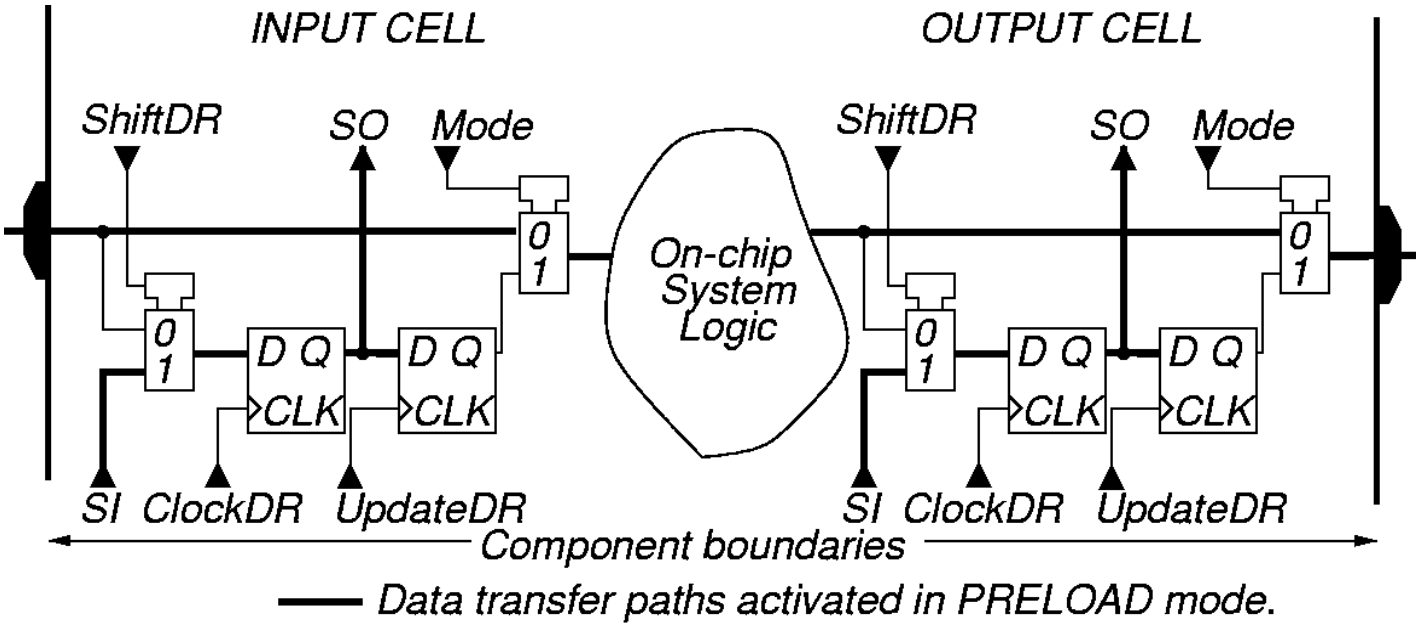
*Get snapshot of normal chip output signals (**monitoring mode**)*

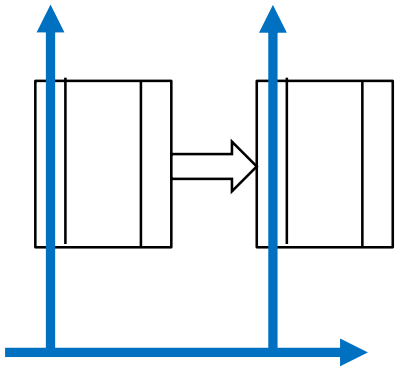


Boundary Scan Working Modes

PRELOAD mode:

Put data on boundary scan chain before next instruction

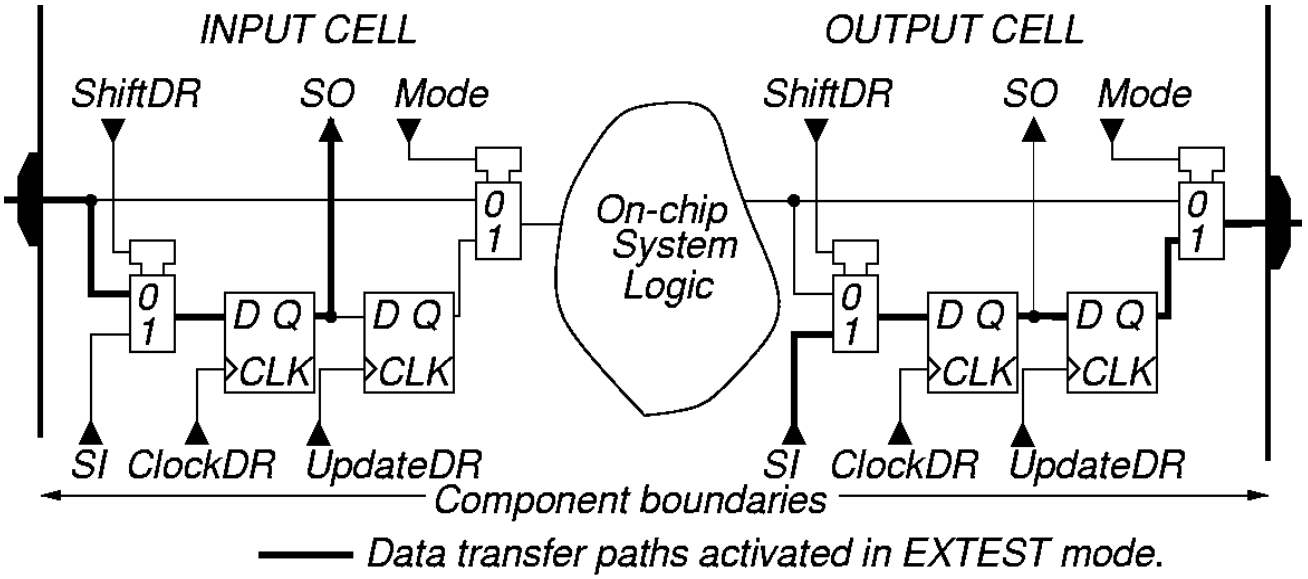


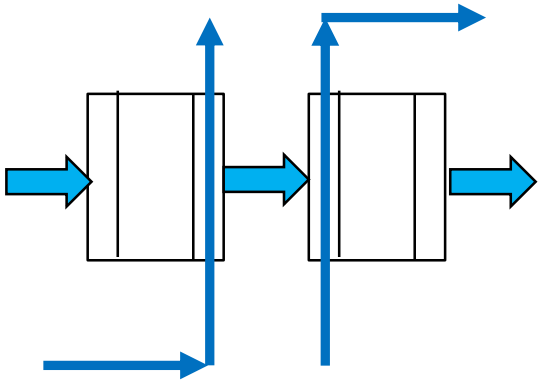


Boundary Scan Working Modes

EXTEST instruction:

Test off-chip circuits and board-level interconnections

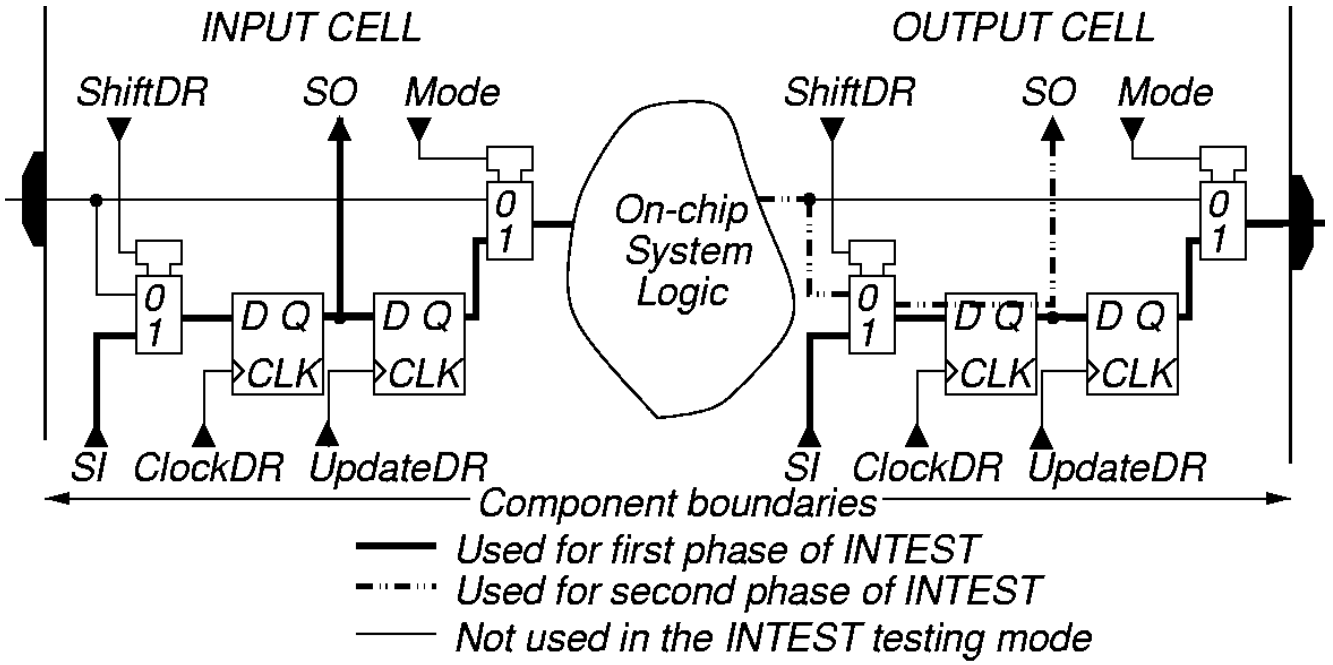


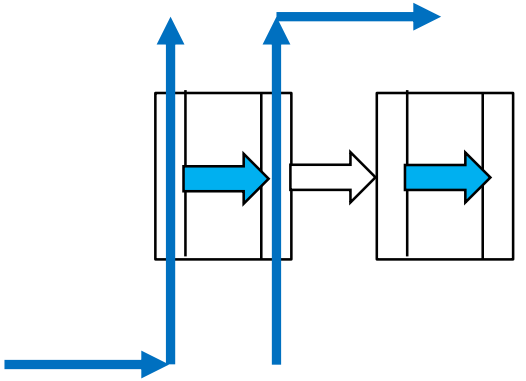


Boundary Scan Working Modes

INTEST instruction

Feeds external test patterns in and shifts responses out

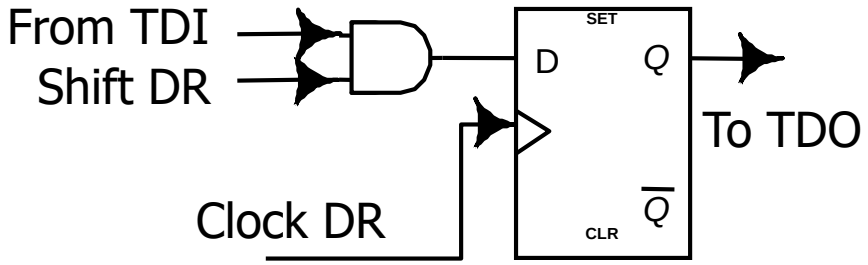


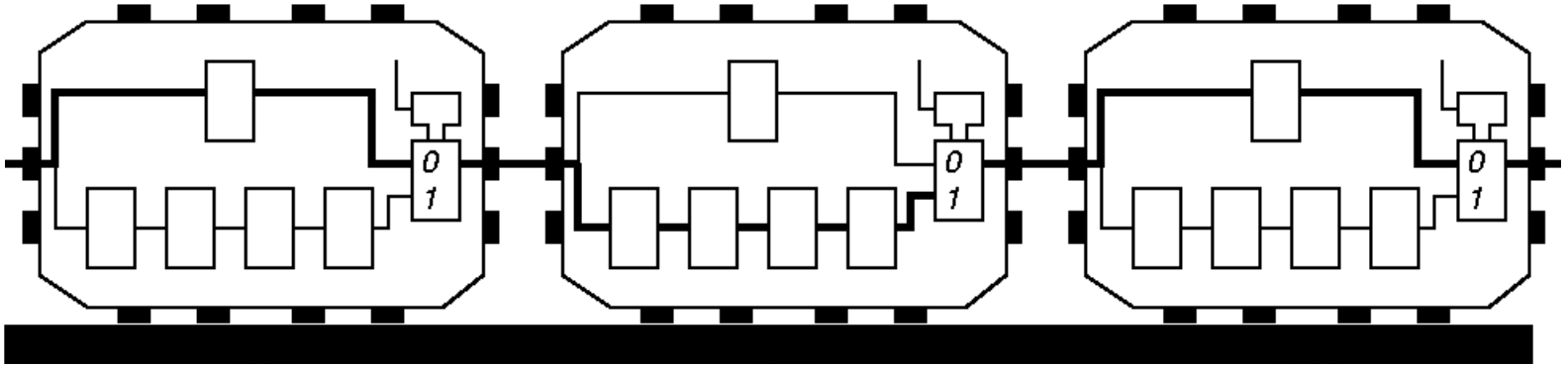


Boundary Scan Working Modes

Bypass instruction:

Bypasses the corresponding chip using 1-bit register





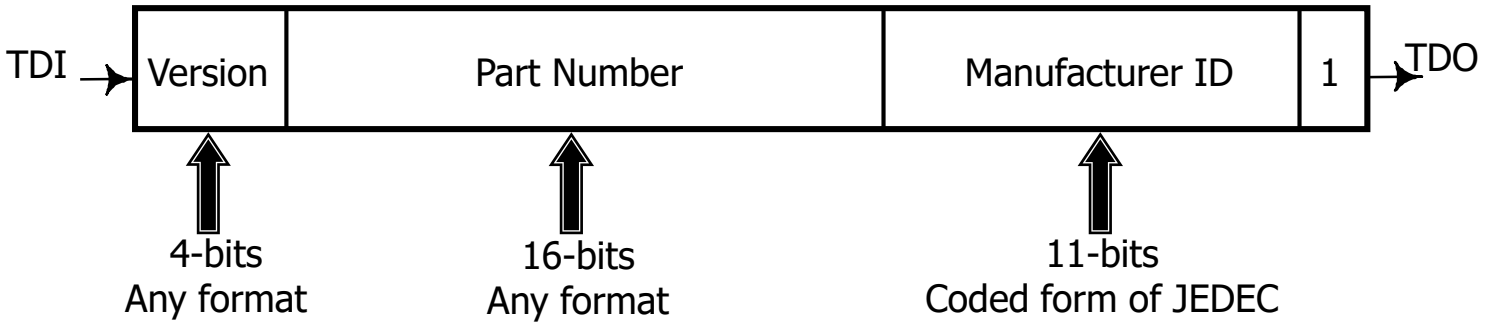
Boundary Scan Working Modes

IDCODE instruction:

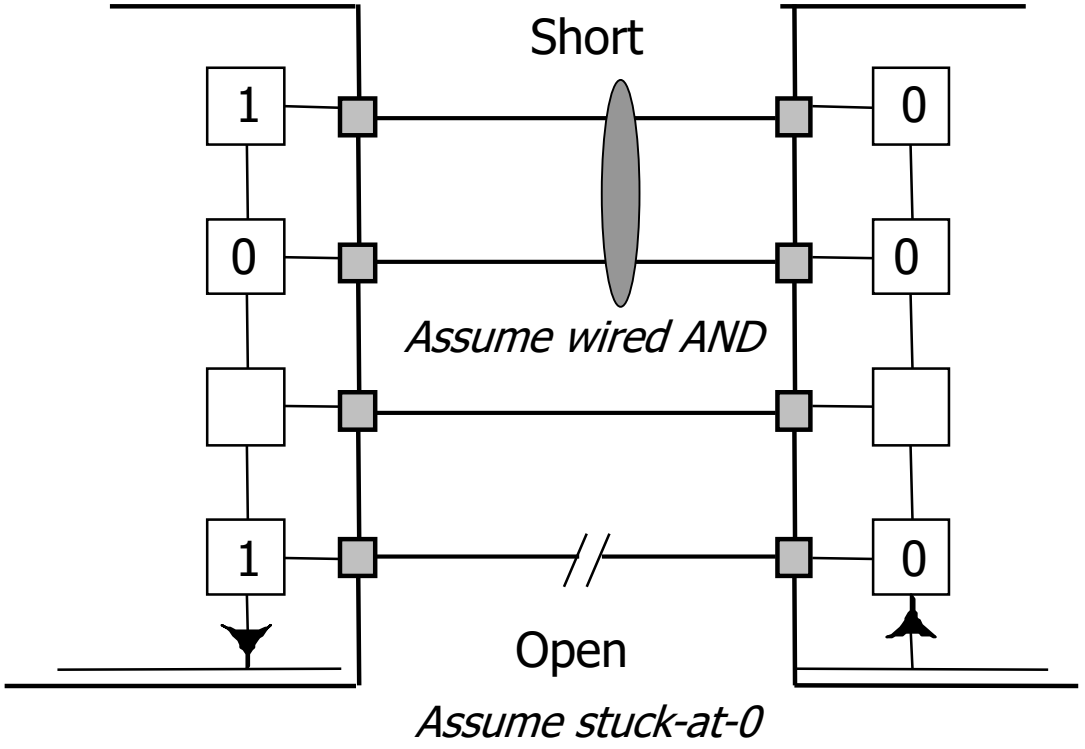
Connects the component device identification register serially between TDI and TDO in the Shift-DR TAP controller state

Allows board-level test controller or external tester to read out component ID

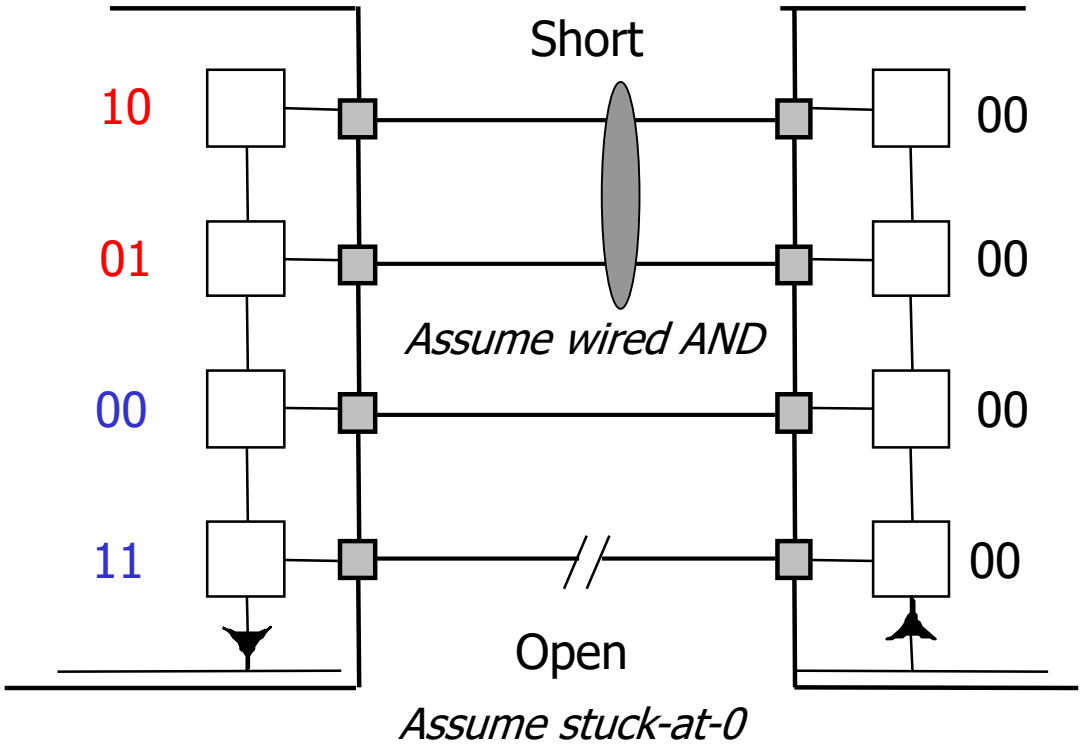
Required *whenever a JEDEC identification register is included in the design*



Fault Detection with Boundary Scan



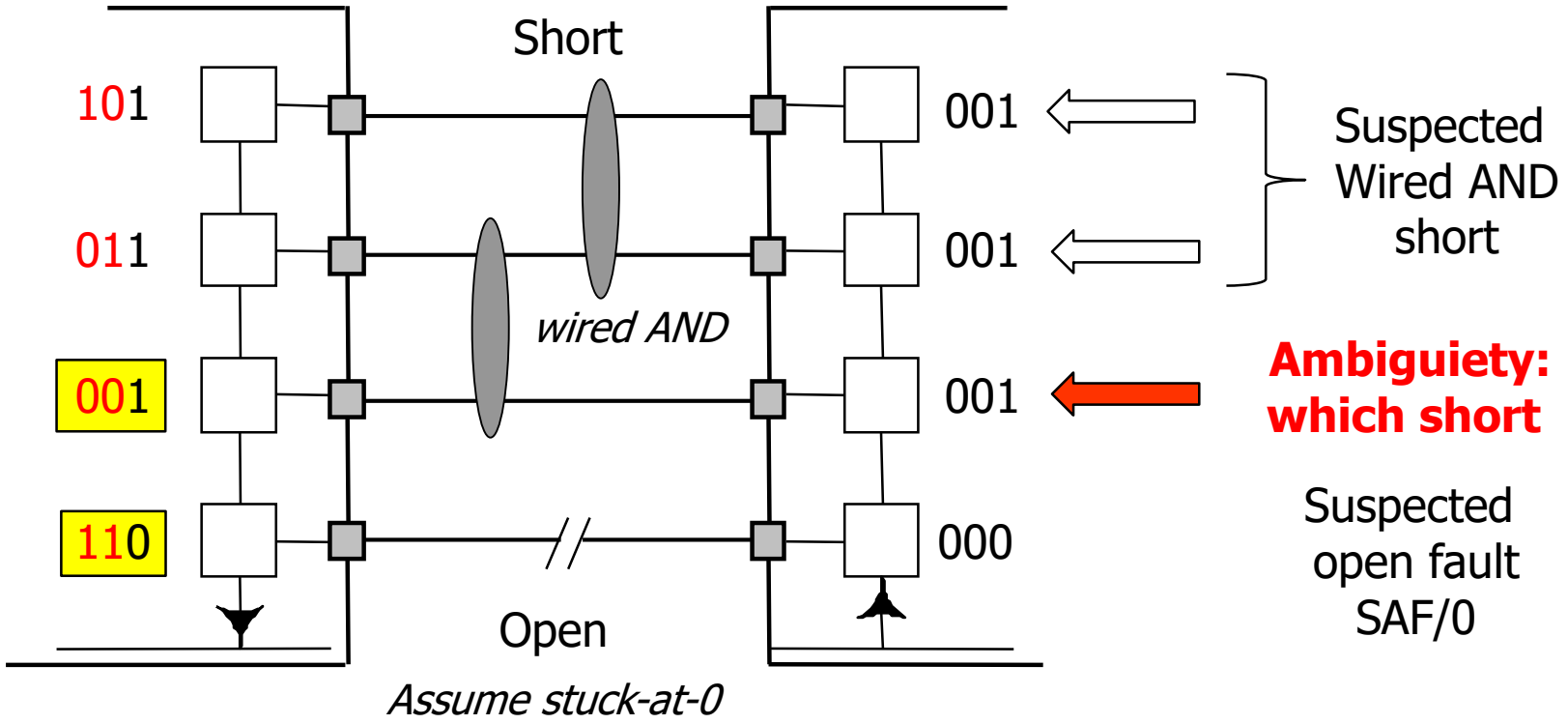
Any Bridge Detection with Boundary Scan



The problem is:
which fault
Open
(SAF/0)
or
Short

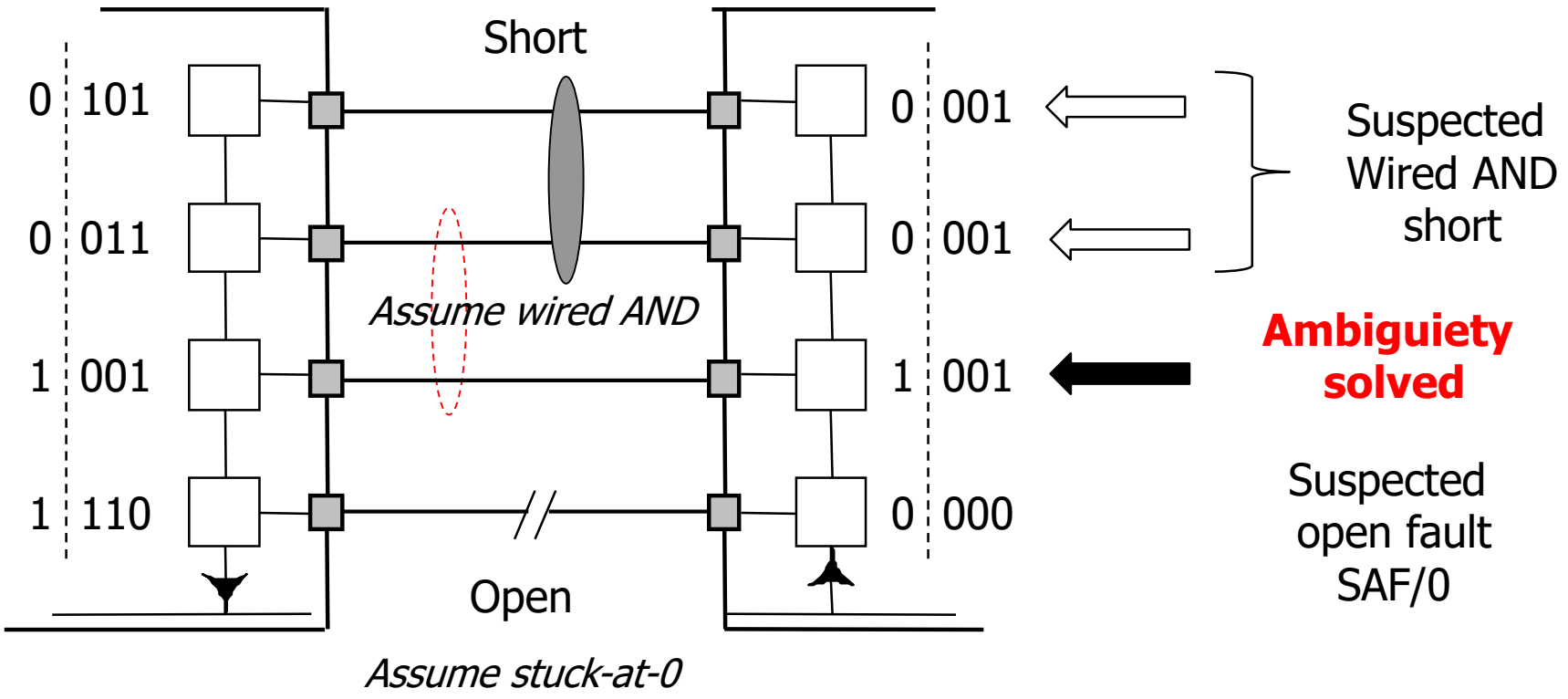
Kautz showed in 1974 that a sufficient condition to detect any pair of short circuited nets was that the "horizontal" codes must be unique for all nets. Therefore the test length is $]\log_2(N)[$

Any Fault Detection with Boundary Scan



All 0-s and all 1-s are forbidden codes because of stuck-at faults
Therefore the final test length is $]\log_2(N+2)[$ (for testing SAF without masking by shorts)

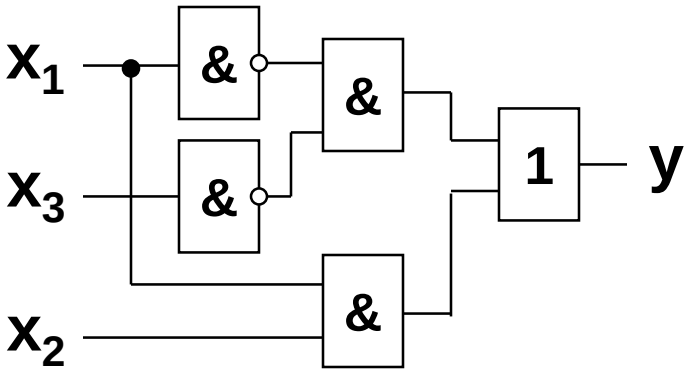
Fault Diagnosis with Boundary Scan



To improve the diagnostic resolution we have to add one bit more

Synthesis of Testable Circuits

$$y = \overline{x_1}\,\overline{x_3} \vee x_1 x_2$$



Test generation:

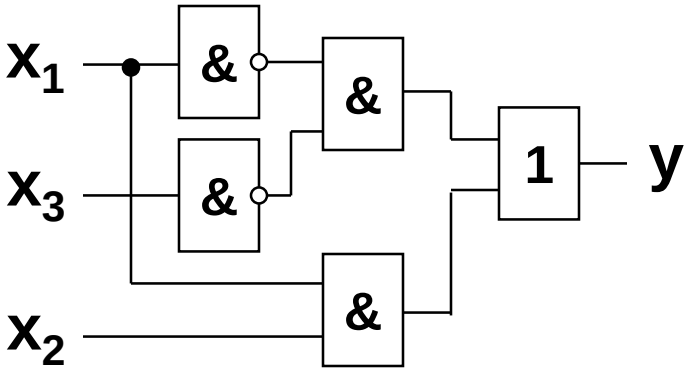
$y = \overline{x_1}\,\overline{x_3} \vee x_1 x_2$				x_1	x_2	x_3	y
0	1	1	0	1	0	0	0
1	0	0	1	0	1	1	0
1	1	0	0	0	0	1	1
0	0	1	1	1	1	1	1

4 test patterns are needed

Synthesis of Testable Circuits

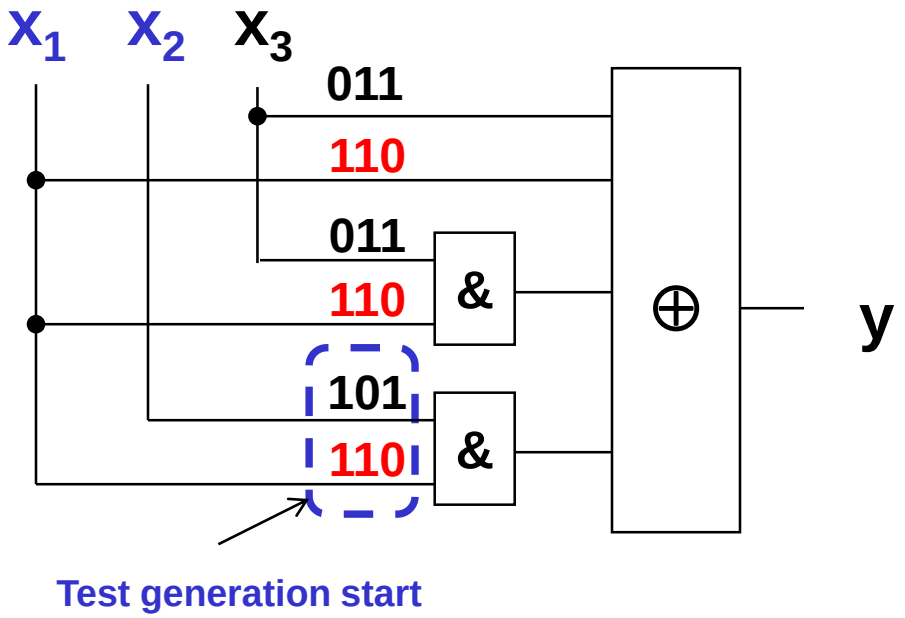
Two implementations for the same circuit:

$$y = \overline{x_1 x_3} \vee x_1 x_2$$



Here:

4 test patterns are needed



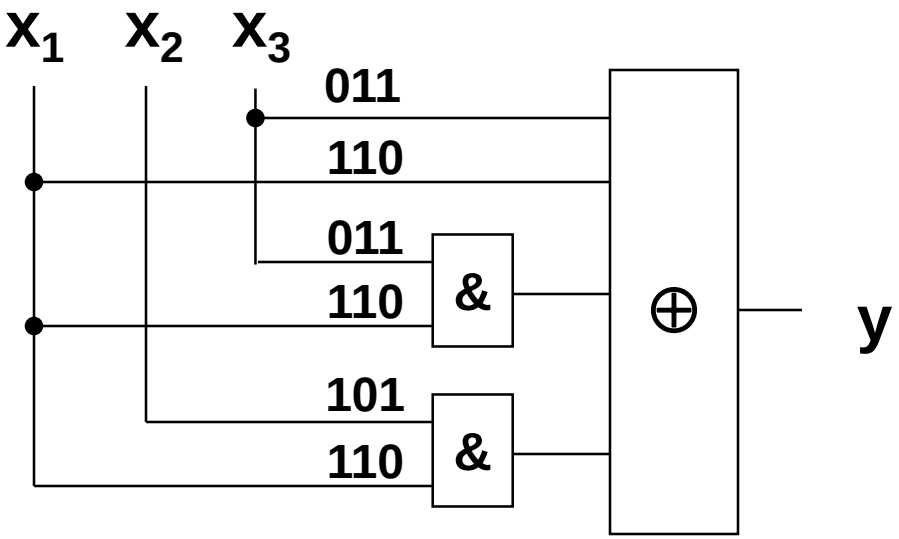
Here:

Only 3 test patterns are needed

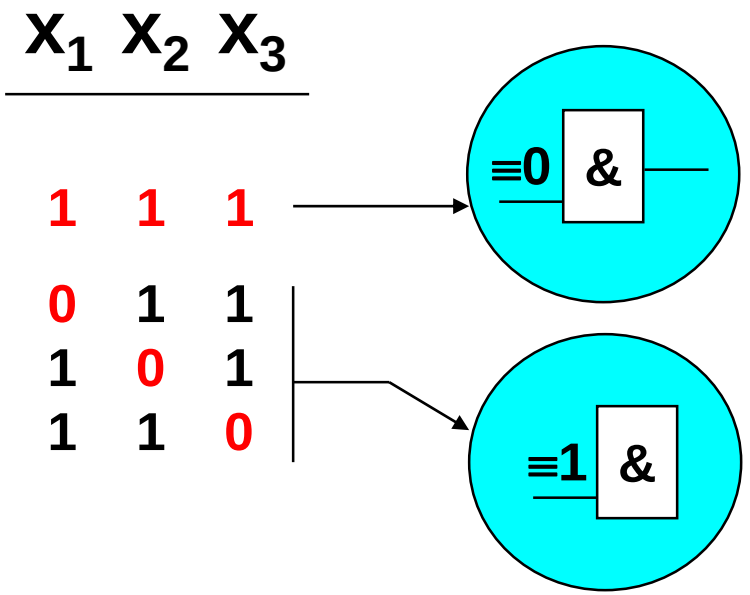
Synthesis of Testable Circuits

Test generation method:

$$y = 1 \oplus x_3 \oplus x_1 \oplus x_1 x_3 \oplus x_1 x_2$$



Roles of test patterns:



Synthesis of Testable Circuits

Given: $y = \overline{x_1}\,\overline{x_3} \vee x_1 x_2$

$$y = c_0 \oplus c_1 x_3 \oplus c_2 x_2 \oplus c_3 x_2 x_3 \oplus c_4 x_1 \oplus c_5 x_1 x_3 \oplus c_6 x_1 x_2 \oplus c_7 x_1 x_2 x_3$$

Calculation of constants:

f_i	x_1	x_2	x_3	y	Σ	
f_0	0	0	0	1	1	$C_0 = f_0 = 1$
f_1	0	0	1	0	1	$C_1 = f_0 \oplus f_1 = 1$
f_2	0	1	0	1	0	$C_2 = f_0 \oplus f_2 = 0$
f_3	0	1	1	0	0	$C_3 = f_0 \oplus f_1 \oplus f_2 \oplus f_3 = 0$
f_4	1	0	0	0	1	$C_4 = f_0 \oplus f_4 = 1$
f_5	1	0	1	0	0	$C_5 = f_0 \oplus f_1 \oplus f_4 \oplus f_5 = 1$
f_6	1	1	0	1	1	$C_6 = f_0 \oplus f_2 \oplus f_4 \oplus f_6 = 1$
f_7	1	1	1	1	0	$C_7 = f_0 \oplus f_1 \oplus f_2 \oplus f_3 \oplus f_4 \oplus f_5 \oplus f_6 \oplus f_7 = 0$

New circuit:

$$y = 1 \oplus x_3 \oplus x_1 \oplus x_1 x_3 \oplus x_1 x_2$$