

# Scan Diagnosis (Logic)

Yi-Shing Chang

Principal Engineer, FTE/GEMS/Intel

Adjunct Professor, GIAT, NTU

# Logic Diagnosis

Why diagnosis?

What kinds of professions perform diagnosis?

Logic fault diagnosis or fault isolation is the process of analyzing the failing logic portions of an integrated circuit to isolate the cause of failure

- A process of narrowing down the possible locations of the defects



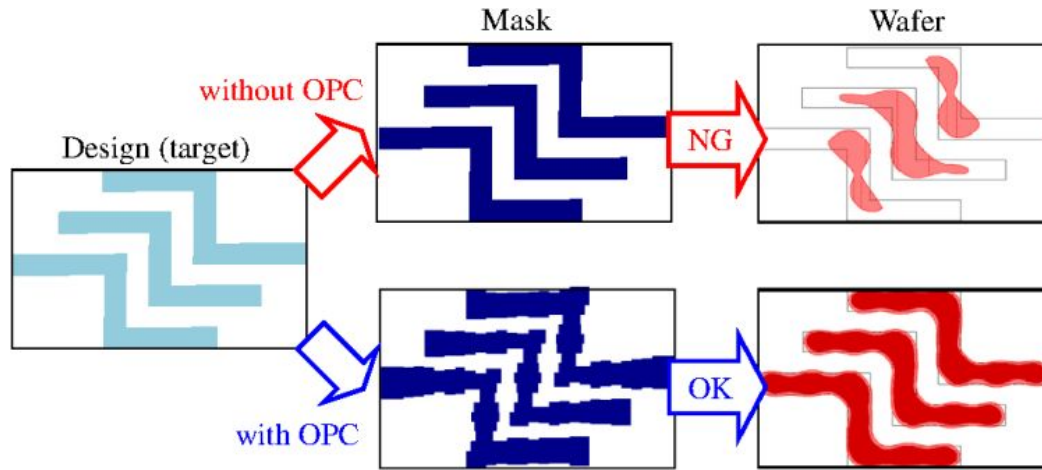
# How Can Chip Fail

Failure comes from many sources

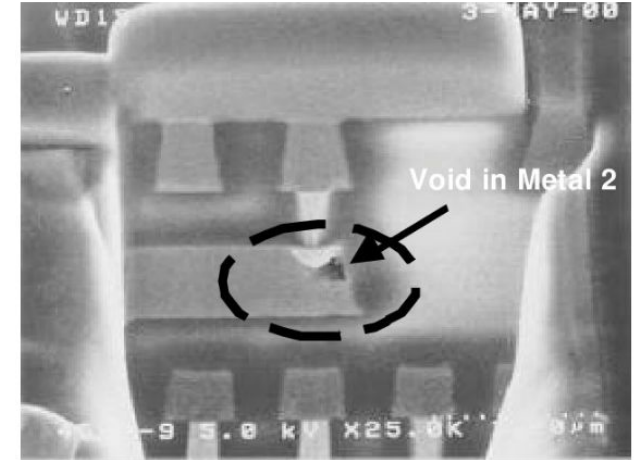
- Design Bugs: number of bugs  $\sim$  number of lines of RTL code
- Circuit Marginality
  - Timing related
  - Noise Induced
  - Operating condition related: extreme temperatures, outer space and etc.
- Manufacturing imperfection  $\rightarrow$  Design Rule Check  $\rightarrow$  Design for Manufacturing
  - Optical Proximity Correction (OPC)
  - Mask error
  - Void/bridge
  - Manufacturing process related, e.g. edge dice fail rate usually higher
- Random defects
  - Foreign particles from air or human: Clean Room



# How Can Chip Fail

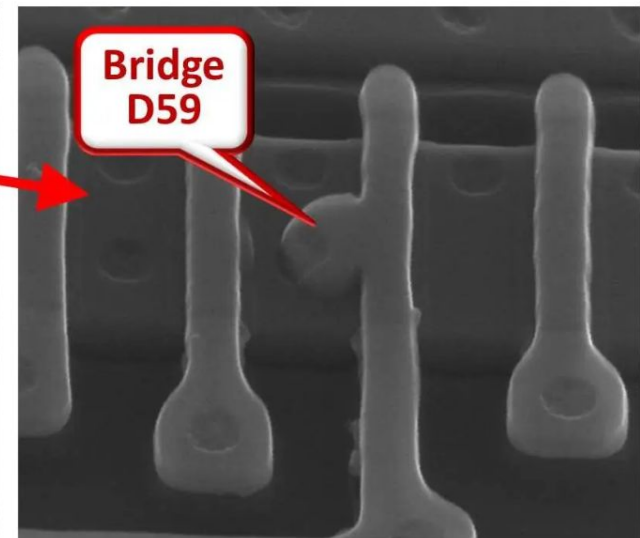
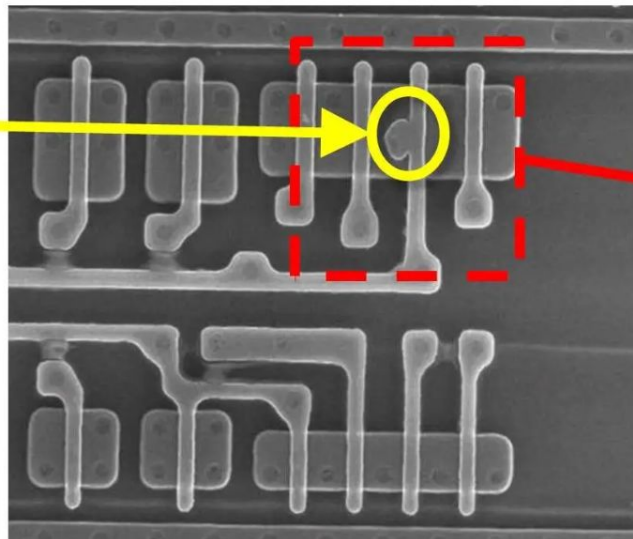
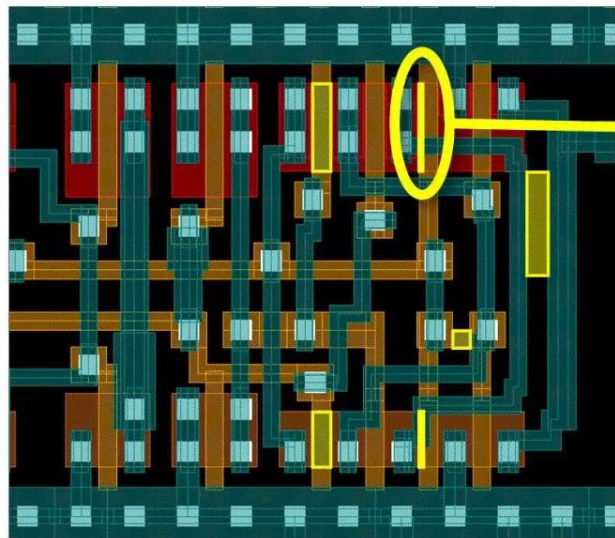


<https://semiconductorian.wordpress.com/2020/11/29/qua-trinh-phat-trien-va-san-xuat-chip/>



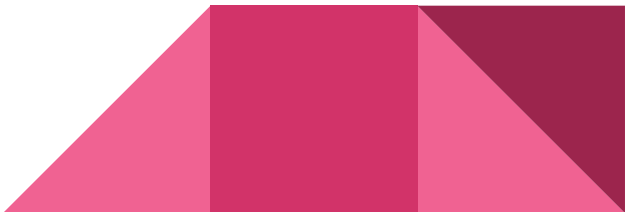
Venkataraman, and Drummonds. "Poirot: Applications of a logic fault diagnosis tool." *IEEE Design & Test of Computers* 18.1 (2001): 19-30.

suspect	score	type	cell=add
1.1	100	CELL	/top/core1/alu
...	...	...	...
1.6	100	BRIDGE	D59/MP8:G-MN8:D/1.0



Cell-aware diagnosis narrows down the location of a failure at the transistor level. Image from P. Maxwell, et.al, "Cell-Aware Diagnosis: Defective Inmates Exposed in their Cells", European Test Symposium (ETS) 2016.

# Main Purpose for Logic Diagnosis

- **Identifying Design or Functional Issues:** Logic diagnosis helps detect and pinpoint errors in the design or functionality of digital circuits, such as combinational or sequential logic, to ensure correct behavior.
  - **Fault Localization:** It helps identify the exact location of a fault in a circuit, whether it's a hardware fault (e.g., broken connections, defective gates) or a design issue.
  - **Failure Analysis:** Logic diagnosis supports understanding why a circuit or system has failed by analyzing test results and narrowing down the cause of failure.
  - **Improving Test Coverage:** By identifying patterns and weaknesses in the test process, logic diagnosis ensures that testing procedures are robust, improving the quality of future test cases.
- 

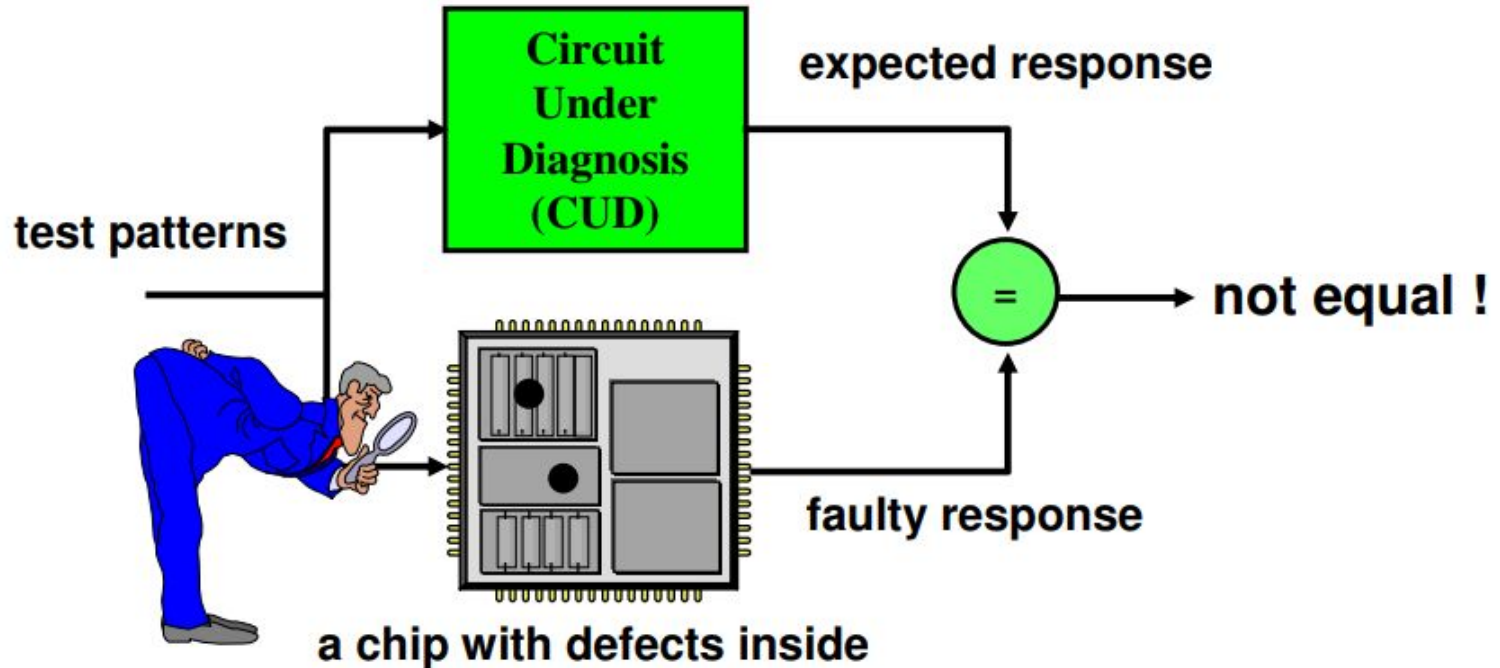
# Main Purpose for Logic Diagnosis

- **Yield Improvement:** In semiconductor manufacturing, logic diagnosis helps identify and correct manufacturing defects, improving production yield by addressing frequent or systemic issues.
- **Debugging in Post-Silicon Validation:** After chip fabrication, logic diagnosis aids in debugging and correcting issues that might not have been detected during simulation or pre-silicon testing.
- **Reducing Time-to-Market:** By quickly identifying and fixing logic issues, logic diagnosis helps speed up the design and manufacturing process, leading to faster delivery of reliable products.
- **Optimizing Circuit Design:** It can reveal inefficiencies or unnecessary complexity in the logic, enabling design optimization and enhancing performance.

In summary, logic diagnosis plays a critical role in ensuring the reliability, functionality, and performance of digital circuits in both the design and manufacturing phases. Plus pFA is very expensive and time consuming e.g. pFA for Advanced Technology node(s) is usually US\$1.5-7k/unit and take 5-30 days/unit



# Fault Diagnosis



**Question: Where are the fault locations ?**



# Combinational Logic Diagnosis

## Cause-Effect Analysis

- Fault Dictionary based paradigm

## Effect-Cause Analysis

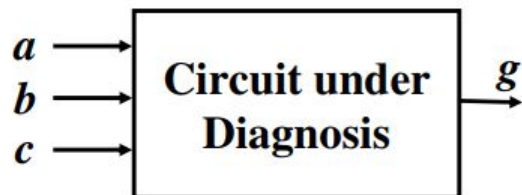
- Directly examine the failing signatures to derive the fault candidates through Boolean reasoning



# *Cause-Effect Analysis*

- **Fault dictionary (pre-analysis of all causes)**
  - **Records test response of every fault under the applied test set**
  - **Built by intensive fault simulation process**
- **A chip is diagnosed (effect matching)**
  - **By matching up the failing syndromes observed at the tester with the pre-stored fault dictionary**

# Fault Dictionary Example

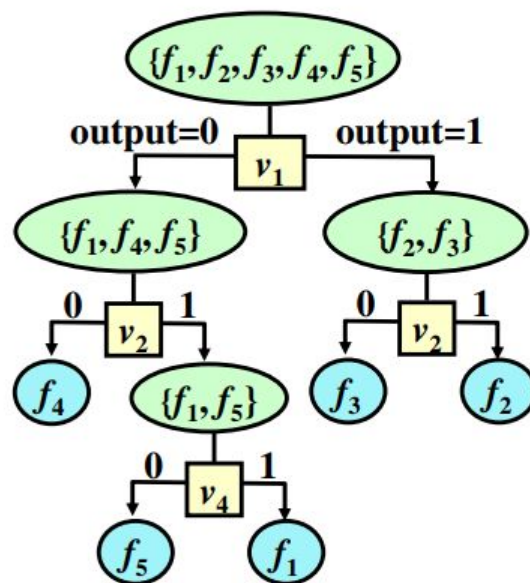


(a) Circuit under diagnosis

Circuits	Test vectors in terms of $(a, b, c)$				
	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$
fault-free	0	0	0	0	1
$f_1$	0	1	1	1	1
$f_2$	1	1	1	0	1
$f_3$	1	0	0	1	1
$f_4$	0	0	1	0	0
$f_5$	0	1	1	0	1

(b) Full-response dictionary

*A diagnosis session:  
traverse from a path from root to a leaf*



(c) Diagnostic tree

# Fault Dictionary Limitation

Size: The size of fault dictionary depends on the number of faults, test vectors, and outputs  $\sim O(F.V.O)$

- With compression & compaction, this problem can be relieved to some extent

Unmodeled Fault: Realistic defects may not behave as single stuck-at faults



# Fault Dictionary Reduction – P&R

(a) Full-response table

Fault	Output Response ( $z_1, z_2$ )			
	$t_1$	$t_2$	$t_3$	$t_4$
$f_1$	10	10	11	10
$f_2$	00	00	11	00
$f_3$	00	00	00	00
$f_4$	01	00	00	01
$f_5$	01	00	01	01
$f_6$	01	00	01	01
$f_7$	10	00	10	00
$f_8$	11	11	11	11

Fault	Pass (0) or Fail (1)			
	$t_1$	$t_2$	$t_3$	$t_4$
$f_1$	1	1	0	1
$f_2$	1	0	0	1
$f_3$	1	0	1	1
$f_4$	1	0	1	0
$f_5$	1	0	1	0
$f_6$	1	0	1	0
$f_7$	1	0	1	1
$f_8$	0	1	0	1

VLSI Test (b) Pass-fail dictionary

(c) P&R compression dictionary

Fault ID	Pass-fail + Extra outputs			
	$t_1$	$t_2$	$t_3$	$t_4$
$f_1$	1 1	1	0 1	1
$f_2$	1 0	0	0 1	1
$f_3$	1 0	0	1 0	1
$f_4$	1 0	0	1 0	0
$f_5$	1 0	0	1 1	0
$f_6$	1 0	0	1 1	0
$f_7$	1 1	0	1 0	1
$f_8$	0 1	1	0 1	1

$f_1, f_2, f_3, f_7, f_4, (f_5, f_6), f_8$   
 Response of  $z_1$       Response of  $z_2$   
 $f_1, f_2, (f_3, f_7), (f_4, f_5, f_6), f_8$

# Cause-Effect Analysis

Directly examine the syndrome of the failing chip to derive the fault candidates through Boolean reasoning on the circuit under diagnosis (CUD)

- Does not assume fault models -> can handle non-stuck-at faults
- Can be adapted to multiple fault cases

Disadvantages: takes longer because a unique round of analysis is required for each failure

Because logic analysis is used to guide physical failure analysis (pFA), which is very time-consuming (days or weeks), the analysis time is not important. Instead, diagnosis accuracy and resolution are much more important

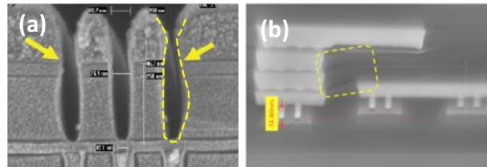
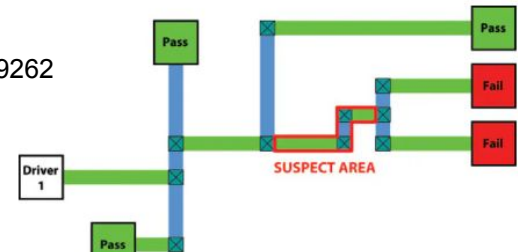


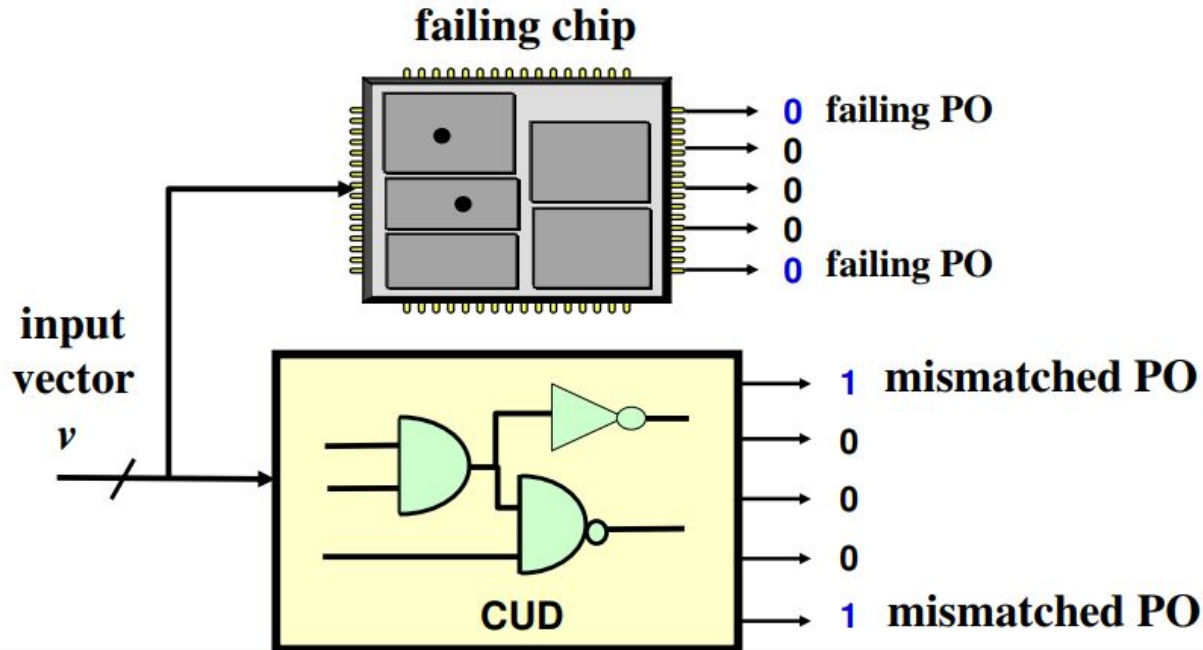
Figure 1 The cross-section SEM images of (a) the deformed post-etch via

DOI:10.1109/IPFA.2018.8452485Corpus ID: 52159262



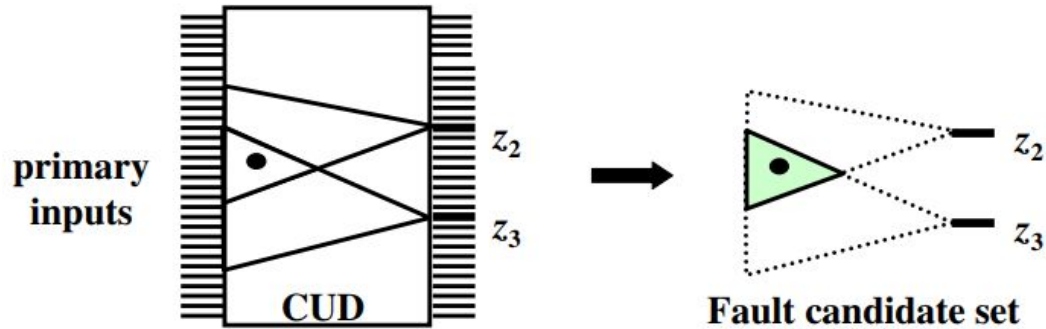
# Terminology: Mismatched Output

*Effect-cause analysis does not build fault dictionary  
It predicts fault locations by analyzing CUD from mismatch PO's*

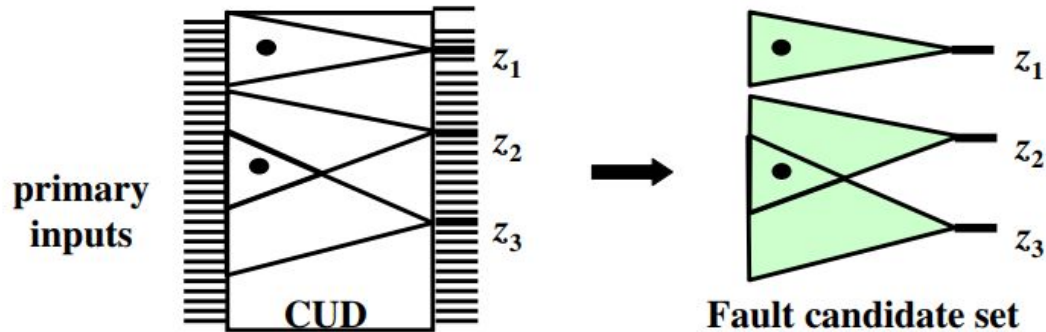




# Structural Pruning – Intersection or Union?



(a) Cone intersection.



(b) Cone union when there are multiple faults.

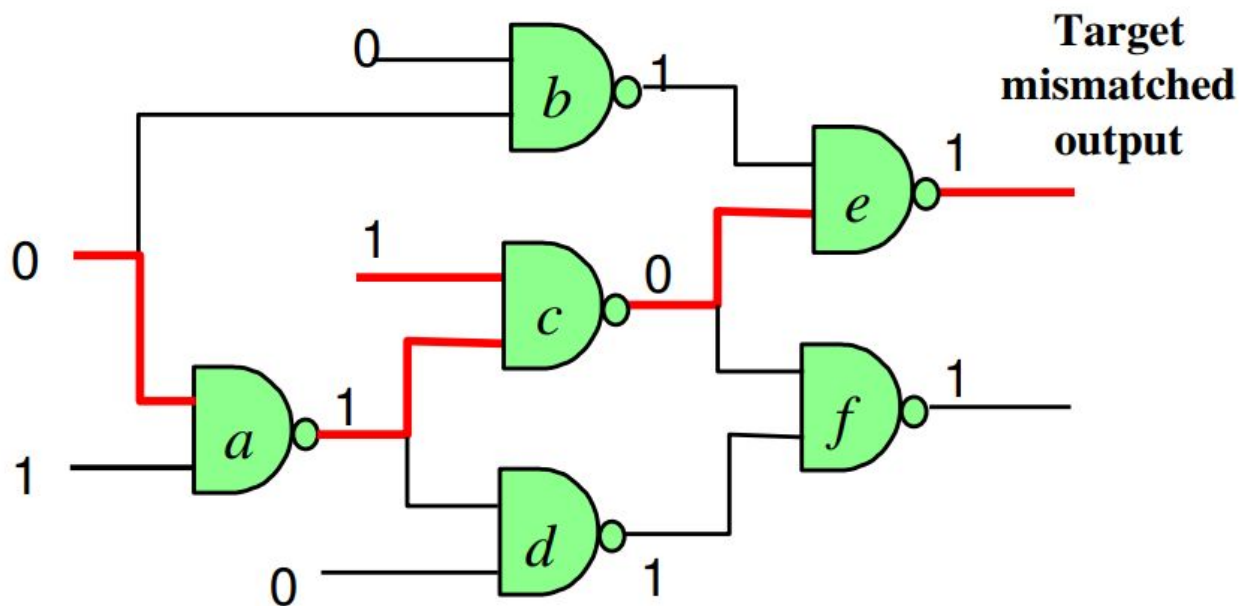


## ***Backtrace Algorithm***

- ❑ **Trace back from each mismatched PO**
  - To find out suspicious faulty locations
- ❑ **Functional Pruning**
  - During the traceback, some signals can be disqualified from the fault candidate set based on their signal values.
- ❑ **Rules**
  - (1) At a controlling case (i.e., 0 for a NAND gate): Its fanin signals with non-controlling values (i.e., 1) are excluded from the candidate set.
  - (2) At a non-controlling case (i.e., 1 for a NAND gate): Every fanin signal remains in the candidate set.

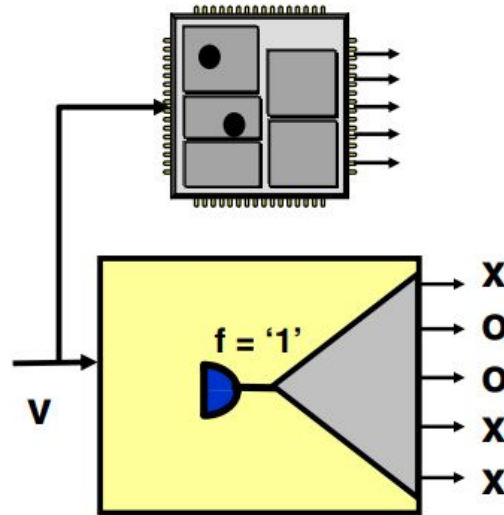
## Backtrace Example

*All suspicious fault locations are marked in red.*

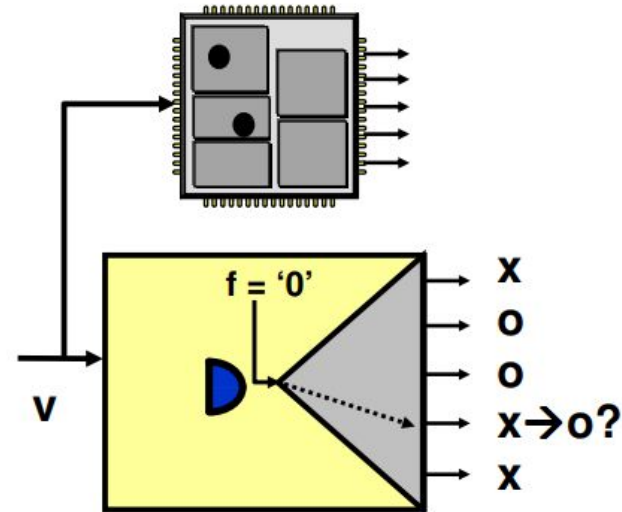


# Terminology – Injection

*An injection at a signal  $f$  flips its current value which could create value-change events downstream.*

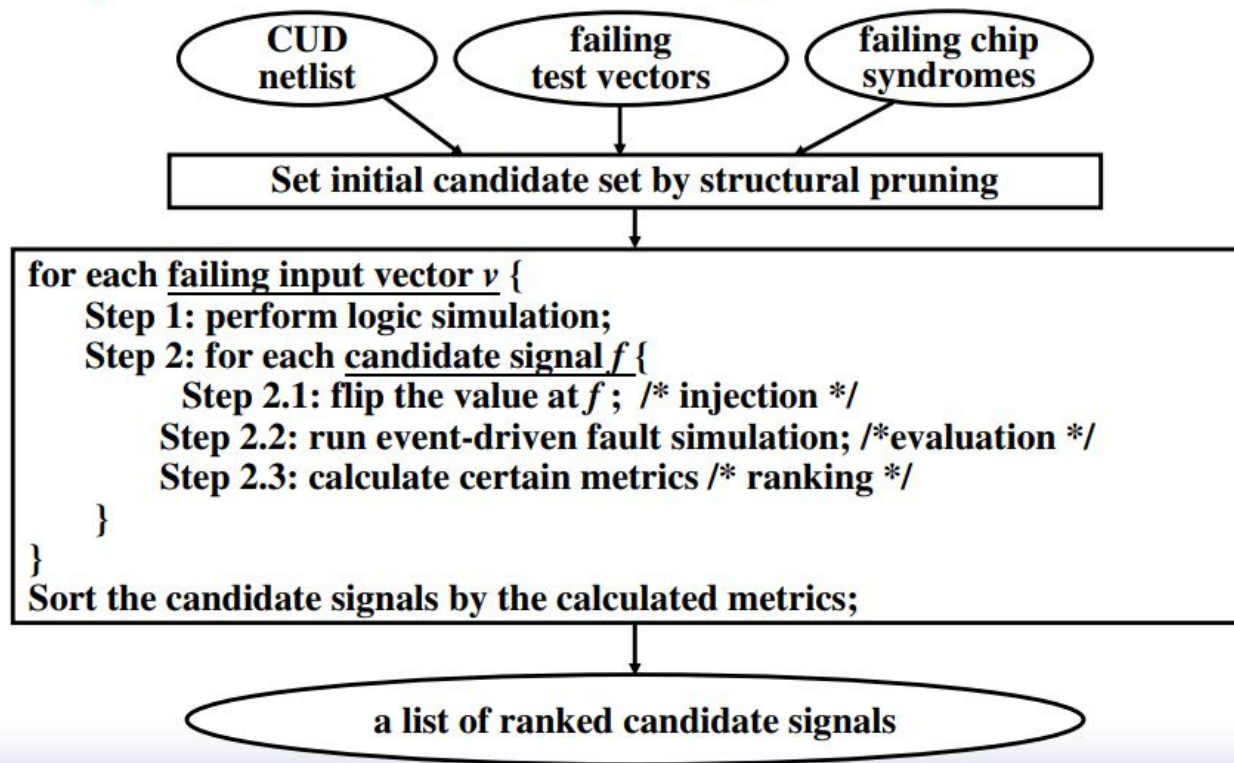


O: correct output  
X: failing output

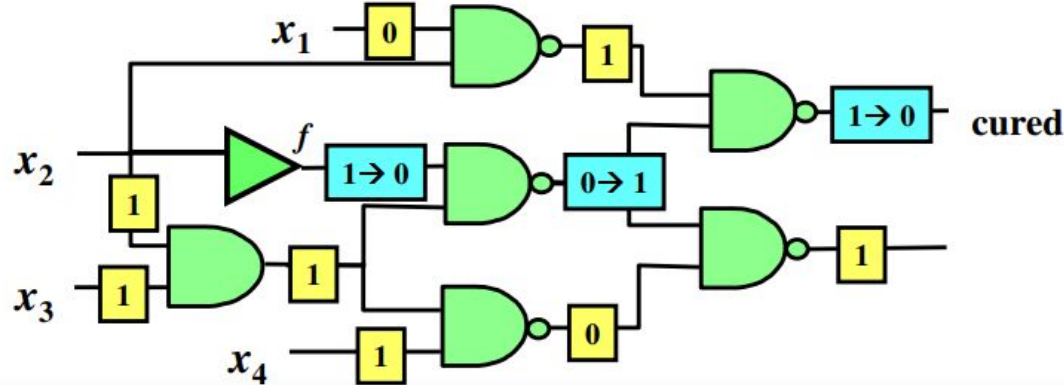
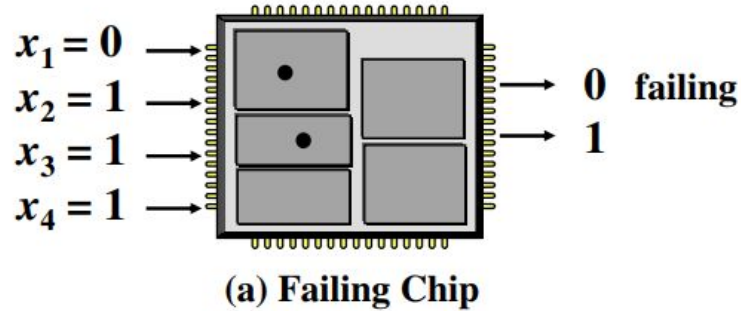


A mismatched output  
could be fixed by the injection!

## Detailed Computation – Inject-and-Evaluate Paradigm



## Example of Curable Vector



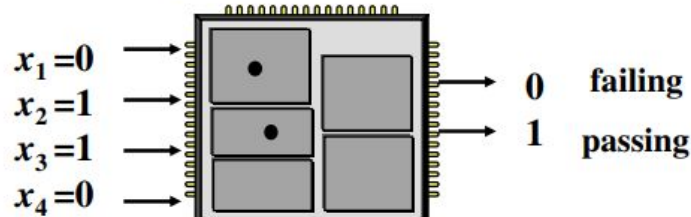
(b) Circuit Under Diagnosis

A mismatched output is called **curable output** of a signal  $f$ , if the mismatch can be resolved by an injection at  $f$ .  
A signal with more curable outputs is regarded as being more likely to be a fault location -> a **ranking metric**

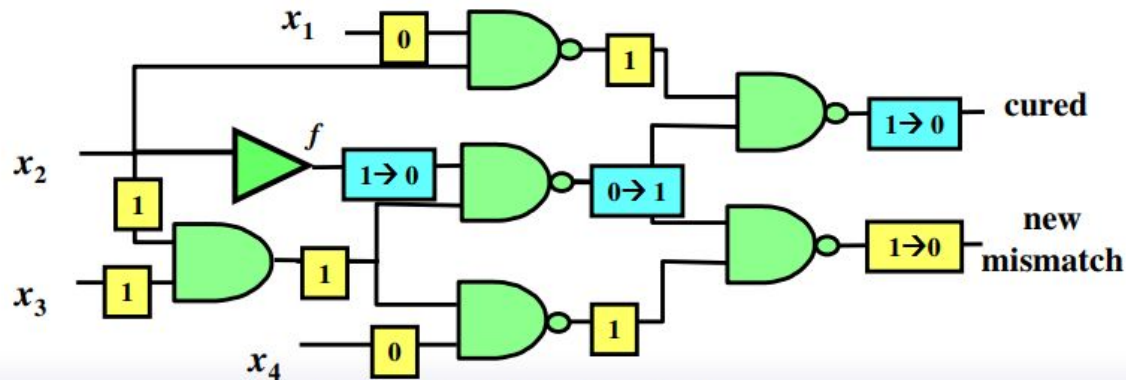
# Reward-and-Penalty Heuristic

*Rank1: curable vector count*

*Rank2 = (curable output count – 0.5 \* new mismatched output count)*



(a) Failing Chip.



(b) Circuit Under Diagnosis.



# Scan Chain Diagnosis

Scan chains are “almost” must-have design features for logic circuit testing and diagnosis

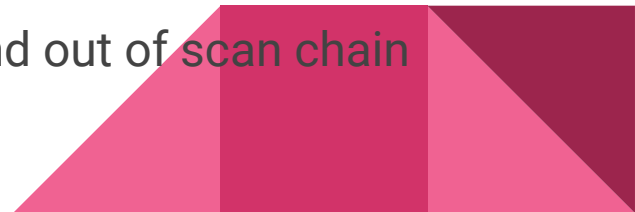
DFx: Design For **T**est, **D**ebug, **D**iagnosis, **M**anufacturing, etc

What if scan chains fail themselves

- Garbage in garbage out

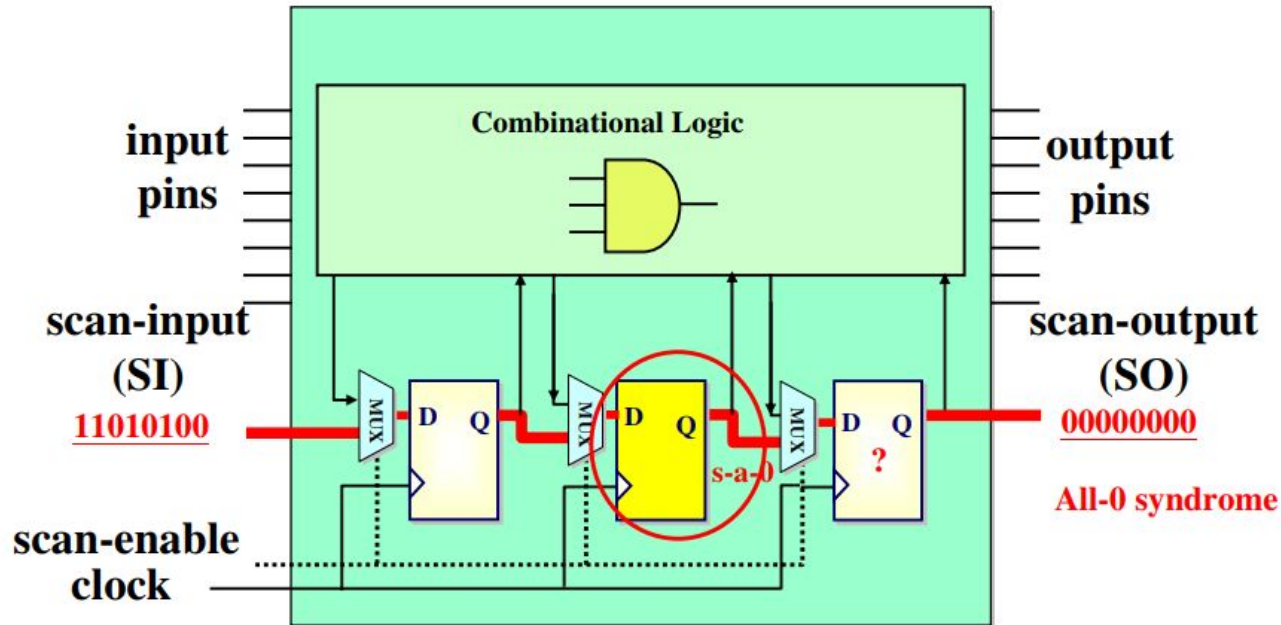
How to test the scan chain?

- Flush test: a set of random patterns are shifted in and out of scan chain



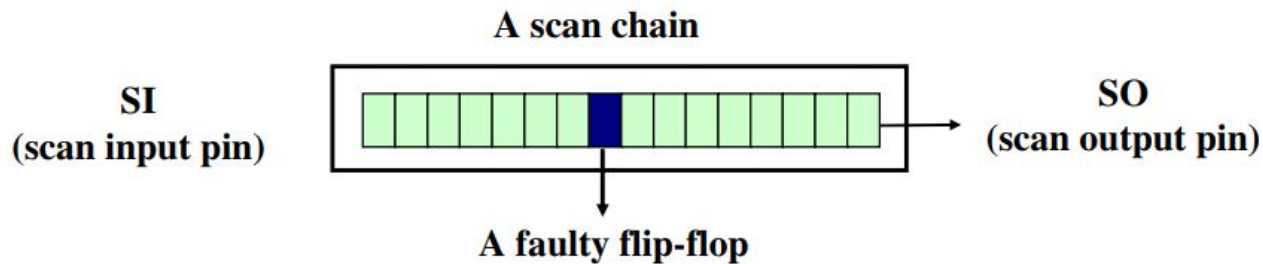
# A Stuck-At Fault In the Chain

Effect: A **killer** of the scan-test sequence





## Example: Faulty Syndrome of a Scan Chain



Fault Type	Scan-In Pattern	Observed Syndrome
Stuck-at-0	1100110011001100	<u>0000000000000000</u>
Stuck-at-1	1100110011001100	<u>1111111111111111</u>
Slow-to-Rise	1100110011001100	1 <u>000</u> 1 <u>000</u> 1 <u>000</u> 1 <u>000</u>
Slow-to-Fall	1100110011001100	1101110111011100

The rightmost bit goes into the scan first

The rightmost bit gets out of the scan first

A underlined bit in the observed image is failing.

# Scan Chain Diagnosis Method

## Hardware-based

- require extra hardware overhead, usually not acceptable in many products
- If defects occur in the extra control hardware, diagnosis becomes more complicated

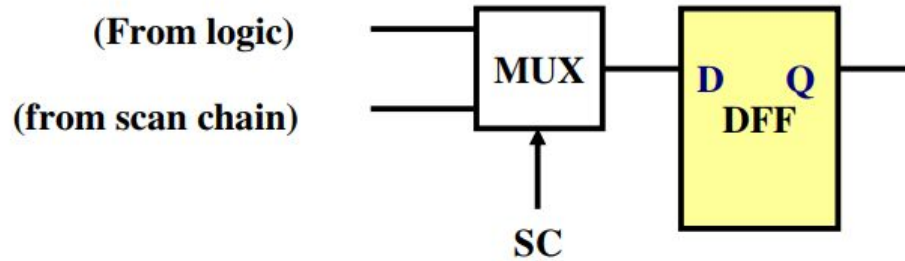
## Simulation-based

## Dictionary-based

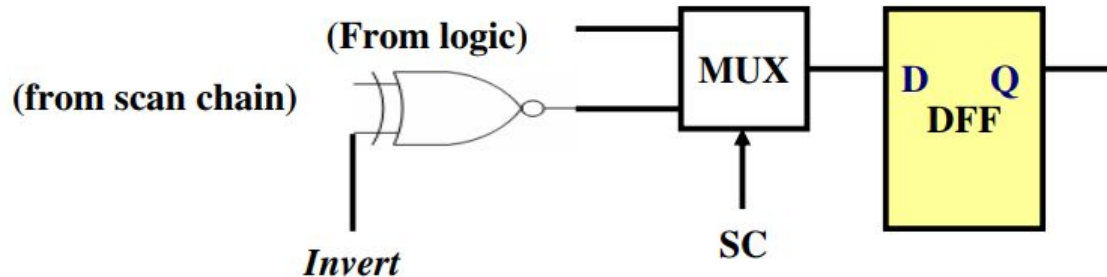
## Probability-based – Signal Profiling based



## Augmentation of a Flip-Flop for Easy Diagnosis



**(a) A normal scan flip-flop.**



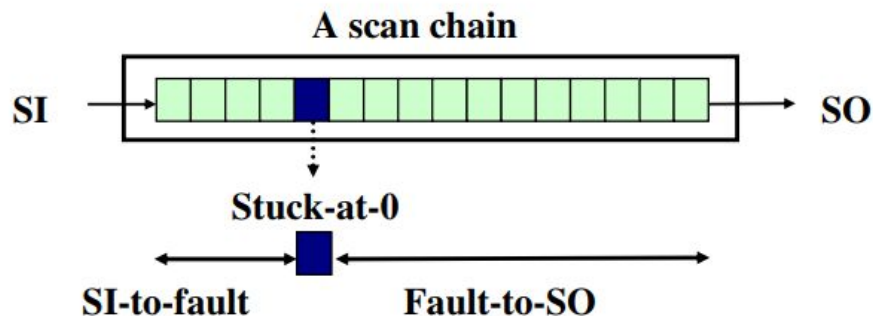
**(b) A modified scan flip-flop for easy inversion.**

# Diagnosis procedure

1. Scan in a flush pattern (e.g. all-1) into the scan chain
2. Invert the scan chain by setting signal invert to one
3. Shift out the scan chain values to see if there is a s@0 FF in the scan chain
4. Repeat step 1 to 3 by flushing all-0 pattern to see if there is a s@1 FF



## Fault Location via Inversion Operation



- (1) Original bitstream pattern = (111111111111111)
- (2) After scan-in: snapshot image = (11110000000000000)
- (3) After inversion: snapshot image = (0000011111111111)
- (4) After scan-out: observed image = (0000011111111111)

*The fault location is at the edge between 0's and 1's*

# Simulation-based Method

No extra overhead

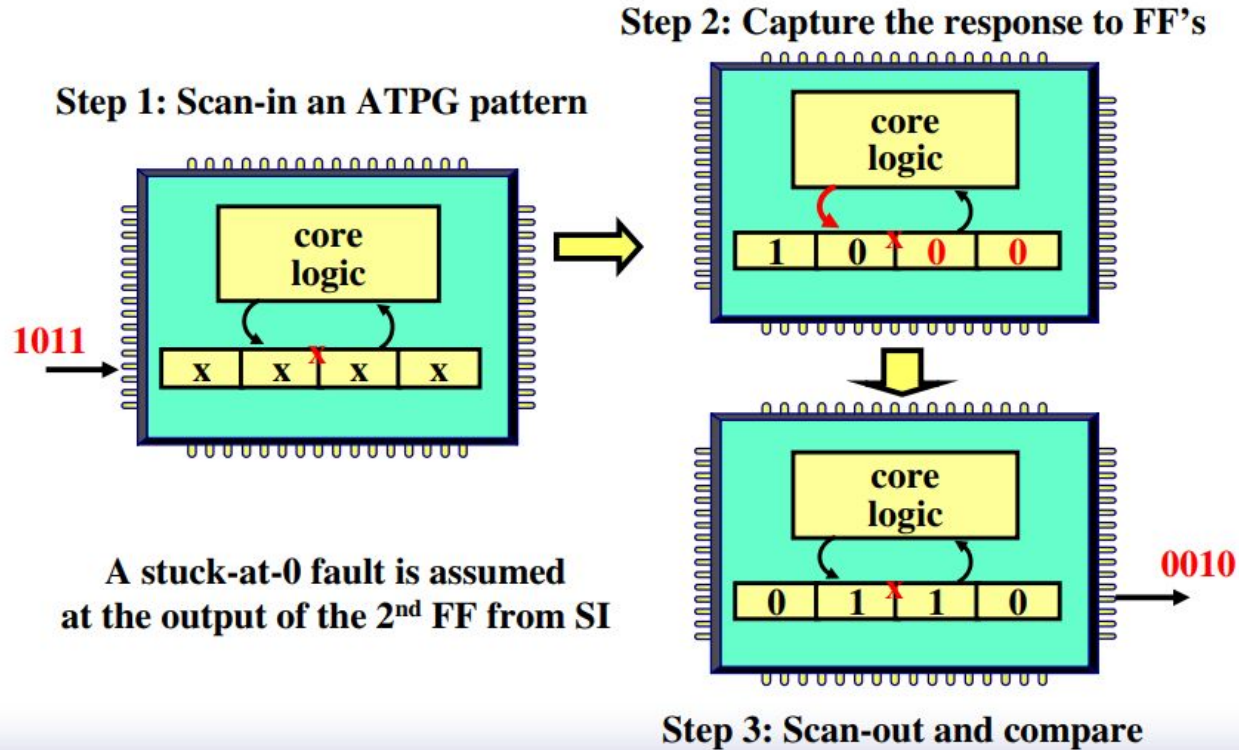
Flush test patterns, while ineffective for fault location, can be used for determine the fault type

Two Steps:

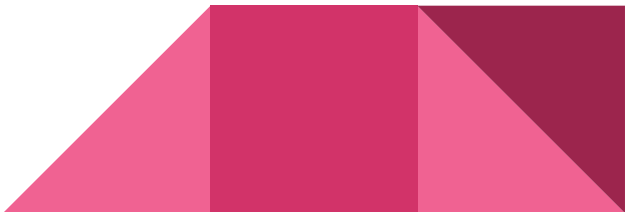
1. Use flush test patterns to identify faulty chains and determine the fault model for each faulty chain
2. Using scan patterns and simulation to locate the suspect faulty scan cells



# Modified Inject-and-Evaluate Paradigm



# Simulation Based Scan Chain Diagnosis

1. Run the flush test to guess the type of the faults
  2. Pick one scan vector and simulate the scan-capture-scan on the CUD to derive the fault-free responses
  3. Pick one possible fault candidate and Inject the fault effect into the scan chain
  4. Simulate the scan-capture-scan on the CUD to derive the failing responses
  5. Compare the failing response with the fault-free response. Accumulate the matching score for each fault candidate
  6. Go back to step 3 if there are more candidates
  7. Go back to step 2 if there are more scan vectors
  8. Rank candidates based on the matching scores
- 



# Test Sequence

Scan chain test->Scan ATPG test-> Transition ATPG test

Stop on first fail or Force Flow

Fail data log collection -> running diagnosis tool automatically

It seems straightforward. What are the challenges for modern ICs?

- The accuracy and resolution of diagnosis tools -> enable faster defect localization. HOW?
  - The number of failing cycle captured in the log
  - Too many candidates
  - Model size vs Simulation time
  - The goal is to identify the root cause of yield loss-> how to separating devices with systemic issues from those with random defects ?
  - Scan chain compression -> One-hot or Bypassed patterns
  - Layout Aware diagnosis: effective for bridge and void
- Yield Loss and Improvement