

**Quiz No. 5 is
NEXT WEEK**

Lecture 23

EE 421 / CS 425

Digital System Design

Fall 2024

Shahid Masud

Topics

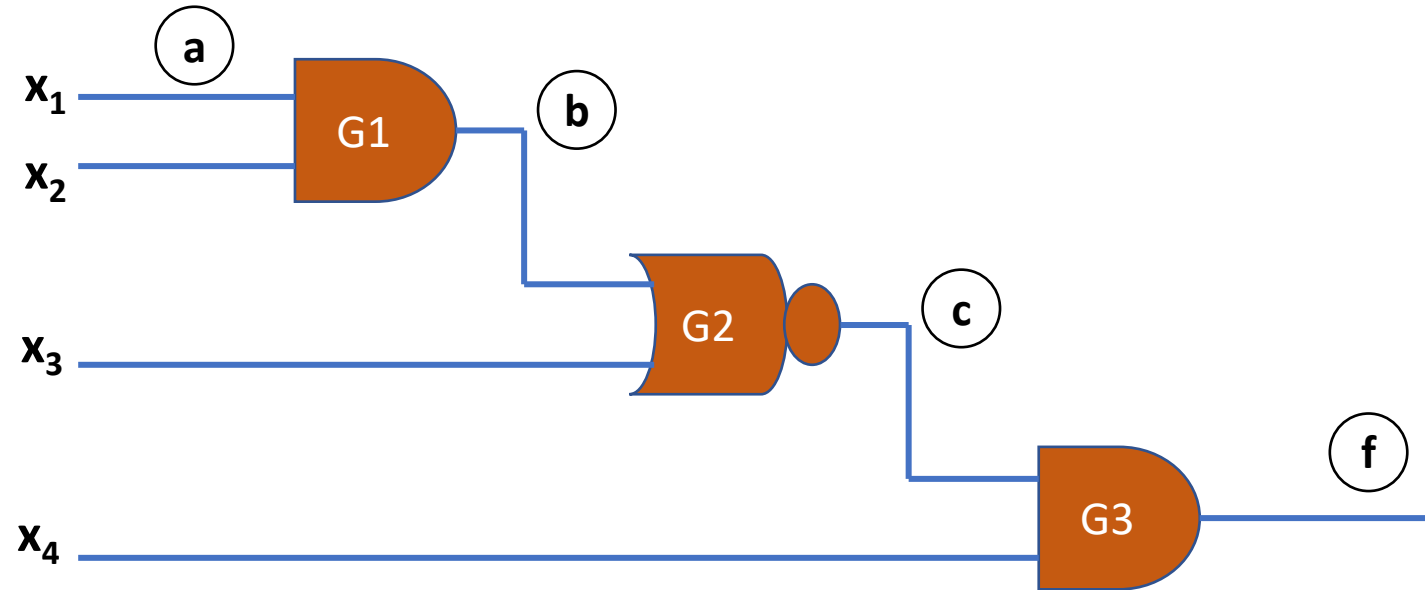
- **Recap** – Path Sensitization Method for Fault Test Generation
- Examples of XOR based test set generation
- Design for testability
- BIST and SCAN technique
- Pseudorandom test vector generation through ALFSR

**Quiz No. 5 is
NEXT WEEK**

Three Steps in Path Sensitization Method

- **Fault Excitation:** Which vector to be induced to detect the suspected SA0 or SA1 fault at the suspicious path
- **Fault Propagation:** Identify path/s through which fault can be propagated to the observable output
- **Back tracking:** Move back from output towards all inputs and assign appropriate test values

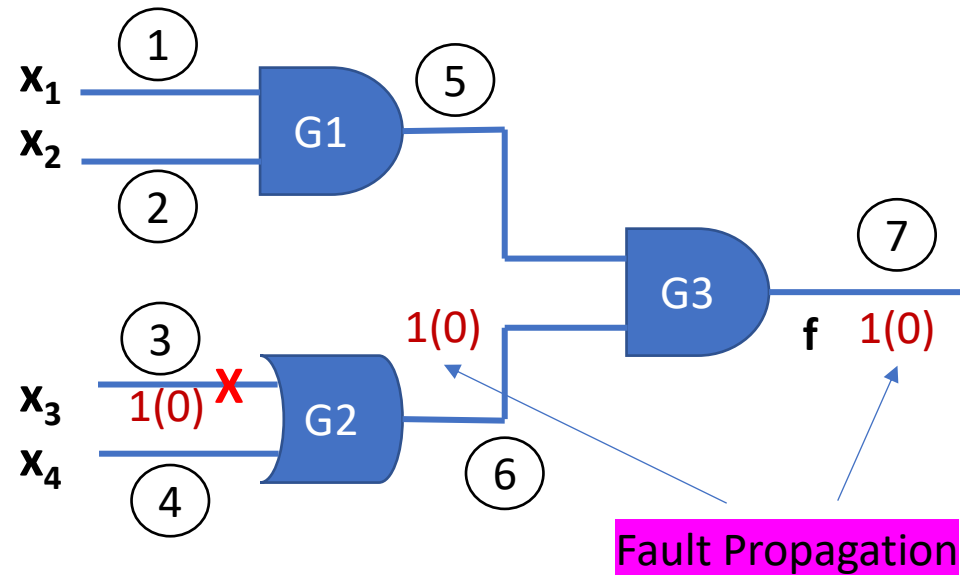
Path Sensitization – how to



To sensitize a path through an input of AND gate or NAND gate, all other inputs must be set to '1'

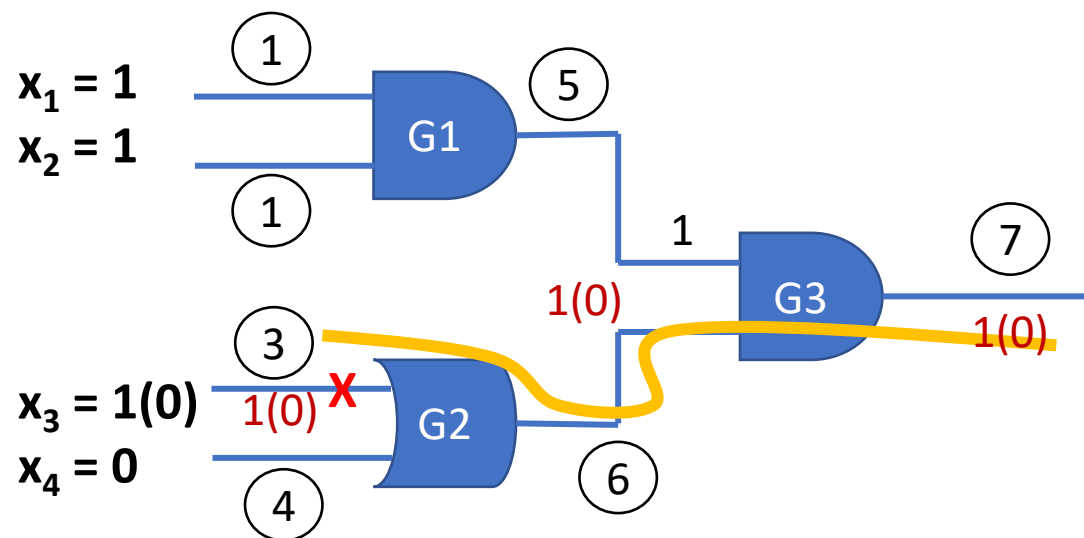
To sensitize a path through an input of OR gate or NOR gate, all other inputs must be set to '0'

Path Sensitization – Example 1



Purpose: To detect **SA0** fault at **wire 3** connected to input of OR gate
 Input x_3 is selected opposite to Stuck-At fault (eg. SA0), written as **$x_3=1(0)$**
This is fault generation or excitation

Test Vectors for Example 1



Sensitized path = 3 \rightarrow 6 \rightarrow 7

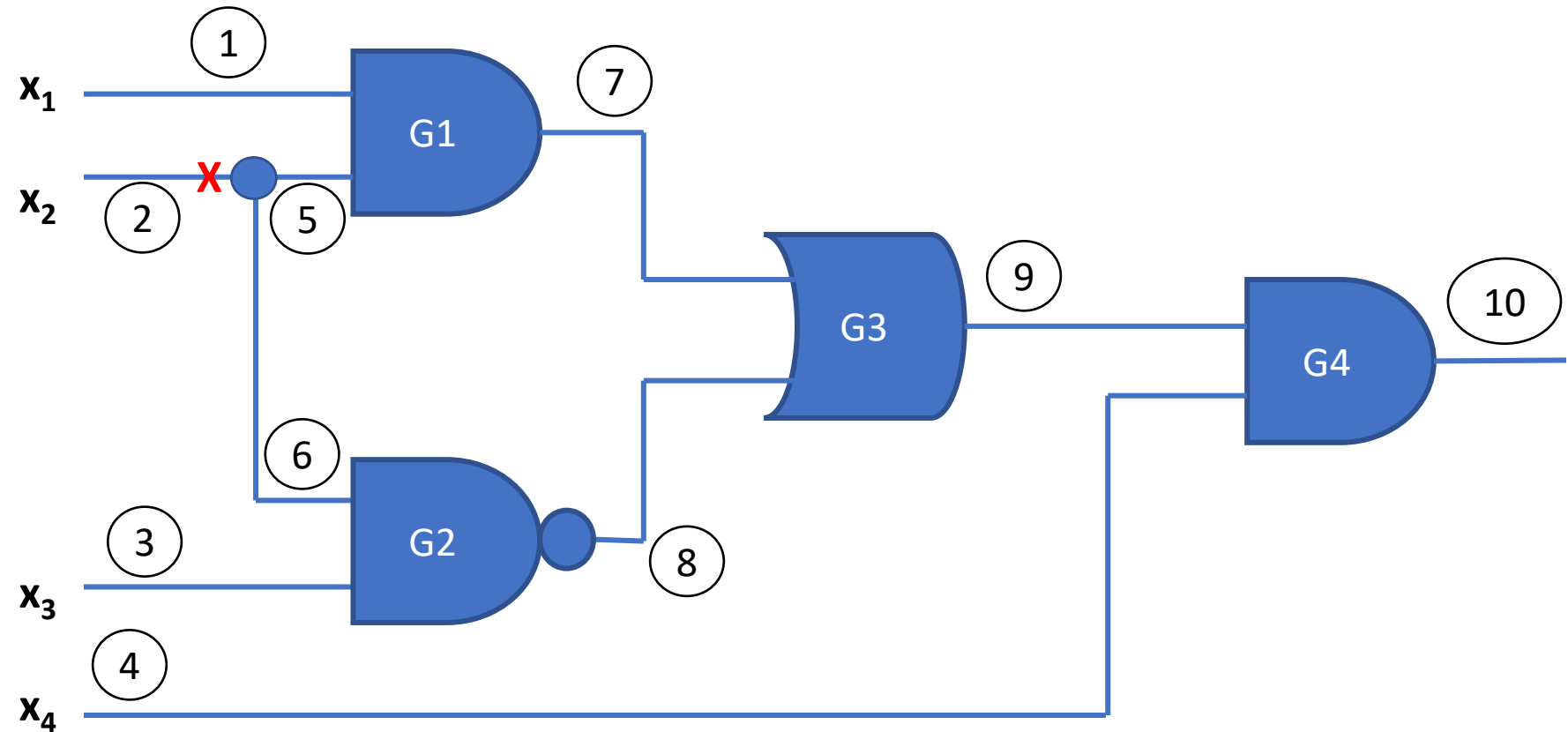
Back tracing reveals inputs to all gates to ensure fault propagation

Required test vector to detect SA0 at wire 3 is "1110"

Path Sensitization – Example 2

To detect SA1 fault at wire 2

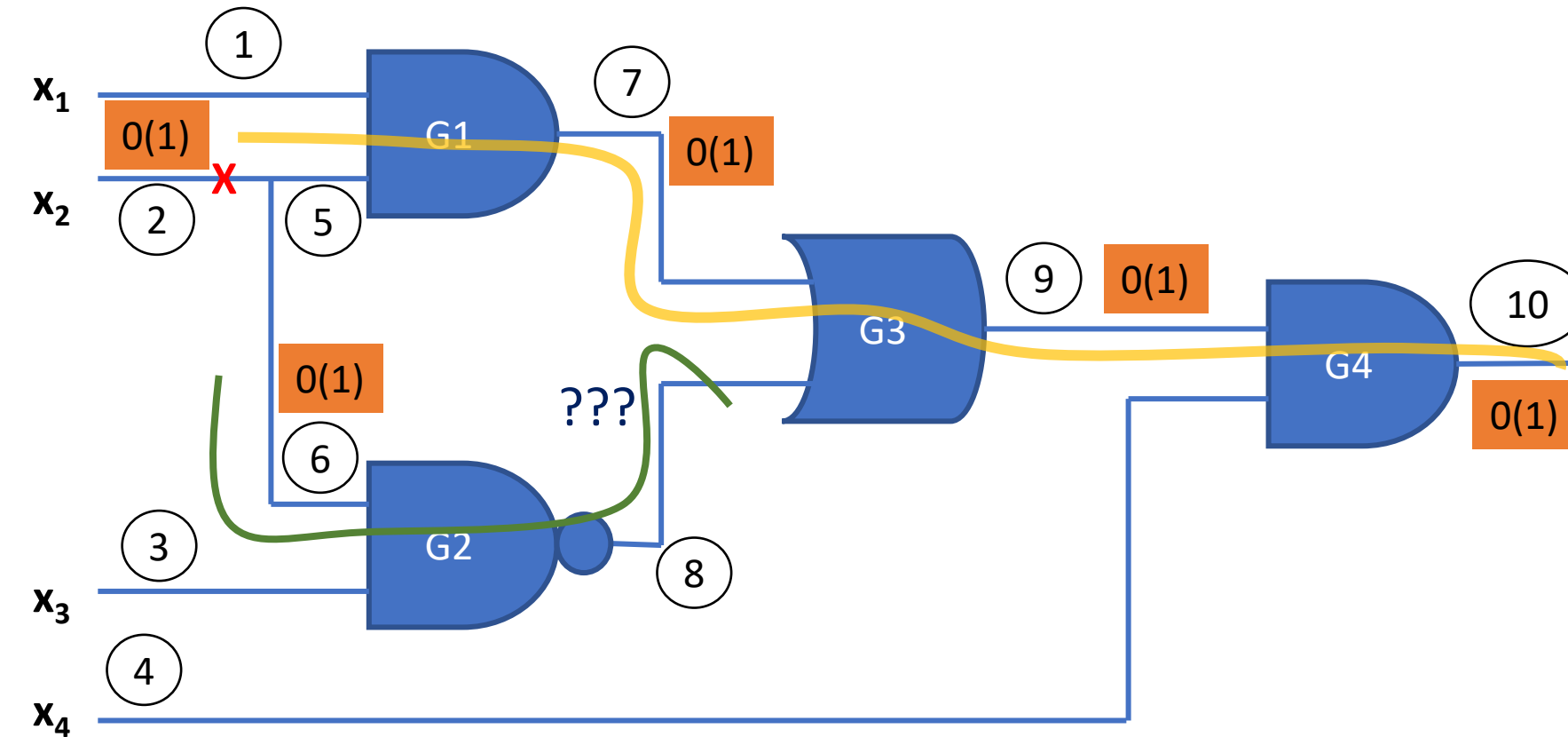
Two possible paths can be excited



continued

Detect SA1 at path 2

Fault propagation by supplying input $x_2=0(1)$



Case 1:

Back tracking reveals:

First Selected path = $2 \rightarrow 7 \rightarrow 9 \rightarrow 10$

But path 8=1 due to path 2 input x_2
This is not correct to have '1' at
Path 9. Hence '0' cannot be justified
at line 8 and line 2 simultaneously

This situation is **'Inconsistent'** hence
Some other path is thus required

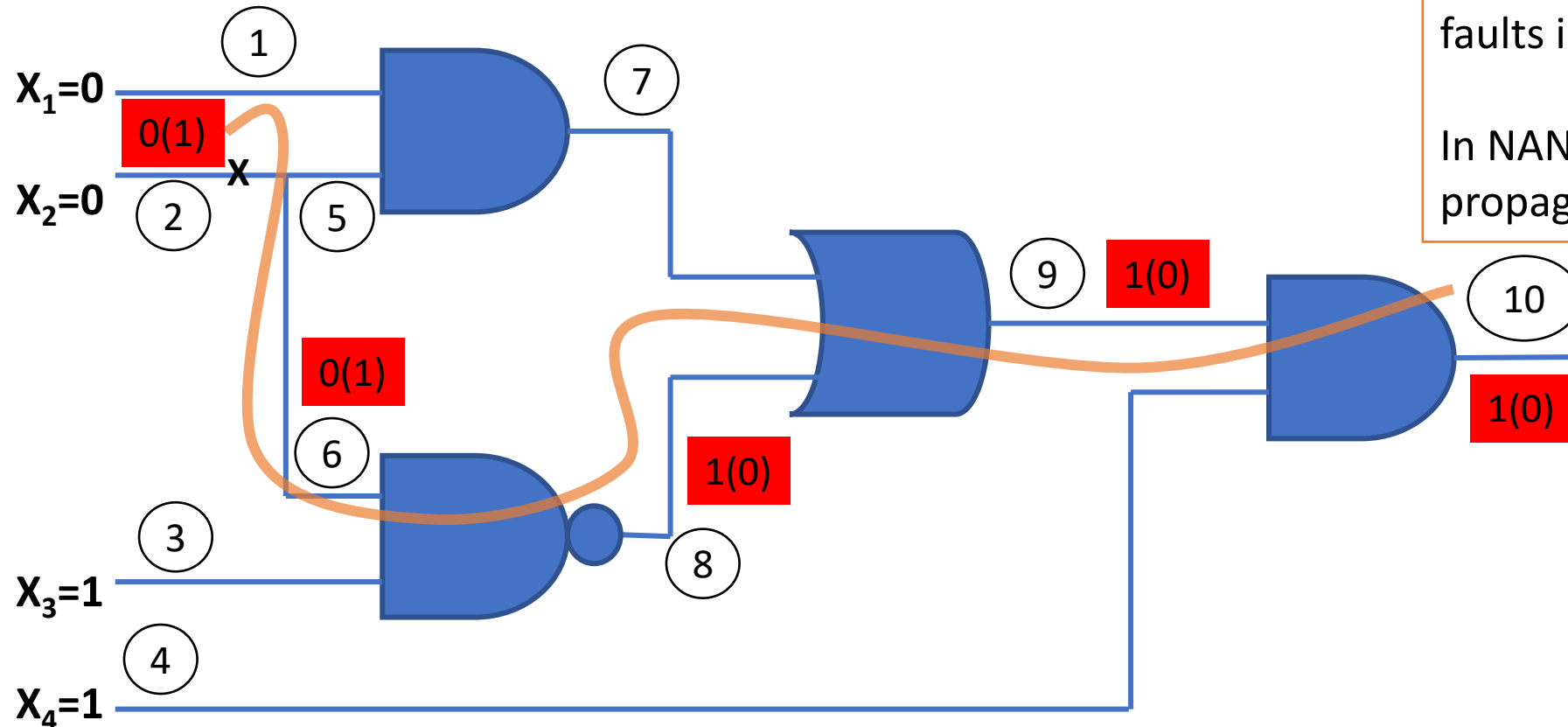
Continued – final test vectors

Selected path = 2 → 6 → 8 → 9 → 10

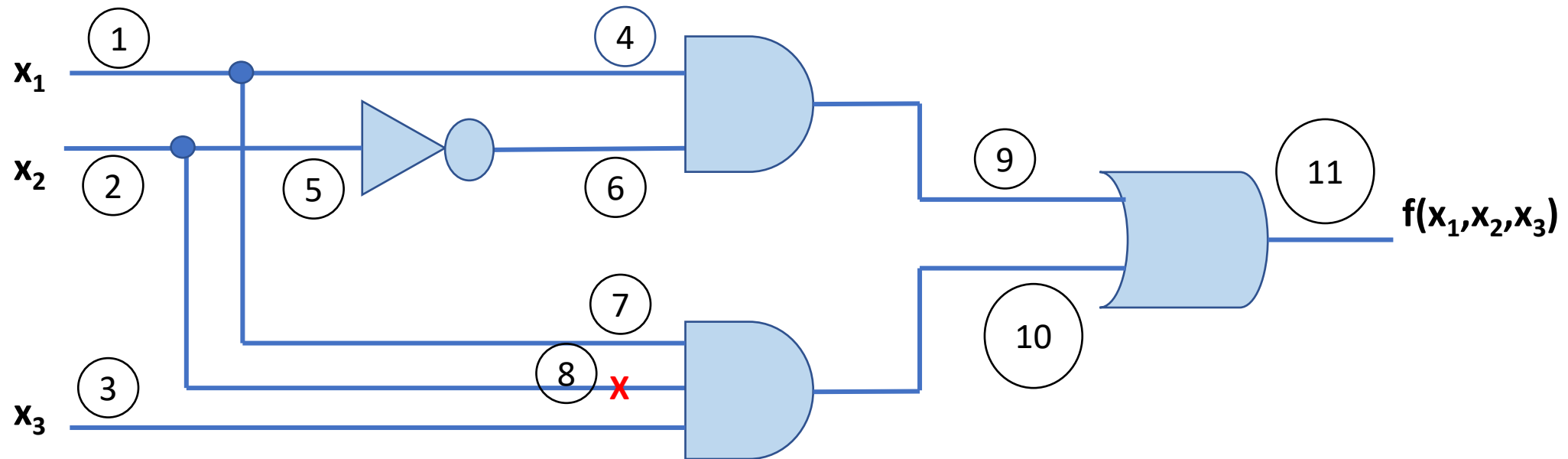
Test Vector = "0011"

This test vector can also reveal other faults in wires 6, 8 and 9

In NAND and NOR, reverse fault is propagated



Untestable Fault



Look at SA1 fault on path 8

This fault cannot be distinguished (sensitized) by changing inputs x_1 to x_3

Mathematically:

An untestable fault exists when $f^{8/1} \oplus f^8 = 0$

This condition means it is not possible to test this path

EXOR Method for Fault Test Generation

EXOR Method for Fault Test Generation

Define faulty and fault-free circuit as Boolean expressions

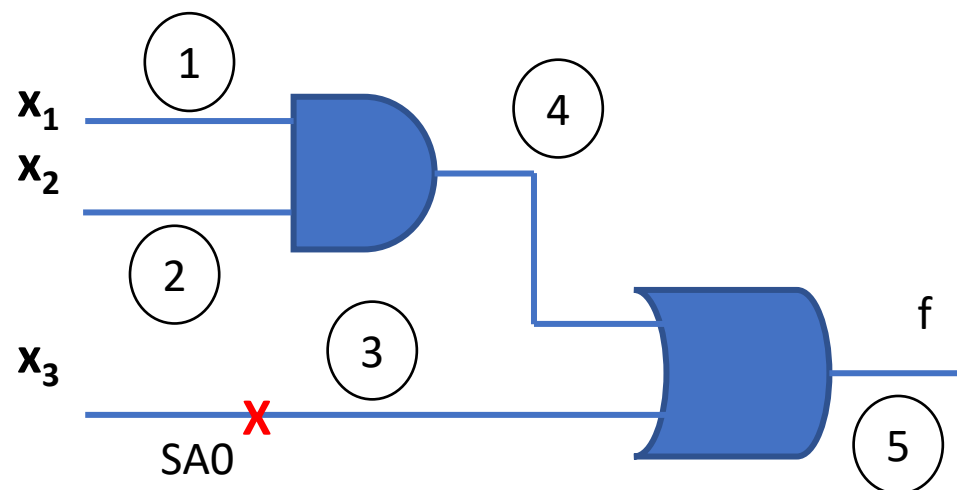
The **Exclusive OR of the two functions should be ZERO if they are same** i.e. NO FAULT

The **Exclusive OR of the two functions should be ONE if there is fault,** means different fault propagation

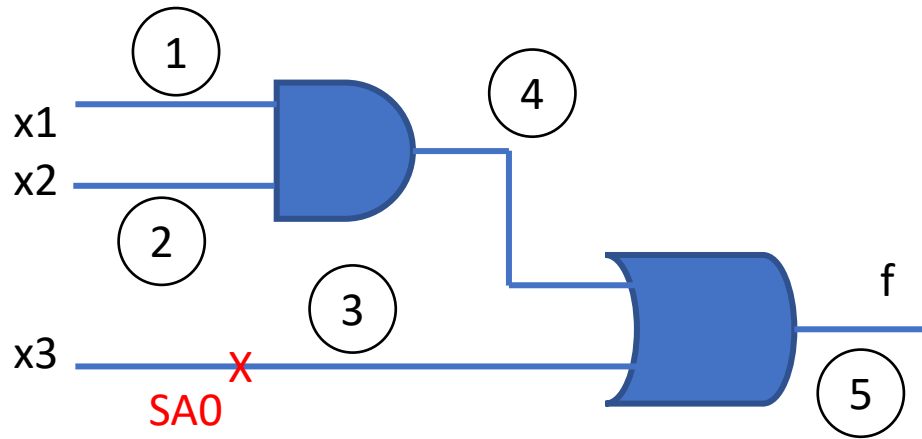
Premise:

Faulty Circuit must produce a different response from a fault-free circuit

Example SA0 fault on wire 3, input to OR gate in circuit below:



EXOR method using Boolean Algebra



$$\begin{aligned} F^{1/0} &= f \oplus f^{1/0} \\ &= (x1.x2 + x3) \oplus (x3) \\ &= x1.x2.x3' \end{aligned}$$

\Rightarrow test vector = 110

Similarly, test for 3/0 is:

$$\begin{aligned} F^{3/0} &= (x1.x2 + x3) \oplus (x1.x2) \\ &= (x1' + x2').x3 \\ &= (x1'.x3 + x2'.x3) \end{aligned}$$

Thus tests are: 001, 011, 101

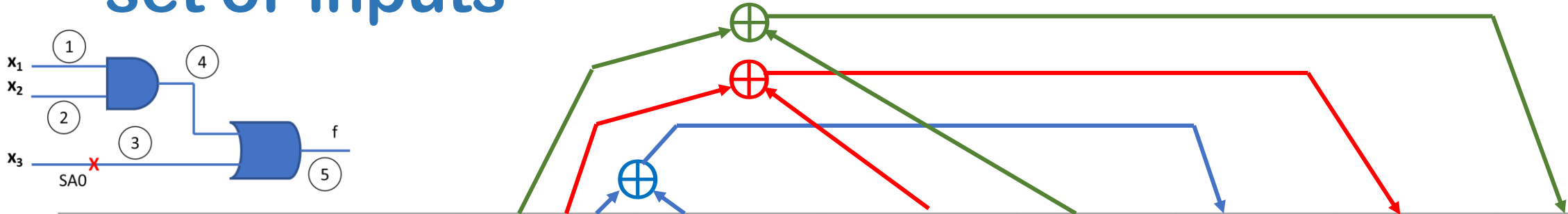
Solution

$$\begin{aligned}
 f \oplus f^{3/0} &= (x_1 x_2 + x_3) \oplus (x_1 x_2) \\
 &= (x_1 x_2 + x_3)' \cdot (x_1 x_2) + (x_1 x_2 + x_3) \cdot (x_1 x_2)' \\
 &= \cancel{0} \oplus \left(\underline{(x_1 x_2)'} \cdot \underline{(x_3)'} \right) \underline{(x_1 x_2)} + \underline{(x_1 x_2 + x_3)} \underline{(\overline{x_1} + \overline{x_2})} \\
 &= 0 + \boxed{\overline{x_1} x_3 + \overline{x_2} x_3}
 \end{aligned}$$

EXOR Method – The steps involved

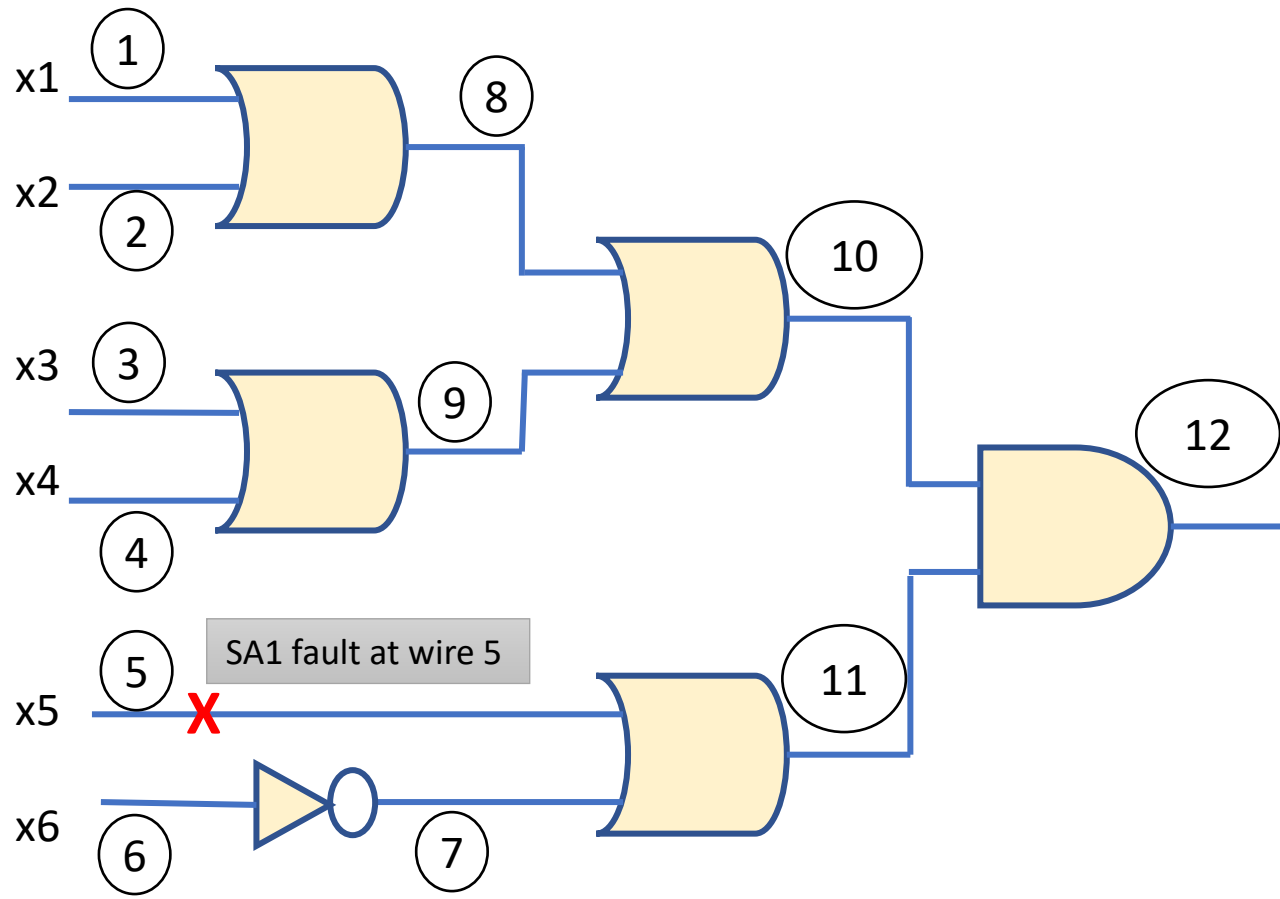
- Step 1: Construct truth table of fault-free f and faulty $f^{p/d}$
- Step 2: Compute $f \oplus f^{p/d}$ for each row of the truth table
- Step 3: Tests (input vectors) for fault p/d are indicated by the ones in the columns corresponding to $f \oplus f^{p/d}$
- Step 4: By expressing f and $f^{p/d}$ in Boolean algebra, an expression that gives all tests for p/d can be determined

Draw Fault Table – shows a set of faults and a set of inputs



Tests (inputs)			f output	$f^{1/0}$	$f^{2/1}$	$f^{3/0}$	$f \oplus f^{1/0}$	$f \oplus f^{2/1}$	$f \oplus f^{3/0}$
x_1	x_2	x_3		SA0 at 1	SA1 at 2	SA0 at 3			
0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0	0	1
0	1	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	1
1	0	0	0	0	1	0	0	1	0
1	0	1	1	1	1	0	0	0	1
1	1	0	1	0	1	1	1	0	0
1	1	1	1	1	1	1	0	0	0

Checking Testability through EXOR Equations



$$f = (x1 + x2 + x3 + x4)(x5 + x6')$$

$$f^{6/0} = (x1 + x2 + x3 + x4).1$$

$$\text{Test vector } F^{6/0} = f \oplus f^{6/0}$$

$$= [(x1 + x2 + x3 + x4).(x5 + x6')] \oplus [(x1 + x2 + x3 + x4).1]$$

$$= A.B + A.B'$$

$$\text{Note: } (x1 + x2 + \dots).(x5 + \dots).(x1 + x2 + \dots)' = 0$$

$$= (x1 + x2 + x3 + x4)(x5'.x6)$$

$$= x1x5'.x6 + x2x5'.x6 + x3x5'.x6 + x4x5'.x6$$

Many test vectors are there for $f^{6/0}$

Eg. 100001, 010001, 001001, 000101

Solving – find test set

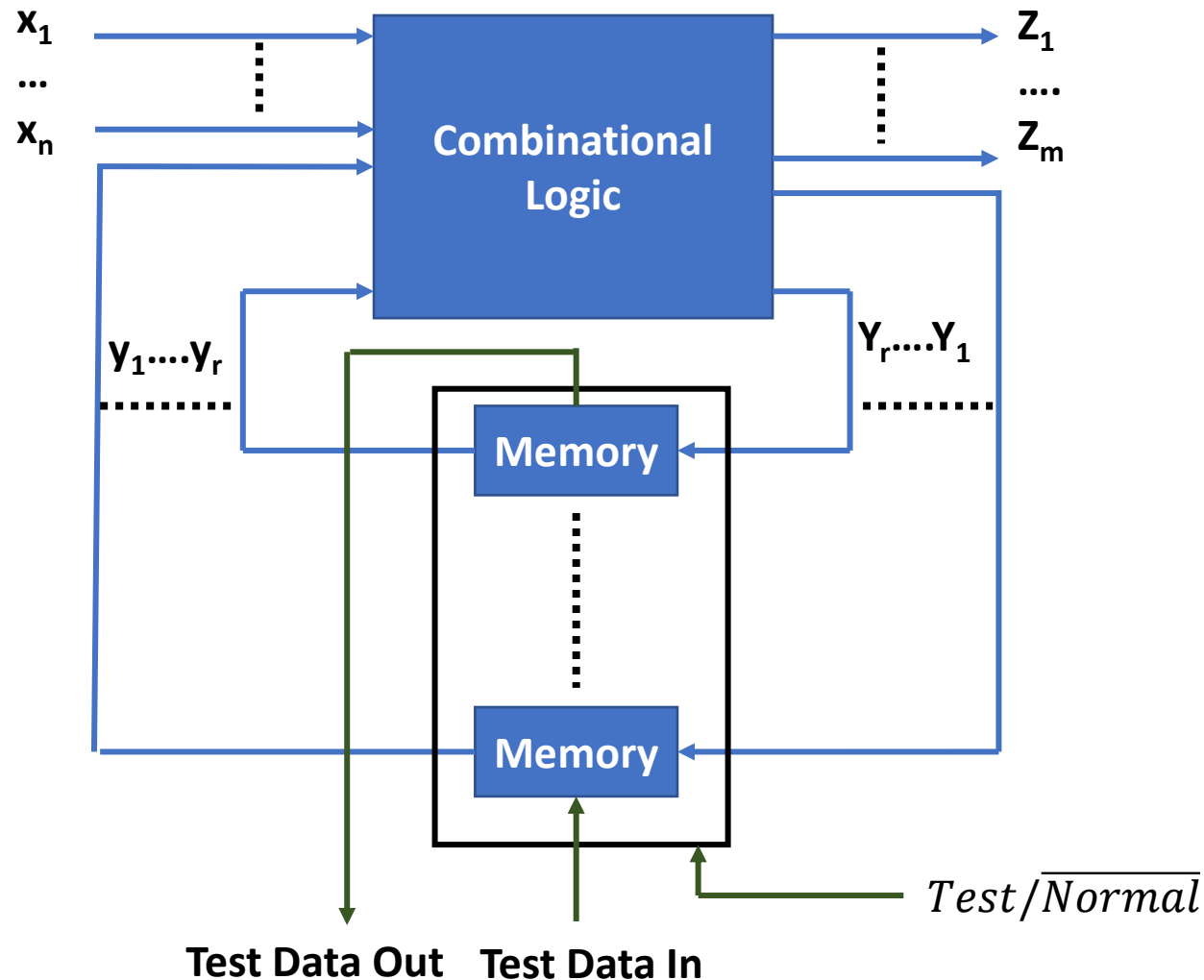
$$\begin{aligned}
 F^{6/0} &= f \oplus f^{6/0} = [(x_1 + x_2 + x_3 + x_4)(x_5 + \bar{x}_6)] \oplus [(x_1 + x_2 + x_3 + x_4)] \\
 &= [(x_1 + x_2 + x_3 + x_4)(x_5 + \bar{x}_6)]' \cdot [x_1 + x_2 + x_3 + x_4] \\
 &\quad + \underline{[(x_1 + x_2 + x_3 + x_4)(x_5 + \bar{x}_6)]} \cdot \underline{(x_1 + x_2 + x_3 + x_4)'} \\
 &= \underline{[(x_1 + x_2 + x_3 + x_4)'] + (x_5 + \bar{x}_6)'} \cdot \underline{(x_1 + x_2 + x_3 + x_4)} \\
 &= (x_5 + \bar{x}_6)' \cdot (x_1 + x_2 + x_3 + x_4) \\
 &= (x_5' \cdot x_6)(x_1 + x_2 + x_3 + x_4) = x_1 \bar{x}_5 x_6 + x_2 x_5' x_6 \\
 &\quad + x_3 \bar{x}_5 x_6 + x_4 \bar{x}_5 x_6
 \end{aligned}$$

Design for Testability – Built-In Self Test BIST

Design for Testability – Built-In Self Test BIST

- Design for testability means that the **designer** introduces such features in design that can be **later used for testing and debugging of circuits inside the chip**
- Special circuit elements that participate in the application of test vectors and capturing of results are **integrated into the design**
- Both sequential and combinational circuits can be tested

Sequential Machines with Scan Circuits



- Test/Normal' isolates flipflops from internal Combinational logic and inputs $x_1 \dots x_n$
- Special I/O Test Data In and Test Data out
Provide a known signature pattern that is Driven into the scan chain formed by connecting Inputs and outputs of all flipflops
- The received data at Test Data Out is compared With string provided at Test Data In after r clock cycles

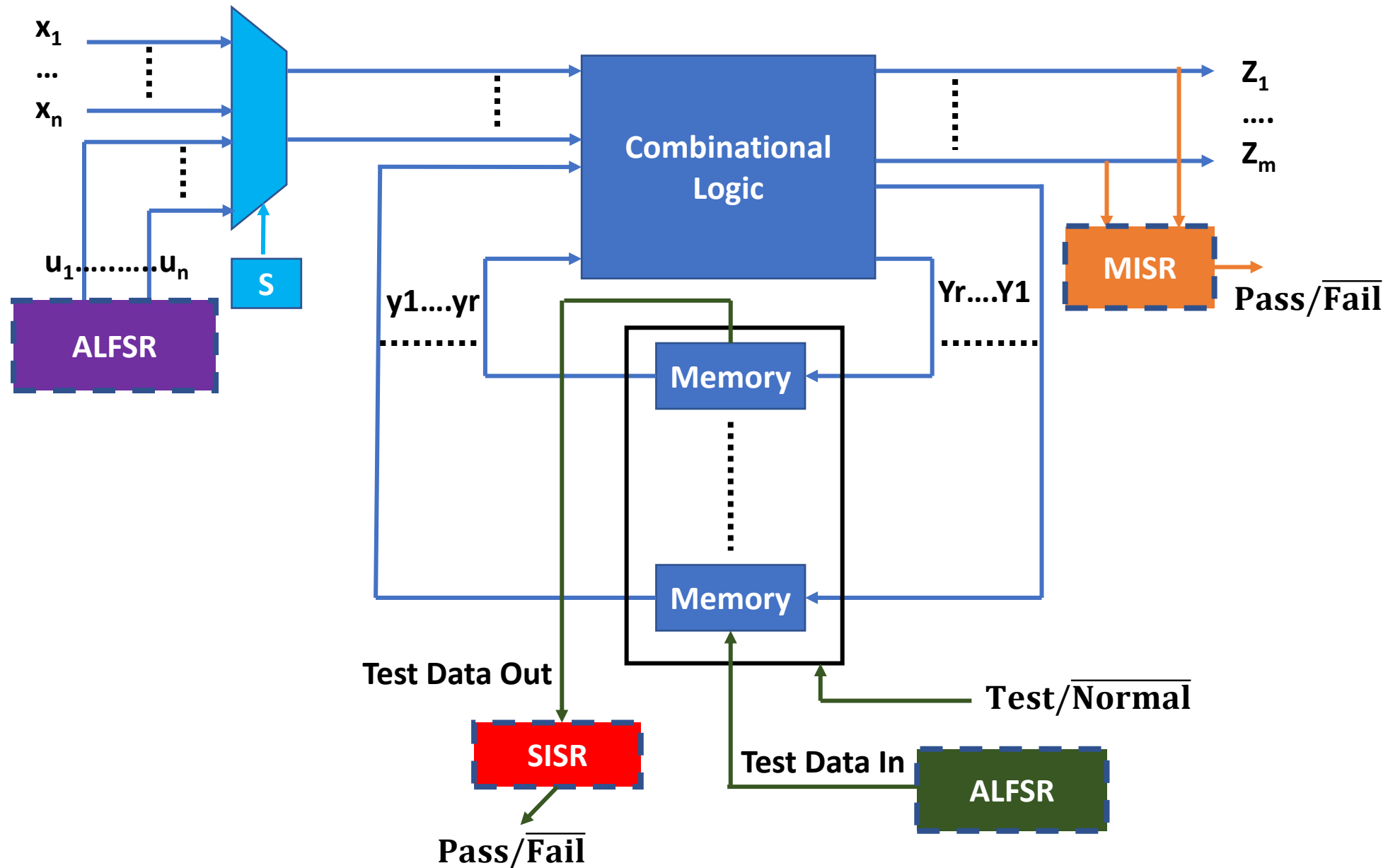
Scan Path operation for testing state machines

- Allows testing of combinational and sequential logic
- Set $Test/\overline{Normal}$ mode to Test
- Shift pattern into y_1, y_2, \dots, y_n into flip flops through Scan Input to put them into known defined state
- Apply input pattern through primary inputs x_1, x_2, \dots, x_n
- Observe primary outputs z_1, z_2, \dots, z_m
- Set $Test/\overline{Normal}$ to Normal mode
- Clock the circuit to force one transition,
- And so on

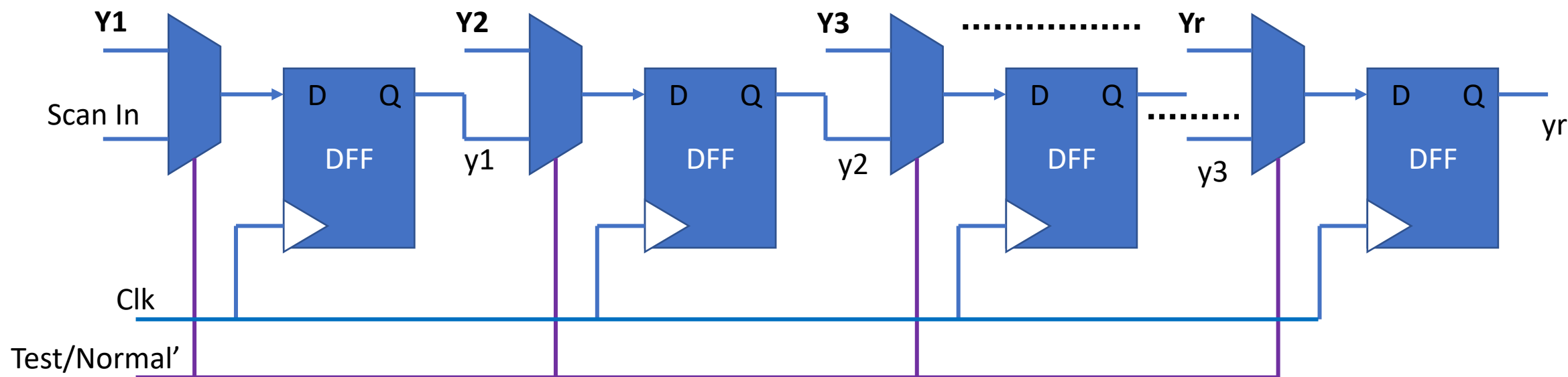
BIST – Built In Self Test

- Test pattern generator at Scan Input port as well as response capture and comparison **circuitry is placed inside the chip**. Thus, testing can be done without any other external mechanism, setting appropriate test mode on chip
- Pseudorandom test patterns are generated using ALFSR (**Autonomous Linear Feedback Shift Register**) circuits. These are applied at Scan In as well as inputs to combinational logic blocks
- Data from primary output is captured and compressed by MISR (**Multi Input Signature Register**)
- Scan Output is captured and compressed by SISR (**Serial Input Shift Register**)

Designing with BIST Circuits



Scan Path Register Design



- y_1, y_2, \dots, y_r are inputs to combinational logic
- A 2:1 Multiplexer is added to the input of each flip flop
- In Normal' mode, the Y_1, Y_2, \dots, Y_r inputs are connected to the D input of flip flops
- In Test mode, each flip flop is connected to the output of previous flipflop, forming scan chain
- Serial shift register formed in Test mode is called 'Scan Path'

Pseudorandom Test Pattern Generation

- Uses **ALFSR** – Autonomous Linear Feedback Shift Register
- An n-stage ALFSR produces a periodic **pseudorandom sequence** of n-bit binary numbers according to a special generating function that is realized through feedback lines and XOR gates within the ALFSR
- $a_{n-1}, a_{n-2}, \dots, a_0$ are the outputs of the n flip flops of the n-bit shift register with 'an' the input to the shift register equal to the exclusive OR of the feedback signals:

$$\text{i.e. } a_n = \sum_{i=0}^{n-1} a_i c_i$$

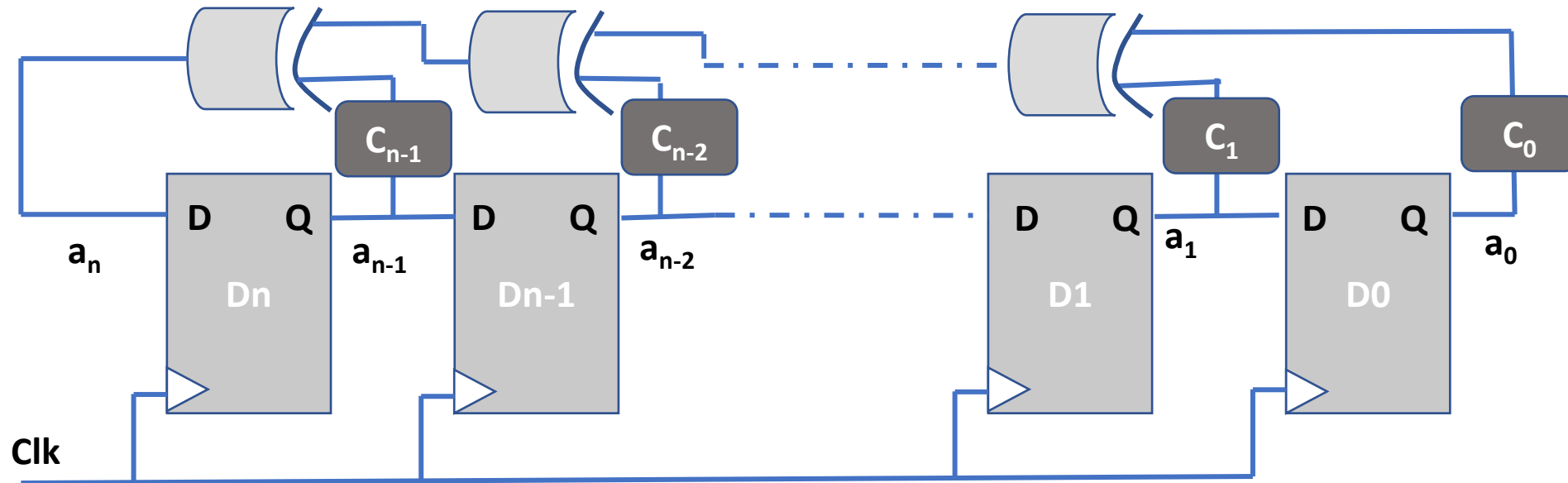
$$= a_0 c_0 \oplus a_1 c_1 \oplus a_2 c_2 \oplus \dots \oplus a_{n-1} c_{n-1}$$

The coefficients c_0, c_1, \dots, c_{n-1} represent the primitive polynomial $p(x)$

$C_i=1$ means flip flop output a_i is connected to flip flop input through the Exclusive OR gate

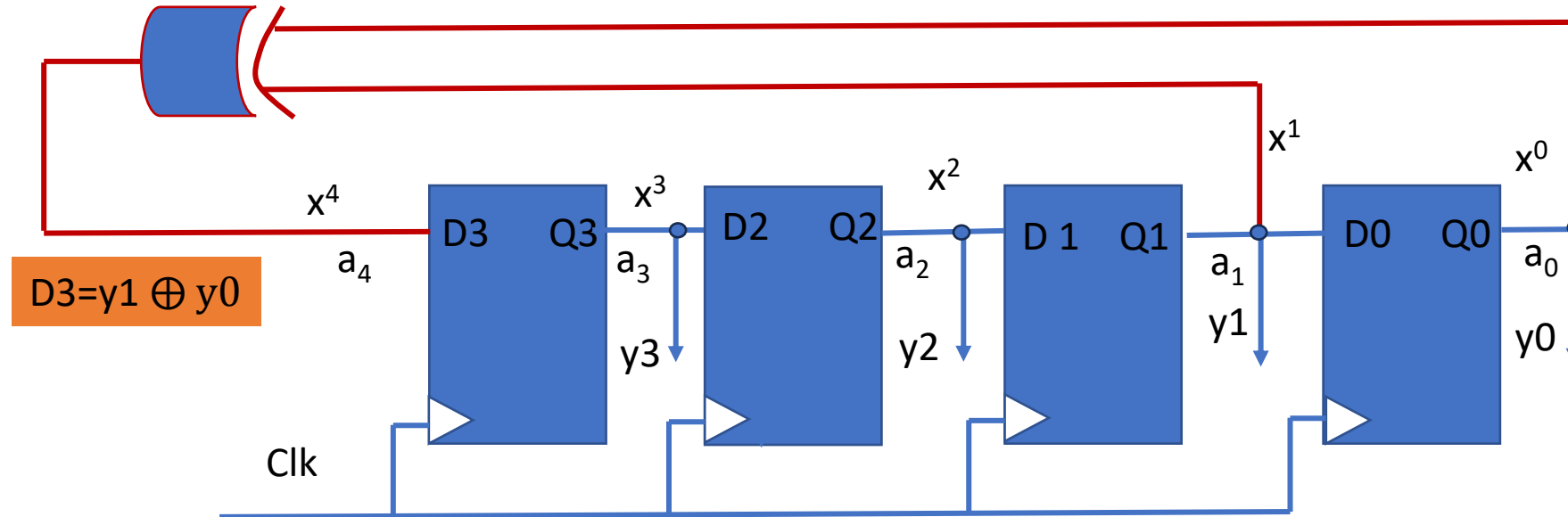
$C_i=0$ means flip flop output is NOT connected to the feedback circuit and there is no exclusive or gate at this point

General Structure of ALFSR



$$f(x) = a_n x^n + \dots + a_3 x^3 + a_2 x^2 + a_1 x^1 + a_0$$

Check ALFSR Sequence with given circuit or polynomial and seed value



Polynomial

$$P(x) = x^4 + x + 1$$

$$\text{Or } f(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0x^0$$

Use seed value $y_3y_2y_1y_0 = 1000$ that is preset at start

Generated sequence from this ALFSR circuit

$$D3 = y1 \oplus y0$$

Clock	D3=(y1⊕y0)	y3	y2	y1	y0
1	0	1	0	0	0
2	0	0	1	0	0
3	1	0	0	1	0
4	1	1	0	0	1
5	0	1	1	0	0
6	1	0	1	1	0
7	0	1	0	1	1
8	1	0	1	0	1
9	1	1	0	1	0
10	1	1	1	0	1
11	1	1	1	1	0
12	0	1	1	1	1
13	0	0	1	1	1
14	0	0	0	1	1
15	1	0	0	0	1
16	REPEAT	1	0	0	0

Seed Value 1000