Design For Test (DFT)

Dr. Adam Teman
EnICS Labs, Bar-Ilan University

24 June 2020

Emerging Nanoscaled
Integrated Circuits and Systems Labs

Tradition of Excellence
Bar-Ilan University
אוניברסיטת בר-אילן

Overview

- Introduction
- The Testing Process
- Design For Test
- Built-In Self Test
- Boundary Scan IEEE STD 1149.1

Introduction

Testing vs. Verification

Cost of Testing

- **Design for testability (DFT)**
  - Hardware design styles or added hardware that reduces test generation complexity
  - Chip area and performance overhead
- **Software processes of test**
  - Test generation and fault simulation
  - Test programming and debugging
- **Manufacturing test**
  - Automatic test equipment (ATE) capital cost
    - ~$5M initial cost + ~$2M per year
  - Test center operational cost
    - ~5 cent/second (~$1.5M/year for 24hour operation)

Teradyne UltraFLEXplus

Source: Teradyne

Cost of NOT Testing

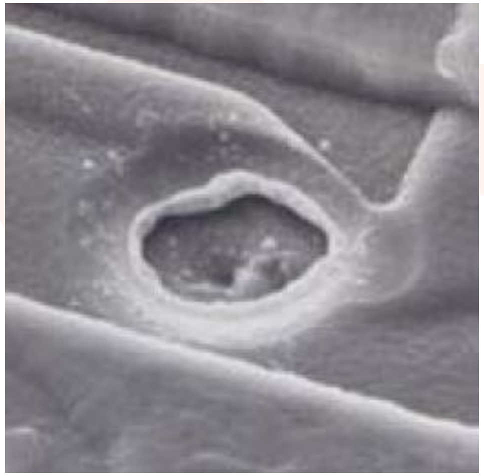
- **Cost of defective ICs**
  - IC $
  - IC on a PCB (printed circuit board) $$
  - IC on a PCB in a system $$$
  - IC on a PCB in a system in field $$$$
  - ….or more

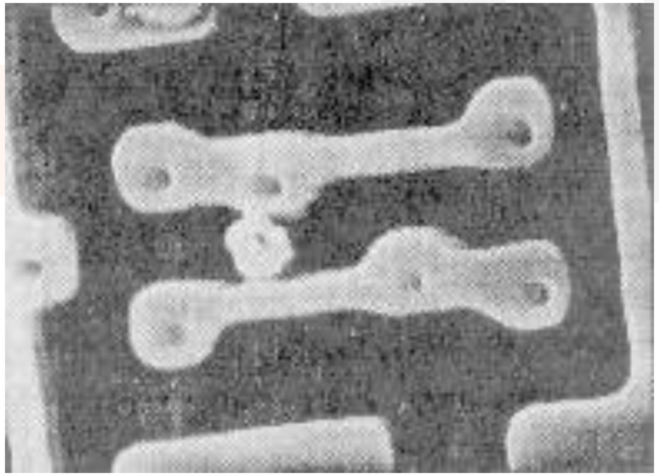
- **The most expensive defect is the one that wasn't detected inline**
- **Detect defective parts as soon as possible!**

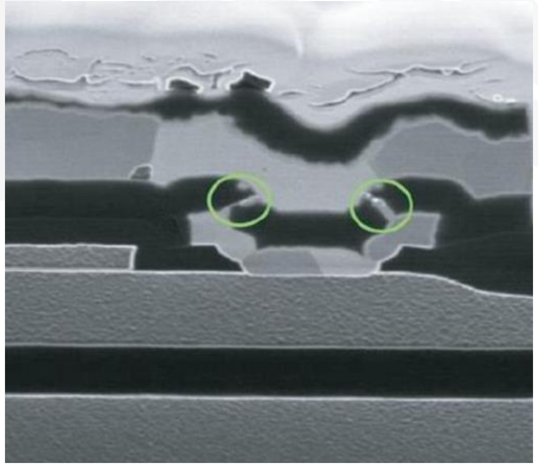
Source: Easy Pace Learning

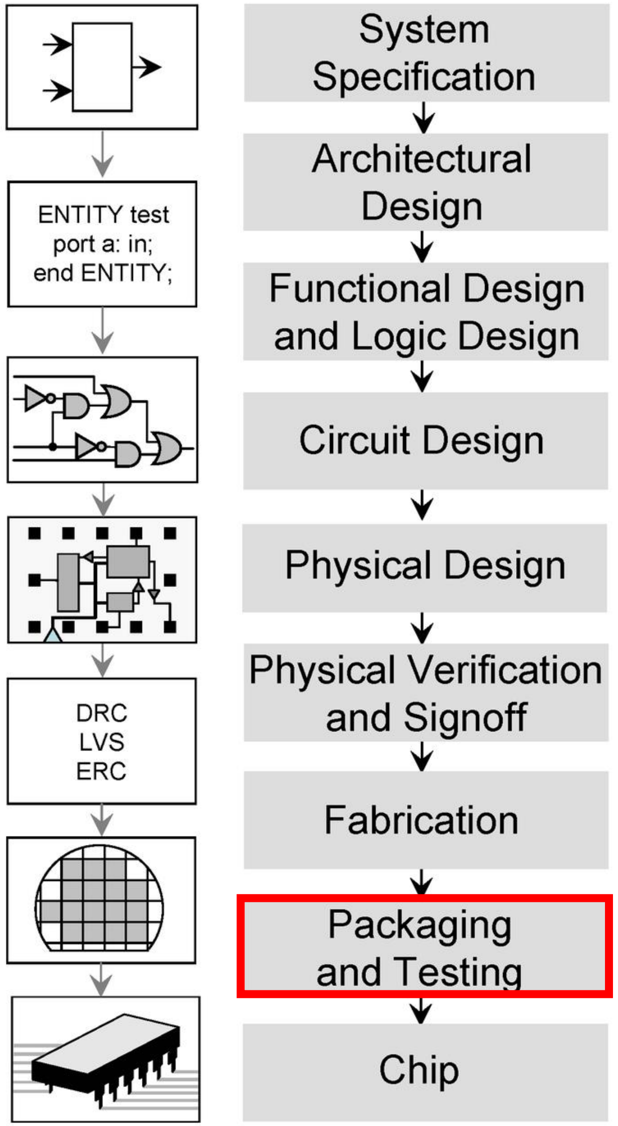
Production Testing

- Testing after chip fabrication in order to detect manufacturing defects that may impact functionality or electrical parameters.
- Manufacturing test ideally would check every node in the circuit to prove it is operational.







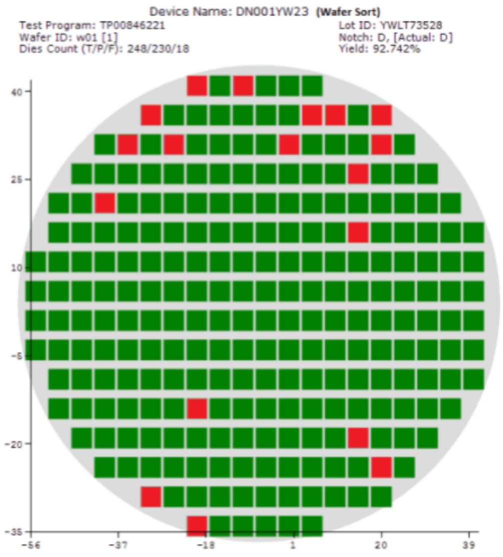


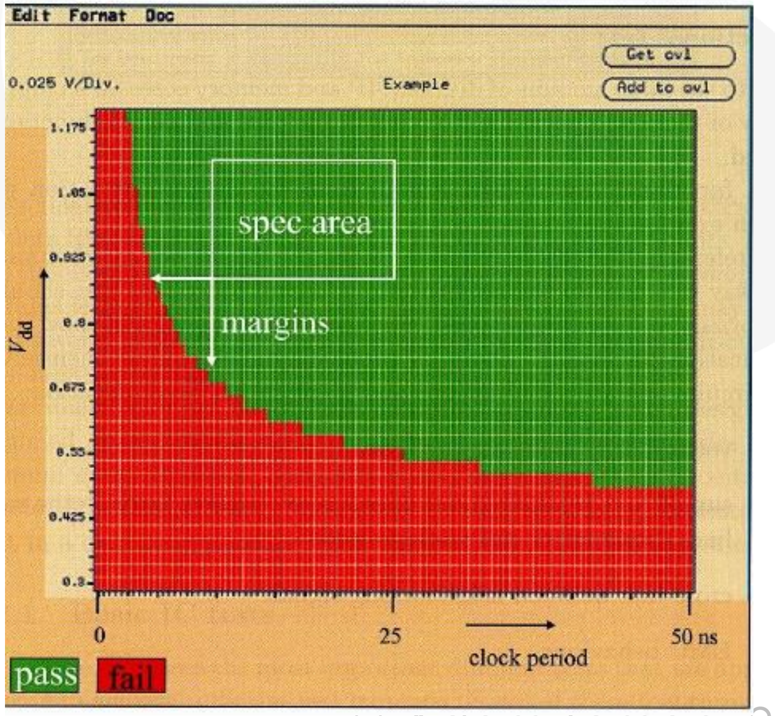
Production Test Flow

Production Test Flow

- Wafer level testing (Wafer Sort)
- Assembly & Packaging
- Open/Short test
- Packaged device test
- Burn-In (@ elevated voltage and temperature)
- Final Test (pass/fail) and Bin Sorting

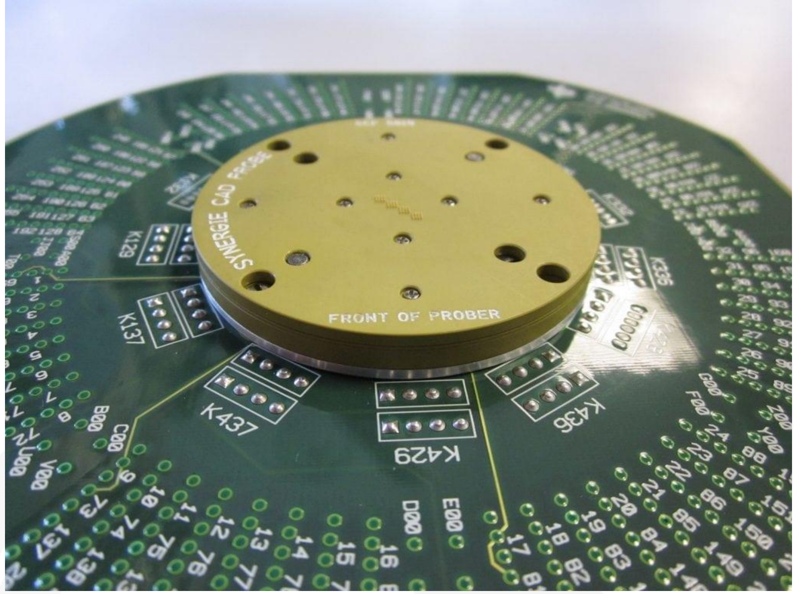
- Parametric Tests (voltage, temperature and clock)
  - Shmoo plot



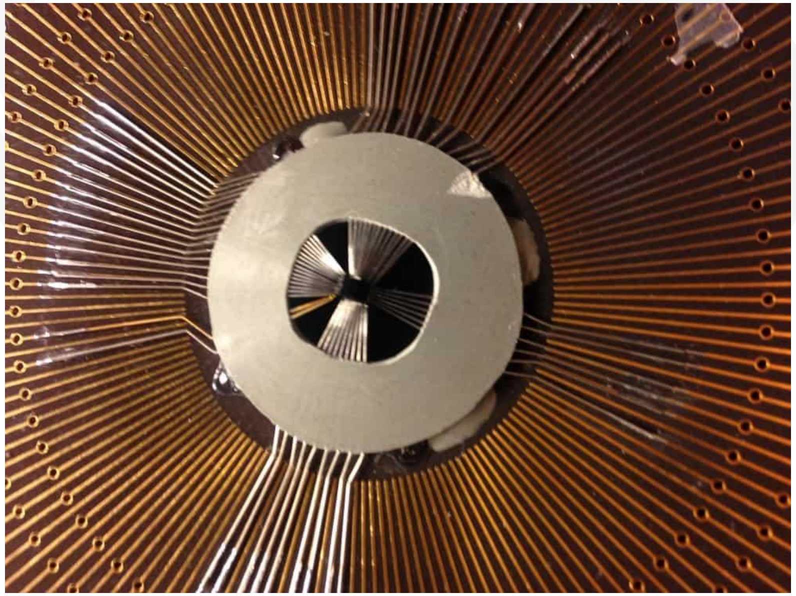


Wafer Sort

- **Wafer sort or probe test**
  - Done before wafer is *scribed* and cut into chips
  - Includes test site characterization – specific test devices are checked with specific patterns to measure gate threshold, poly sheet resistance, etc.
- **Probe card**
  - Custom built PCB to allow performing wafer sort
  - Modern probe cards can test an entire 12" wafer with one touchdown
  - Can contact several dies in parallel (~1-16)
  - Camera in the wafer prober allows alignment



Source: Synergie-CAD



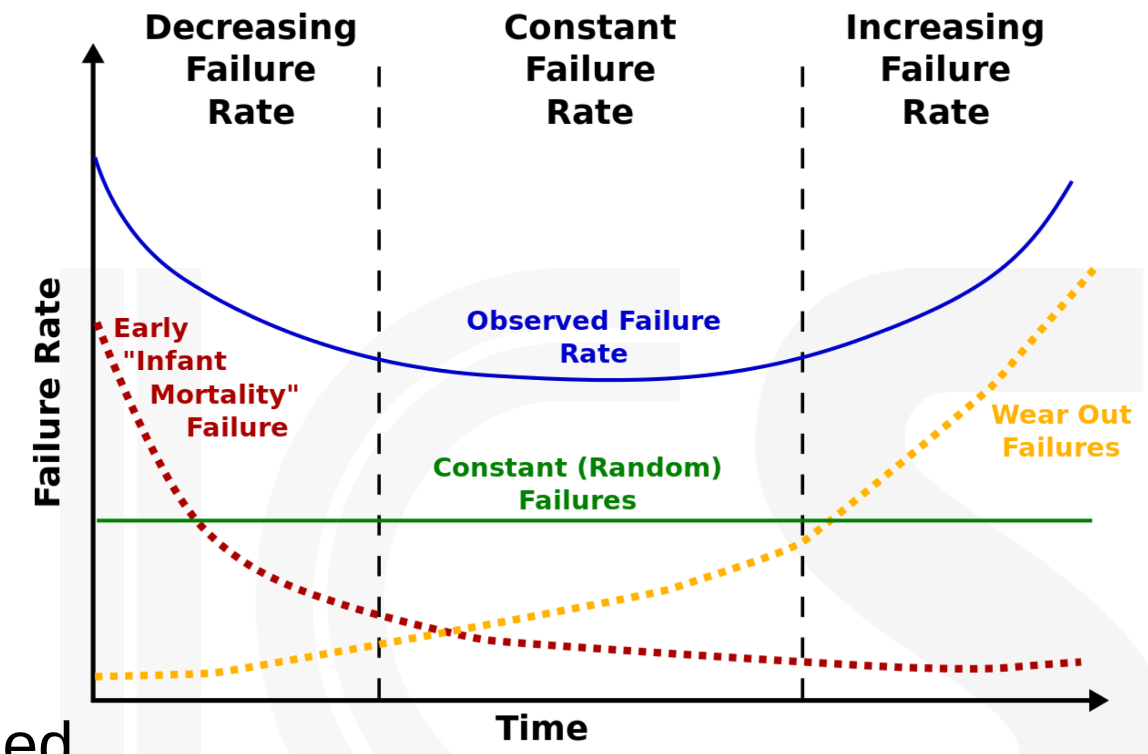
Source: Anysilicon

Electrical Testing

- **DC Parametric Tests**
  - DC contact test - Calculates pin resistance
  - Power consumption test - Measure max current at worst case temperature
  - Output short circuit test - Measure current driven when output short circuited
  - Output drive current test - Measure current for ‘1’ and ‘0’ outputs
  - Threshold test - Measure VIH, VIL of input pads
- **AC Parametric Tests**
  - Rise/Fall time tests
  - Setup/Hold time tests
  - Propagation delay tests

Burn-in or Stress Test

- **Process:**
  - Subject chips to high temperature and over-voltage supply, while running production tests
  - For example: 125C for 168 hours
- **Catches:**
  - *Infant mortality cases* – these are damaged or weak (low reliability) chips that will fail in the first few days of operation – burn-in causes bad devices to fail before they are shipped to customers
  - *Freak failures* – devices having same failure mechanisms as reliable devices

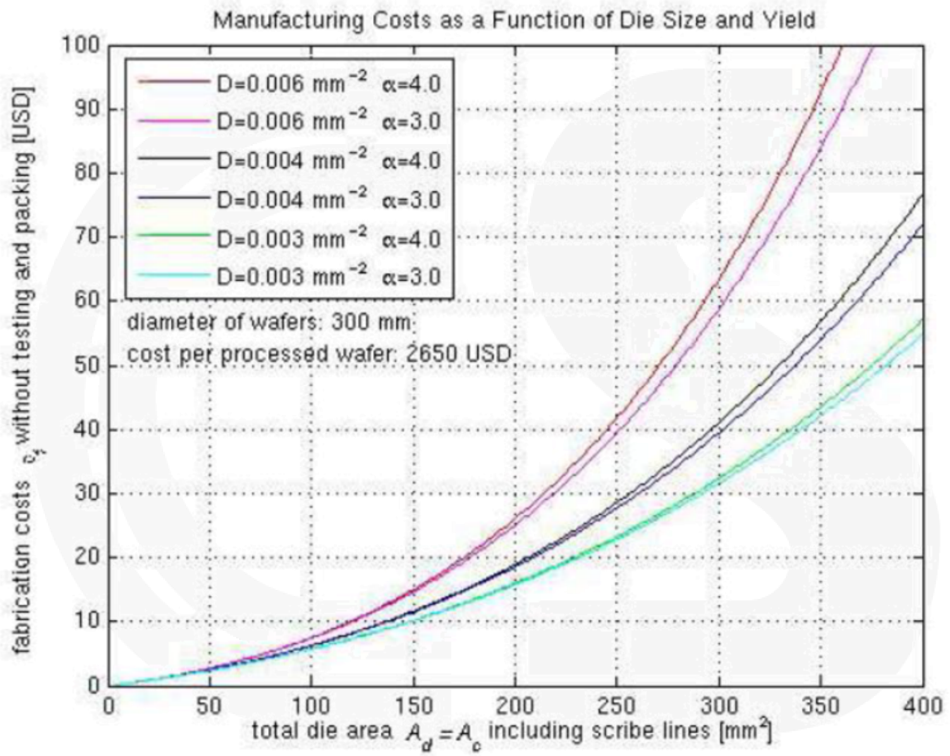


The well known "Bath Tub Curve"

Source: wikipedia

Yield & Cost

- **Die Yield** $= \frac{\#\, good\ dies}{\#\, manufactured\ dies} \approx \left(1 + \frac{D \cdot A}{\alpha}\right)^{-\alpha}$
  - D – Defect Density – average number of defects per unit of chip area
  - A – Chip area, α – Clustering Factor
  - ex. $AxD=1$, $\alpha=0.5$ → $Y=0.58$
- $Die\ Cost = \frac{Wafer\ Cost}{Dies\ per\ Wafer * Die\ Yield}$
- $IC\ Cost = \frac{Die\ Cost + Testing\ Cost + Packaging\ Cost}{Final\ Test\ Yield}$
- **Defect Level (DL)**
  - The ratio of faulty chips among the chips that pass test, measured in DPM (<<500 DPM)



$$DL = 1 - Y^{(1-FC)},$$
$$0 < DL < 1 - Y$$

The Testing Process

Various Processes During Testing

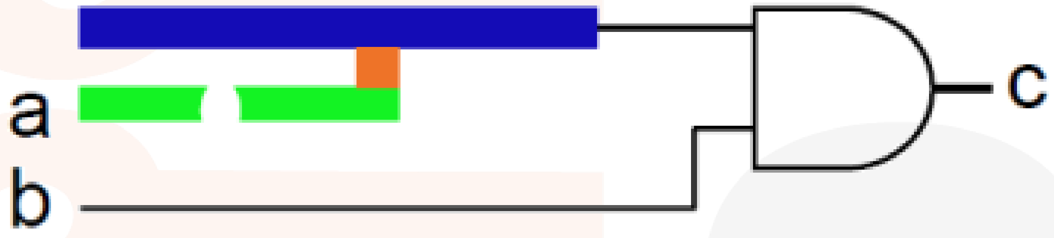
- **Fault Modeling**
  - Abstract the physical defects and define a suitable logical fault model
  - Limits/simplifies the scope of test generation
- **Test Generation**
  - Giving a circuit and a set of faults F, determine a set of test vectors T that detects all faults in F.
- **Fault Simulation**
  - Given a circuit, a set of faults F, and a set of test vectors T, determine the faults in F that are tested by the vectors in T.
- **Design for Testibility (DFT)**
  - Formulate a set of design rules that result in a circuit that will be easily testable.

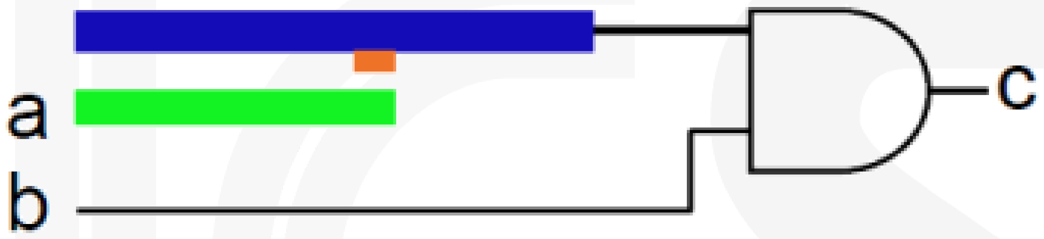
Defects – Faults – Errors

- **Defect:** *physical phenomenon*
  - A defect is the unintended difference between the implemented hardware and its intended design.
  - Defects are due to shorts, opens, etc., during manufacturing or throughout the lifetime of a device.
- **Fault:** *abstract representation*
  - A fault is a model of the influence of the defect on the circuit operation (e.g., a node is stuck at “0” or “1”)
- **Error:** *operational result*
  - An error is the incorrect circuit response (wrong output signal) under the presence of faults (or design errors).

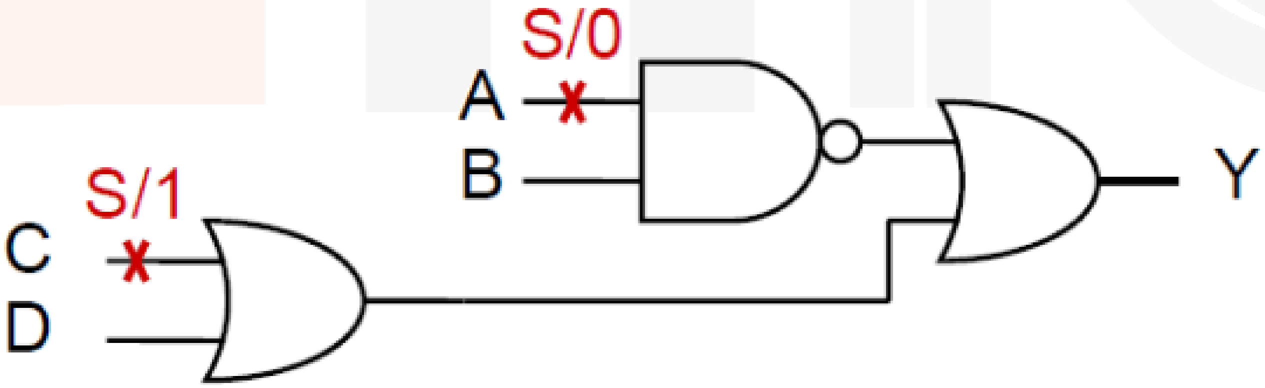
Defects – Faults – Errors

- Different types of defects may cause the same fault



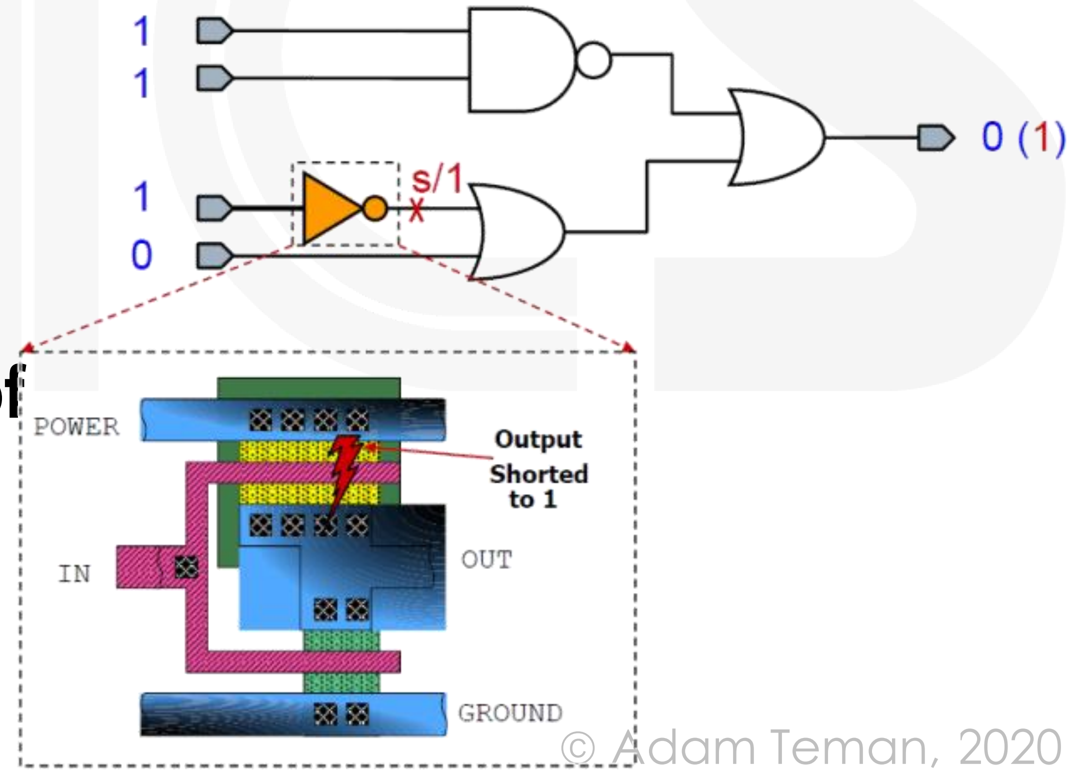


- Different types of faults may cause the same error



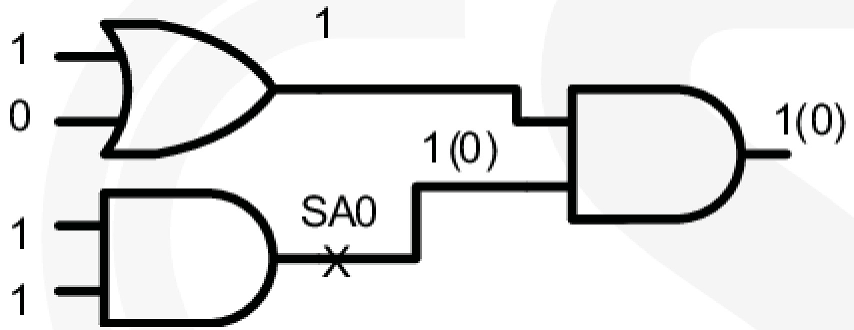
Why do we need a Fault Model?

- **The number of physical defects in a chip can be way too large**
  - Difficult to count and analyze all possible faults
- **Fault models abstract away physical defects into a logical model**
  - Drastically reduce the number of faults to be considered
  - Enable test generation and fault simulation
  - Enable evaluation of fault coverage and comparison of test results
- **Fault models can be done at various levels of abstraction to trade off accuracy vs. number of possible faults**
  - e.g., behavioral, functional, structural, switch-level, geometric

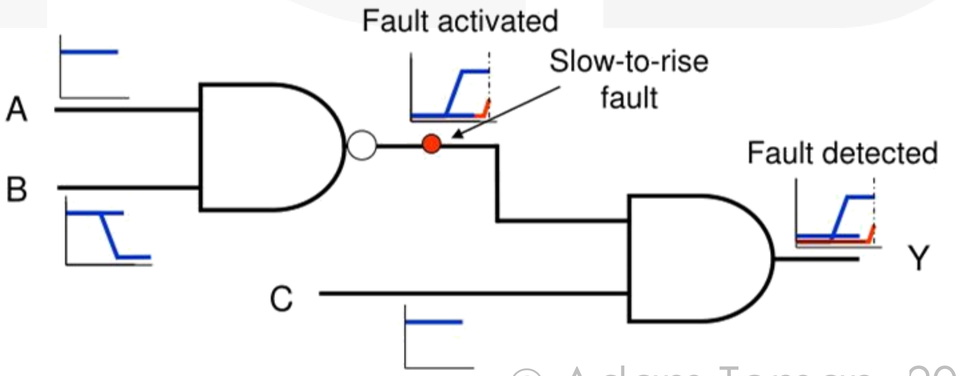


Commonly used Fault Models

- **Stuck At Fault**
  - Assume all failures cause nodes to be “stuck-at” 0 or 1
  - Static model (as opposed to at-speed test)
  - Independent of process technology
  - Not quite true, but works well in practice

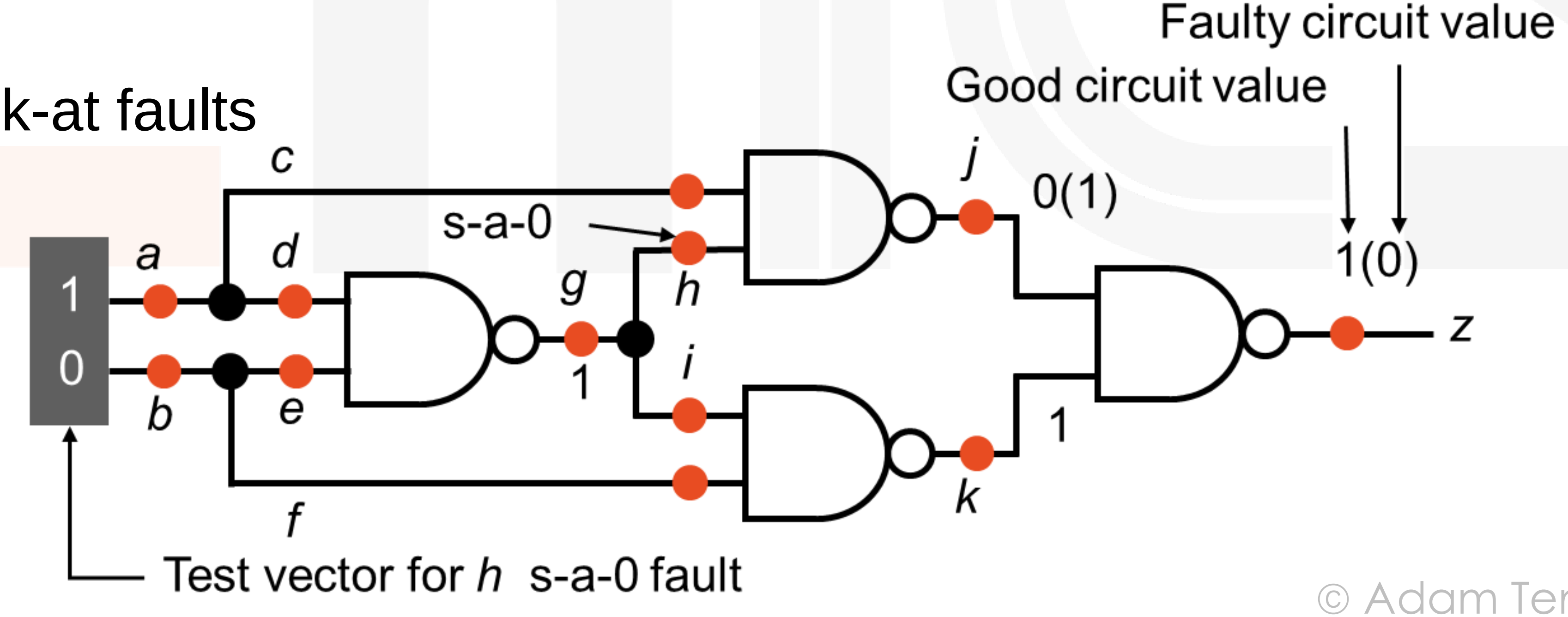


- **Transition/Delay Fault**
  - “Slow to Rise” or “Slow to Fall” fault
  - Signal propagation delays that are outside the circuit specifications
  - Dynamic model, tested at-speed
- **Other fault models**
  - Transistor opens and shorts
  - Bridging faults



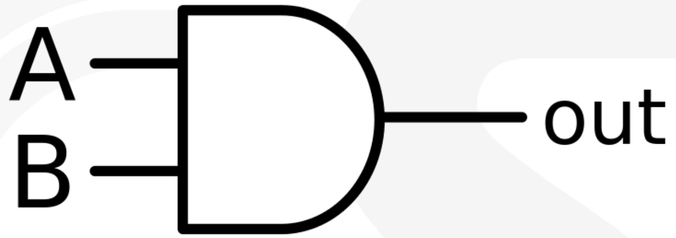
Stuck-at Fault (s/0, s/1)

- **Three properties define a single stuck-at fault**
  - Only one line is faulty, fanout stems and branches considered separate lines
  - The faulty line is permanently set to 0 or 1
  - The fault can be at an input or output of a gate
- **Example: XOR circuit has:**
  - 12 fault sites
  - 24 single stuck-at faults



Stuck-at Fault Testing Examples

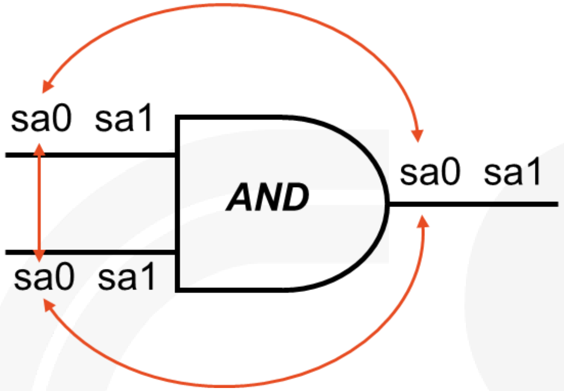
- **2-input AND:**
  - We have six possible faults: A/0, A/1, B/0, B/1, F/0, F1
  - But we only need three test vectors to test them:
    - 01: detects A/1, F/1
    - 10: detects B/1, F/1
    - 11: detects A/0, B/0, F/0
  - An *N*-input AND gate only needs *N*+1 test vectors

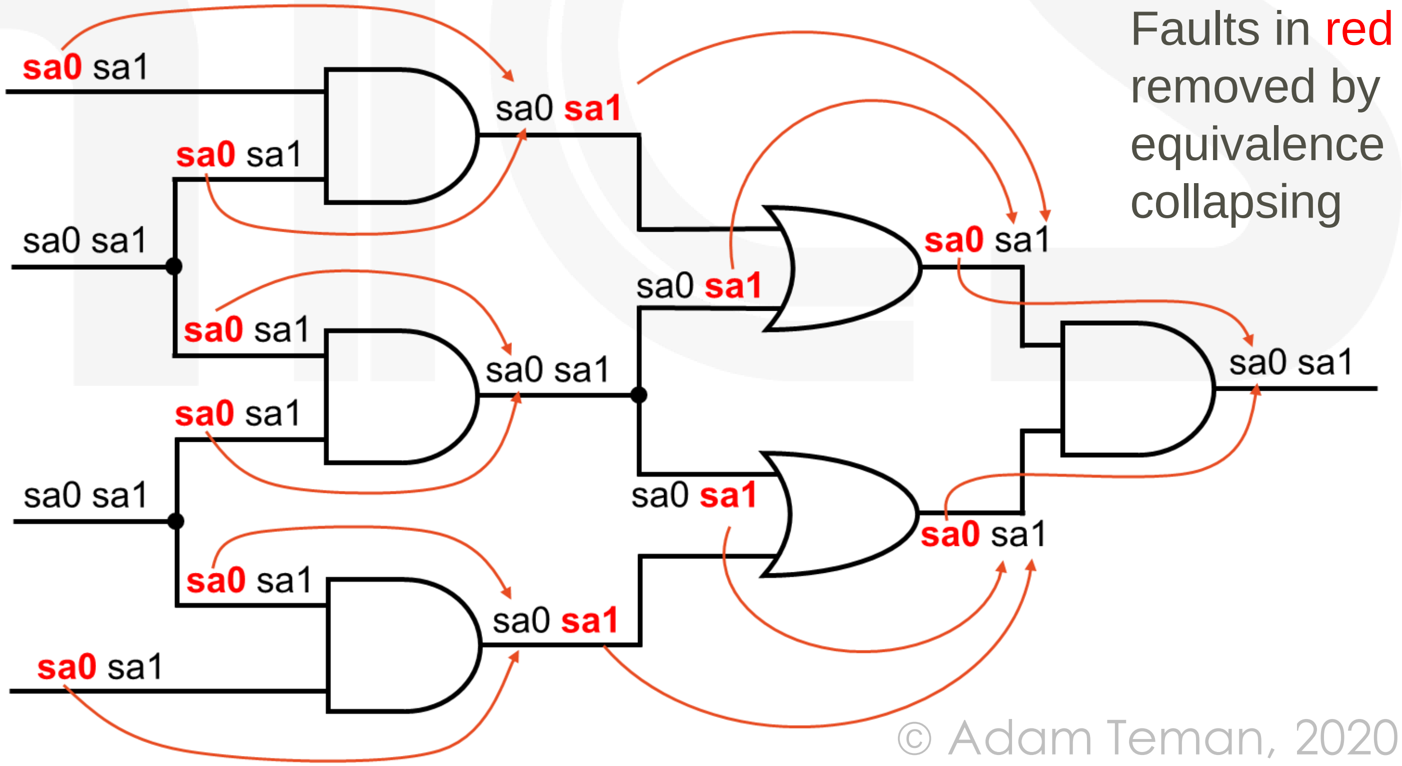


- **3-input XOR:**
  - We can detect all single stuck-at faults with 000 and 111
    - 000: tests all inputs s-a-1, and output s-a-1
    - 111: tests all inputs s-a-0, and output s-a-0
  - An *N*-input XOR gate needs only *2* test vectors!

Fault Equivalence and Collapsing

- **Number of fault sites in a Boolean gate circuit is**
  - #PI + #gates + # (fanout branches)
- **Fault equivalence:**
  - Two faults *f1* and *f2* are equivalent if all tests that detect *f1* also detect *f2*.
  - e.g., input s-a-0 and output s-a-0 in an AND gate.
- **Equivalent faults can be *collapsed* into a single fault.**



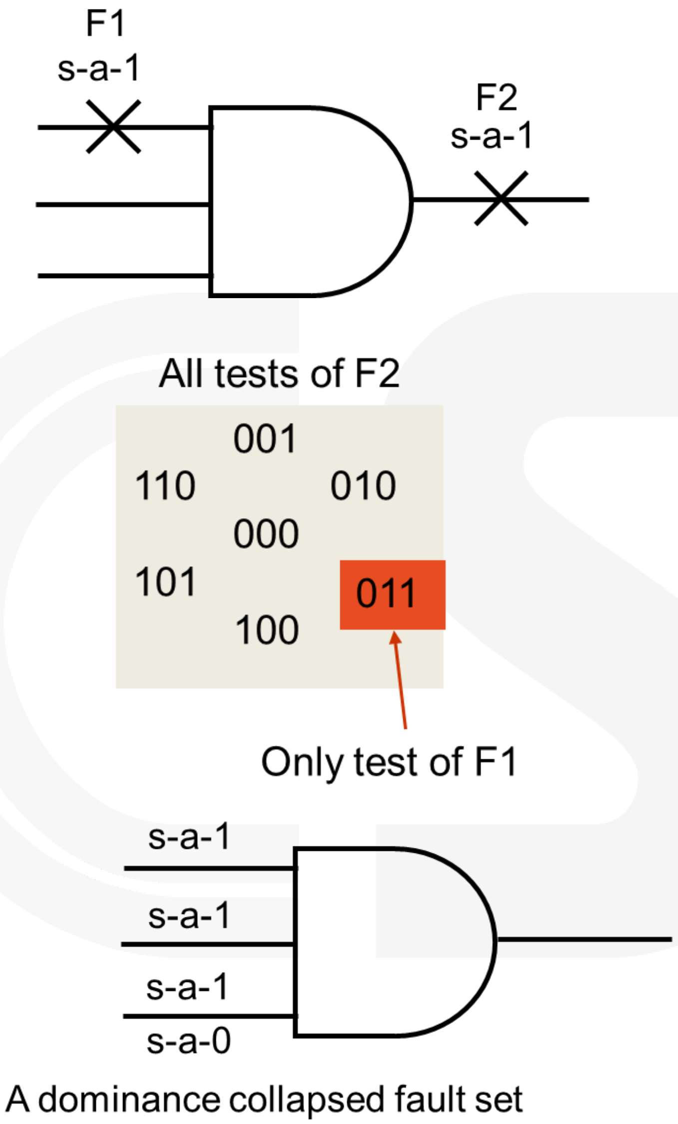


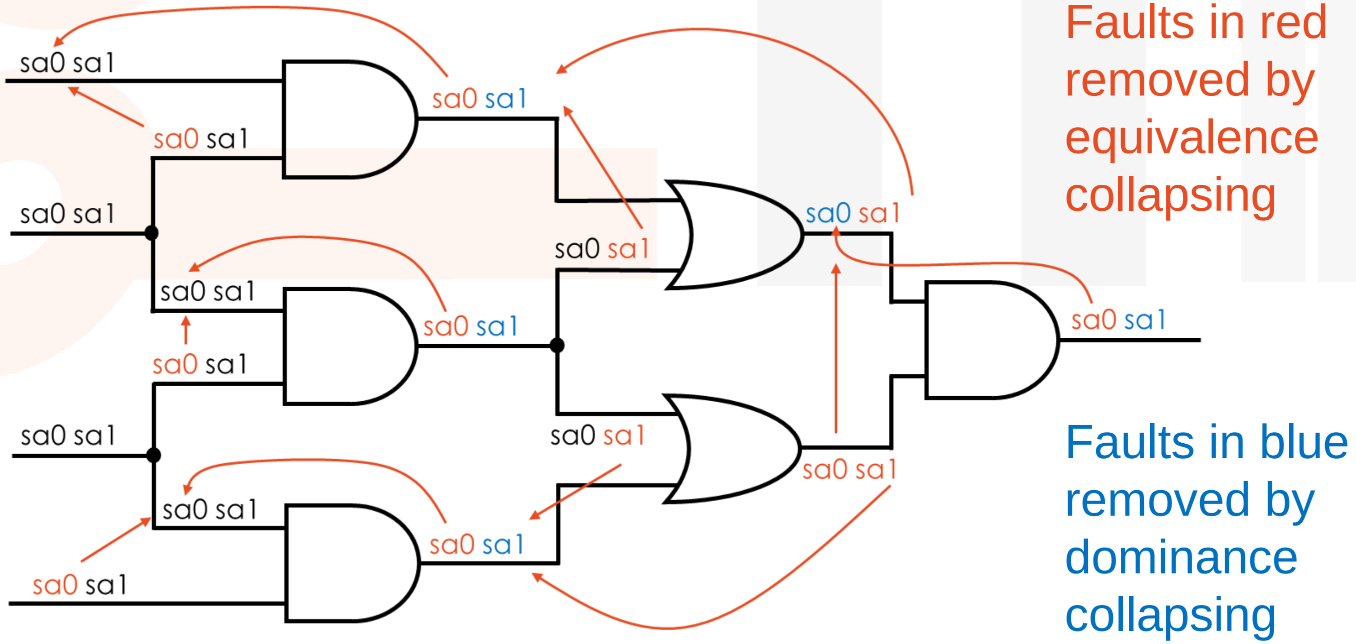
Faults in red removed by equivalence collapsing

$$\text{Collapse ratio} = \frac{20}{32} = 0.625$$

Fault Dominance

- **If all tests of some fault *f1* detect another fault *f2*, then *f2* is said to *dominate* *f1*.**
  - i.e., the tests for *f1* are a subset of those for *f2*
- **We can then remove (collapse) *f2***
  - i.e., testing *f1* is sufficient to test *f2*

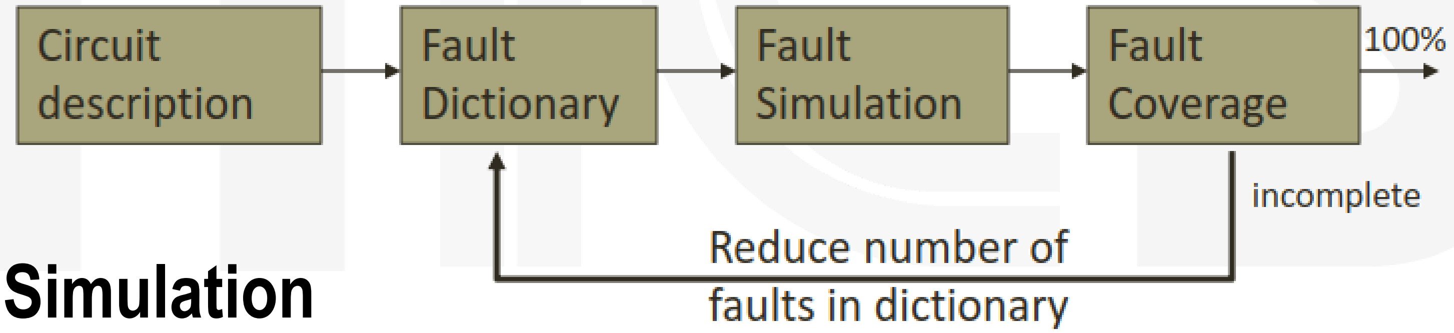




Fault Simulation and Coverage

- **Given a circuit, a set of test vectors T, and a fault list F, fault simulation computes the faults in F detected by T.**
- **Fault Coverage is the measure of the ability of a test to detect a given defect**

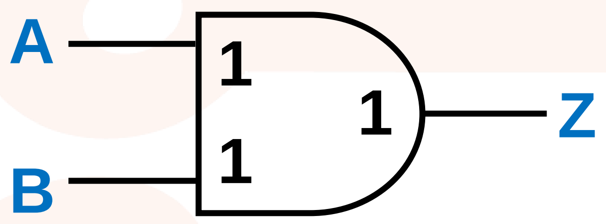
$$Fault\ Coverage = \frac{Number\ of\ Detected\ Faults}{Number\ of\ Possible\ Faults}$$



- **Trivial Approach: Serial Fault Simulation**
  - For each fault in the fault list, inject one fault to the netlist
  - Simulate the modified netlist and compare response to fault-free netlist
  - Not feasible! Requires *huge* number of simulations.

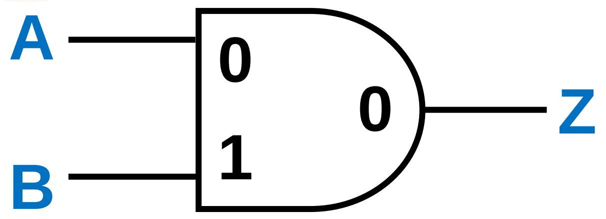
Deductive Fault Simulation

- For each test vector mark the faults that it detects at each line.
- Propagate through the circuit in *event-driven* fashion.
- Let's start with the simple example of a 2-input AND Gate



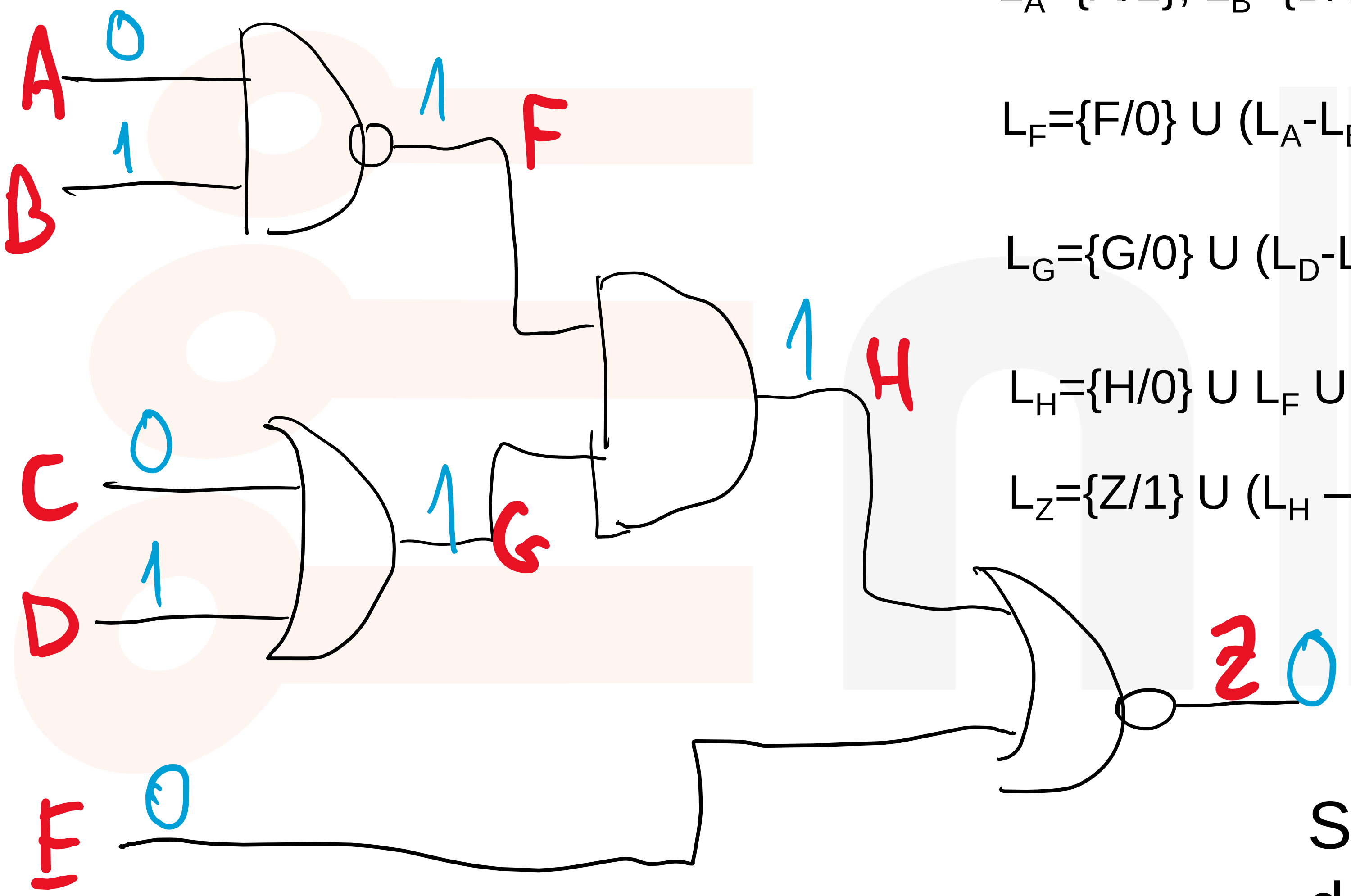
$L_A=\{A/0\}$
$L_B=\{B/0\}$
$L_Z=L_A \cup L_B \cup \{Z/0\} = \{A/0, B/0, Z/0\}$

- If one of the inputs shouldn't change, we mark it with a "bar" and it is equivalent to the set difference:



$L_A=\{A/1\}$
$L_B=\{B/0\}$
$L_Z=L_A \cup \overline{L_B} \cup \{Z/1\} = (L_A-L_B) \cup \{Z/1\} =\{A/0, Z/0\}$

Deductive Fault Simulation



L_A={A/1}, L_B={B/0} , L_C={C/1} , L_D={D/0} , L_E={E/1}

L_F={F/0} U (L_A-L_B)={F/0, A/1}

L_G={G/0} U (L_D-L_C)={G/0, D/0}

L_H={H/0} U L_F U L_G = {H/0, F/0, A/1, G/0, D/0}

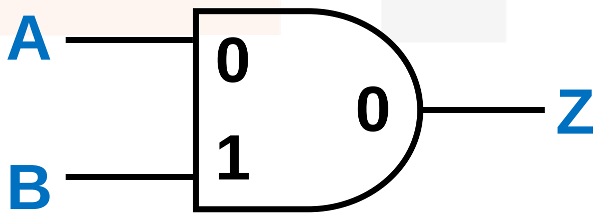
L_Z={Z/1} U (L_H – L_E)= {Z/1, H/0, F/0, A/1, G/0, D/0}

So T1= {01010} was able to detect 6 faults!

Concurrent Fault Simulation

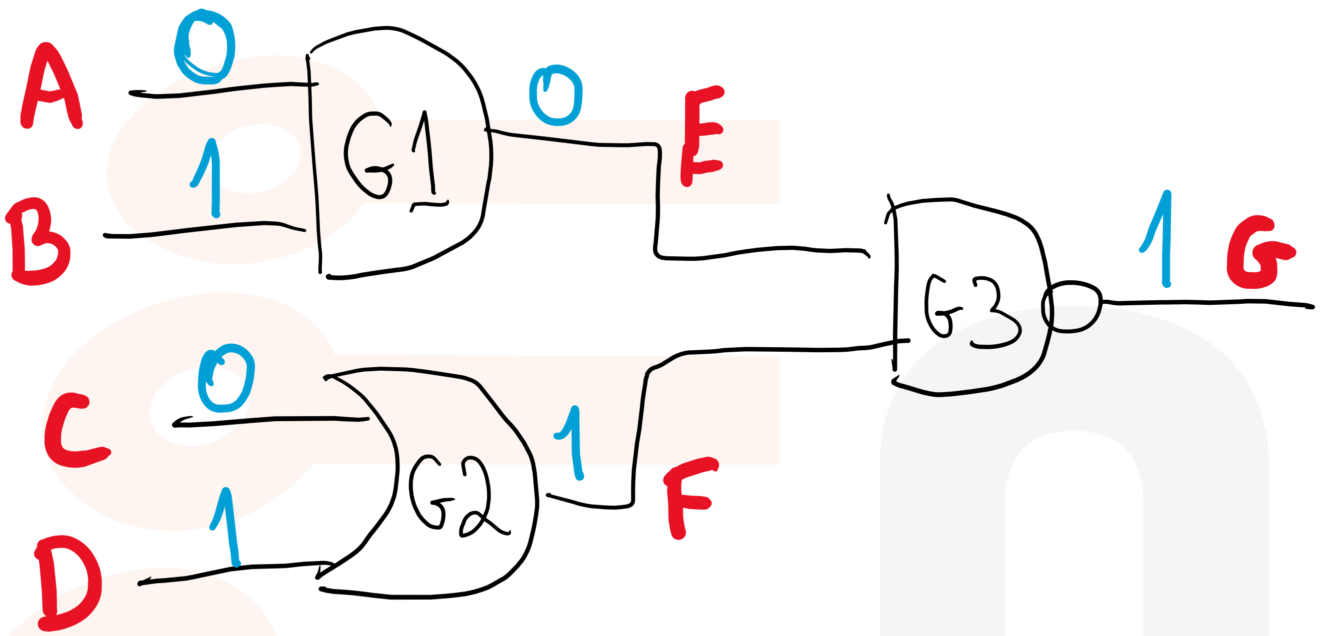
- Concurrently simulate the *fault-free* and *faulty* versions of the circuit for a given vector.
- For every gate, we maintain a concurrent fault list:
  - <fault : input values : output value>

- Example of a 2-input AND gate:



A/1	:	11	:	1
B/0	:	00	:	0
Z/1	:	01	:	1

Concurrent Fault Simulation



Gate G1

A/1	11	1
B/0	00	0
E/1	01	1

Gate G2

C/1	11	1
D/0	00	0
F/0	00	0

Gate G3

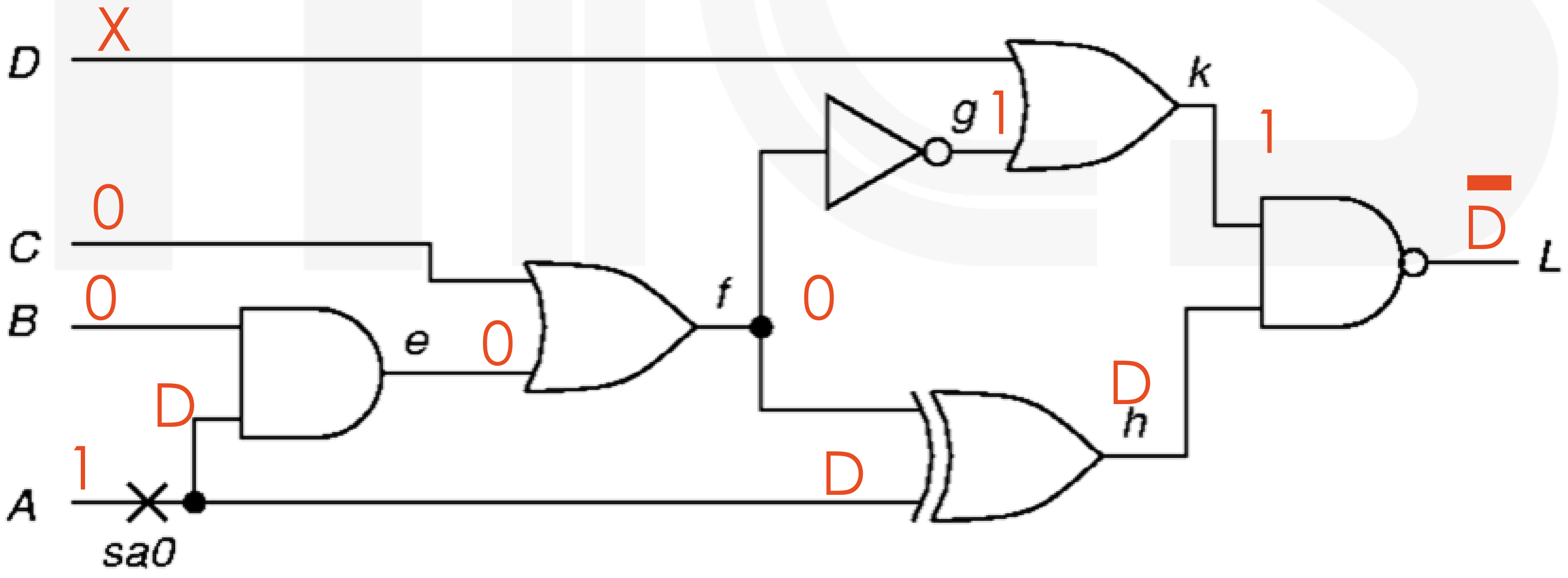
E/1	11	0	Local Response
F/0	00	1	
G/0	01	0	
A/1	11	0	From G1
E/1	11	0	
D/0	00	1	From G2
F/0	00	1	

Test Vector Generation

- **So we saw how to run fault simulation and recognize all the faults that can be detected by a given vector.**
- **Therefore, we can:**
  - Randomly choose a test vector
  - Collect all faults detected by the same vector
  - Reduce fault list (dictionary)
  - Repeat until all faults are detected
- **However, how can we find the test vector for testing a specific fault?**
- **The general approach has three stages:**
  - Control: Excite the fault
  - Observe: Propagate the fault through the network all the way to the output
  - Detect: Output value of good and faulty circuit

D-Algorithm Concept (Roth, 1966)

- **Fault Sensitization (=Assignment)**
  - Assert the opposite of the fault at the fault site.
  - This creates a D or D'
- **Fault Propagation (=Forward Drive)**
  - Propagate the fault to a primary output
- **Line Justification (=Backward Trace)**
  - Find the primary inputs that will ensure this propagation



Points to note

- **Path sensitization is not as simple as the example shows**
  - Fanout and reconvergence may cause conflicts during backward trace
- **This has led to the development of many sophisticated algorithms**
  - PODEM (Goel, 1981)
  - FAN (Fujiwara, 1986)
  - Others
- **Test generation is slower than fault simulation**

Design For Test

Controllability and Observability

- **DFT Mantra**
  - To provide controllability and observability
- **Controllability**
  - The ability to bring a system to any given state.
  - In chip design, this means that any desired value can be produced at the internal signals of the circuit by controlling the primary inputs,.
- **Observability**
  - The ability to evaluate what the current state of the system is.
  - In chip design, this means that any internal signal can be propagated to a primary output.
- **Combinational circuits have inherently high controllability and observability, but the opposite is true for sequential circuits.**

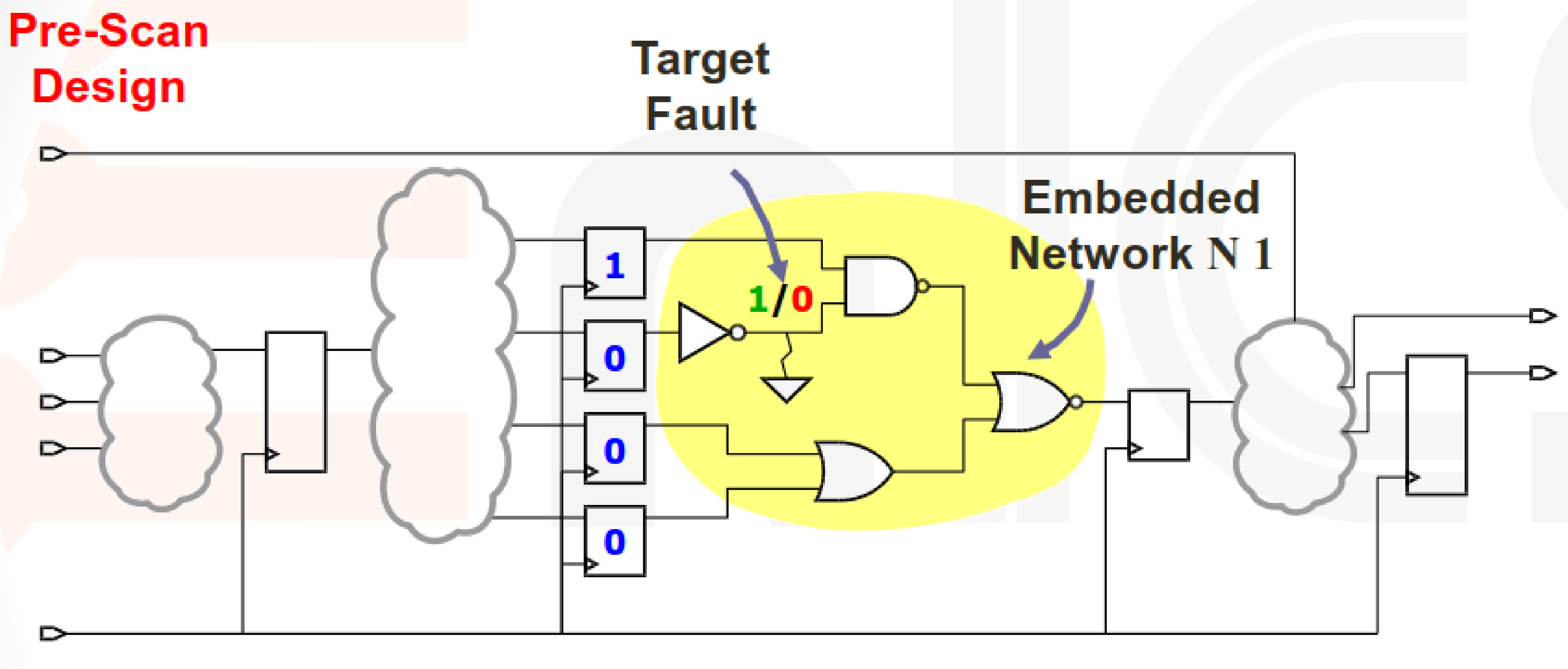
Design For Test

- **Embedding testing circuitry along with the functional Circuit Under Test (CUT) aiming to alleviate the testing process and to enhance testability**

- **We can assist testing by applying embedded Design For Testability (DFT) techniques, e.g.**
  - Adding control and observation points
  - Scan Chains
  - Built-In Self Test (BIST)

The sequential circuit problem

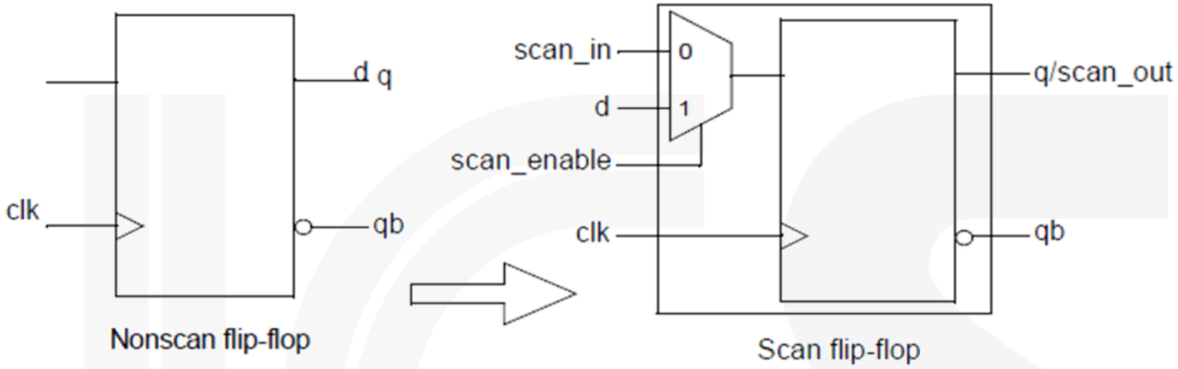
- Sequential designs have limited controllability and observability.

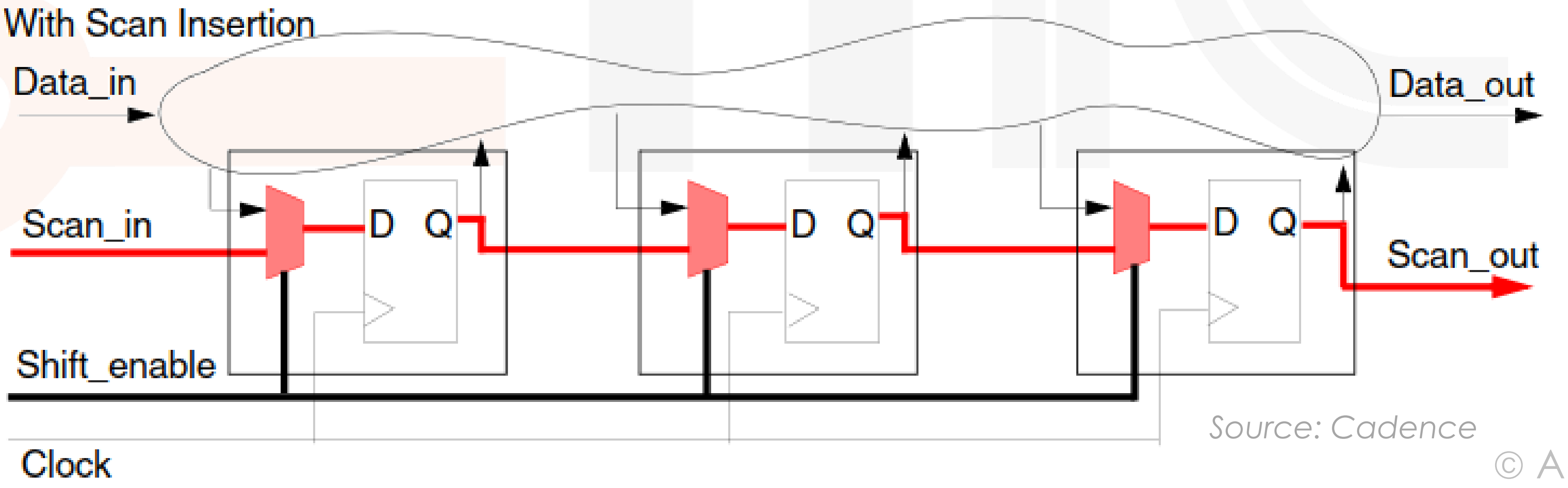


- We can gain controllability/observability by turning our sequential circuits into a collection of combinatorial circuits.

Scan Chain Insertion

- **Replace all flip flops with scan flip flops**
  - FFs become shift register in test mode
- **Control the scan chains with new ports:**
  - Scan In – input to the scan chain
  - Scan Out – output from the scan chain
  - Scan Enable – toggle between test and operational mode

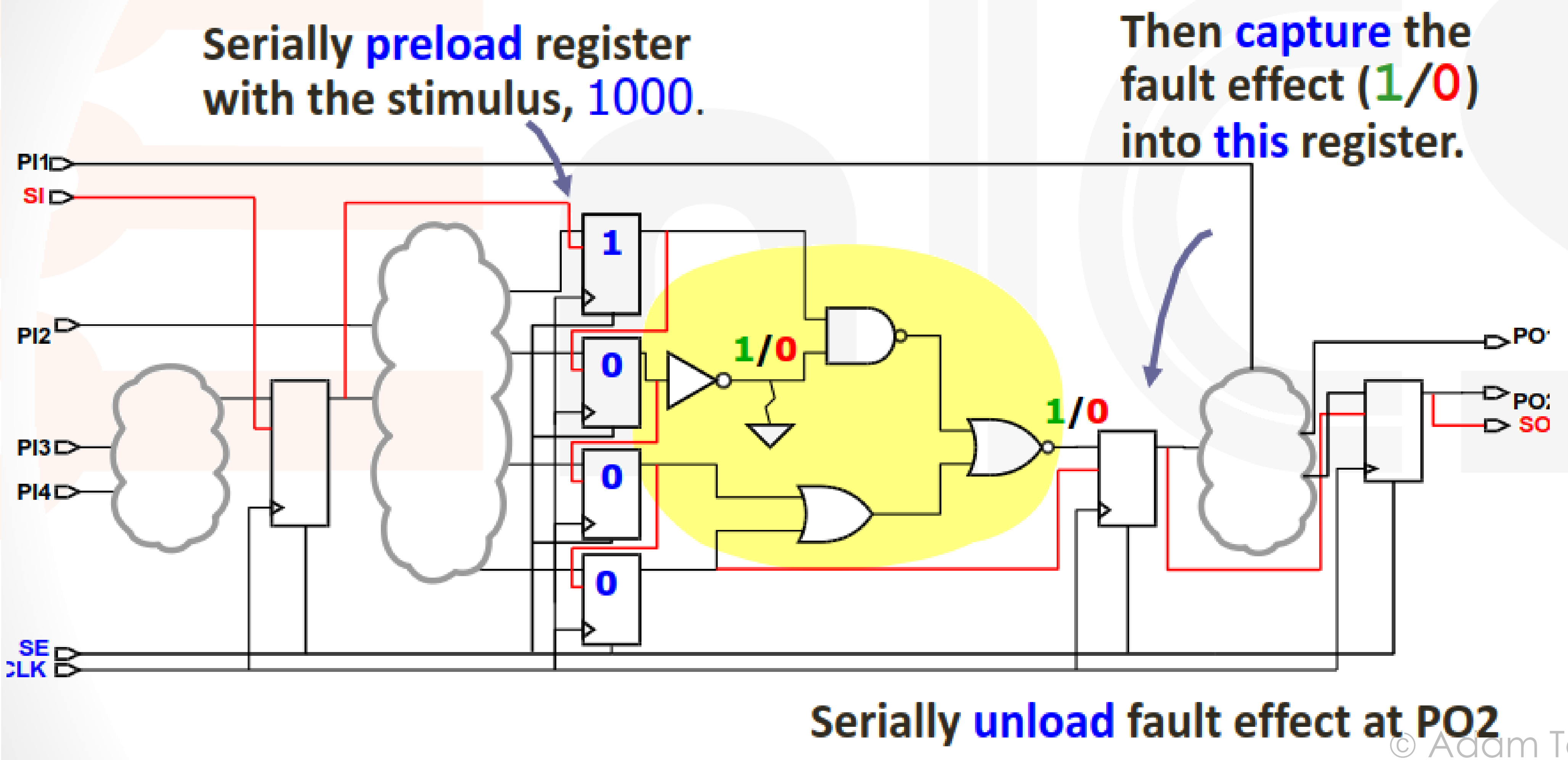




Source: Cadence

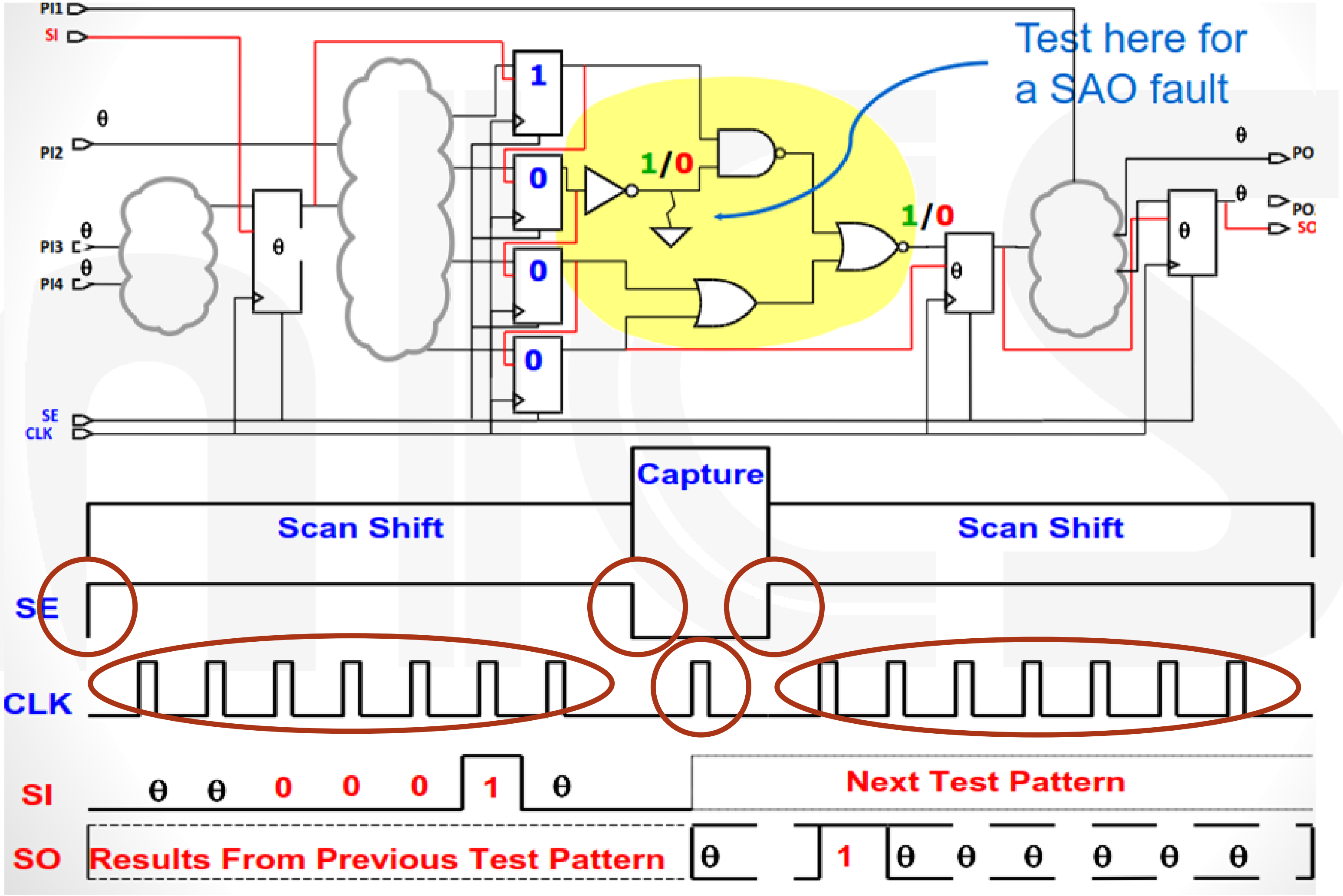
Scan Chain Insertion

- Now we can serially "scan-in" the sanitization vector.



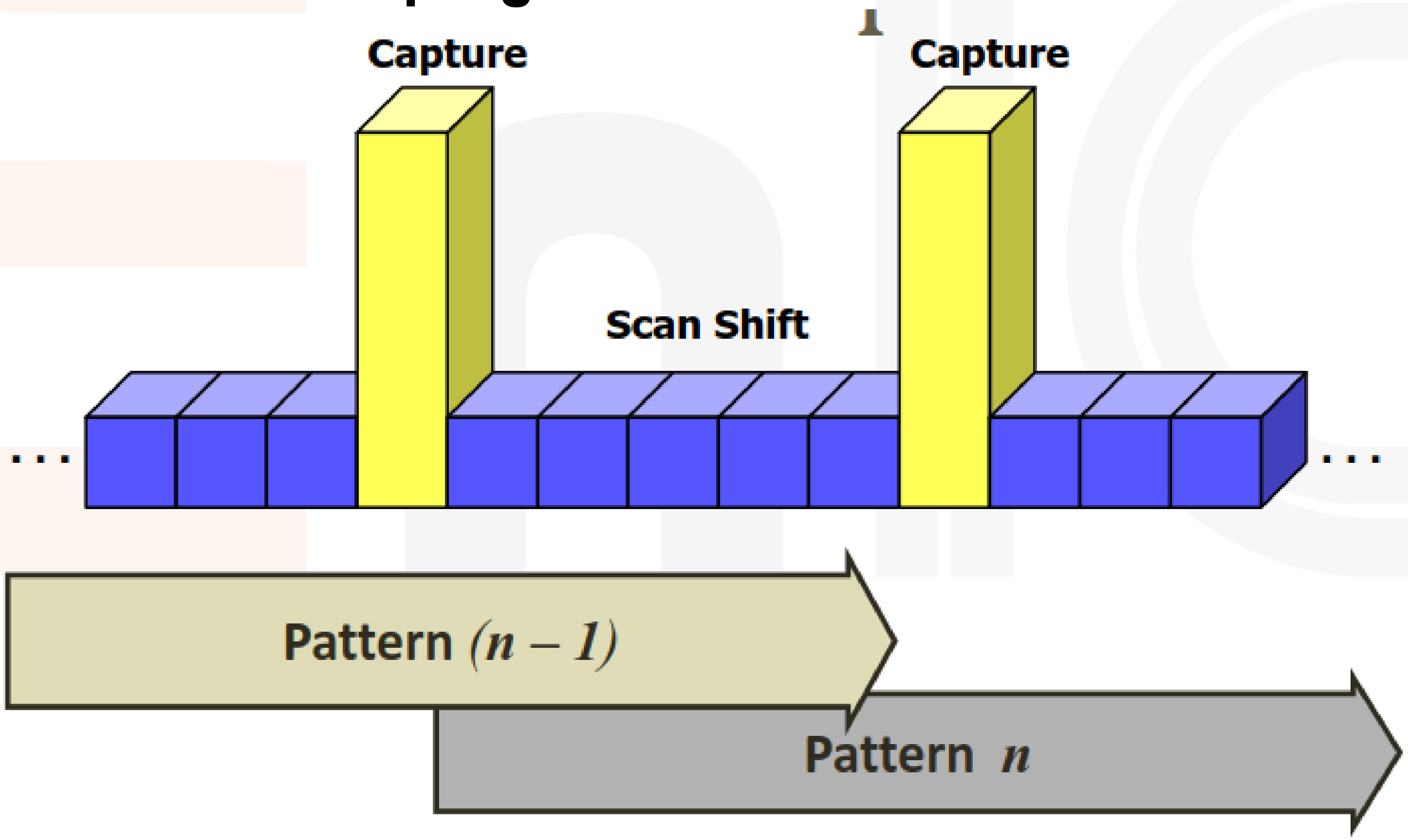
Scan Testing Protocol

- Enable scan (SE=1)
- Shift in state vector
- Disable scan (SE=0)
- Toggle Clock to exercise fault ("Capture")
- Enable scan (SE=1)
- Shift out result vector



Test Patterns Overlap

- Scanning out of previous pattern overlaps scanning in of next - for all but first and last patterns in the test program.



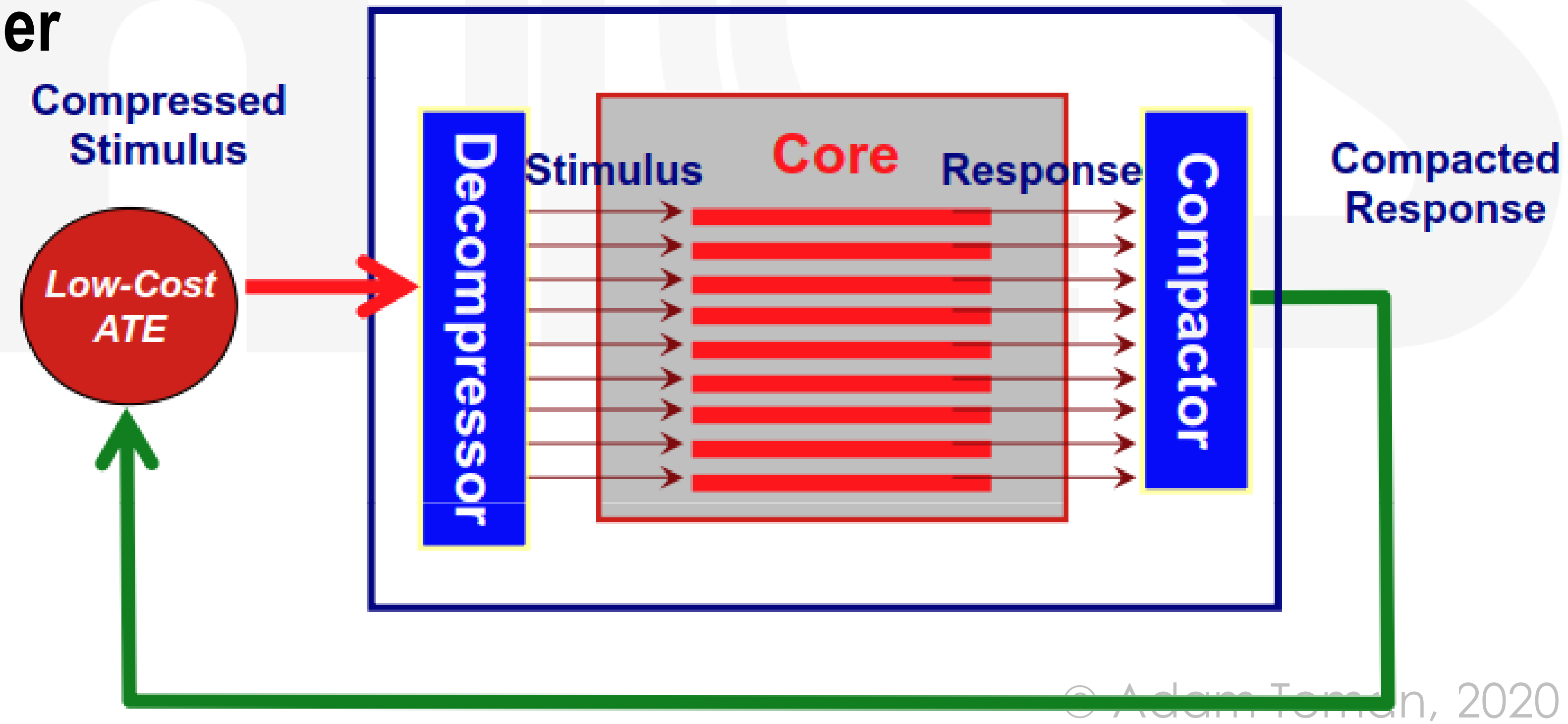
Overlapping Cycles

Scan Chains Implementation

- **Implementation Considerations**
  - The scan clock is typically slower than operational mode (e.g. 100-200 MHz)
  - Usually a number of parallel scan chains are implemented
  - Special care needs to be taken to avoid undesirable conditions during scan, e.g. bus contentions, resets, clock gates, etc.
  - One potential problem associated with scan chains is a temporary excessive current draw due to many flops toggling at the same clock
- **Additional usages:**
  - Debug – scan chain provides visibility to the full internal state of the circuit (hence a long scan chain mode is very useable)
  - Reset propagation

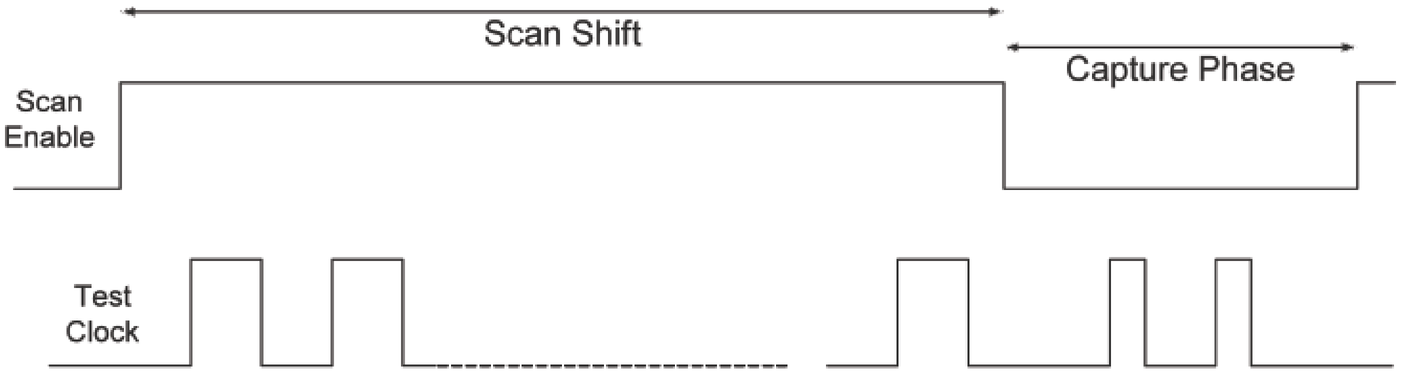
Test Pattern Compression

- Larger designs need to use embedded compression to reduce the pattern volume and test time
- Compression works by dividing the chip's scan chains into smaller balanced chains that are connected between a decompressor and a compactor
- The tester patterns are smaller by a couple orders of magnitude, and only a few primary I/Os need to be connected to the external tester



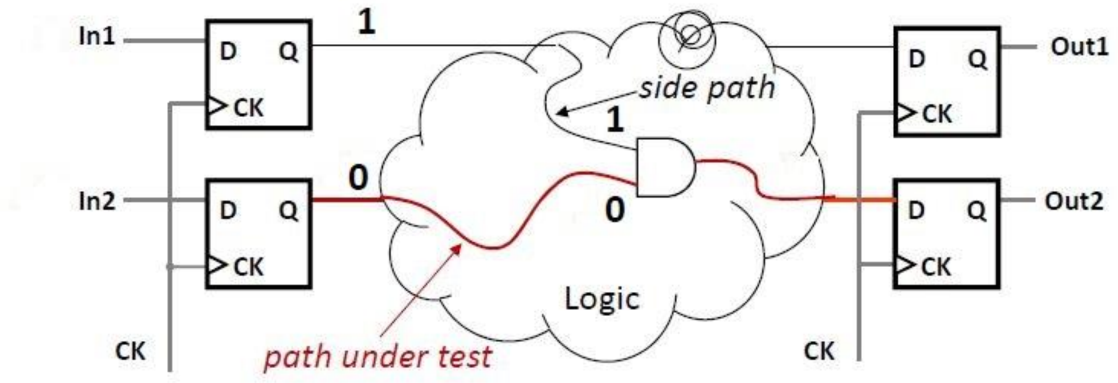
AC Scan (at-speed test)

- **Used to verify:**
  - Transition Faults
  - Delay Faults
- **A path delay fault requires a pair of subsequent vectors to be detected**
  - The first test vector initializes the circuit
  - The second test vector activates the path under test

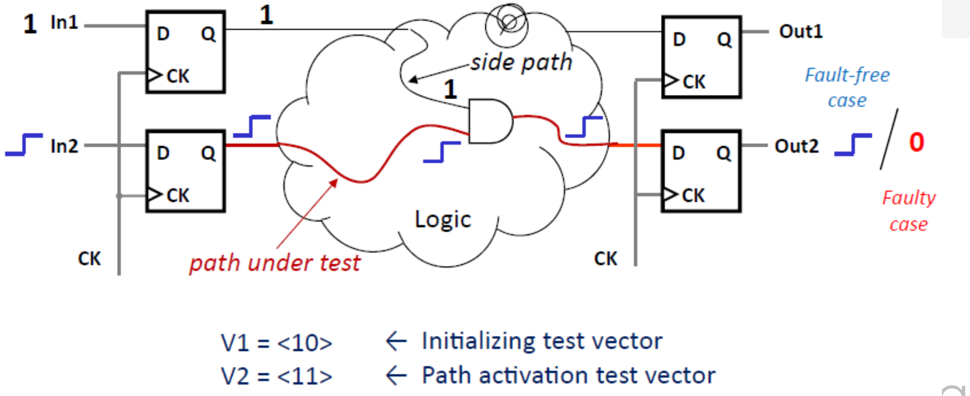


Source: Atrenta

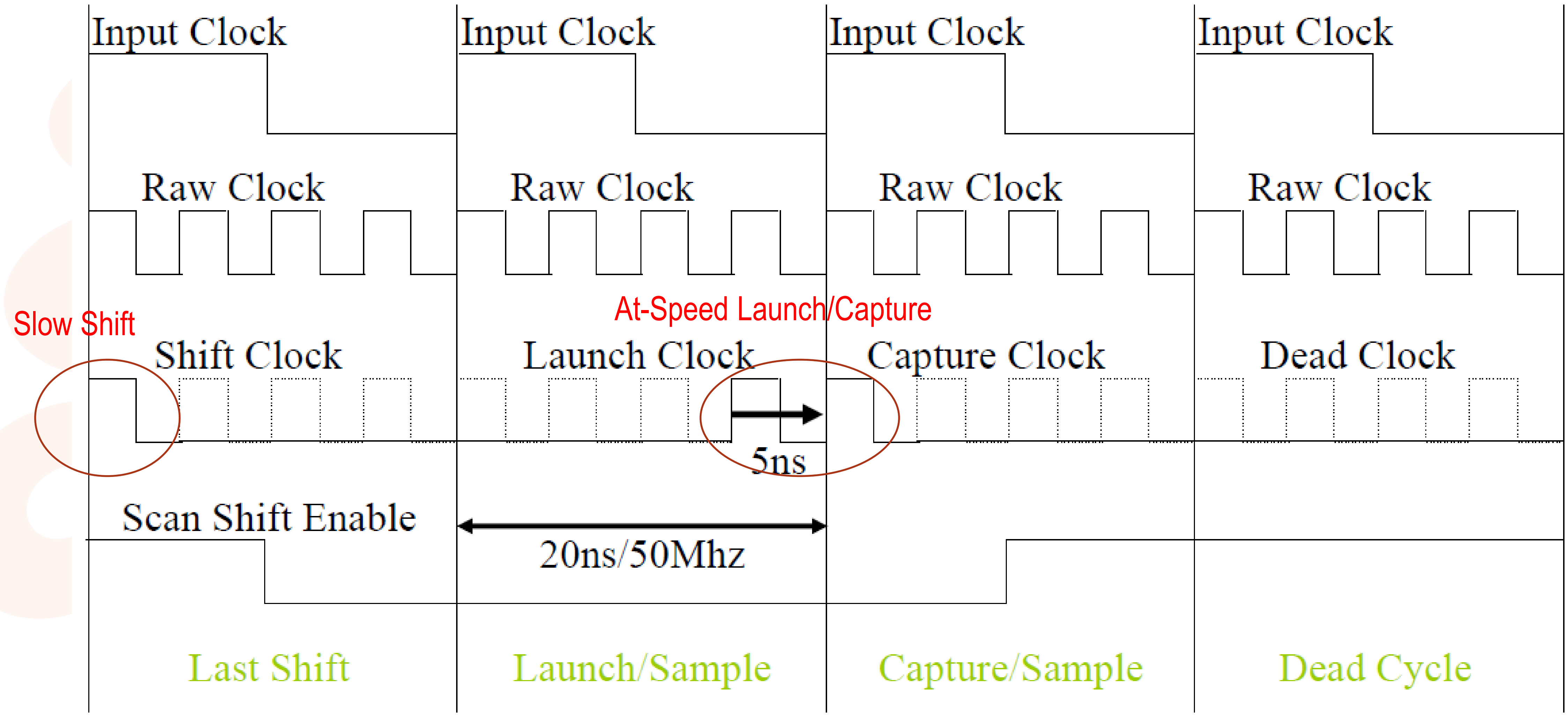
(I) Initialization



(II) Launch & Capture



AC Scan Timing



IDDQ Test

- **Tests quiescent power consumption**
  - Many fabrication defects cause higher static power consumption (orders of magnitude)
  - Difficult for deep sub-micron process due to high leakage current
- **Design Considerations**
  - Pure static logic design
  - Buses
    - No floating nodes
    - No driver conflicts
    - No pull-up or pulldown resistors
  - Separate power supply for analog
- **Advantages**
  - Covers non-stuck-at faults
  - Cheap test equipment
  - Few test vectors
- **Disadvantages**
  - Slow current measurement
  - Difficult to determine proper IDDQ threshold
  - Not practicable for deep sub-micron

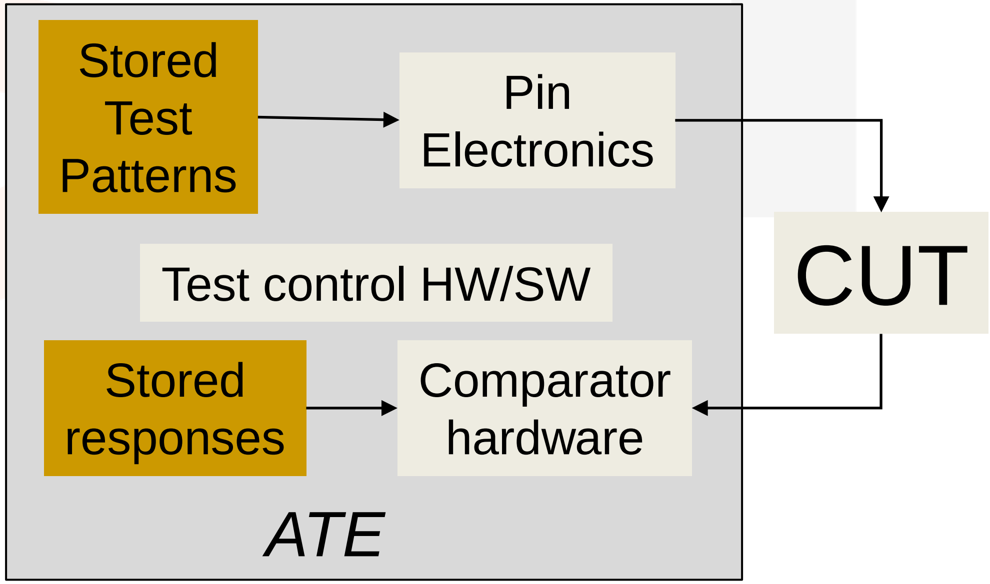
Built-In Self Test

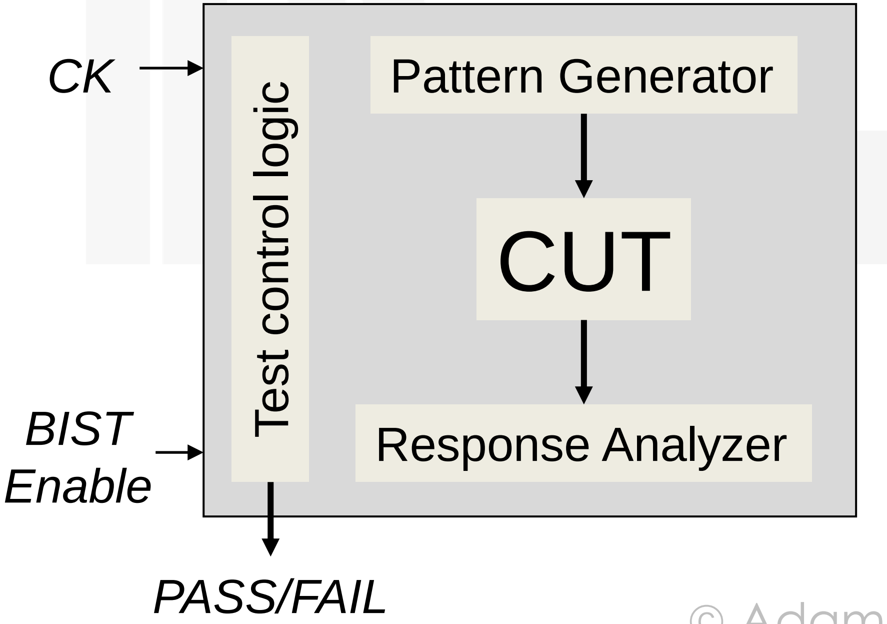
EnICS Emerging Nanoscaled Integrated Circuits and Systems Labs

Tradition of Excellence Bar-Ilan University אוניברסיטת בר-אילן

Built-in Self-Test

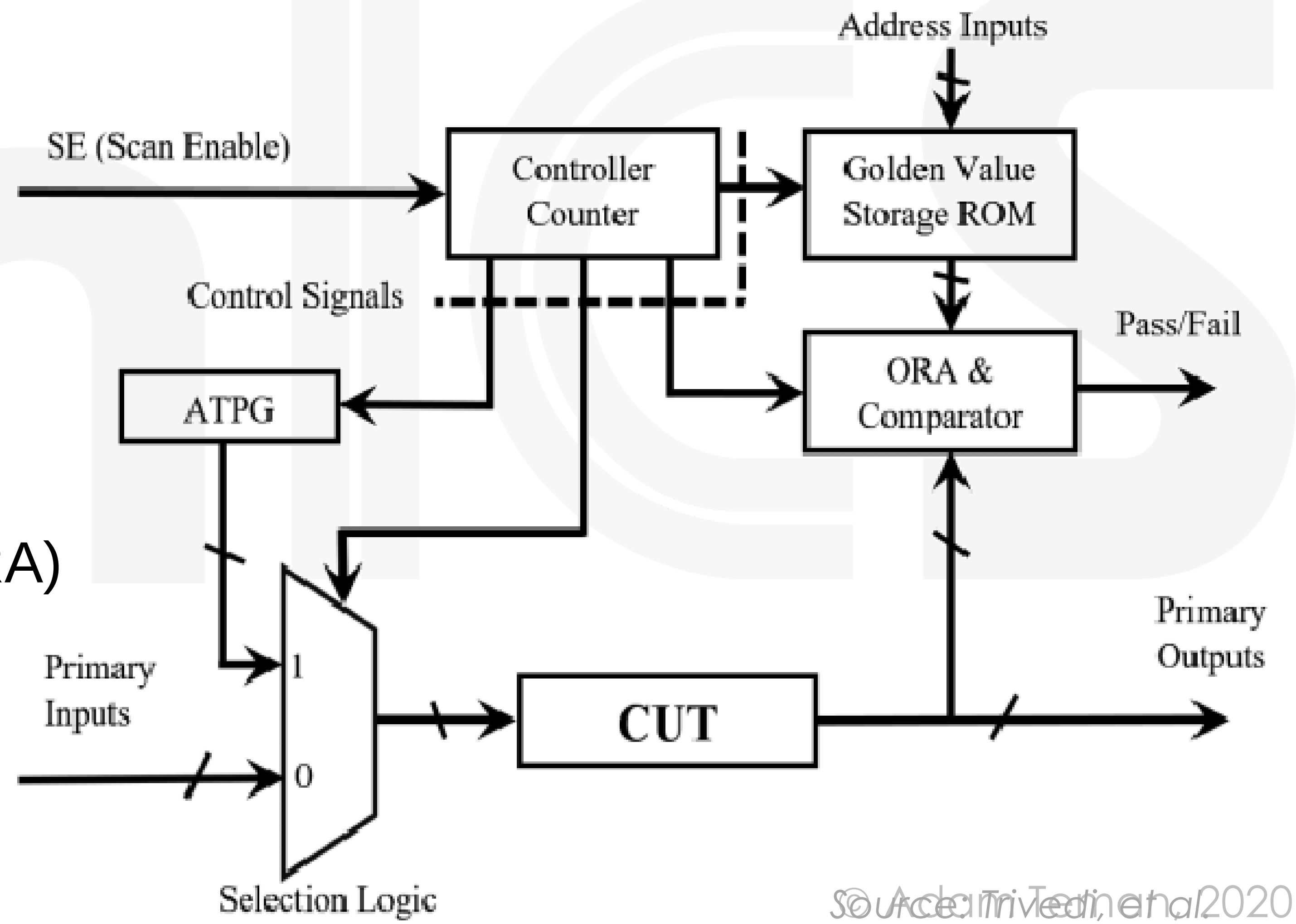
- **Testing with an ATE is expensive**
  - We need to stream in a test vector and stream out the result for evaluation.
- **Instead, add dedicated hardware for test generation and response evaluation**
  - Done on chip for specific blocks (such as memory)
  - Hardware overhead, but runs much faster and can be done in the field.
  - Can run at-speed





BIST Architecture

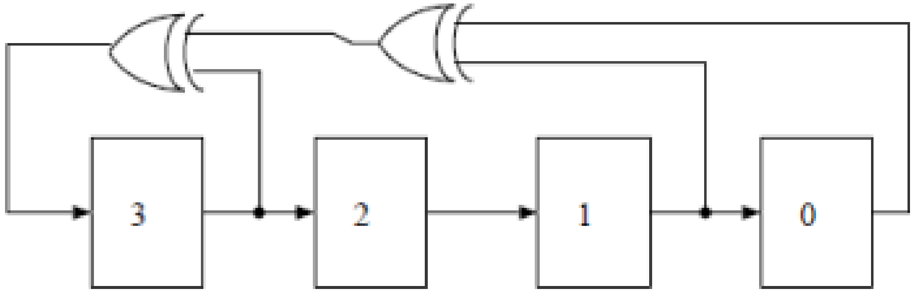
- **Pattern generator**
  - RAM or ROM with stored deterministic patterns
  - Counter
  - Random pattern generator
- **Response Compactor**
  - Compactor – reduces size of output for comparison to ROM
  - Output Response Analyzer (ORA) generates pass/fail

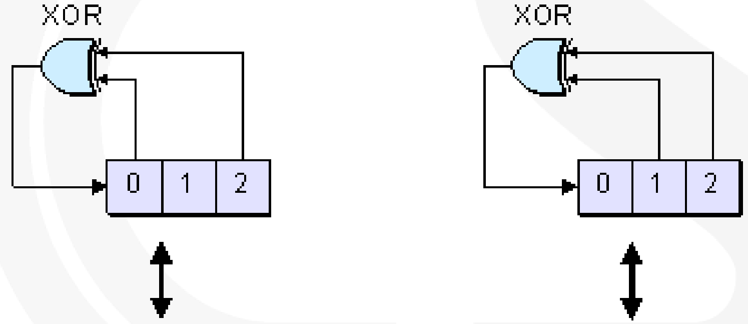


Source: Trivedi, et al.

Linear Feedback Shift Register (LSFR)

- **Pseudo-Random Pattern Generator**
- **Pattern depends on:**
  - Feedback function (XOR, XNOR)
  - Tap Selection
  - Seed (initial value)
- **Properties**
  - Taps described by characteristic polynomial
  - *Primitive Polynomials* cannot be factored.
  - Primitive or Maximal Length provide 2^n-1 unique values
  - Cannot initialize to 0000 (1111 for XNOR)





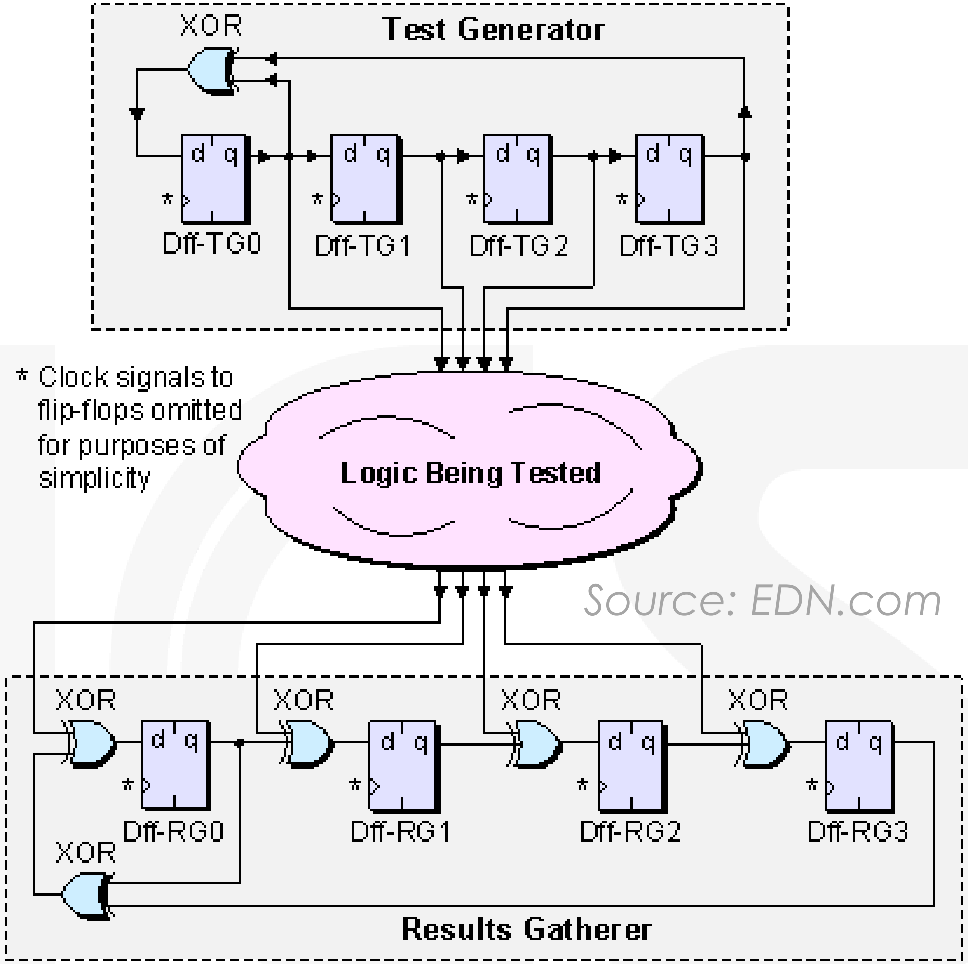
Clock	q0	q1	q2	
--	0	0	1	Initial value
1	1	0	0	
2	1	1	0	
3	1	1	1	
4	0	1	1	
5	1	0	1	
6	0	1	0	
7	0	0	1	
8	1	0	0	
9	1	1	0	
:	:	:	:	

Clock	q0	q1	q2	
--	0	0	1	Initial value
1	1	0	0	
2	0	1	0	
3	1	0	1	
4	1	1	0	
5	1	1	1	
6	0	1	1	
7	0	0	1	
8	1	0	0	
9	0	1	0	
:	:	:	:	

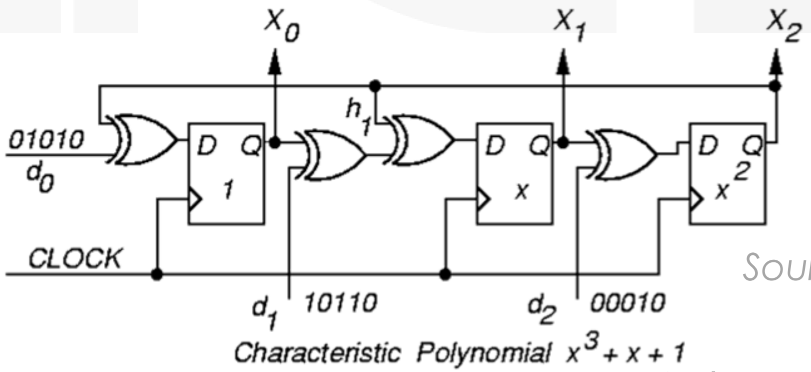
Source: EDN.com

Response Compaction

- **A response compactor (not *compressor*) reduces the response into a *signature* before comparison.**
- **The golden signature is stored on-chip.**
  - Similar to cyclic redundancy check (CRC)
- **This is basically a many-to-one mapping**
  - Can cause aliasing (low probability)
- **Two approaches**
  - *Signature Analyzer*: compact a serial bit stream into an LSFR based compactor.
  - *Multiple-input Signature Register (MISR)*: compact several bit streams in parallel



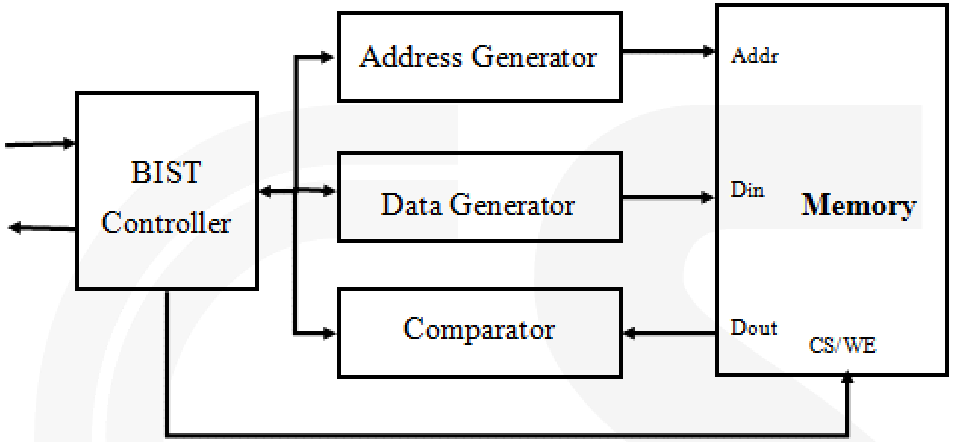
Source: EDN.com

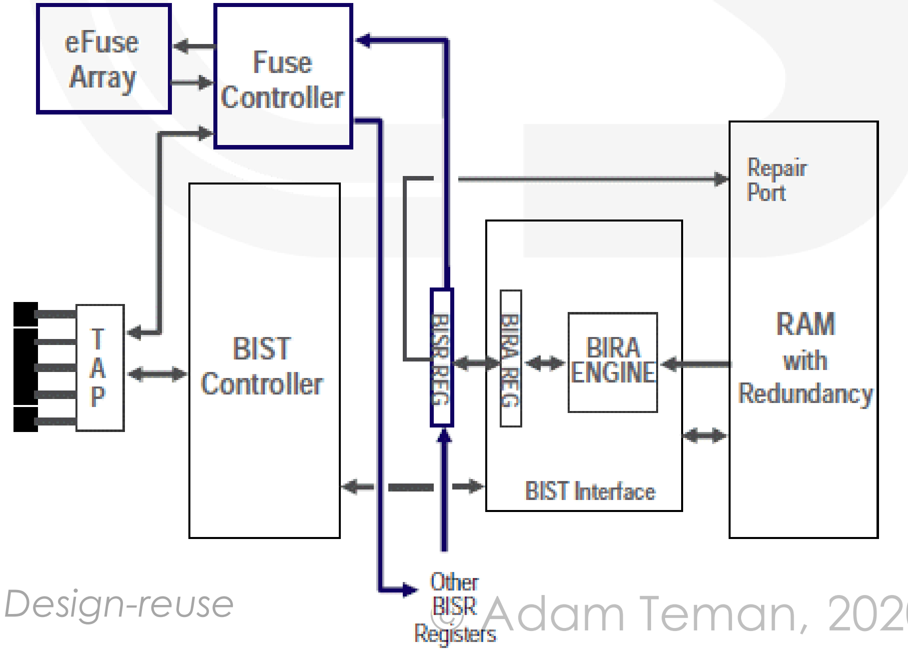


Source: Agarwal & Bushnell

Memory Built In Self Test (MBIST)

- **Many arrays (10's or 100's) of different types and different sizes are scattered in the device**
- **Memory BIST is used to test these arrays**
- **Implements commonly used testing algorithms**
  - e.g., Zero-One, Checkboard, GALPAT, Walking 1/0, March, etc.
- **Built-in Self Repair (BISR)**
  - To avoid yield loss redundant or spare rows and columns are added.
  - Memory repair swaps out faulty rows and/or columns for spare ones

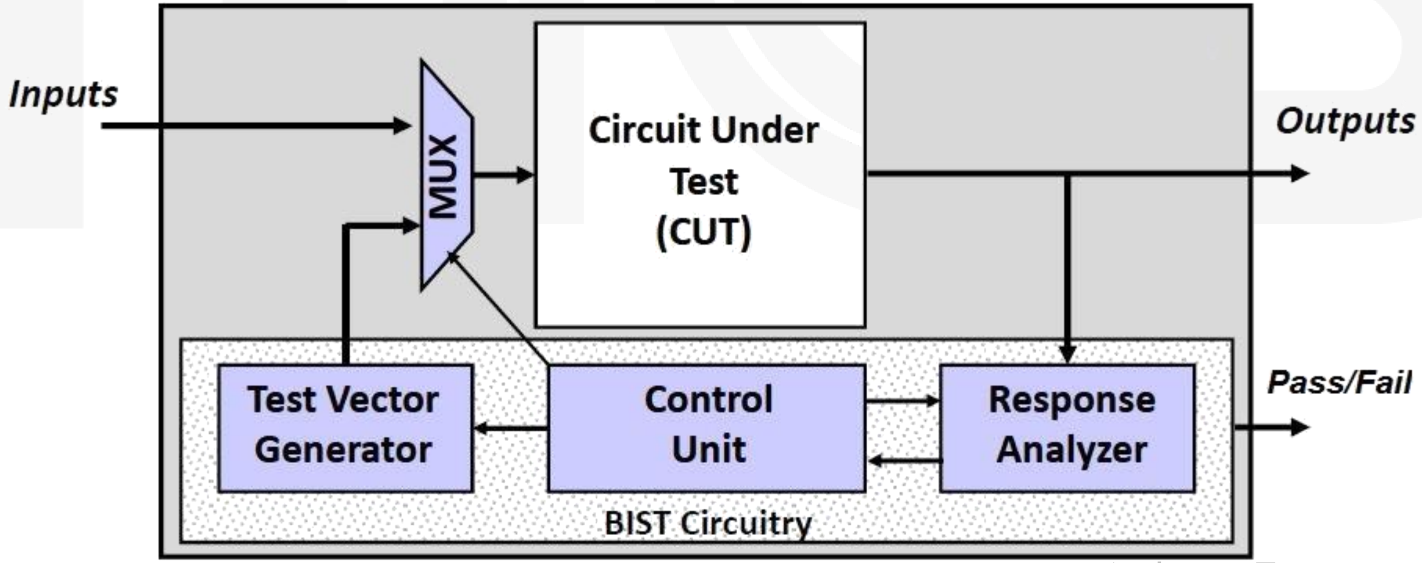




Source: Design-reuse

Logic Built In Self Test (LBIST)

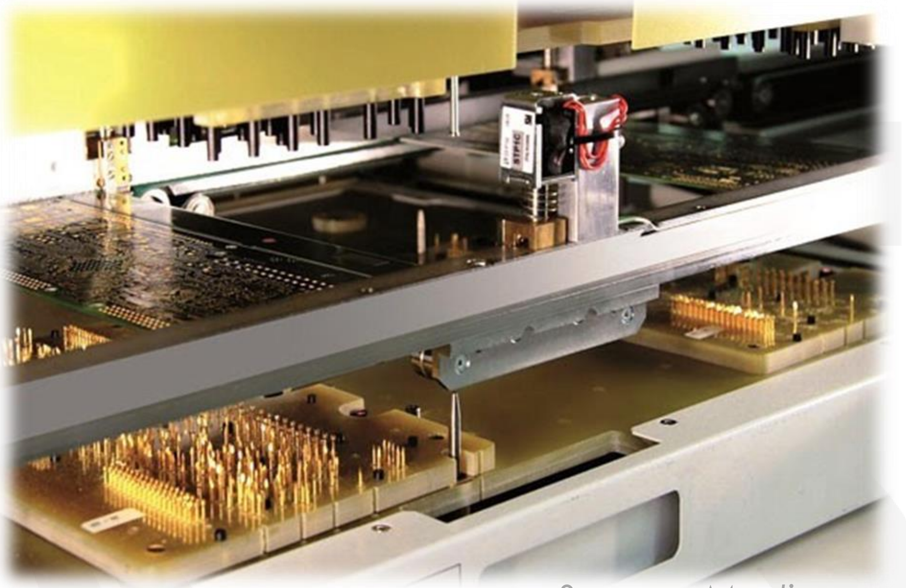
- In built-in self testing the test vectors are generated by an embedded circuit under control of the BIST Controller
- The circuit responses are compacted to a signature
- After the completion of the test, the signature is compared to the expected result to determine a PASS/FAIL result



Boundary Scan
IEEE STD 1149.1

What about board testing?

- **Testing boards is also difficult**
  - Need to verify solder joints are good
  - Drive a pin to 0, then to 1
  - Check that all connected pins get the values
- **Through-hold boards used "bed of nails"**
- **SMT and BGA boards cannot easily contact pins**

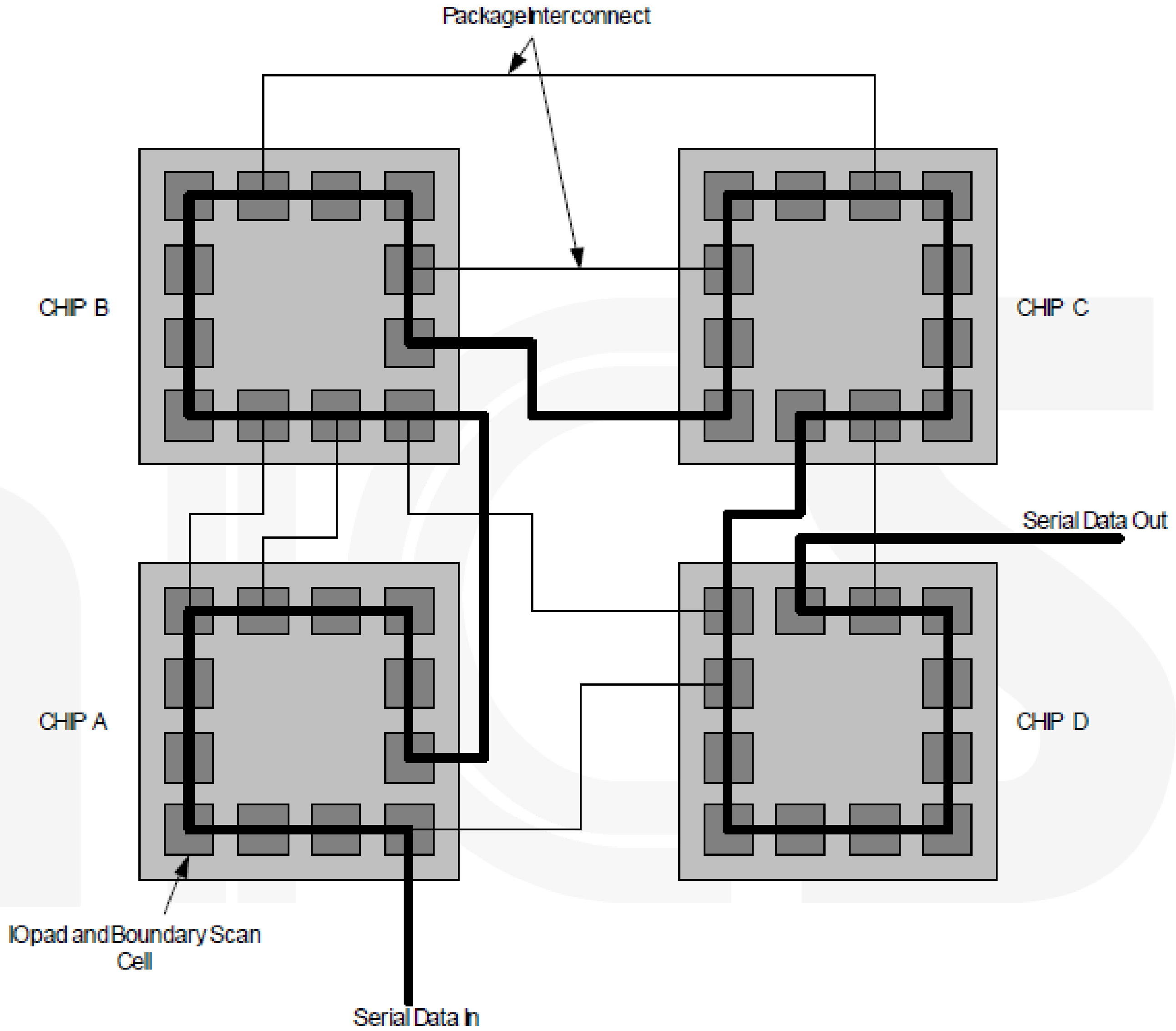


Source: Medium

- **JTAG** (Joint Test Action Group) **Standard**
  - Build capability of observing and controlling pins into each chip to make board test easier
  - IEEE 1149.1

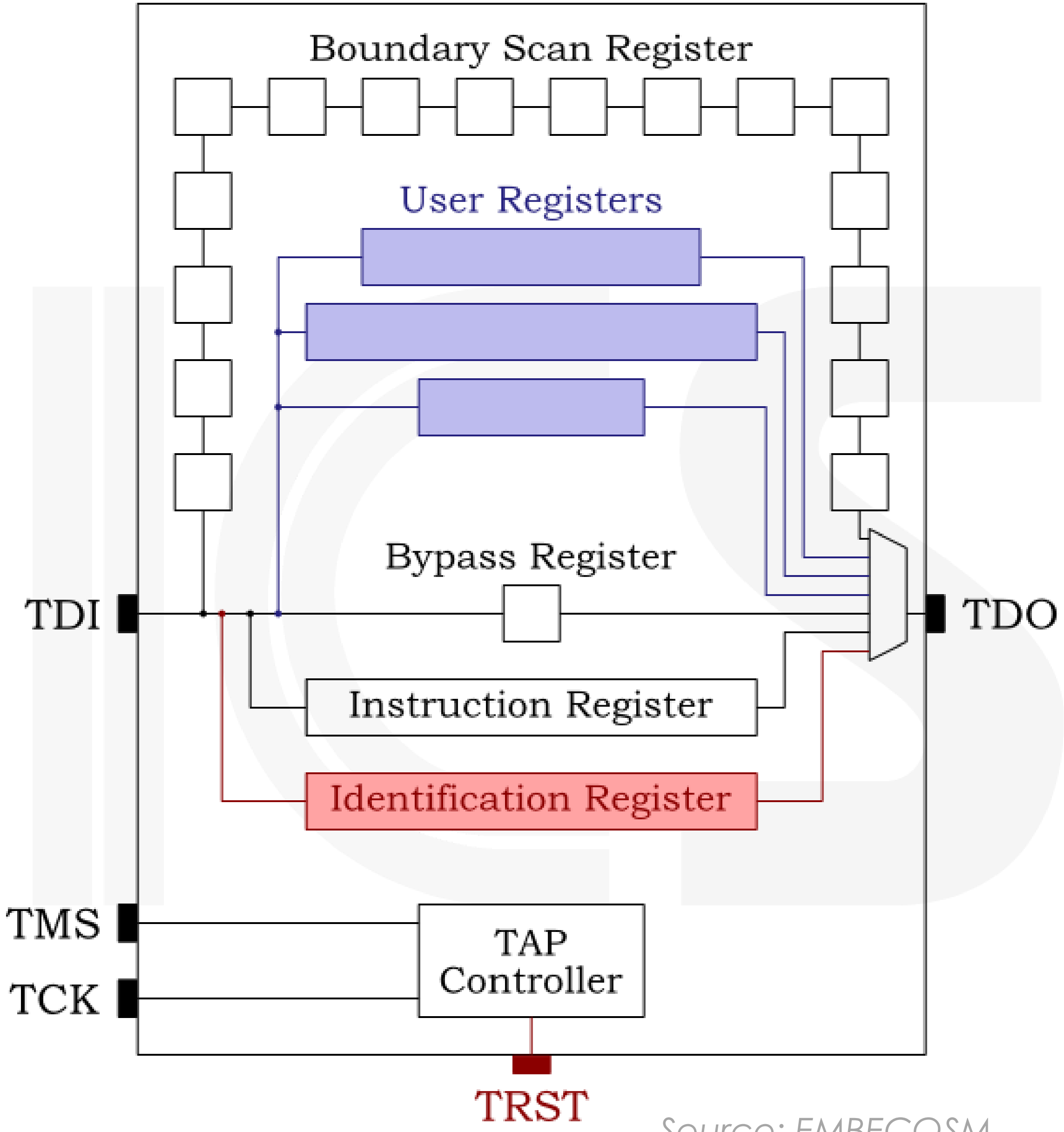
Boundary Scan

- **Used for board (PCB) level testing**
  - Used for debugging/controlling on-chip blocks
- **Implementation**
  - TAP controller (Test Access Port)
  - 4 dedicated IO pins (5th is optional)
    - TCK test clock
    - TMS test mode select
    - TDI serial test data in
    - TDO serial test data out
    - TRST test reset (optional)
  - Allows to drive and sample the device IO pins
  - Implementation is described in BSDL (Boundary Scan Description Language)



1149.1 Wrapper

- **Test Data Registers**
  - Boundary Scan Register
  - Bypass Register
  - Instruction Register
  - Device-ID Register
  - Design Specific Registers
- **Important Test Modes**
  - EXTEST: test the interconnection between devices
  - BYPASS: Forward test to next chip
  - INTEST: Test the internal logic of a chip



Source: EMBECOSM

Main References

- Yehuda Rudin, DFT Lecture, 2018
- Sagi Fisher, DFP Lecture, 2017
- M. Bushnell, V. Agrawal, VLSI Testing Course, U. Auburn
- M. Bushnell, V. Agrawal, “Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits”, Springer 2005
- I. Sengupta, VLSI Physical Design Course, IIT Kharagpur (NPTEL)
- A. Tenca, W. Ruggiero, PCS5030, Univ. Sao Paulo