Lecture 22

EE 421 / CS 425

Digital System Design

Fall 2024

Shahid Masud

Topics

- **Special Features in FPGA**
- Sequential Implementation on CLB
- Memory
- Multipliers
- DSP Slices
- FIR and Symmetric Filters

Recap and Summarize

- **Faults and Testing**
- Examples of Path Sensitization Method
- EXOR Method for Fault Generation
- What is Design for Testability?
- BIST and SCAN technique

Specialized Modules in FPGAs

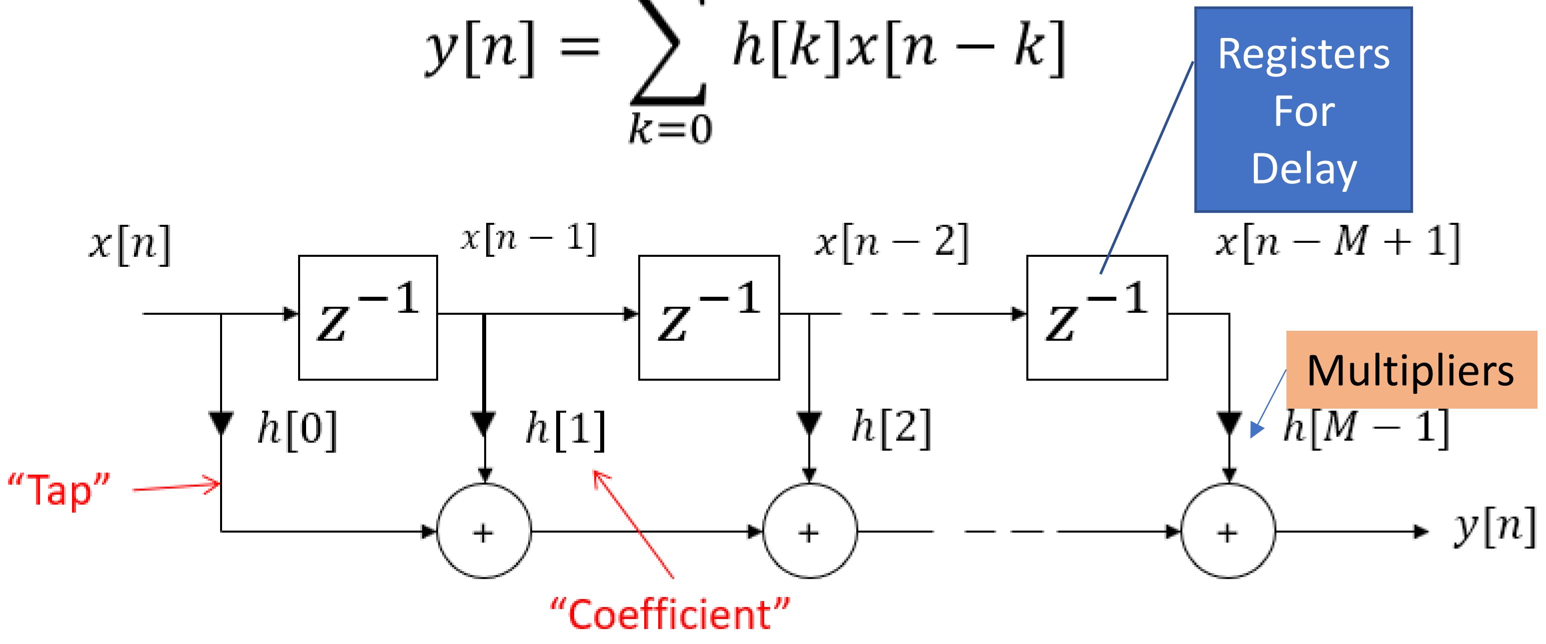
- Dedicated Memory
  - Single Port and Dual Port Embedded Memory Blocks – Block RAM
- Dedicated Arithmetic Units
  - Adders, Multipliers, Multipliers – Accumulators, Fast Carry Logic
- Digital Signal Processing Blocks – DSP Slice
  - FFT Butterfly Modules, FIR / IIR Filters, ALU, Floating Point Arithmetic
  - IP Core Libraries for Encryption, Video Compression, Cloud Applications, etc.
- Embedded Microprocessors
  - PowerPC, Microblaze, NIOS, ARM, MIPS, etc.
- Content Addressable Memory (CAM)
  - used in Branch Prediction, Caches inside CPU
- More and more features keep appearing in new FPGA devices
  - High Speed Interfaces, Security Features, RISC-V Support, etc.

DSP Features in modern FPGA
Example FIR Filter Implementation

FIR Filter Design

- FIR system is easily implemented directly from convolution summation

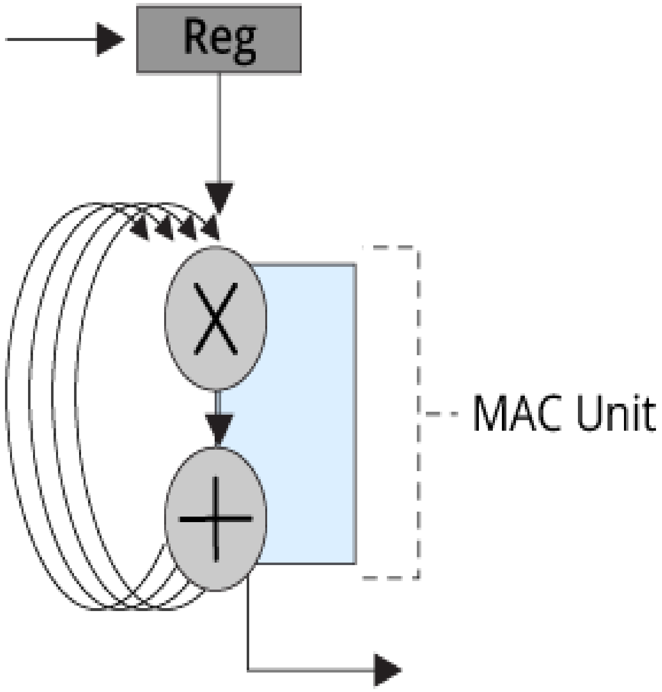
$$y[n] = \sum_{k=0}^{M-1} h[k]x[n-k]$$

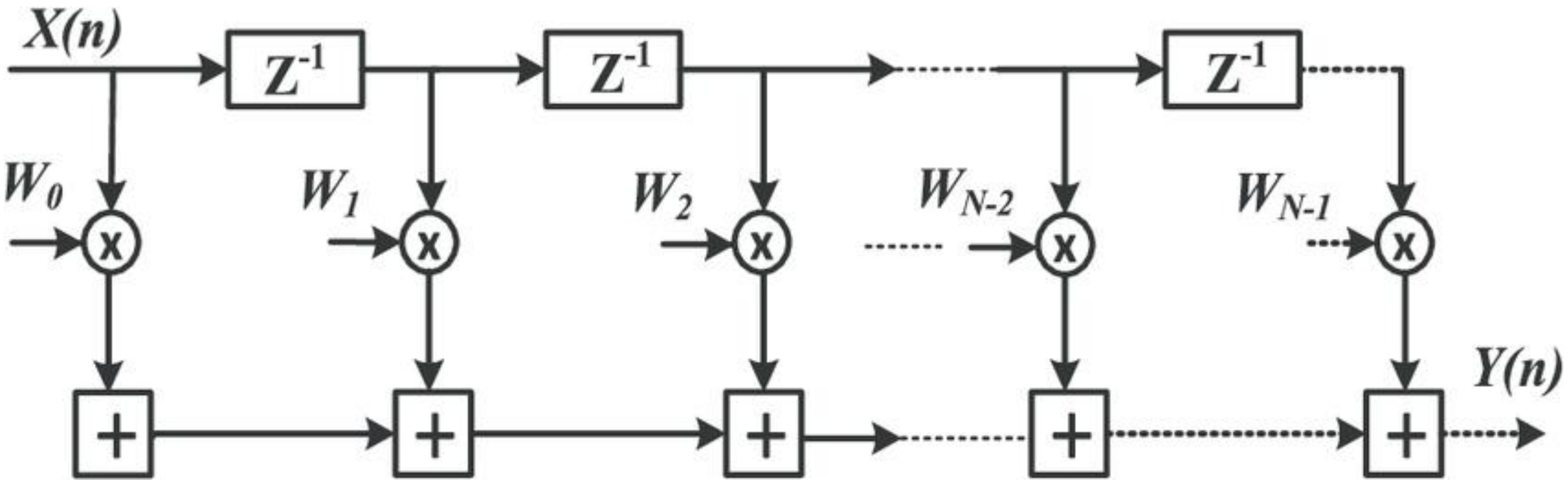


Implementation of DSP Filters

Implementation of FIR filters in Digital Signal Processing

MAC – Multiplier Accumulator

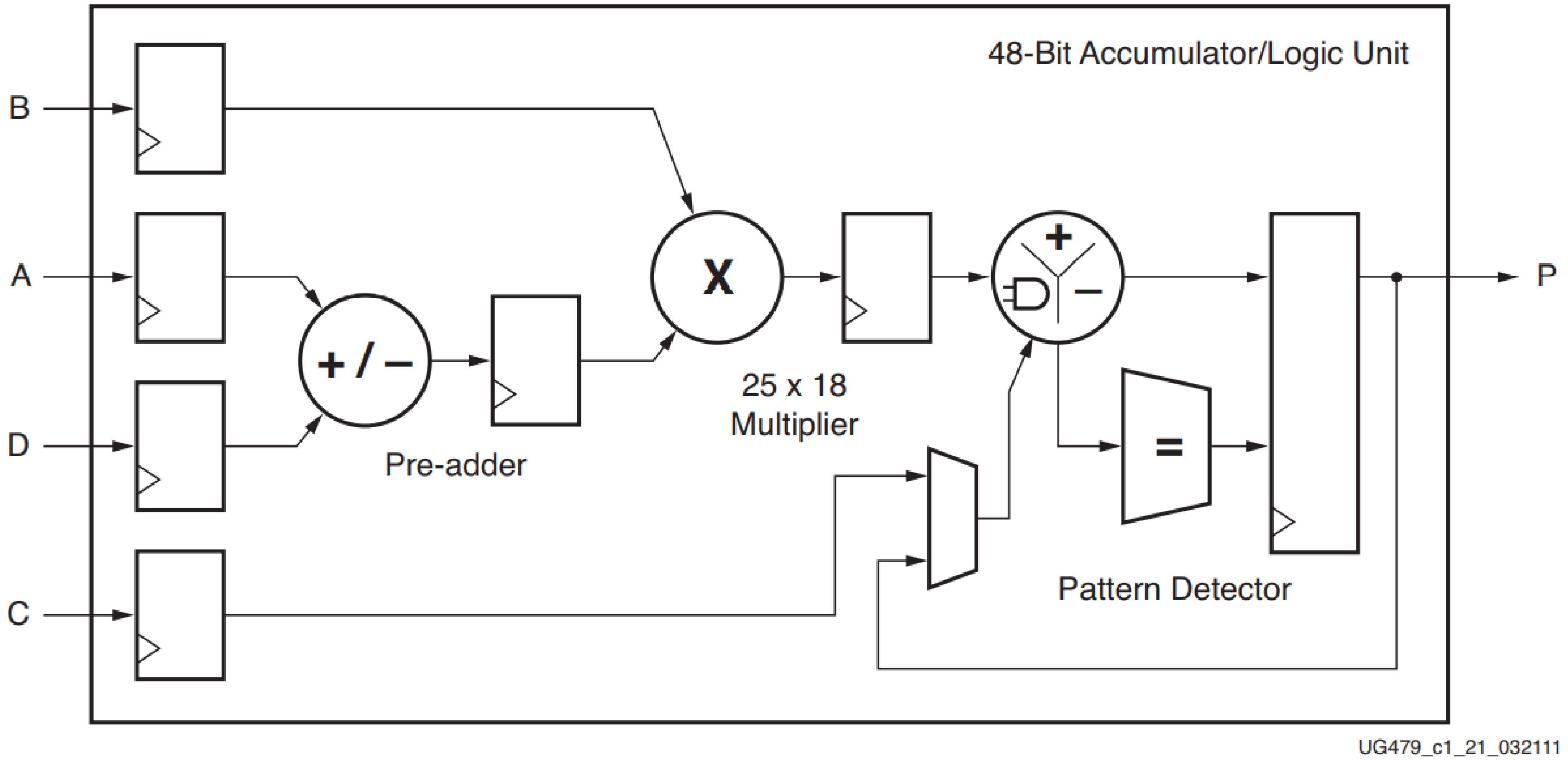




FIF filter mapping on
Software Programmable Device

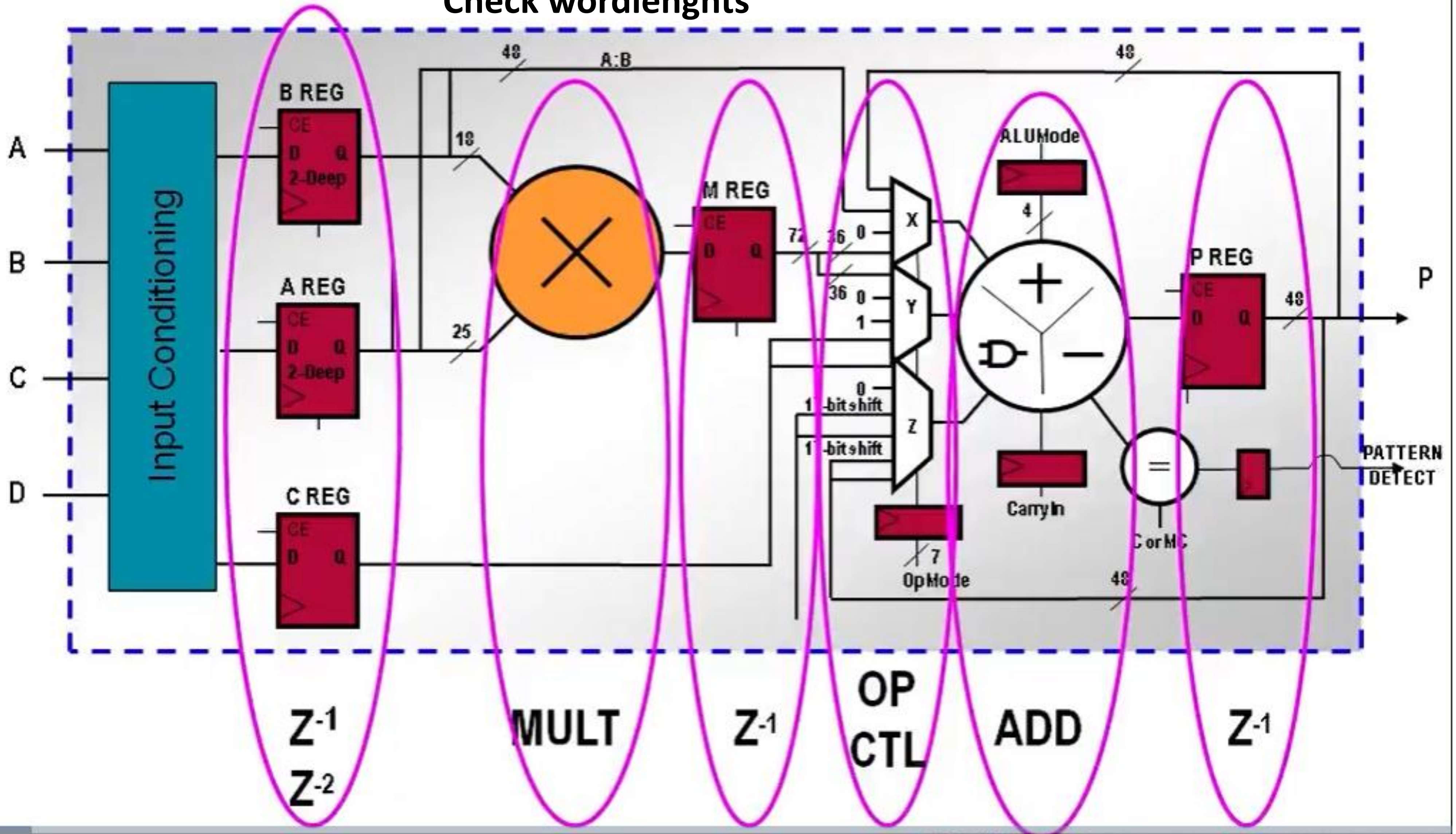
FIF filter mapping on
a configurable Hardware Device

Basic Xilinx DSP48 Slice Architecture



DSP Slice Features

Check wordlenghts



MAC Engine for FIR Filter in FPGA

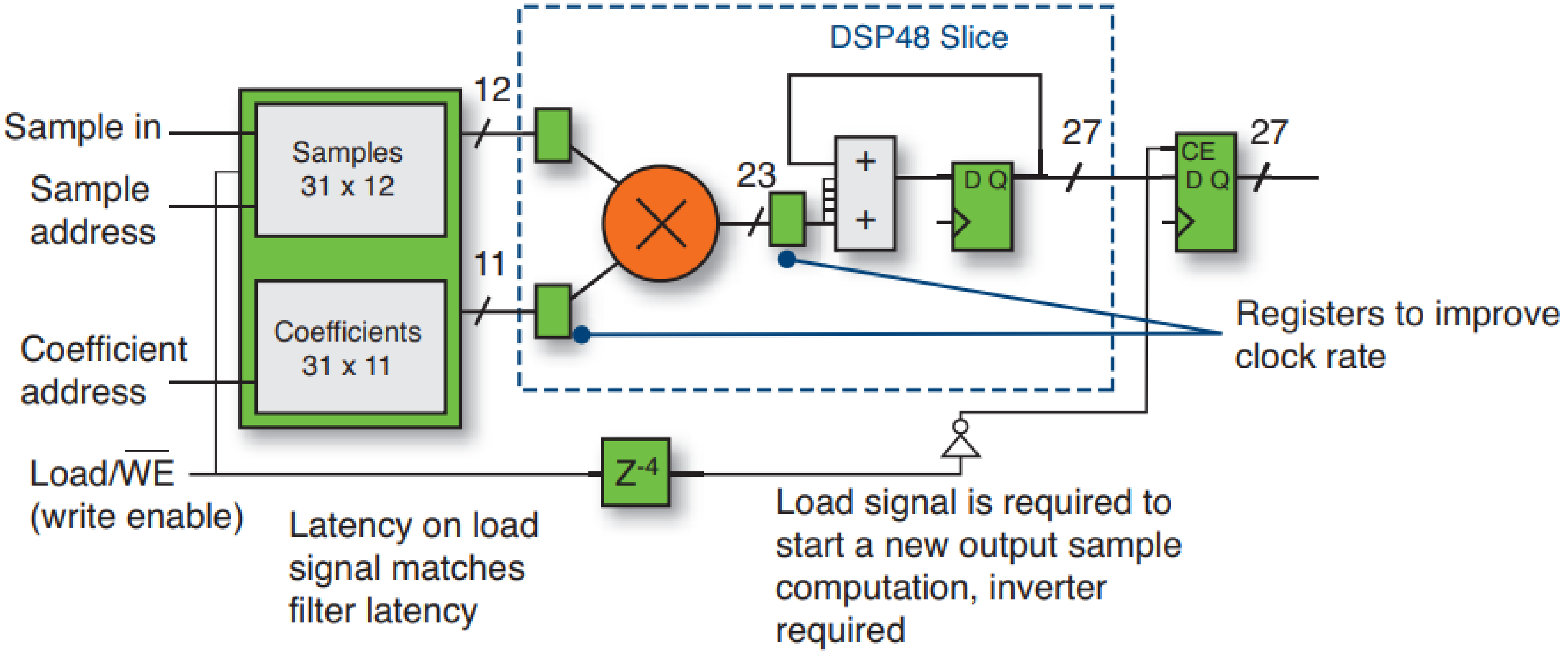
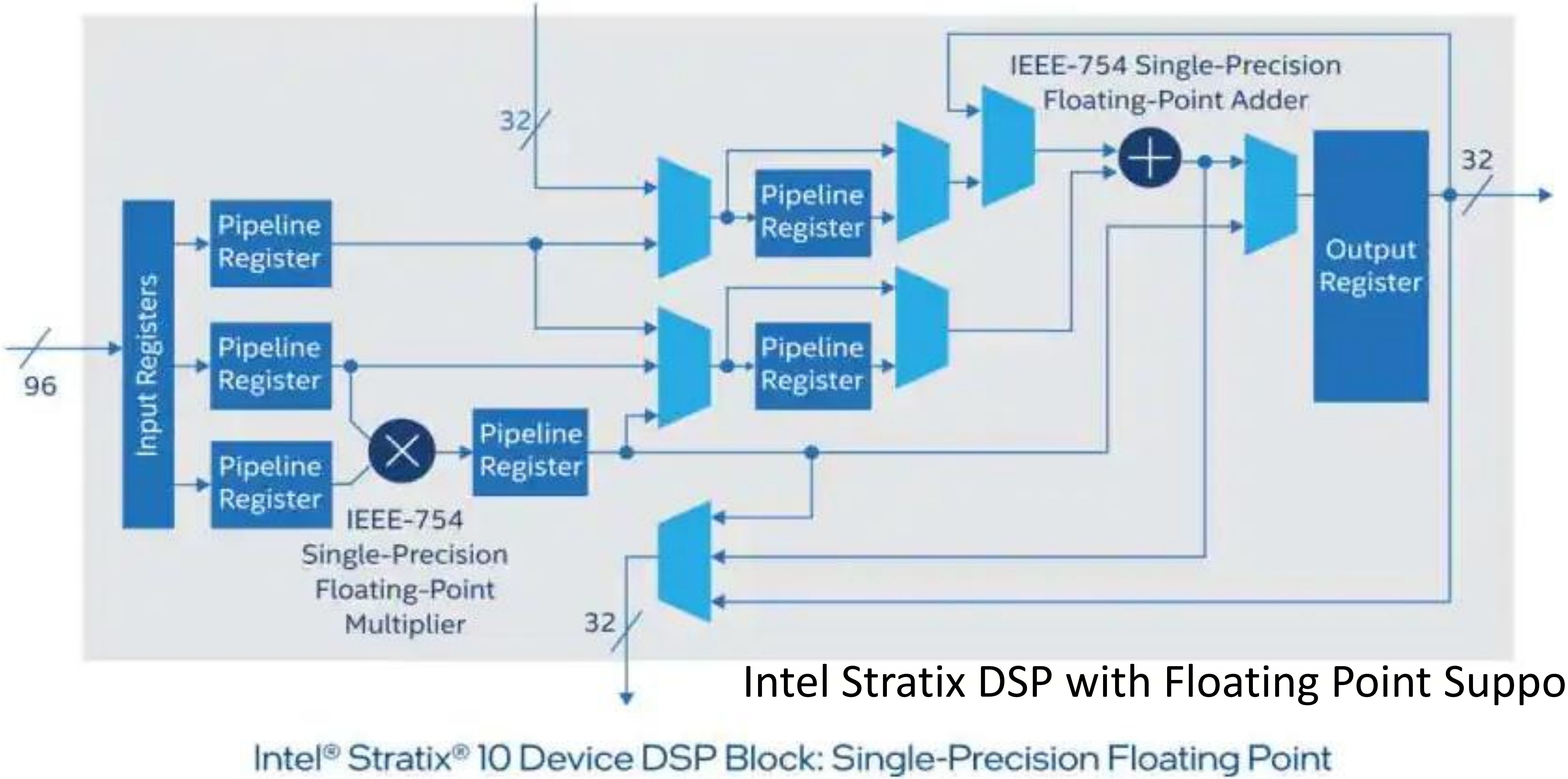


Figure 4 – MAC engine FIR filter in an FPGA

Intel Stratix DSP Slice with Floating Point



Intel Stratix DSP with Floating Point Support

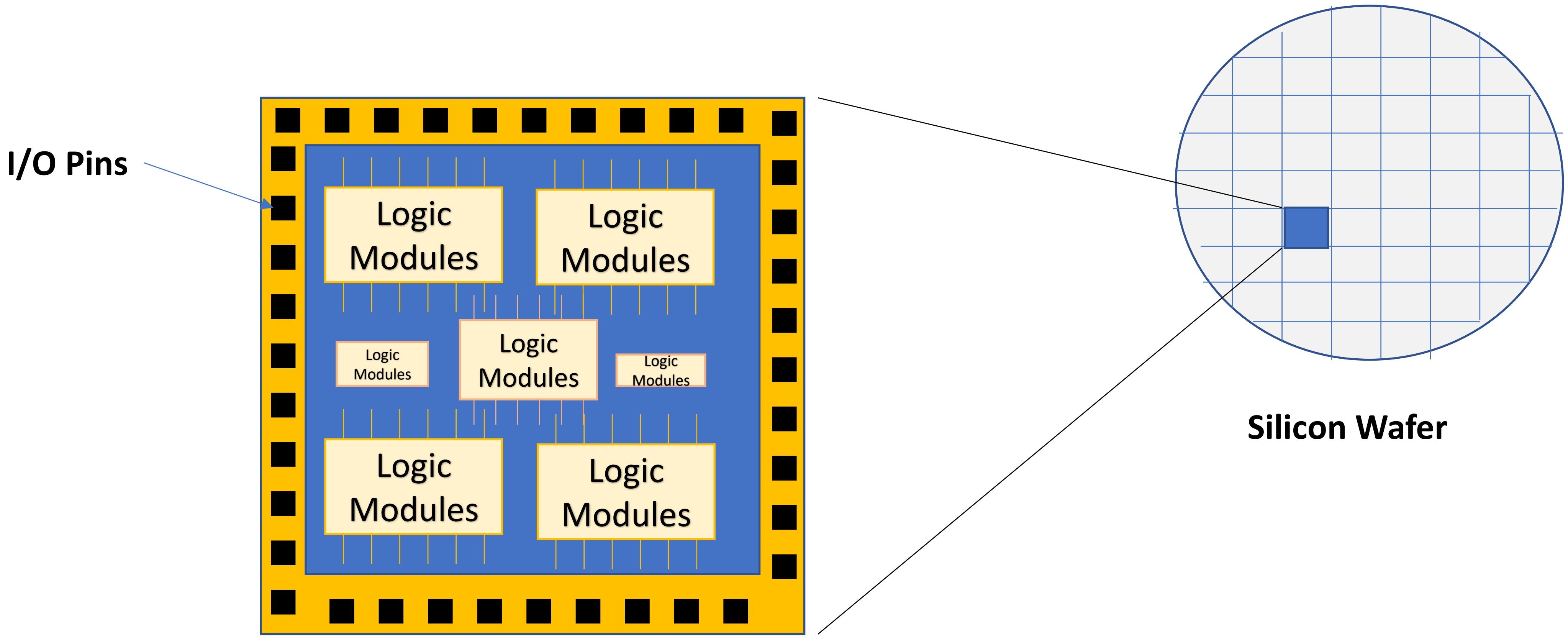
Faults, Testing and Testability

Topics: Faults and Testing, Examples of path sensitization method for fault tests, EX-OR method truth table, EX-OR method Boolean expression, testing of sequential elements using Scan cells

Types of Circuit Failure

- The domain of hardware related failure
- Permanent Failure: Incorrect behaviour at all times
- Intermittent Failure: Occurs randomly for finite time duration
- Transient Failure: Occurs in presence of certain environmental conditions such as high temperature, radiation, etc.
- Reasons for Failure: Wafer defects, impurities in clean room, mask mis-alignment, process imperfections, vibrations in equipment

Faults and Failure in Logic Circuit Chip



Number of internal inputs and outputs is much more than the number of physical I/O pins available

Production testing

- Detection of permanent errors caused by manufacturing defects. Involves two major steps:
  - Test Generation, and
  - Fault Simulation
- Failure modes are called ‘Faults’
- Set of vectors generated to detect ‘Faults’ is called ‘Fault-Simulation’
- ‘Fault-Models’ consider the logic effects that result from the physical faults in a circuit
- When a circuit fails to behave correctly, implies that the logic realized is different from logic that was specified for design

Chip Level Faults

Chip Level Fault Type	Degradation Fault	Open Circuit	Short Circuit
Leakage or Short between package leads	Yes		Yes
Broken or missing wire bonding		Yes	
Surface contamination or moisture	Yes		
Metal migration, stress peeling		Yes	Yes
Metallization		Yes	Yes

Gate Level Faults

Gate Level Fault Type	Degradation Fault	Open Circuit	Short Circuit
Contact Open		Yes	
Gate to Source short circuit	Yes		Yes
Field Oxide Parasitic Device	Yes		Yes
Gate Oxide Flaw, Spiking	Yes		Yes
Mask Misalignment	Yes		Yes

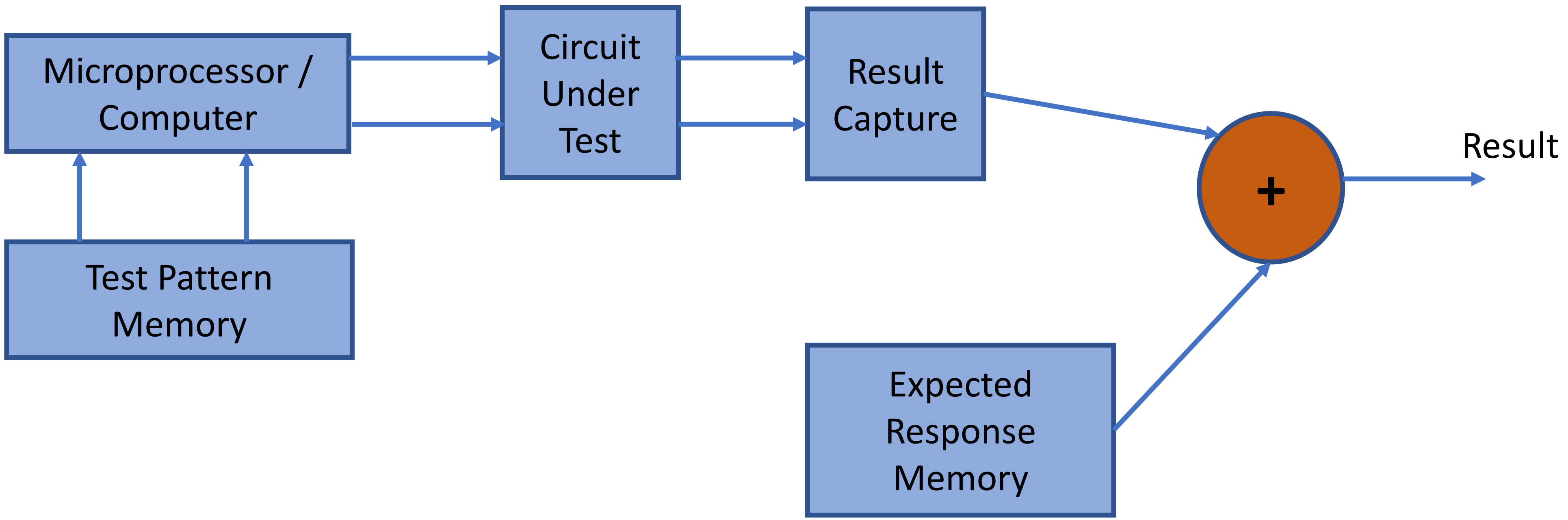
Fault Types

- Stuck Faults: A signal line is shorted to supply or ground permanently
- Bridging Faults: Short circuits in the interconnects between transistors in a logic cell are called bridging faults
- Bridging faults are detected by measuring the quiescent current through the CMOS logic circuit. It takes more time to detect Bridging faults whereas **Stuck-At faults** are easier to locate.
- Problem: The fault sites are typically located in the middle of the logic circuit and their inputs or outputs are not directly accessible from i/o pins
- There are maximum 100s of i/o pins vs the number of gates and their interconnects is in millions

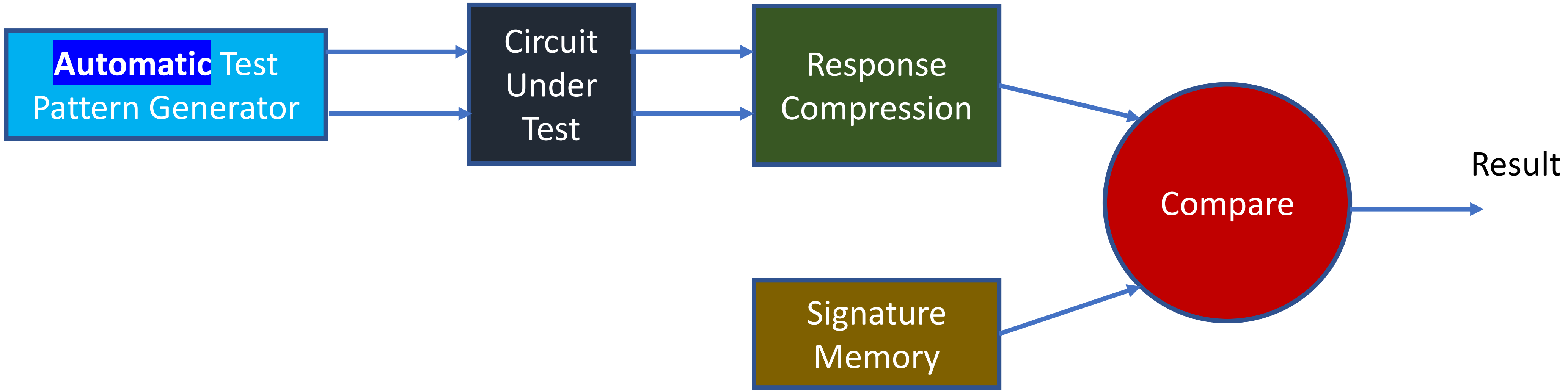
Un-Testable Faults

- Redundant Logic
- Un-Controllable Nets
- Un-Observable Nets that cannot be sensitized through I/O pins

Typical Test Setup



Automated Test Setup



Exhaustive Testing

Requires 2^N test vectors to exhaustively test all input combinations

Exhaustive Testing compares correct and faulty outputs for each input combination

This is a very slow approach

Single Stuck-At Fault Models

Stuck-At Fault Model: Assumes that there is just one stuck-at fault in circuit under test. Hope that single fault removal will remove multiple faults as well.

Stuck at 0 / Stuck at 1 (SA0/SA1) faults: Only two types of logical faults assumed in the model at gate level.

Observability: The degree to which one can observe a node at the output pins of an IC package.

Given that only a limited number of nodes could be directly observed, alternative methods such as JTAG are used to observe all outputs with some delays.

Controllability: Measure of the ease of setting the node to ‘1’ or ‘0’ state. Easiest would be directly settable by an input pin on IC package.

Fault Coverage

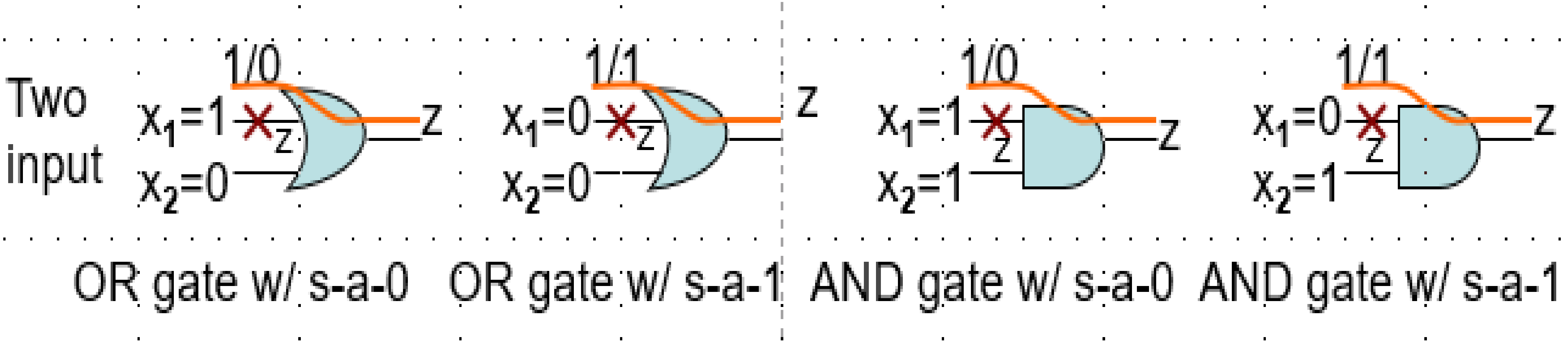
- % Fault Coverage = $\dfrac{No.of\ nodes\ when\ set\ to\ 1\ or\ 0\ result\ in\ detection\ of\ fault}{total\ number\ of\ nodes\ in\ the\ circuit}$

- KN Cycles are needed; K = no. of nodes in the circuit
- N/2 cycles are needed to detect each fault
- N = length of test sequence
- In turn, every node is tested for SA0 and SA1 sequentially i.e. Sequential Fault Grading

Fault Representation

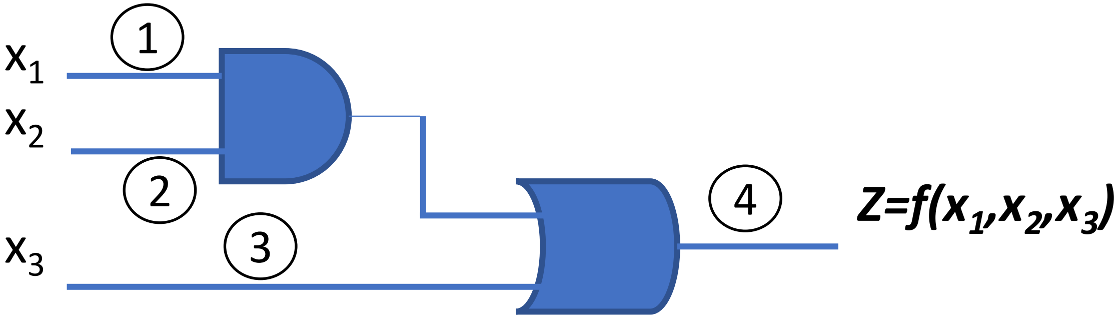
- **$f(x_n) = f(x_1, x_2, x_3, \ldots\ldots, x_n)$** represents a fault-free circuit
- **$f^{p/d}(x_n)$** represents the same circuit with fault p/d;
- Where p is a wire label, d is ‘0’ or ‘1’ representing SA0 or SA1 respectively
- n is the no. of input variables

Different Faults and Input vectors



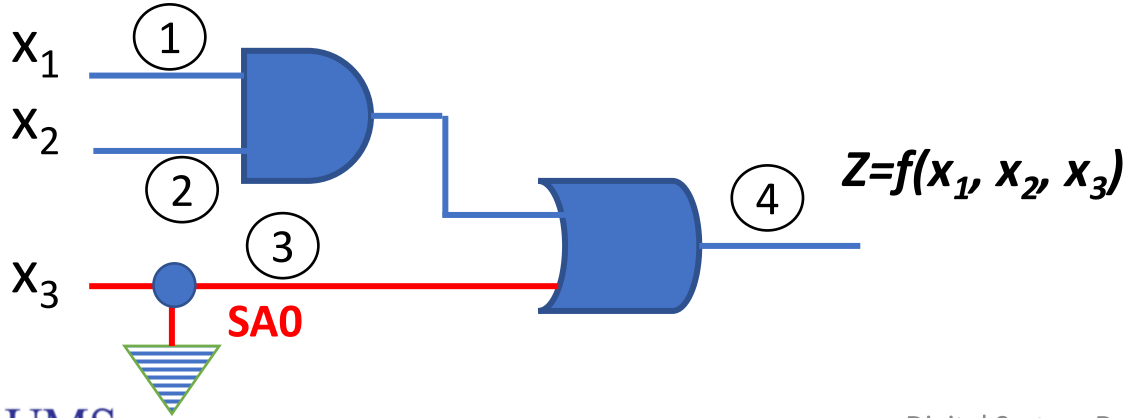
How to observe different faults at outputs of gates?

Example of fault representation



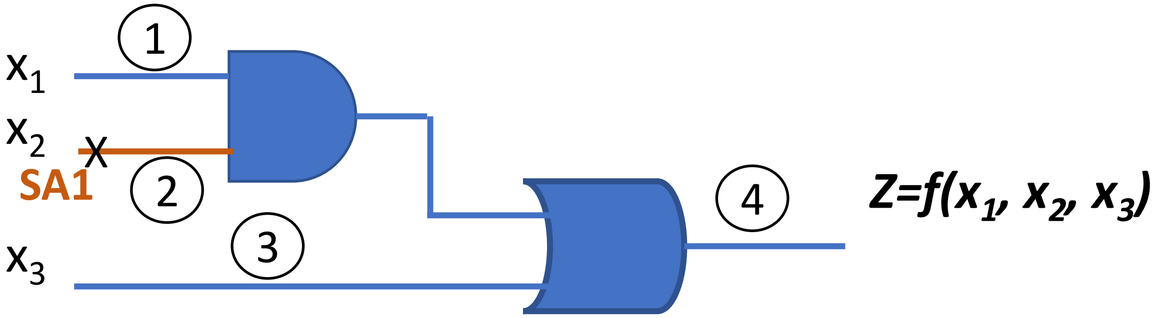
Stuck at 0 (SA0) fault at node 3:

$f^{3/0}\,(x_3) = x_1.x_2$



Stuck at 1 (SA1) fault at node 2:

$f^{2/1}\,(x_2) = x_1 + x_3$



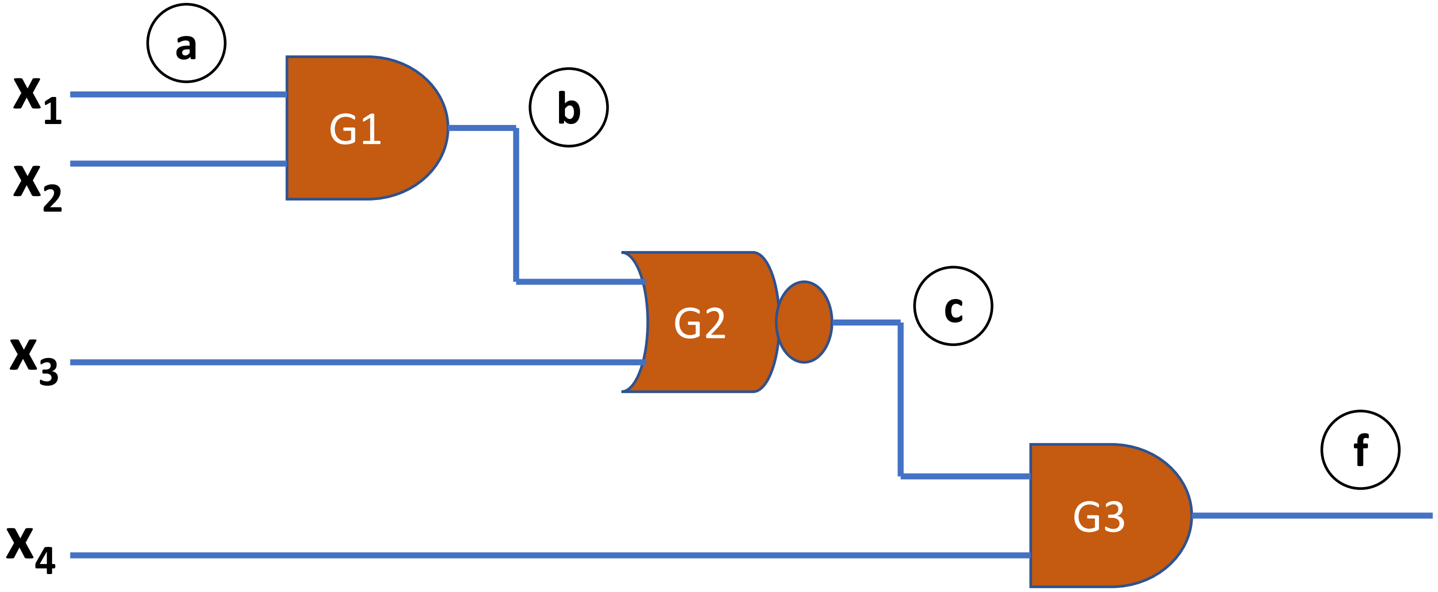
Path Sensitization

- Purpose is to sensitize the path so that inputs can help observe effects of SA0 or SA1 faults at the outputs
- In multi-level circuits, one set of test vector can act as test for faults in several paths

Three Steps in Path Sensitization Method

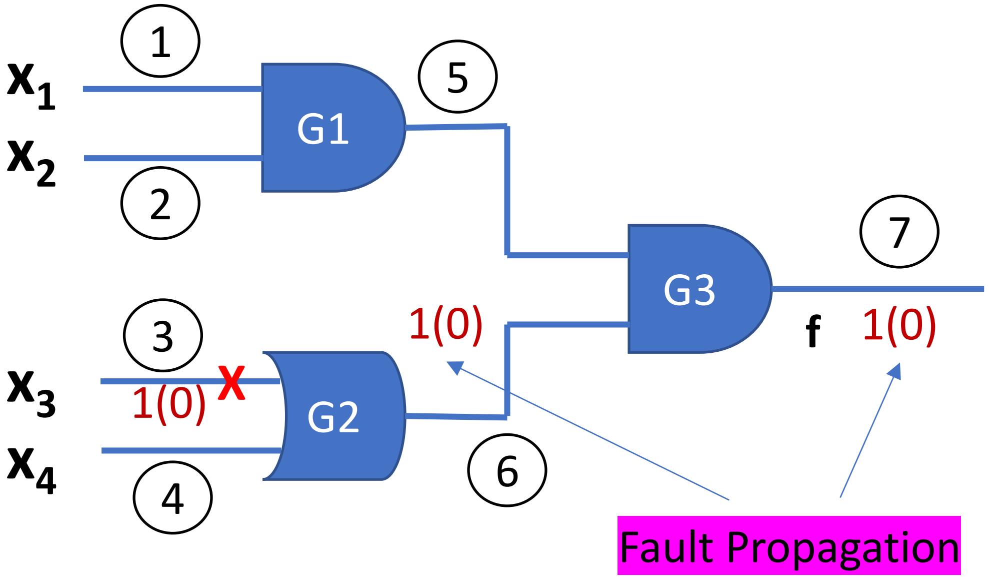
- **Fault Excitation:** Which vector to be induced to detect the suspected SA0 or SA1 fault at the suspicious path
- **Fault Propagation:** Identify path/s through which fault can be propagated to the observable output
- **Back tracking:** Move back from output towards all inputs and assign appropriate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

Path Sensitization – how to



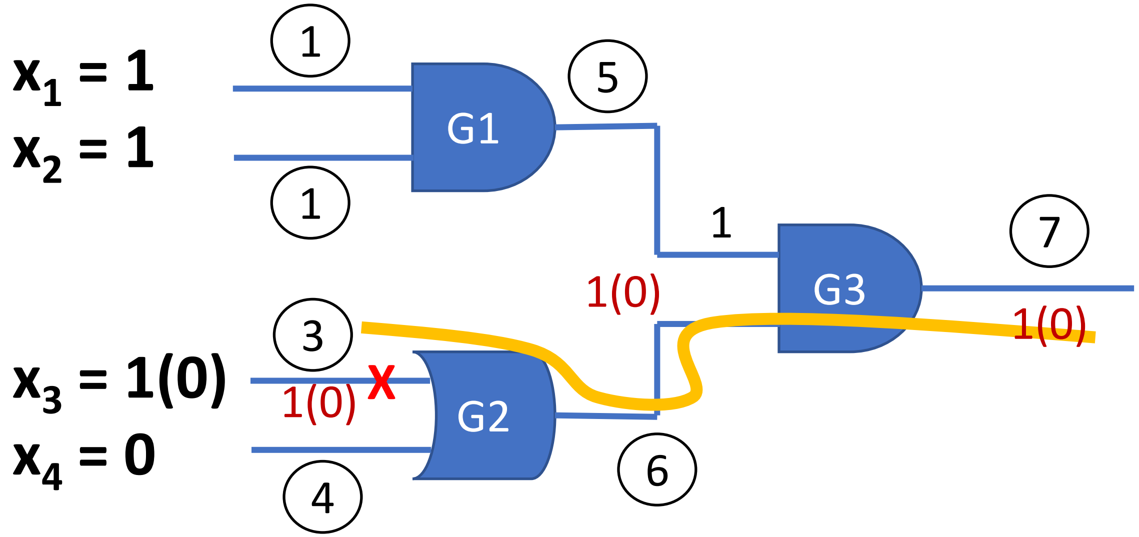
To sensitize a path through an input of AND gate or NAND gate, all other inputs must be set to ‘1’
To sensitize a path through an input of OR gate or NOR gate, all other inputs must be set to ‘0’

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 → 6→ 7

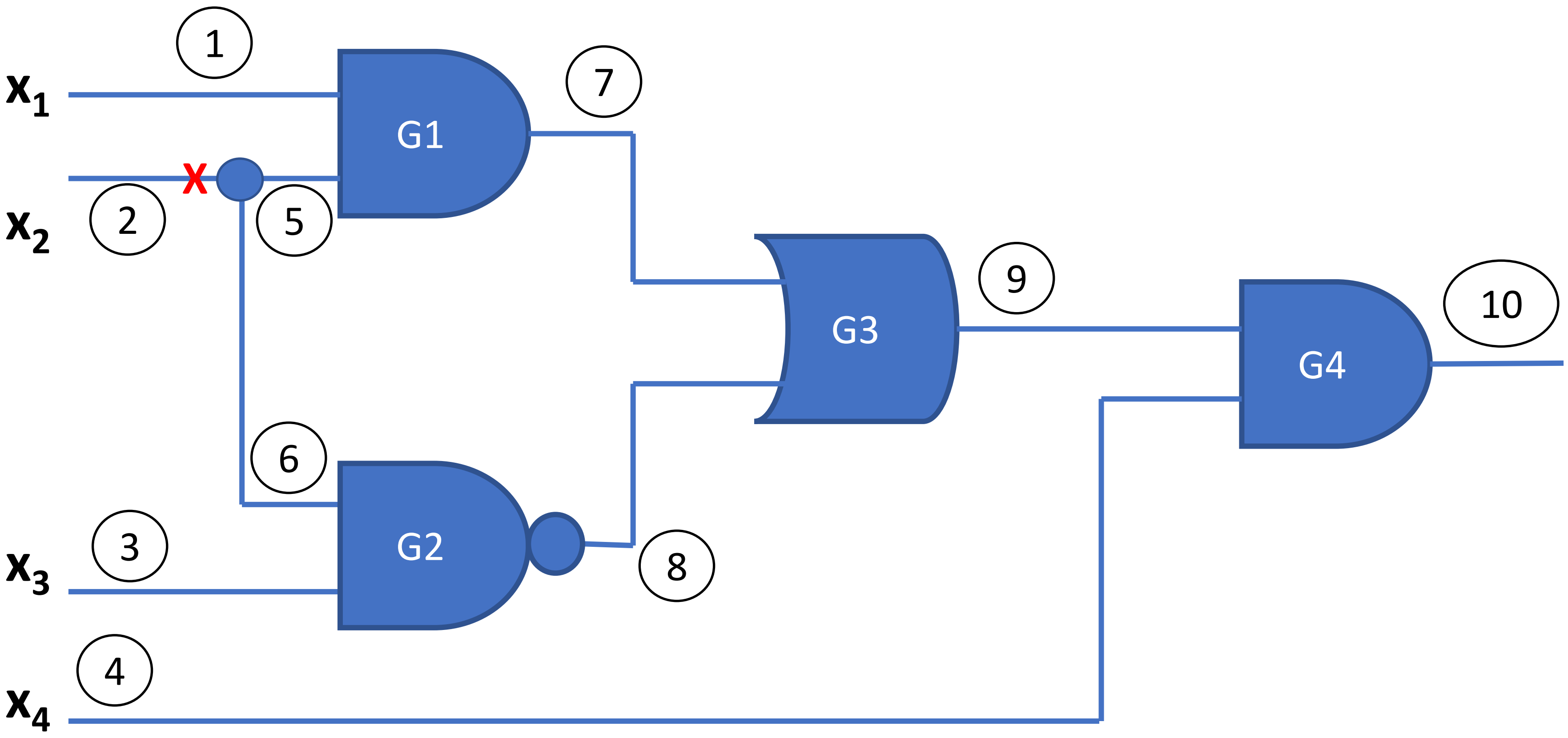
Back tracing reveals inputs to all gates to ensure fault propagation

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

Path Sensitization – Example 2

To detect SA1 fault at wire 2

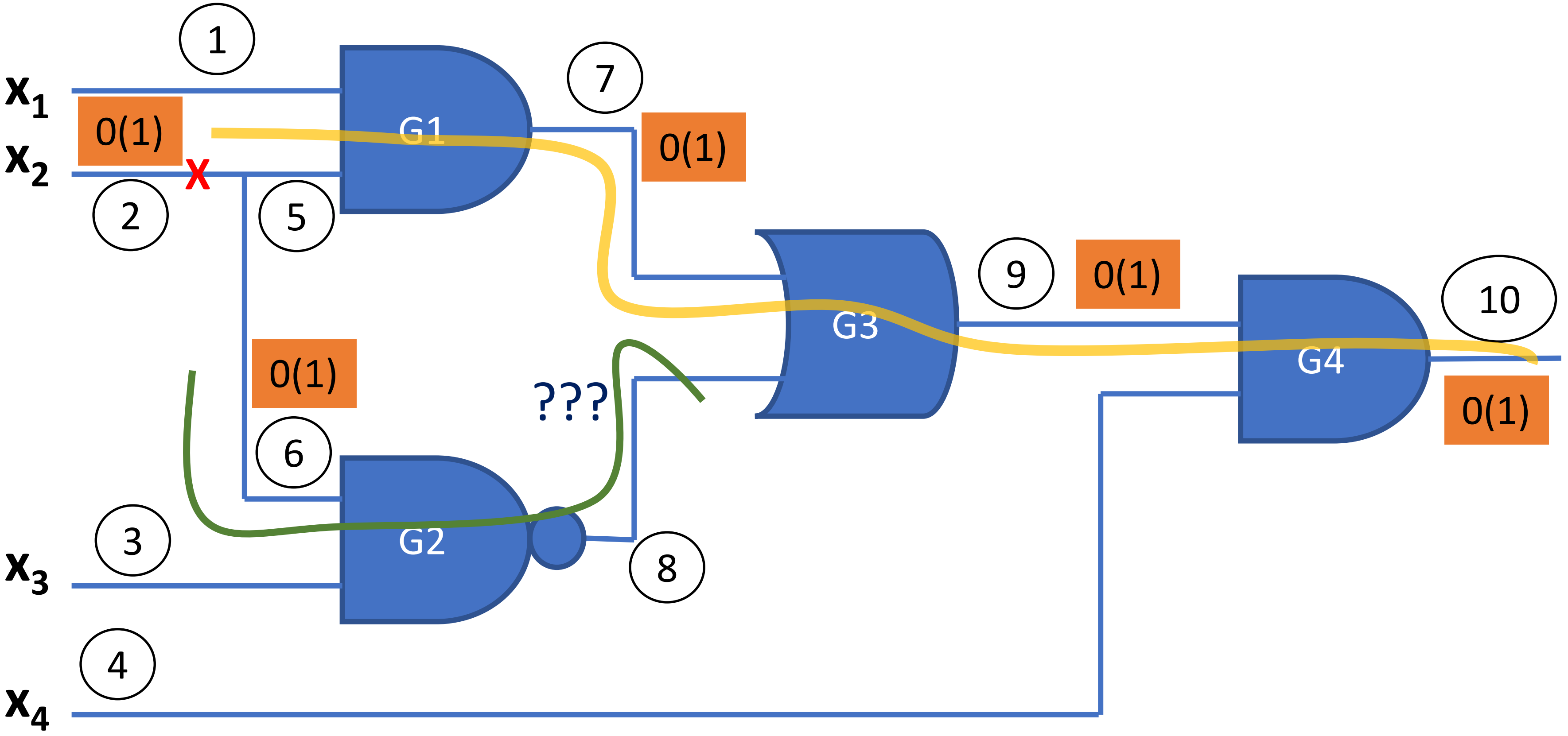
Two possible paths can be excited



continued

Detect SA1 at path 2

Fault propagation by supplying input x_2=0(1)



Case 1:

Back tracking reveals:
First Selected path = 2→7 → 9 → 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

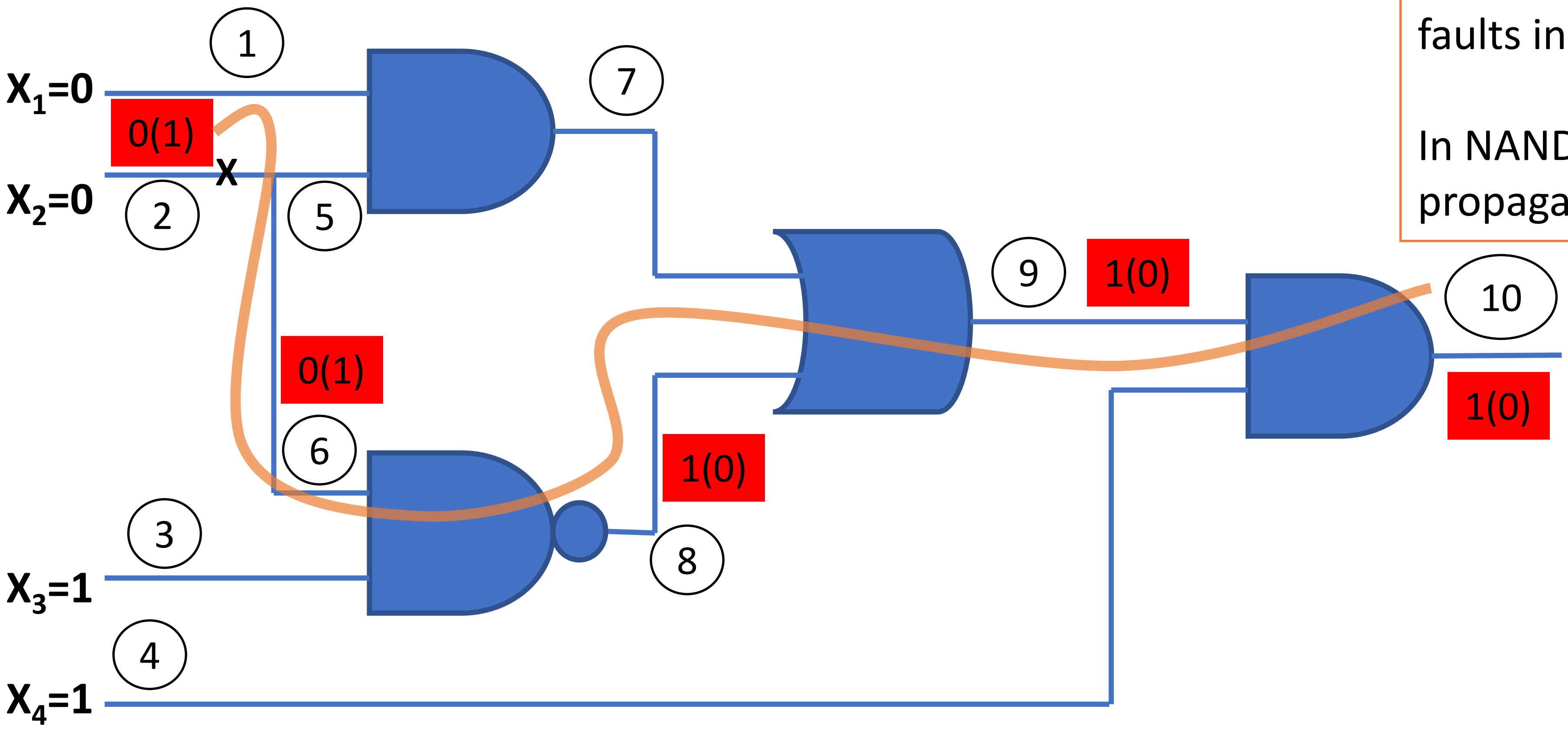
Continued – final test vectors

Selected path = 2 → 6 → 8 → 9 → 10

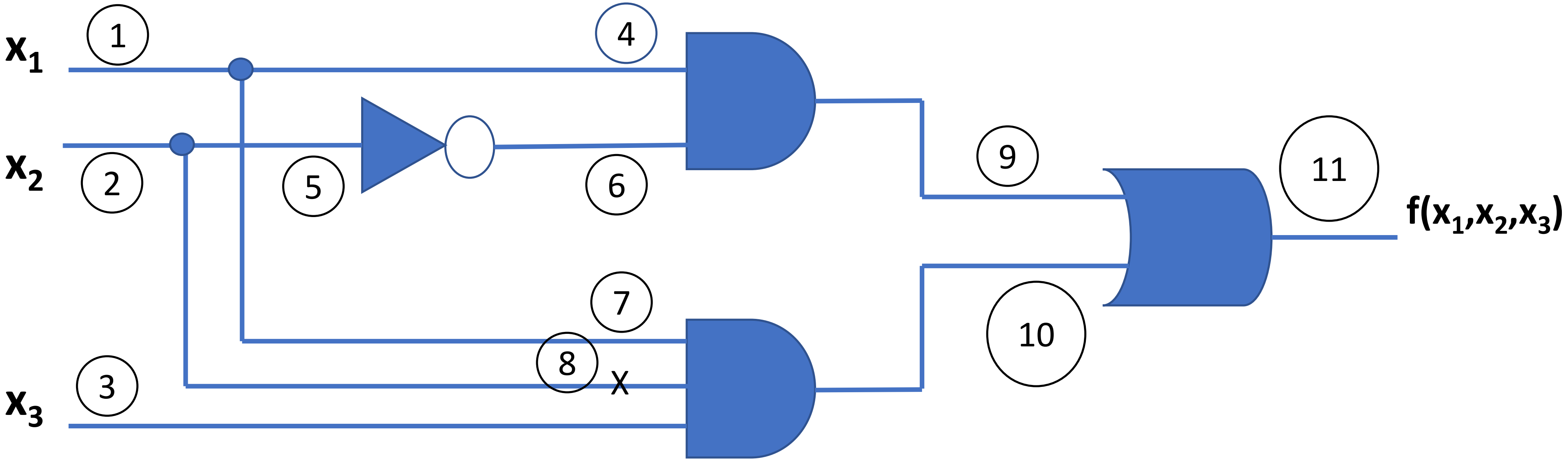
Test Vector = “0011”

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 x1 to x3

Mathematically:

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

This condition means it is not possible to test this path