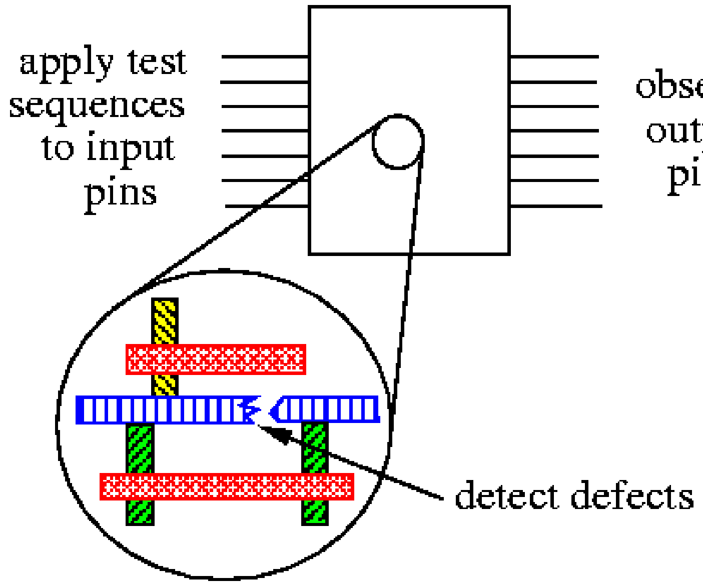
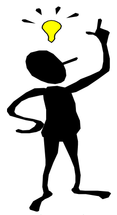
Unit 11: Testing

- Course contents
  - Fault Modeling
  - Fault Simulation
  - Test Generation
  - Design For Testability

One good reference: "VLSI Test Principles and Architectures," by Wang, Wu, and Wen

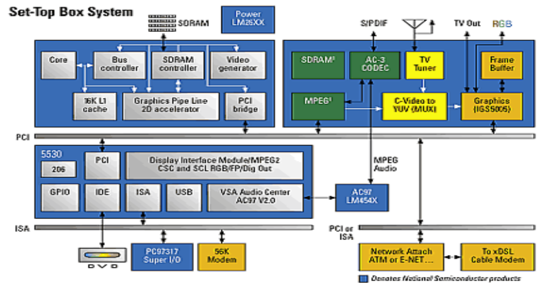


Chip Design & Manufacturing Flow



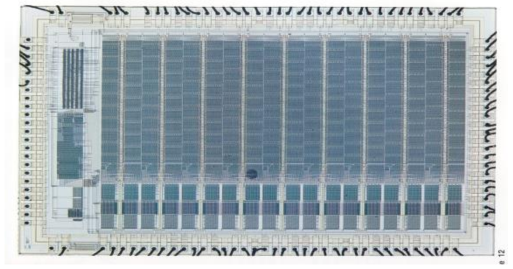
Idea

Architecture Design



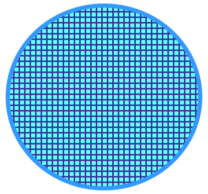
Block diagram

Circuit & Layout Design



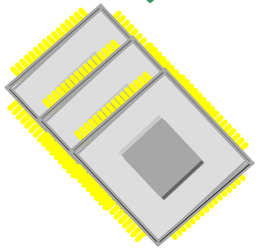
Layout

IC Fabrication



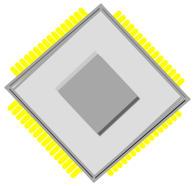
Wafer (hundreds of dies)

Sawing & Packaging

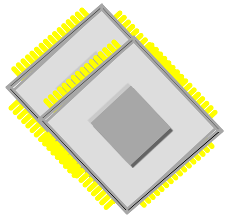


Final chips

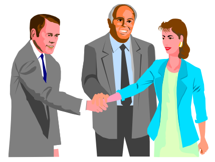
Final Testing



Bad chips



Good chips



customers

Design Verification, Testing and Diagnosis

- Design Verification:
  - Ascertain the design perform its specified behavior
- Testing:
  - Exercise the system and analyze the response to ascertain whether it behaves correctly after manufacturing
- Diagnosis:
  - To locate the cause(s) of misbehavior after the incorrect behavior is detected

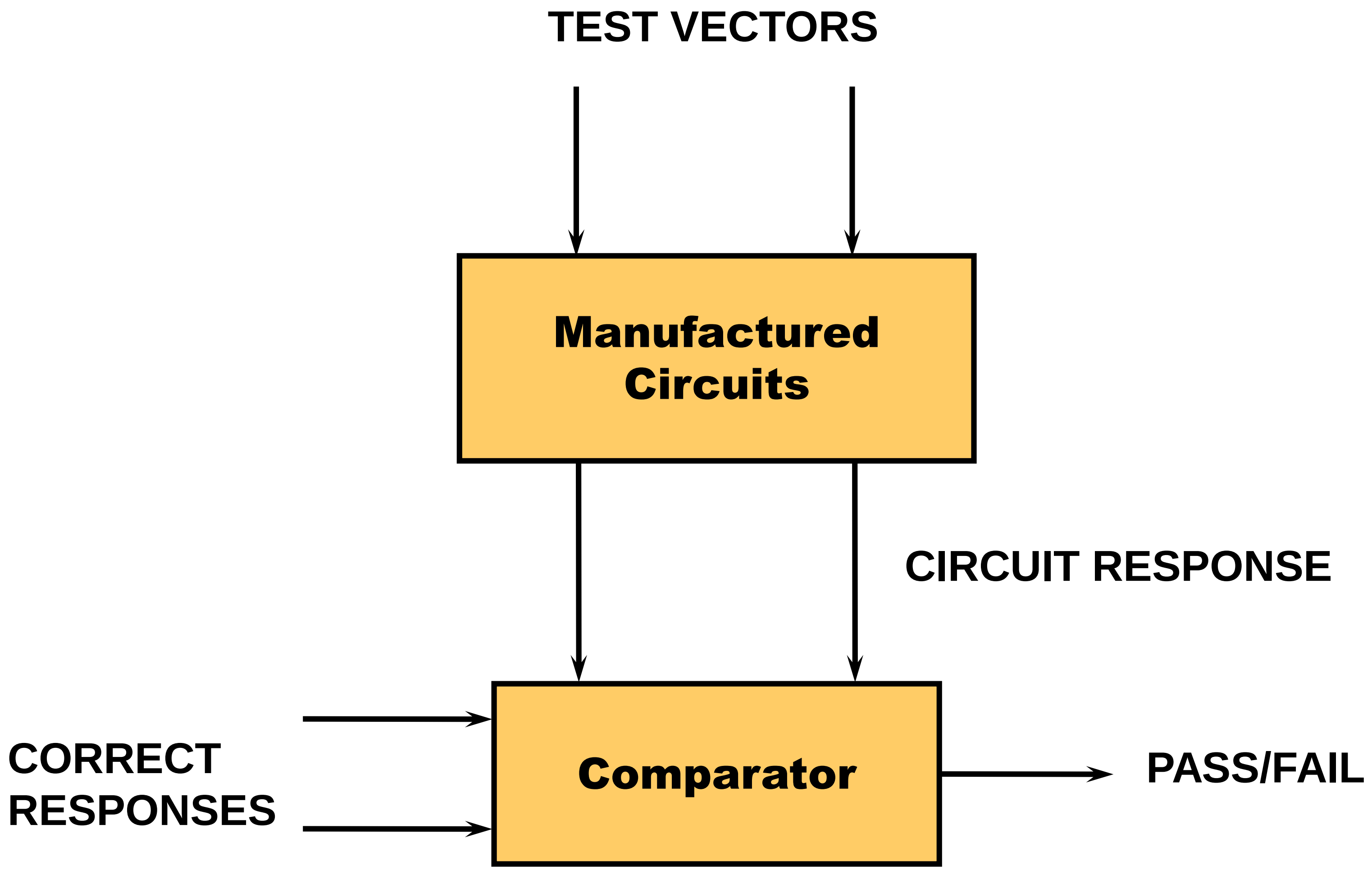
Manufacturing Defects

- Process Defects
  - missing contact windows
  - parasitic transistors
  - oxide breakdown
- Material Defects
  - bulk defects (cracks, crystal imperfections)
  - surface impurities
- Time-Dependent Failures
  - dielectric breakdown
  - electro-migration
- Packaging Failures
  - contact degradation
  - seal leaks

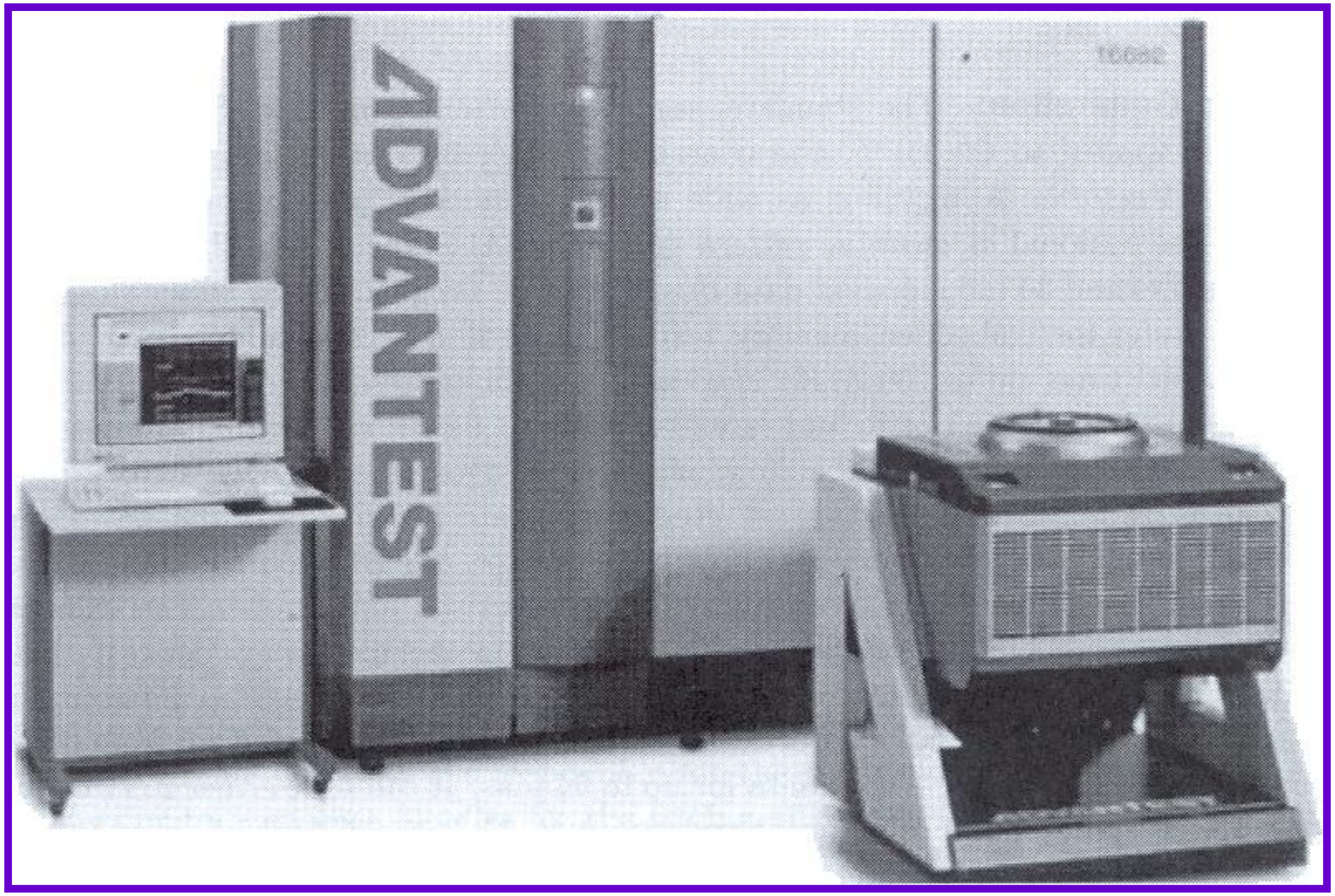
Faults, Errors and Failures

- Fault:
  - Faulty behavior caused by a physical defect in a circuit or a system
  - May or may not cause a system failure
- Error:
  - Manifestation of a fault that results in incorrect circuit (system) outputs or states
  - Caused by faults
- Failure:
  - Deviation of a circuit or system from its specified behavior
  - Fails to do what it should do
  - Caused by an error
- Fault ---> Error ---> Failure

Scenario of Manufacturing Test

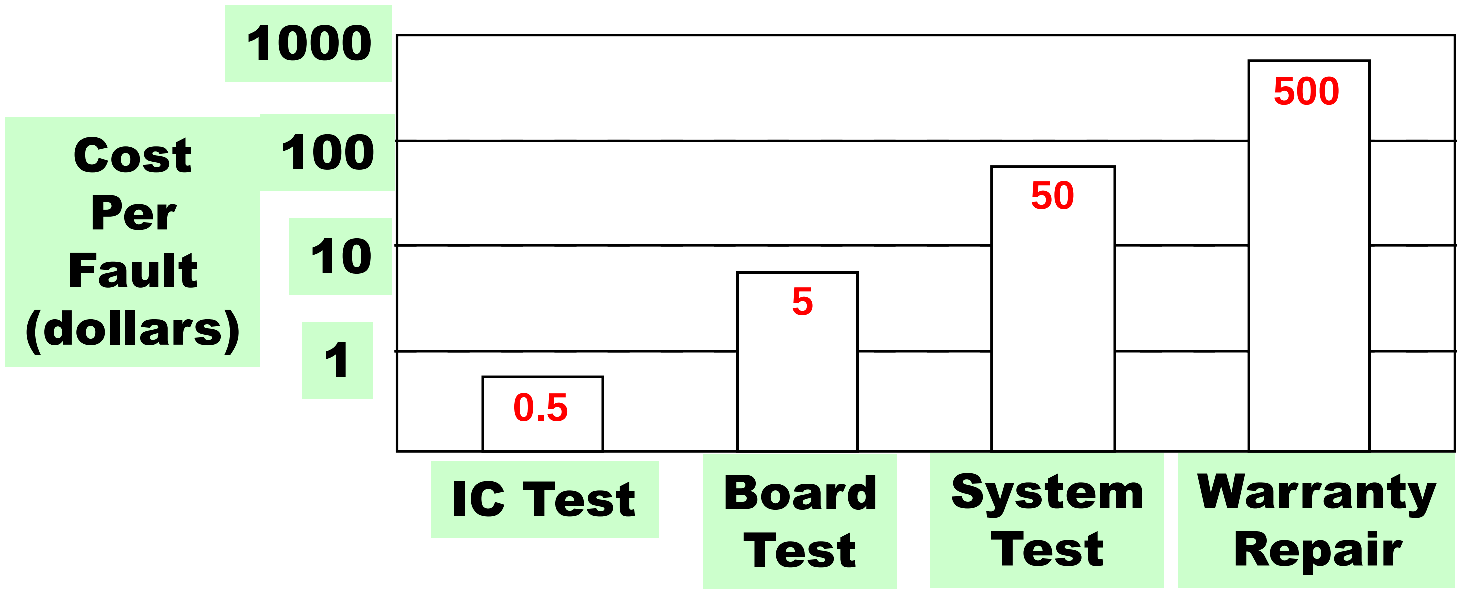


Tester: Advantest T6682



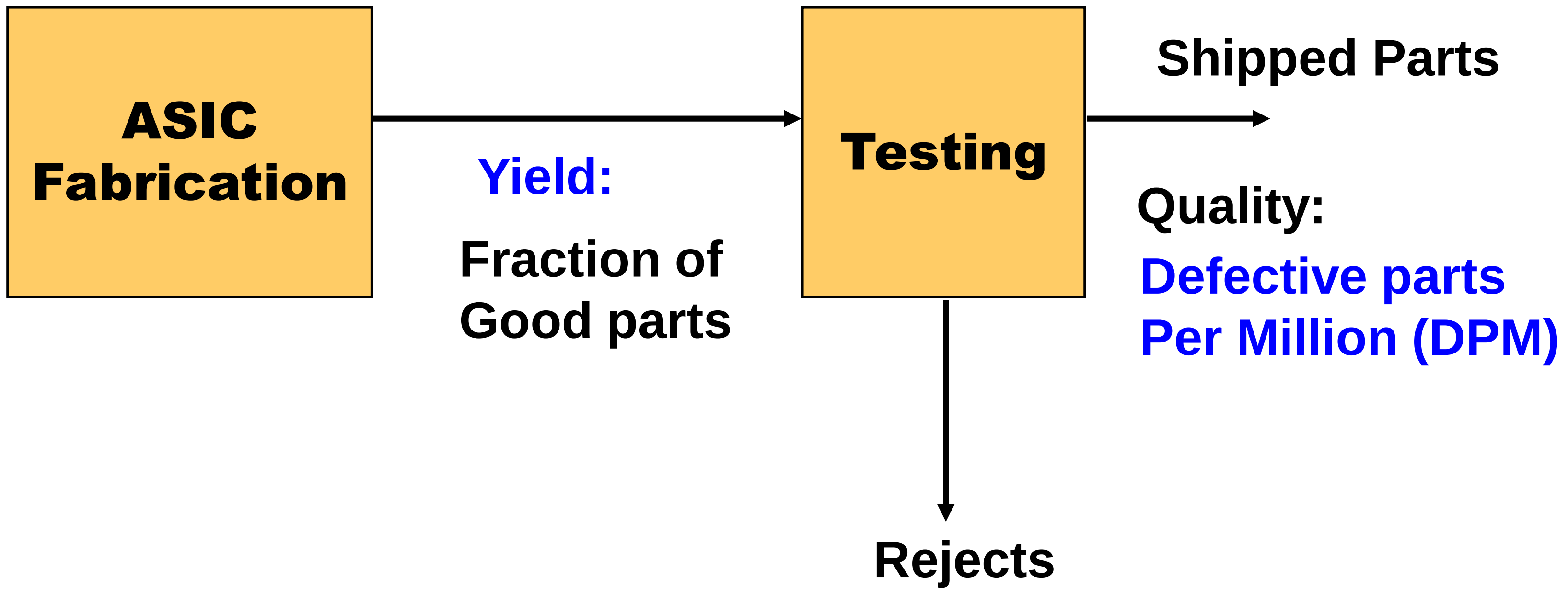
Purpose of Testing

- Verify manufacturing of circuit
  - Improve system reliability
  - Diminish system cost
- Cost of repair
  - Goes up by an order of magnitude each step away from the fab. line



B. Davis, "The Economics of Automatic Testing" McGraw-Hill 1982

Testing and Quality



Quality of shipped part is a function of yield Y and the test (fault) coverage T.

Fault Coverage

- Fault Coverage T
  - Is the measure of the ability of a set of tests to detect a given class of faults that may occur on the device under test (DUT)

$$T = \frac{\text{No. of detected faults}}{\text{No. of all possible faults}}$$

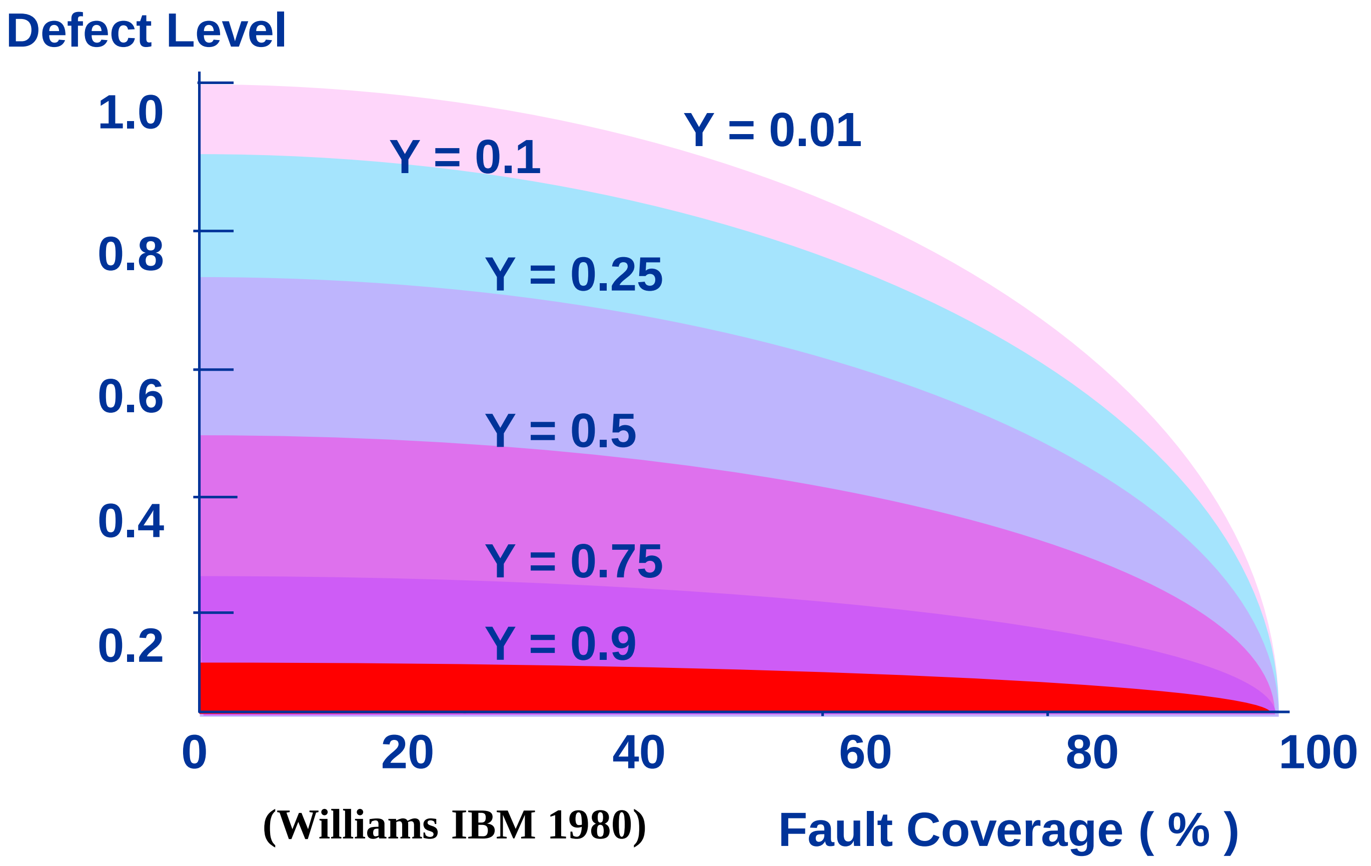
Defect Level

- Defect Level (Reject Rate)
  - Is the fraction of the shipped parts that are defective (the fraction of faulty chips among the chips that pass the test)

$$\boxed{DL = 1 - Y^{(1-T)}}$$

Y: yield
T: fault coverage

Defect Level v.s. Fault Coverage



$\boxed{\text{High fault coverage} \longrightarrow \text{Low defect level}}$

DPM v.s. Yield and Coverage

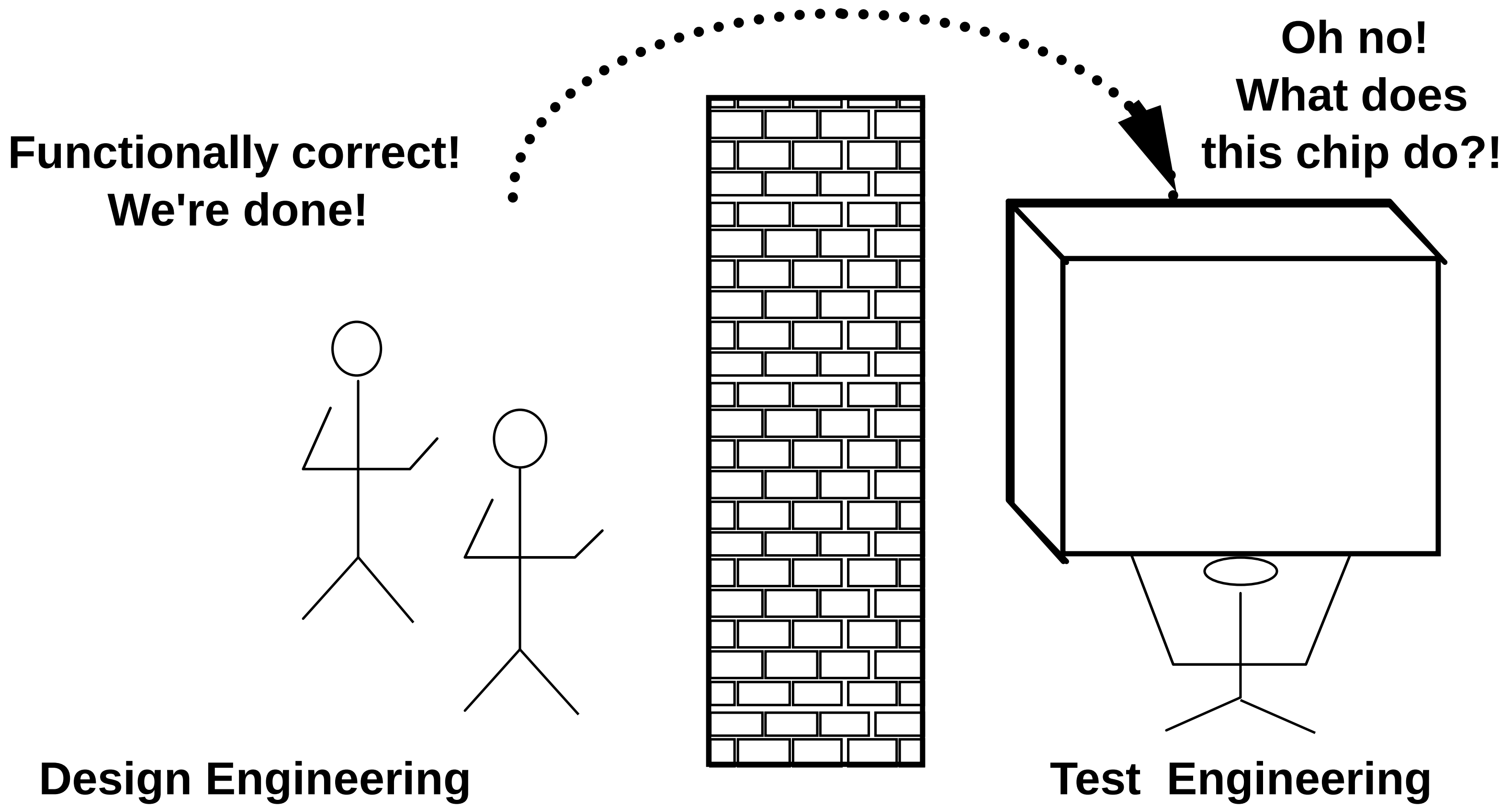
Yield	Fault Coverage	DPM (ppm)
50%	90%	67,000
75%	90%	28,000
90%	90%	10,000
95%	90%	5,000
99%	90%	1,000
90%	90%	10,000
90%	95%	5,000
90%	99%	1,000
90%	99.9%	100

Why Testing Is Difficult ?

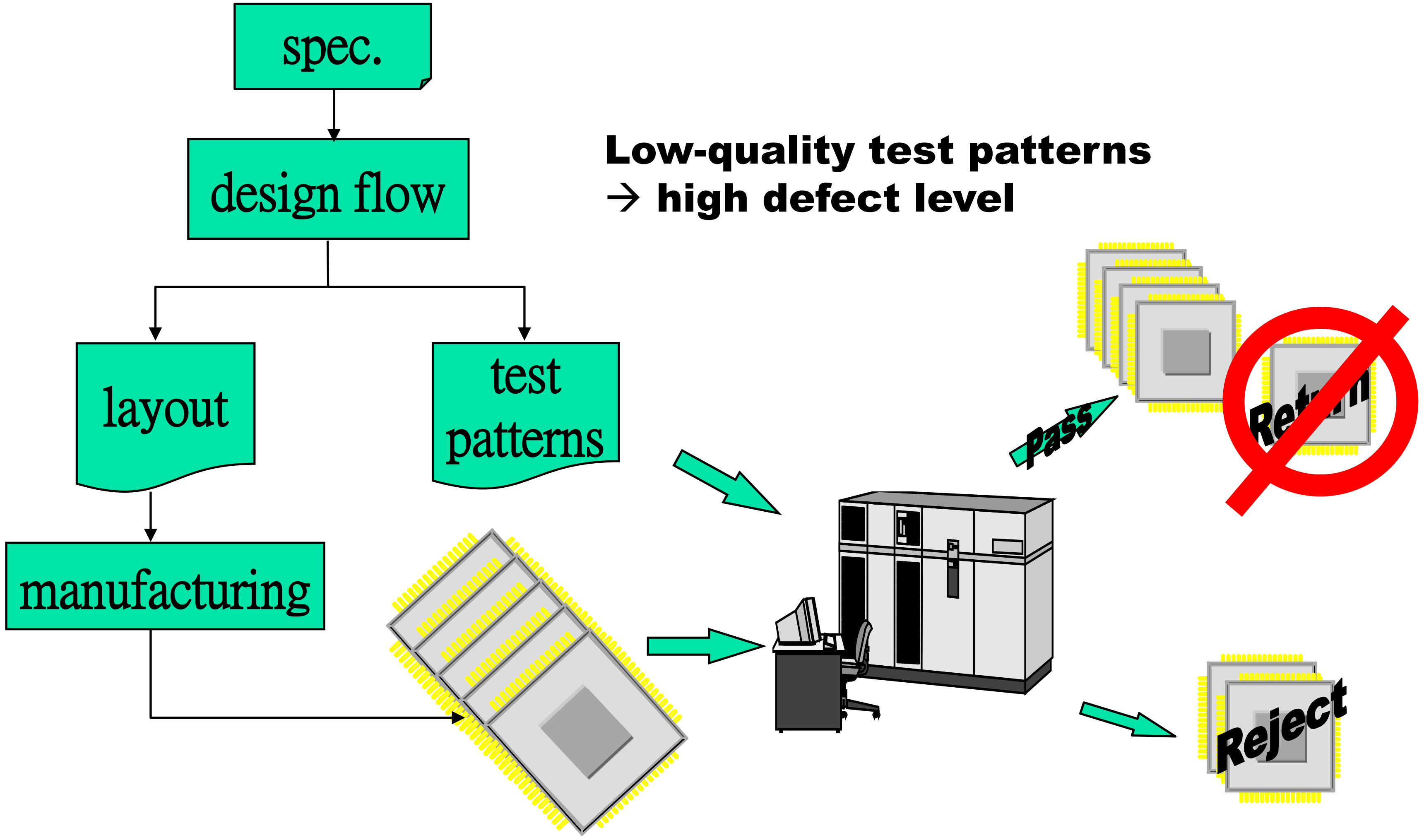
- Test application time can be exploded for exhaustive testing of VLSI
  - For a combinational circuit with 50 inputs, we need $2^{50} = 1.126\mathbf{x}10^{15}$ test patterns.
  - Assume one test per 10^{-7}sec, it takes $1.125\mathbf{x}10^{8}$sec = 3.57yrs. to test such a circuit.
  - Test generation for sequential circuits are even more difficult due to the lack of controllability and observability at flip-flops (latches)
- Functional testing may NOT be able to detect the physical faults

The Infamous Design/Test Wall

30 years of experience proves that test after design does not work!

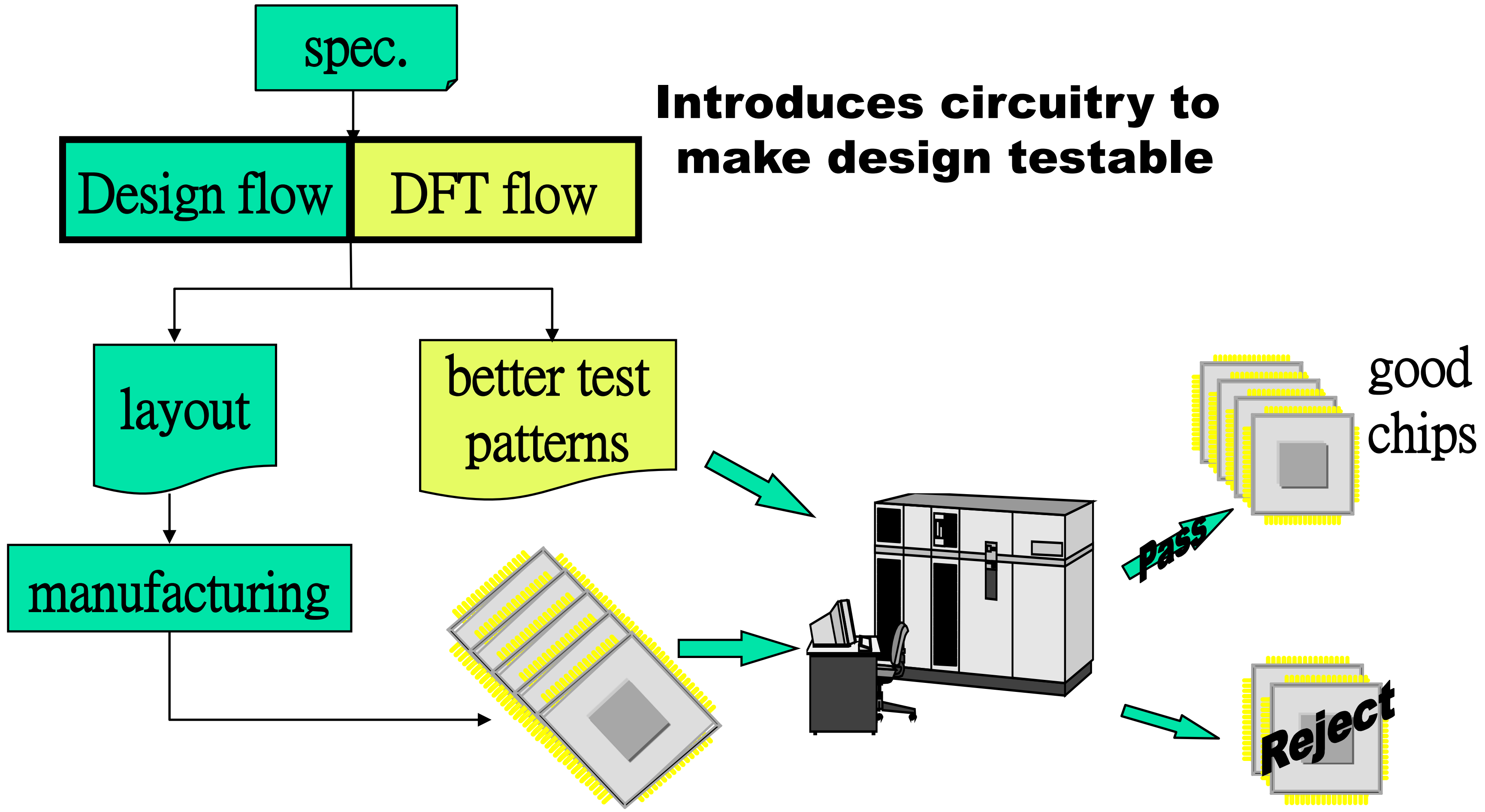

Functionally correct! We're done!
Oh no! What does this chip do?!
Design Engineering
Test Engineering

Old Design & Test Flow



Low-quality test patterns
→ high defect level

New Design and Test Flow



Why Fault Model?

- Fault model identifies target faults
  - Model faults most likely to occur
- Fault model limits the scope of test generation
  - Create tests only for the modeled faults
- Fault model makes effectiveness measurable by experiments
  - Fault coverage can be computed for specific test patterns to reflect its effectiveness
- Fault model makes analysis possible
  - Associate specific defects with specific test patterns

Fault Modeling

- Fault Modeling
  - Model the effects of physical defects on the logic function and timing
- Physical Defects
  - Silicon Defects
  - Photolithographic Defects
  - Mask Contamination
  - Process Variation
  - Defective Oxides

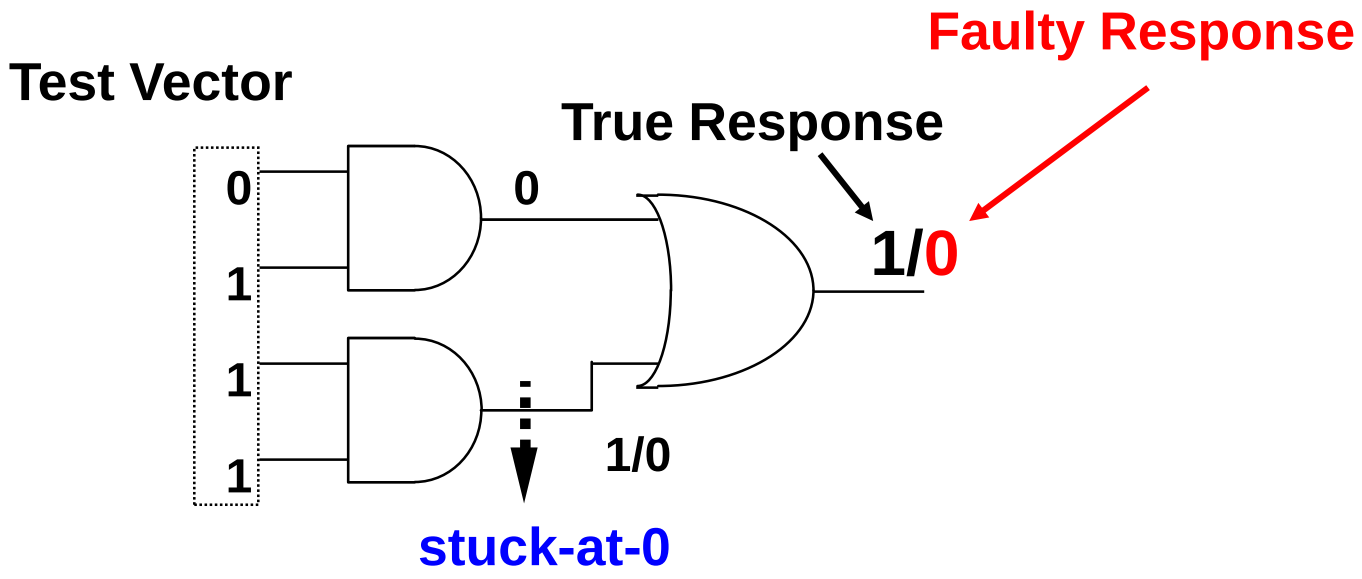
Fault Modeling (cont'd)

- Electrical Effects
  - Shorts (Bridging Faults)
  - Opens
  - Transistor Stuck-On/Open
  - Resistive Shorts/Opens
  - Change in Threshold Voltages
- **Logical Effects**
  - Logical Stuck-at 0/1
  - Slower Transition (Delay Faults)
  - AND-bridging, OR-bridging

Fault Types Commonly Used To Guide Test Generation

- Stuck-at Faults
- Bridging Faults
- Transistor Stuck-On/Open Faults
- Delay Faults
- IDDQ Faults
- State Transition Faults (for FSM)
- Memory Faults
- PLA Faults

Single Stuck-At Fault



Assumptions:

- **Only one line is faulty**
- **Faulty line permanently set to 0 or 1**
- **Fault can be at an input or output of a gate**

Multiple Stuck-At Faults

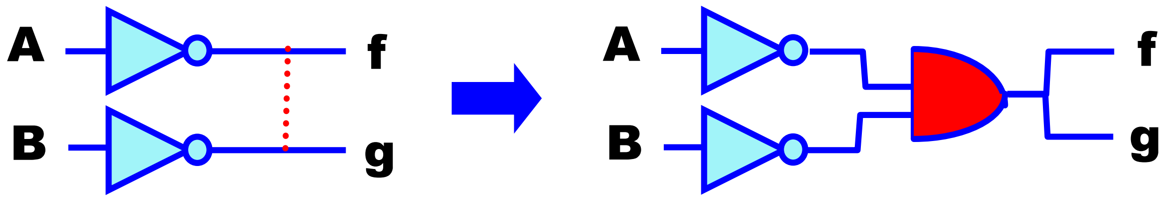
- Several stuck-at faults occur at the same time
  - Important in high density circuits
- For a circuit with k lines
  - there are 2k single stuck-at faults
  - there are 3^k-1 multiple stuck-at faults
    - A line could be stuck-at-0, stuck-at-1, or fault-free
    - One out of 3^k resulting circuits is fault-free

Why Single Stuck-At Fault Model

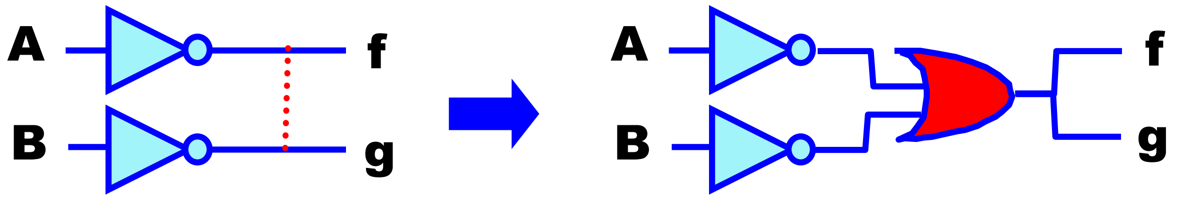
- Complexity is greatly reduced
  - Many different physical defects may be modeled by the same logical single stuck-at fault
- Stuck-at fault is technology independent
  - Can be applied to TTL, ECL, CMOS, BiCMOS etc.
- Design style independent
  - Gate array, standard cell, custom VLSI
- Detection capability of un-modeled defects
  - Empirically many defects accidentally detected by test derived based on single stuck-at fault
- Cover a large percentage of multiple stuck-at faults

Bridging Faults

- Two or more normally distinct points (lines) are shorted together
  - Logic effect depends on technology
  - Wired-AND for TTL



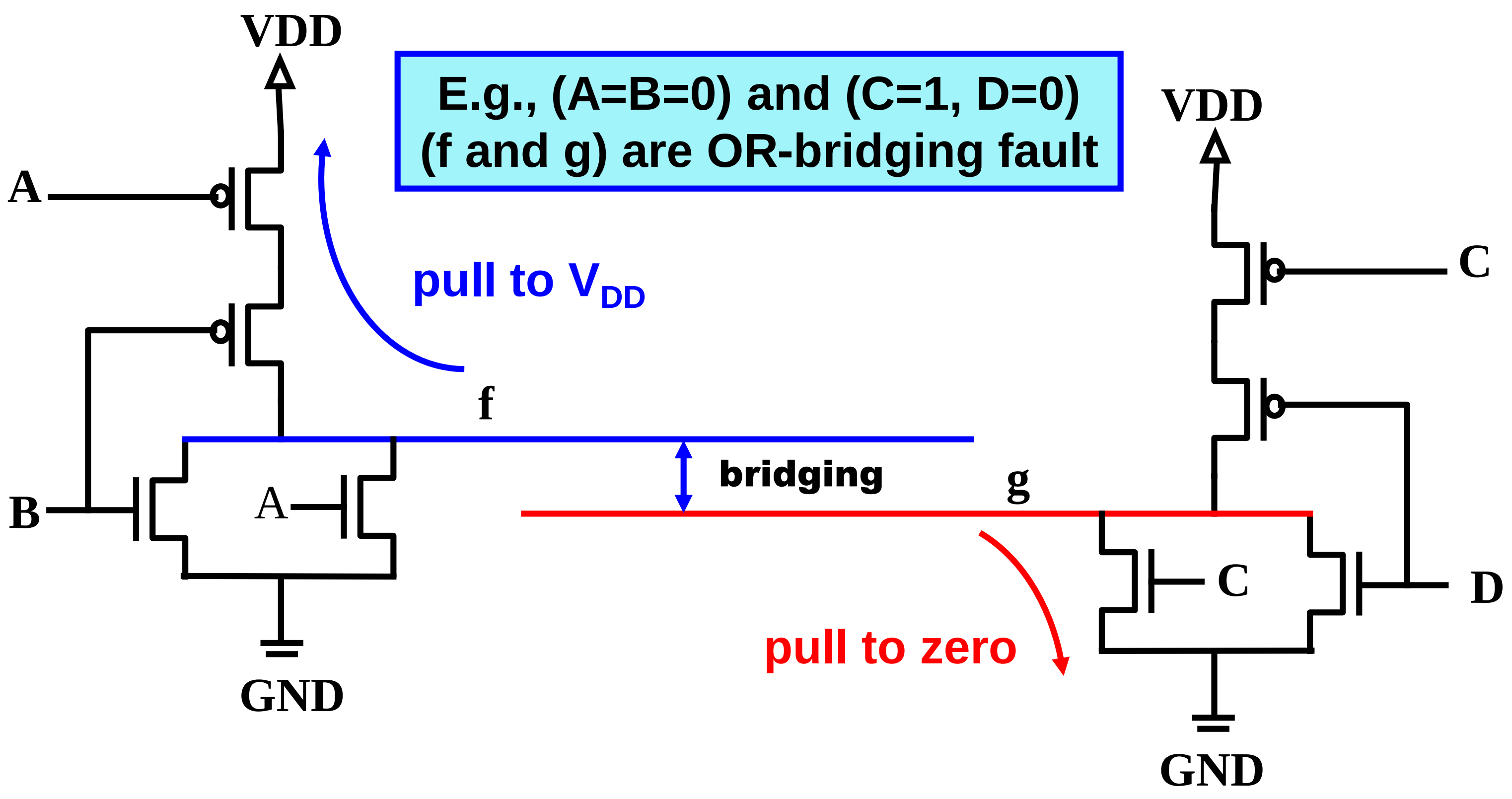
  - Wired-OR for ECL



  - CMOS ?

Bridging Faults for CMOS Logic

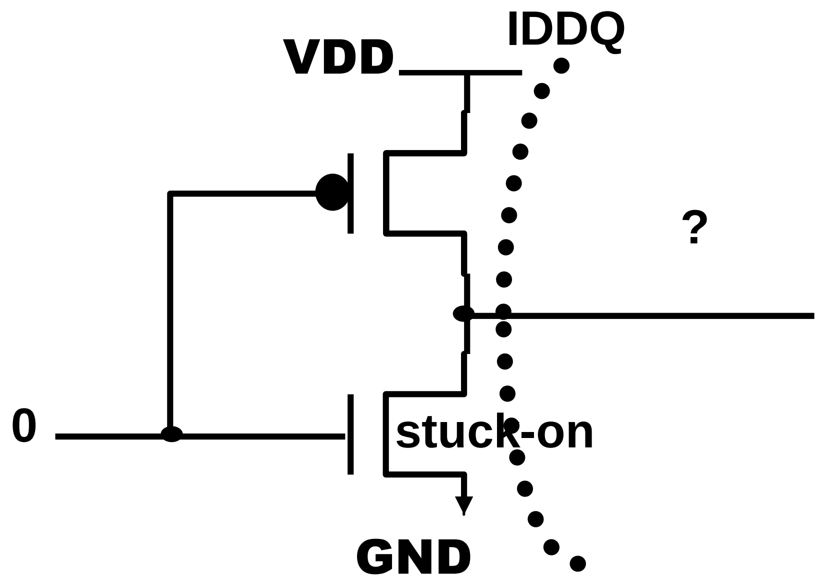
- The result
  - could be AND-bridging or OR-bridging
  - depends on the the inputs



CMOS Transistor Stuck-On (Stuck-Short)

Example:

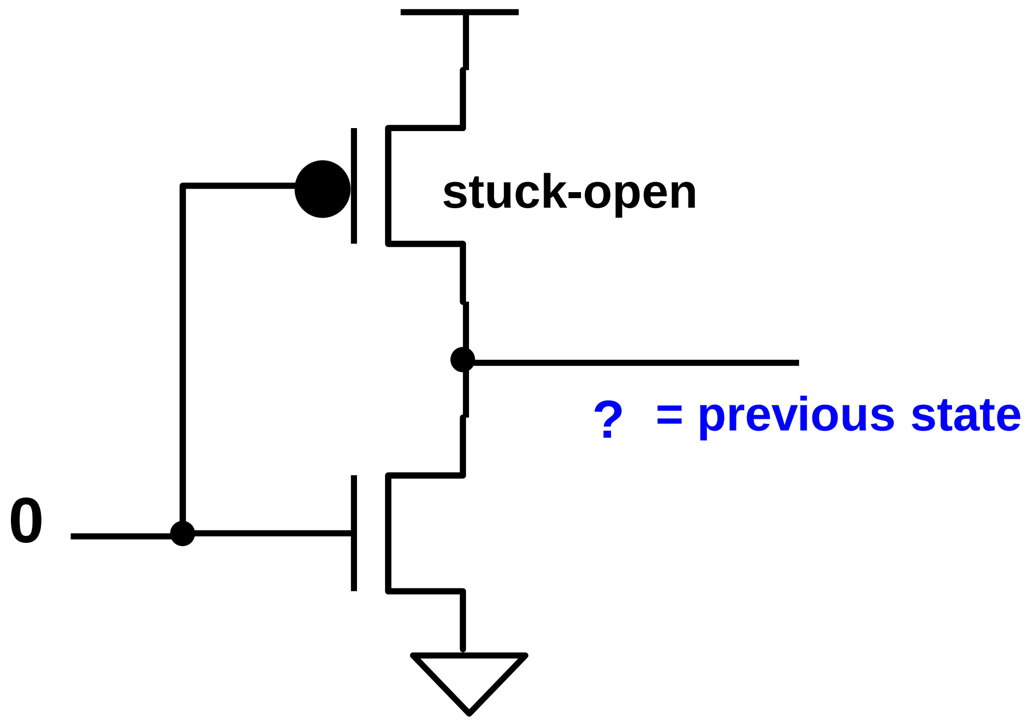
N transistor is always ON



- Transistor Stuck-On
  - May cause ambiguous logic level
  - Depends on the relative impedances of the pull-up and pull-down networks
- When input is low
  - Both P and N transistors are conducting, causing increased quiescent current, called IDDQ fault

CMOS Transistor Stuck-Open (I)

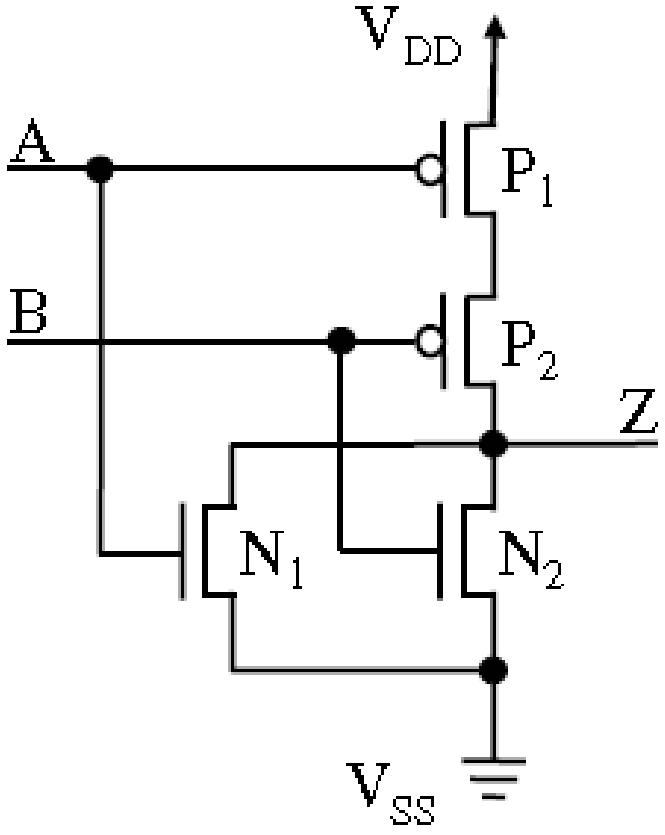
- Transistor stuck-open
  - May cause the output to be floating



CMOS Transistor Stuck-Open (II)

- Stuck-open requires sequence of two vector tests for detection
  - 00 -> 10 detects N_1 stuck-open

2-input CMOS NOR gate



Truth table for fault-free circuit and all possible transistor faults

AB	00	01	10	11
Z	1	0	0	0
N_1 stuck-open	1	0	last Z	0
N_1 stuck-short	I_{DDQ}	0	0	0
N_2 stuck-open	1	last Z	0	0
N_2 stuck-short	I_{DDQ}	0	0	0
P_1 stuck-open	last Z	0	0	0
P_1 stuck-short	1	0	I_{DDQ}	0
P_2 stuck-open	last Z	0	0	0
P_2 stuck-short	1	I_{DDQ}	0	0

Source: VLSI Test Principles and Architectures

Memory Faults

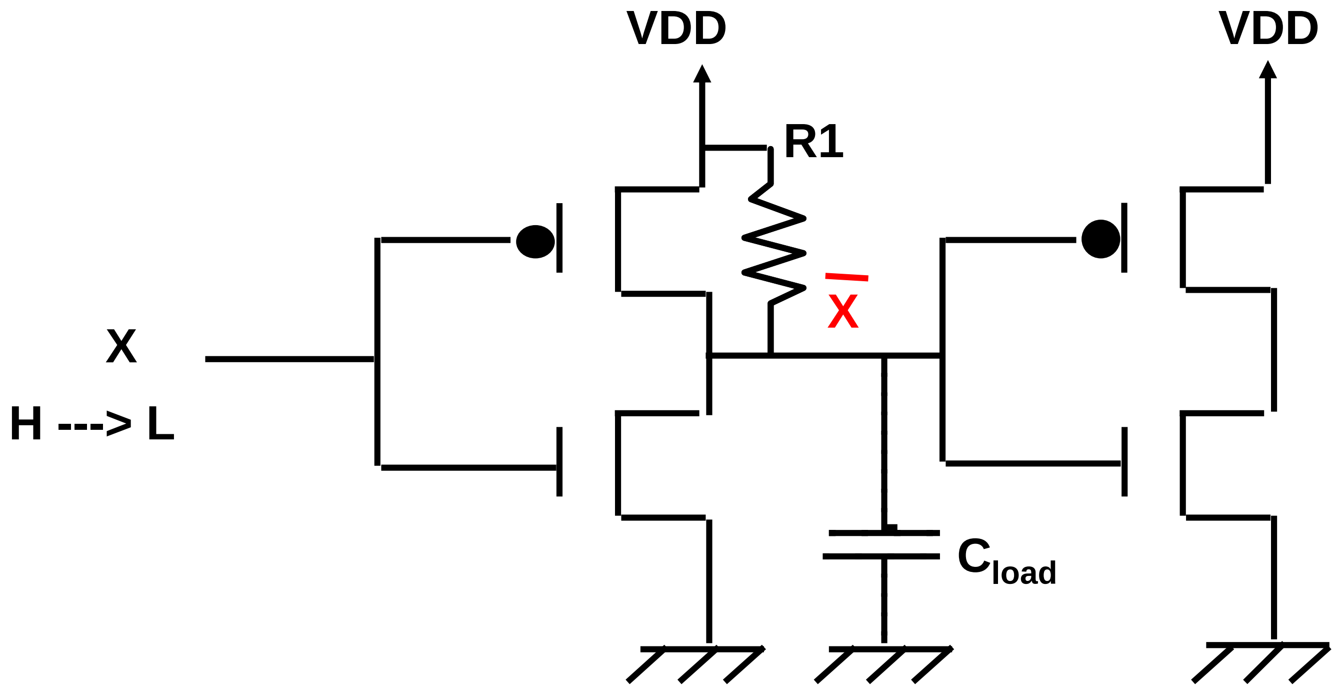
- Parametric Faults
  - Output Levels
  - Power Consumption
  - Noise Margin
  - Data Retention Time
- Functional Faults
  - Stuck Faults in Address Register, Data Register, and Address Decoder
  - Cell Stuck Faults
  - Adjacent Cell Coupling Faults
  - Pattern-Sensitive Faults

Delay Testing

- Chip with timing defects
  - may pass the DC stuck-fault testing, but fail when operated at the system speed
  - For example, a chip may pass testing under 10 MHz operation, but fail under 100 MHz
- Delay Fault Models
  - Gate-Delay Fault
  - Path-Delay Fault

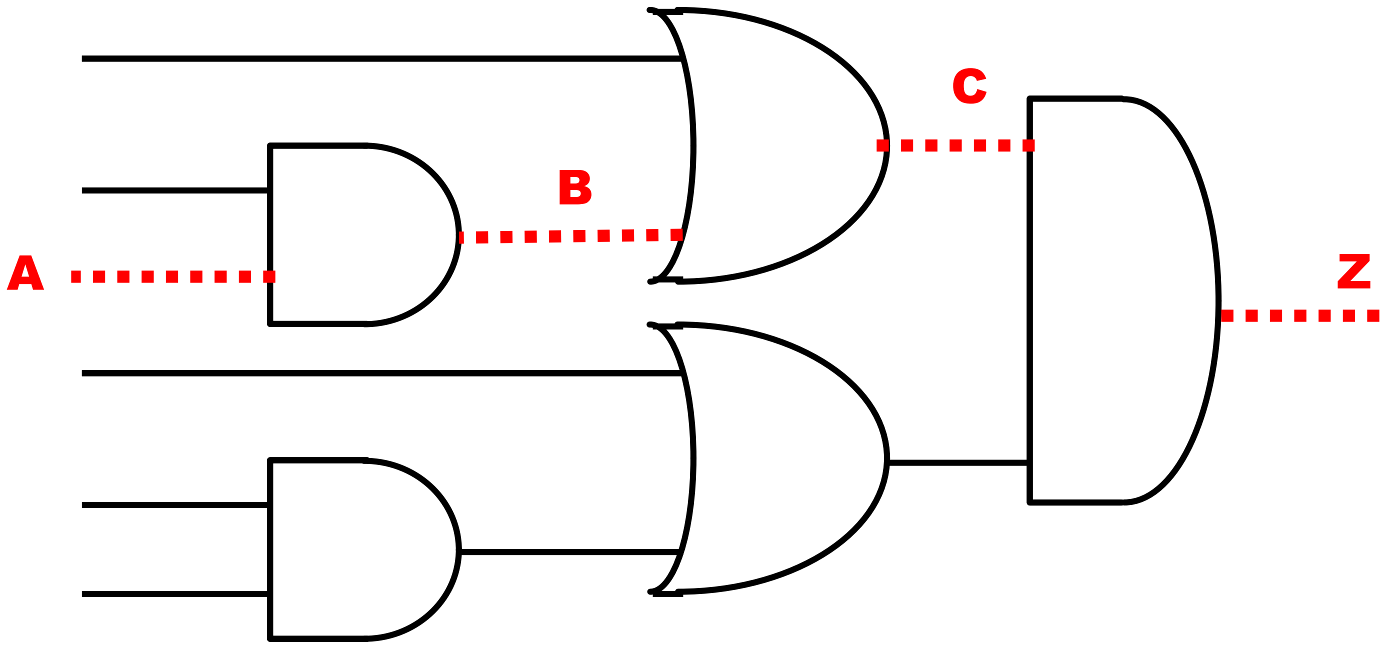
Gate-Delay Fault (Transition Fault)

- Slow to rise, slow to fall
  - $\overline{x}$ is slow to rise when channel resistance R1 is abnormally high



Path-Delay Fault

- Associated with a path (e.g. A-B-C-Z)
  - Whose delay exceeds the clock interval
- More complicated (and important) than gate-delay fault
  - Because the number of paths grows exponentially

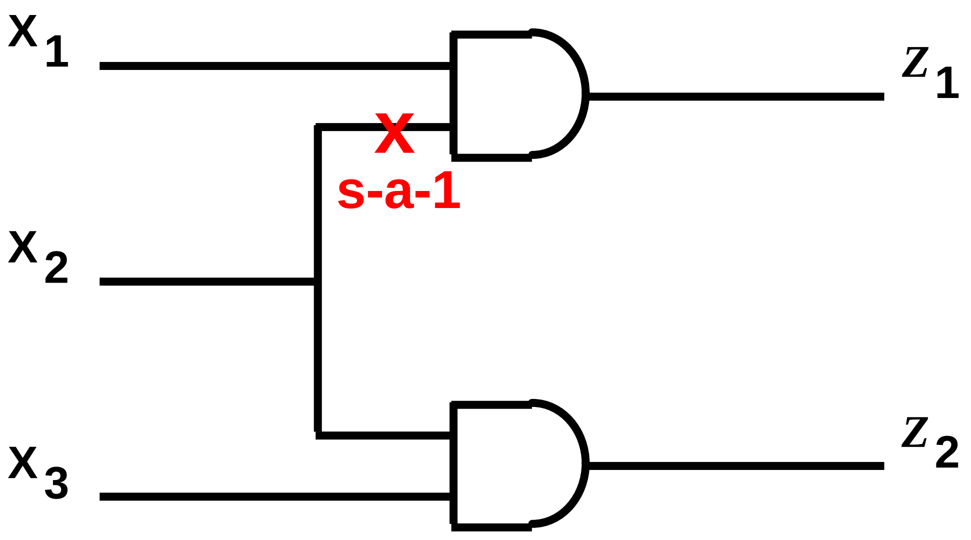


Why Logical Fault Modeling ?

- Fault analysis on logic rather than physical problem
  - Complexity is reduced
- Technology independent
  - Same fault model is applicable to many technologies
  - Testing and diagnosis methods remain valid despite changes in technology
- Tests derived
  - may be used for physical faults whose effect on circuit behavior is not completely understood or too complex to be analyzed
- Stuck-at fault is
  - The most popular logical fault model

Definition of Fault Detection

- A test (vector) *t* detects a fault *f* iff
  - *t* detects *f* ⇔ $z(t) \neq z_f(t)$
- Example



$z_1 = x_1 x_2$ $z_2 = x_2 x_3$

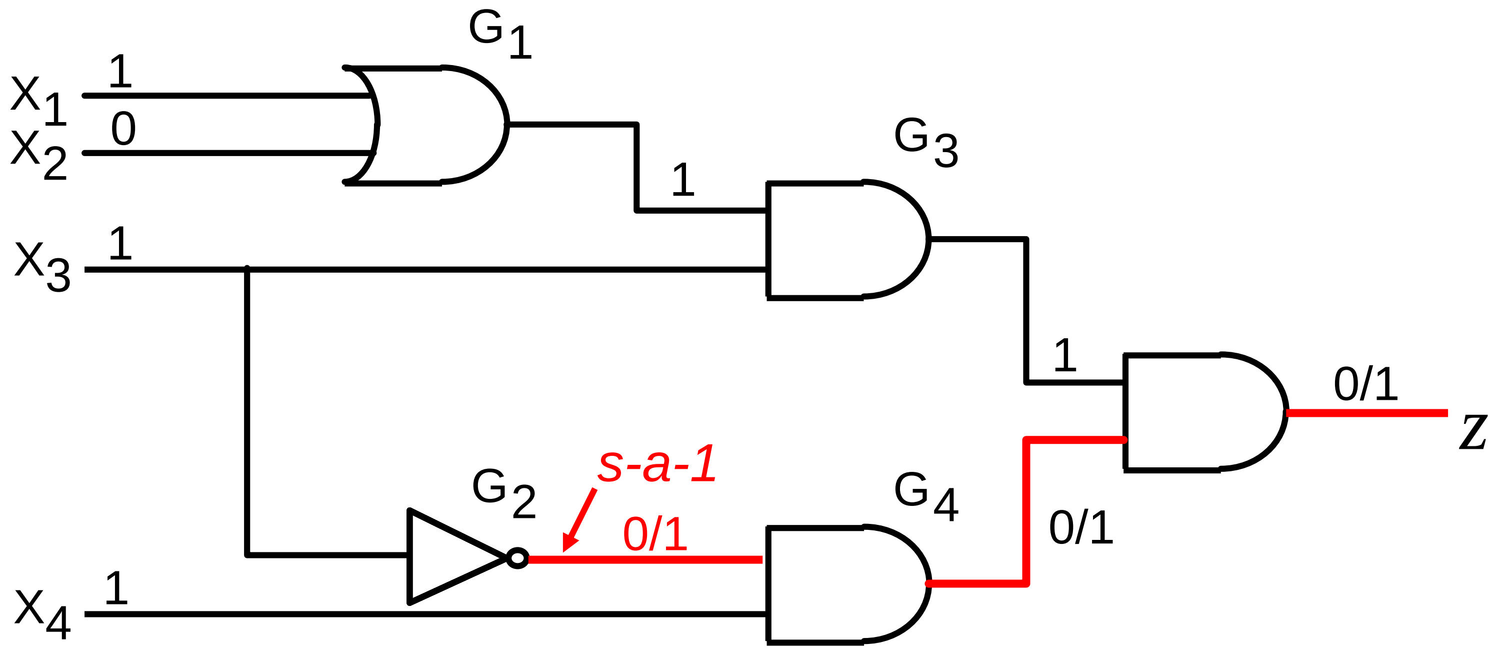
$z_{1f} = x_1$ $z_{2f} = x_2 x_3$

The test (x1,x2,x3) = (100) detects *f* because $z_1(100)=0$ while $z_{1f}(100)=1$

Fault Detection Requirement

- A test *t* that detects a fault *f*
  - Activates *f* (or generate a fault effect) by creating different v and v_f values at the site of the fault
  - Propagates the error to a primary output *w* by making all the lines along at least one path between the fault site and *w* have different v and v_f values
- Sensitized Line:
  - A line whose value in response to the test changes in the presence of the fault *f* is said to be sensitized by the test in the faulty circuit
- Sensitized Path:
  - A path composed of sensitized lines is called a sensitized path

Fault Sensitization



z (1011)=0 *z_f* (1011)=1

1011 detects the fault *f* (G$_2$ stuck-at 1)

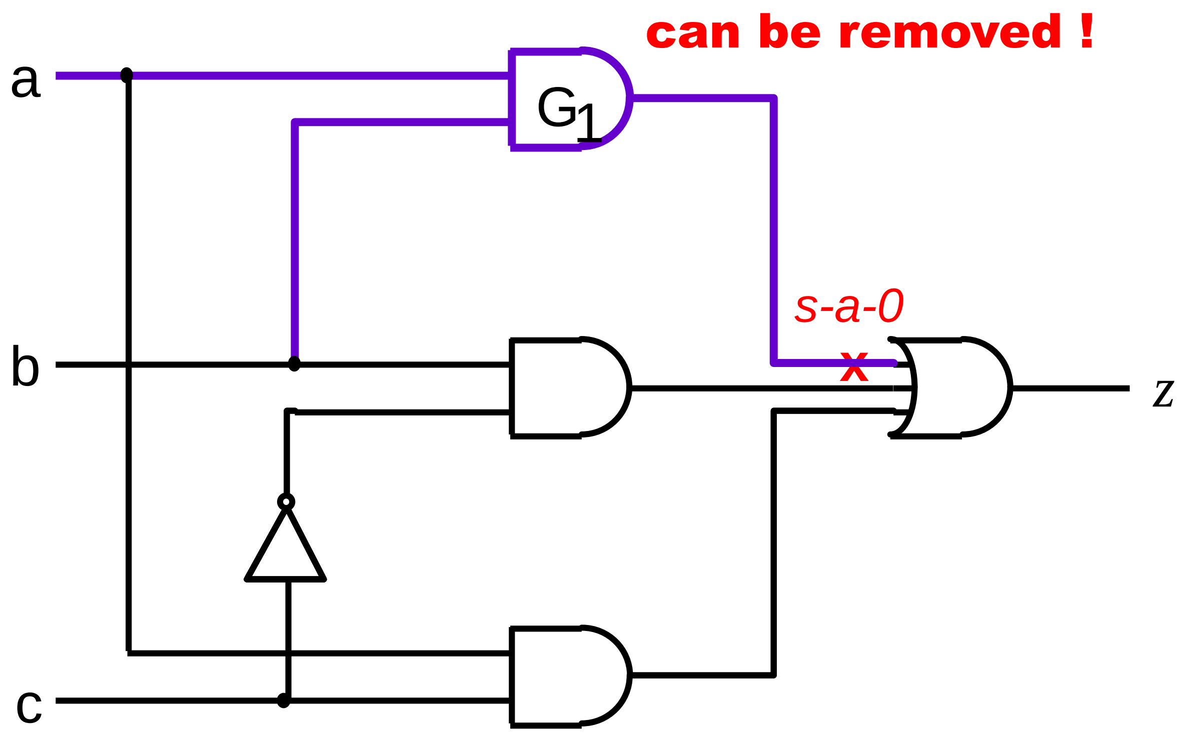
v/v$_f$: *v* = signal value in the fault free circuit

v_f = signal value in the faulty circuit

Detectability

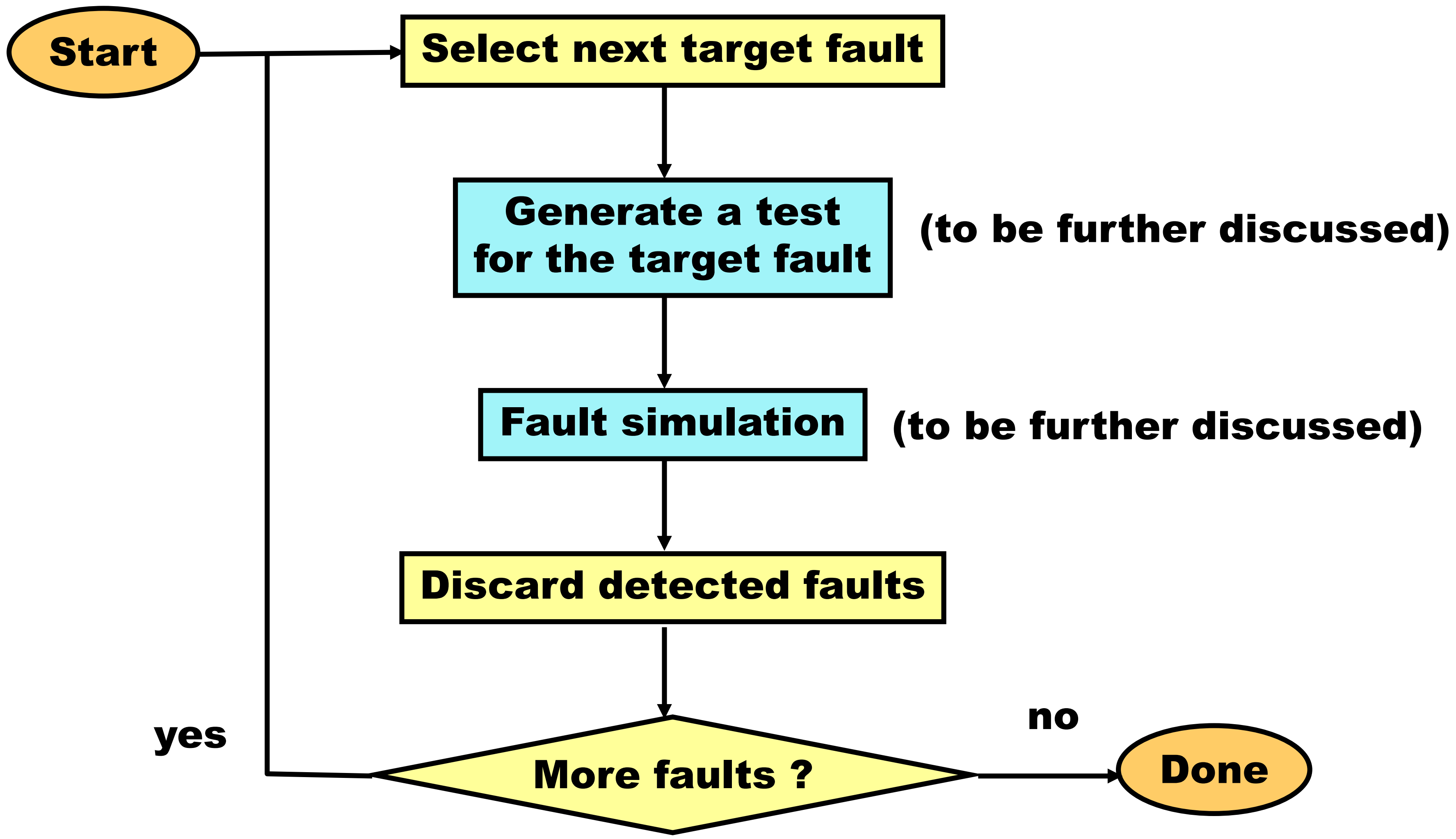
- A fault *f* is said to be detectable
  - if there exists a test *t* that detects *f* ; otherwise, *f* is an undetectable fault
- For an undetectable fault *f*
  - No test can simultaneously activate *f* and create a sensitized path to at least a primary output (fault-free and faulty values remain the same under any input vector)

Undetectable Fault



- G_1 output stuck-at-0 fault is undetectable
  - Undetectable faults do not change the function of the circuit
  - The related circuit can be deleted to simplify the circuit

Typical Test Generation Flow



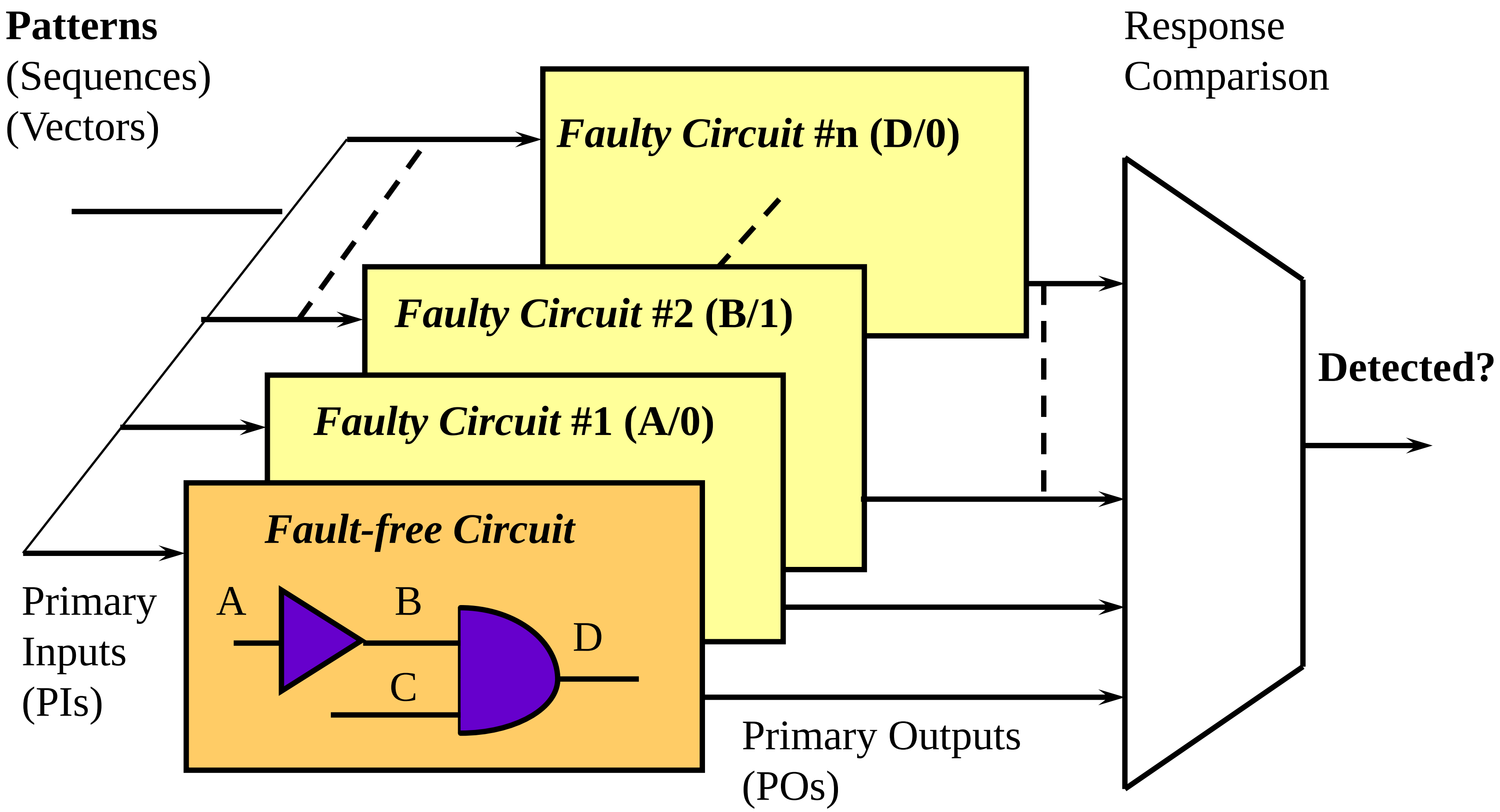
Why Fault Simulation ?

- To evaluate the quality of a test set/rate the effectiveness of a test set
  - i.e., to compute its fault coverage
- Part of an ATPG program
  - A vector usually detected multiple faults
  - Fault simulation is used to compute the faults accidentally detected by a particular vector
- To construct fault-dictionary
  - For post-testing diagnosis

Clarification: Logic and Fault Simulation

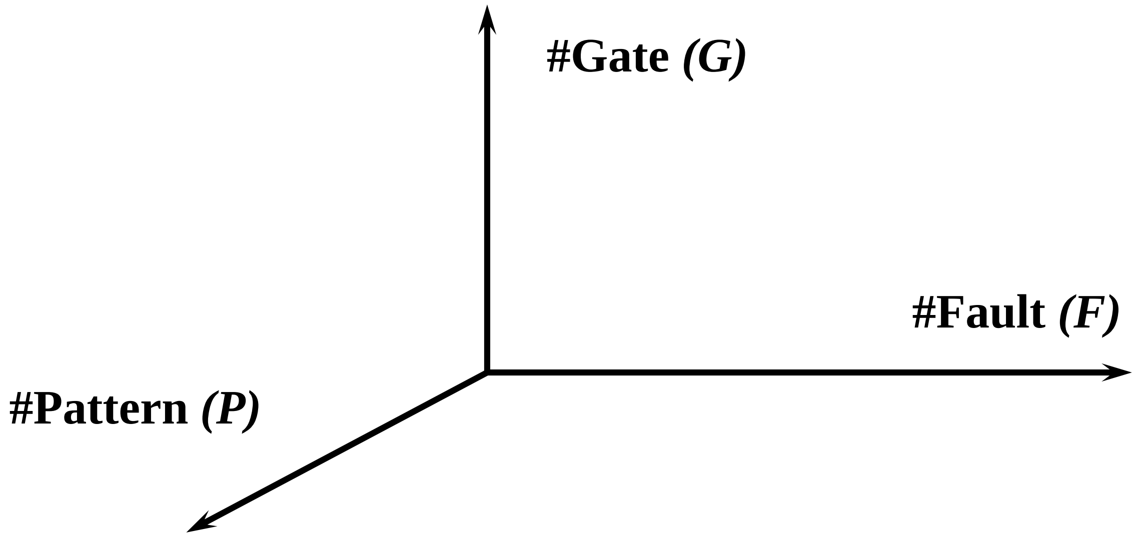
- Simulation is the process of predicting the behavior of a circuit design before it is physically built
- Logic simulation during the design stage
  - helps the designer verify that the design conforms to the functional specs
- Fault simulation during test development
  - is used to simulate faulty circuits
  - Logic simulation is generally referred to as fault-free simulation
- Logic simulation is intended for identifying design errors using the given specs or a known good design as references
  - While fault simulation is concerned with the behavior of fabricated circuits as a consequence of inevitable fabrication process imperfections (assuming the design is functionally correct)

Conceptual Fault Simulation



Logic simulation on both good (fault-free) and faulty circuits

Complexity of Fault Simulation



- **Complexity ~ *$F \cdot P \cdot G$* ~ *$O(G^3)$***
- **The complexity is higher than logic simulation by a factor of *F*, while usually is much lower than ATPG**
- **The complexity can be greatly reduced using**
    - **Fault dropping and other advanced techniques**

General ATPG Flow

- ATPG (Automatic Test Pattern Generation)
  - Generate a set of vectors for a set of target faults
- Basic flow

```
Initialize the vector set to NULL
Repeat
    Generate a new test vector
    Evaluate fault coverage for the test vector (fault simulation)
    If the test vector is acceptable, then add it to the vector set
Until required fault coverage is obtained
```

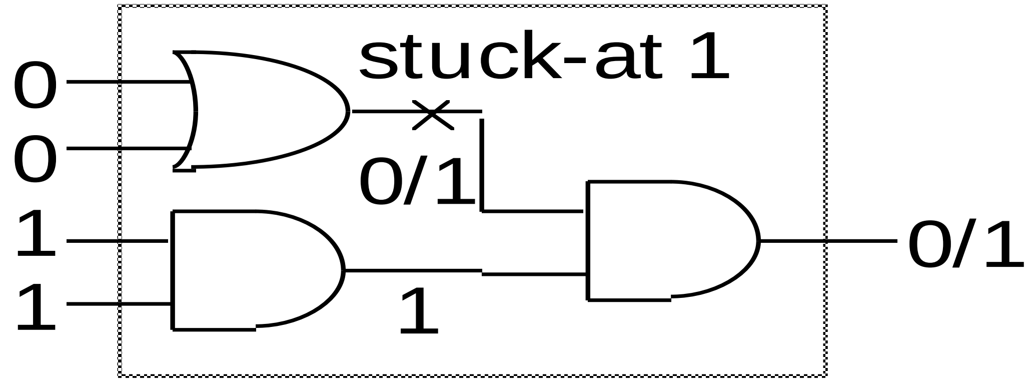
- To accelerate the ATPG
  - Random patterns are often generated first to detect easy-to-detect faults, then a deterministic TG is performed to generate tests for the remaining faults

Combinational ATPG

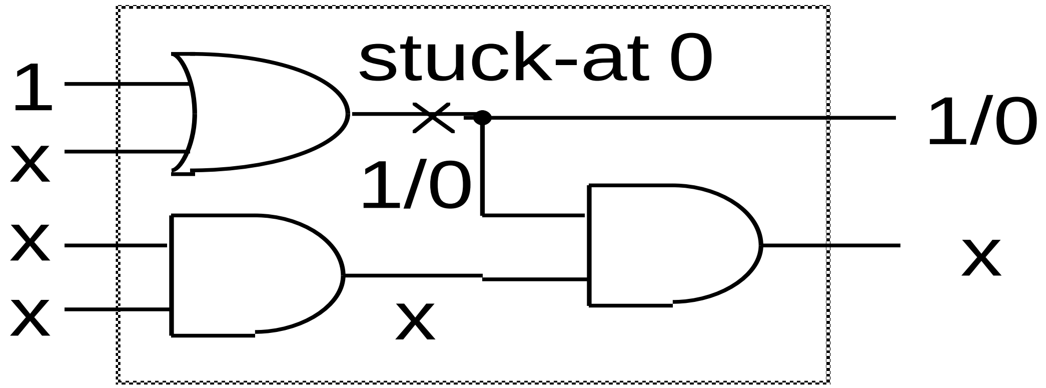
- Test Generation (TG) Methods
  - **Based on Truth Table**
  - Based on Boolean Equation
  - Based on Structural Analysis
- Milestone Structural ATPG Algorithms
  - D-algorithm [Roth 1967]
  - 9-Valued D-algorithm [Cha 1978]
  - PODEM [Goel 1981]
  - FAN [Fujiwara 1983]

A Test Pattern

A Fully Specified Test Pattern (every PI is either 0 or 1)



A Partially Specified Test Pattern (certain PI's could be undefined)



Why DFT ?

- Direct Testing is Way Too Difficult !
  - Large number of FFs
  - Embedded memory blocks
  - Embedded analog blocks

- **Design For Testability is inevitable**
  - Like death and tax

Design For Testability

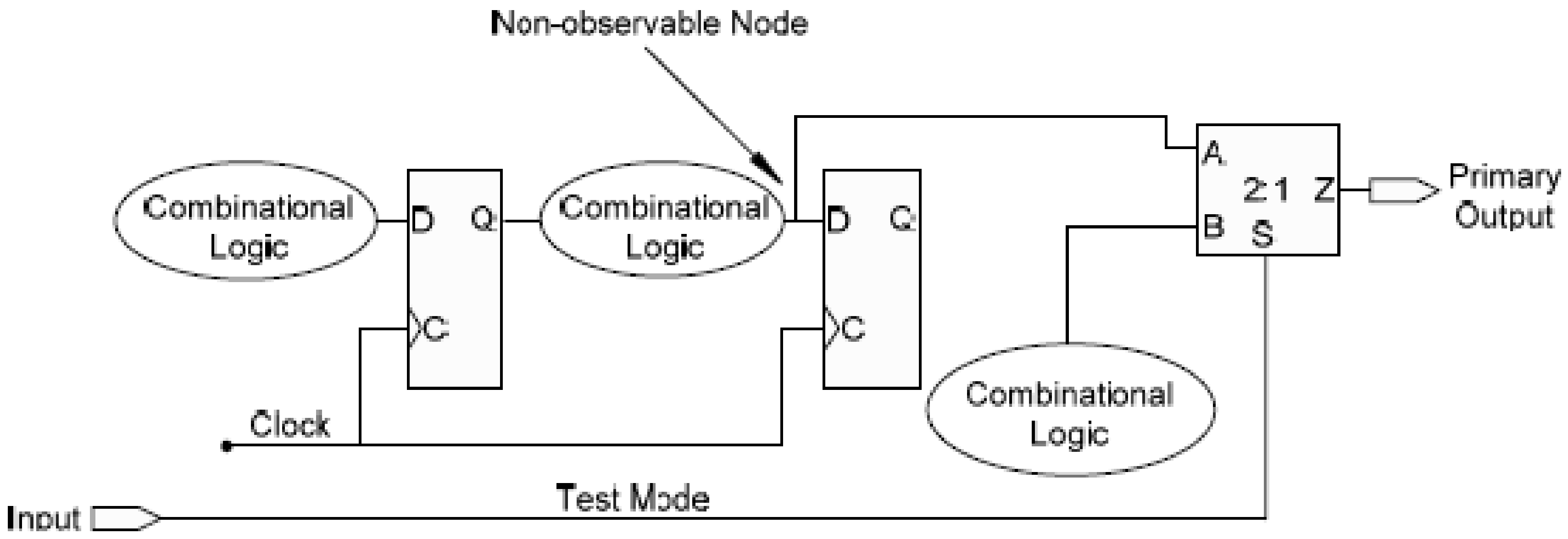
- Definition
  - Design For Testability (DFT) refers to those design techniques that make test generation and testing cost-effective
- DFT Methods
  - Ad-hoc methods
  - Scan, full and partial
  - Built-In Self-Test (BIST)
  - Boundary scan
- Cost of DFT
  - Pin count, area, performance, design-time, test-time

Important Factors

- Controllability
  - Measure the ease of controlling a line
- Observability
  - Measure the ease of observing a line at PO
- DFT deals with ways of improving
  - Controllability
  - Observability

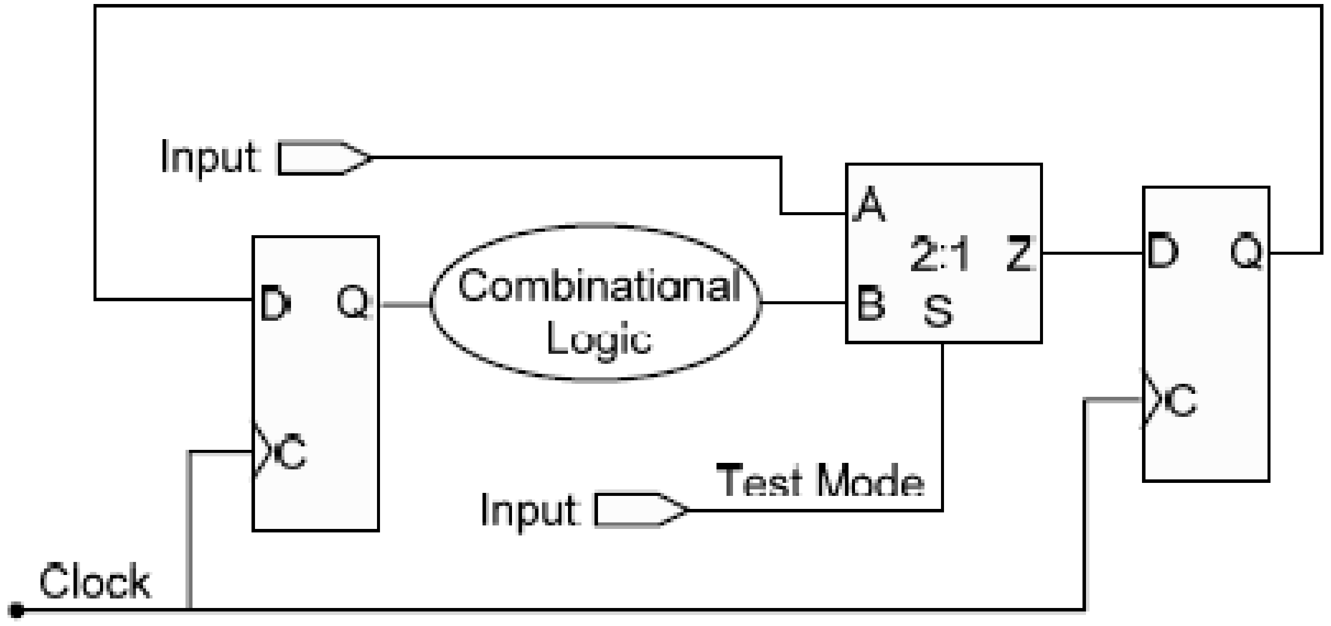
Improving Observability

- Connect non-observable nodes directly to a primary output using a MUX



Improving Controllability

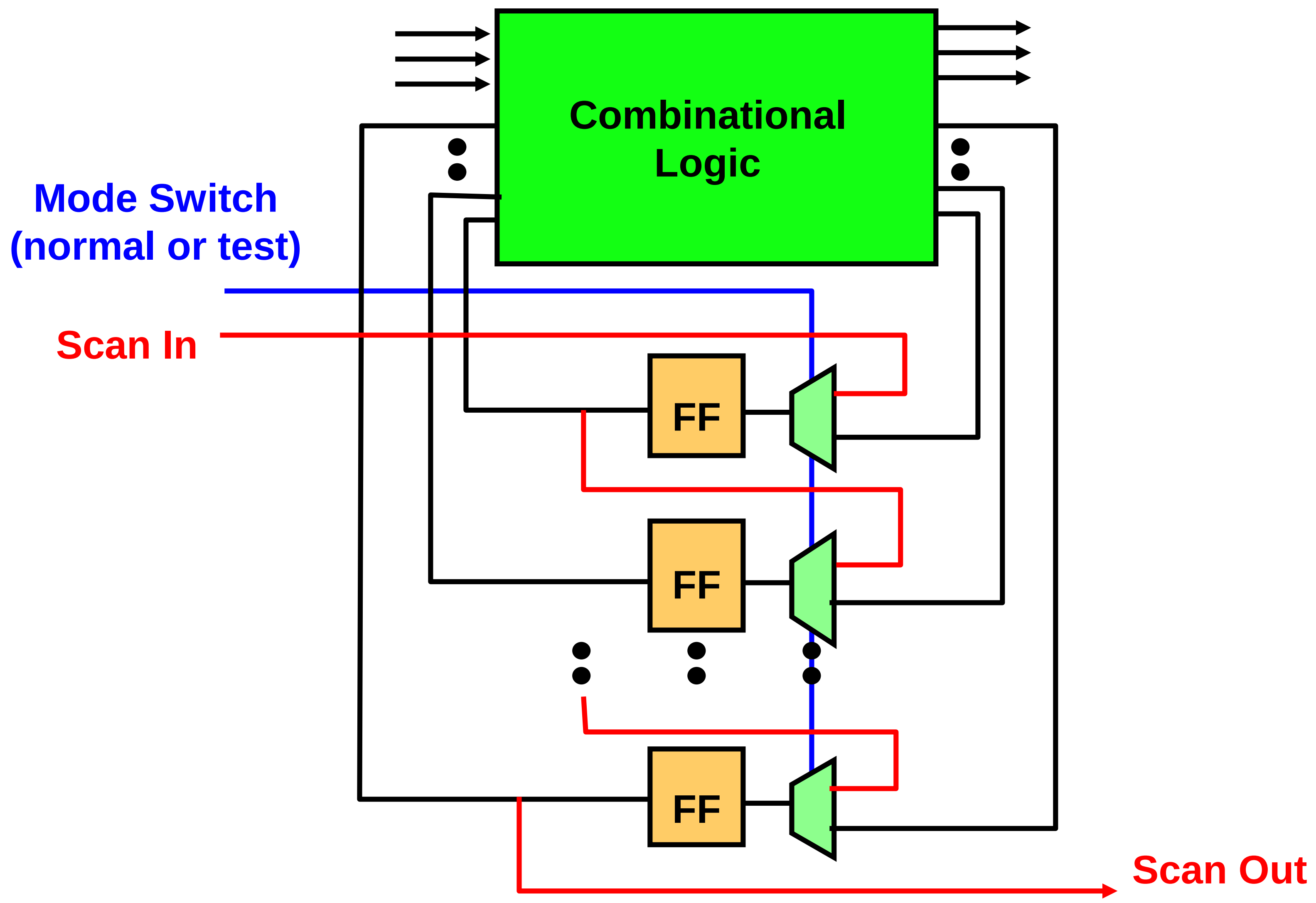
- MUX may also be added to the circuit for improved controllability
  - Inserting a MUX at the D input of the flip-flop allows the circuit to be initialized from a primary input.



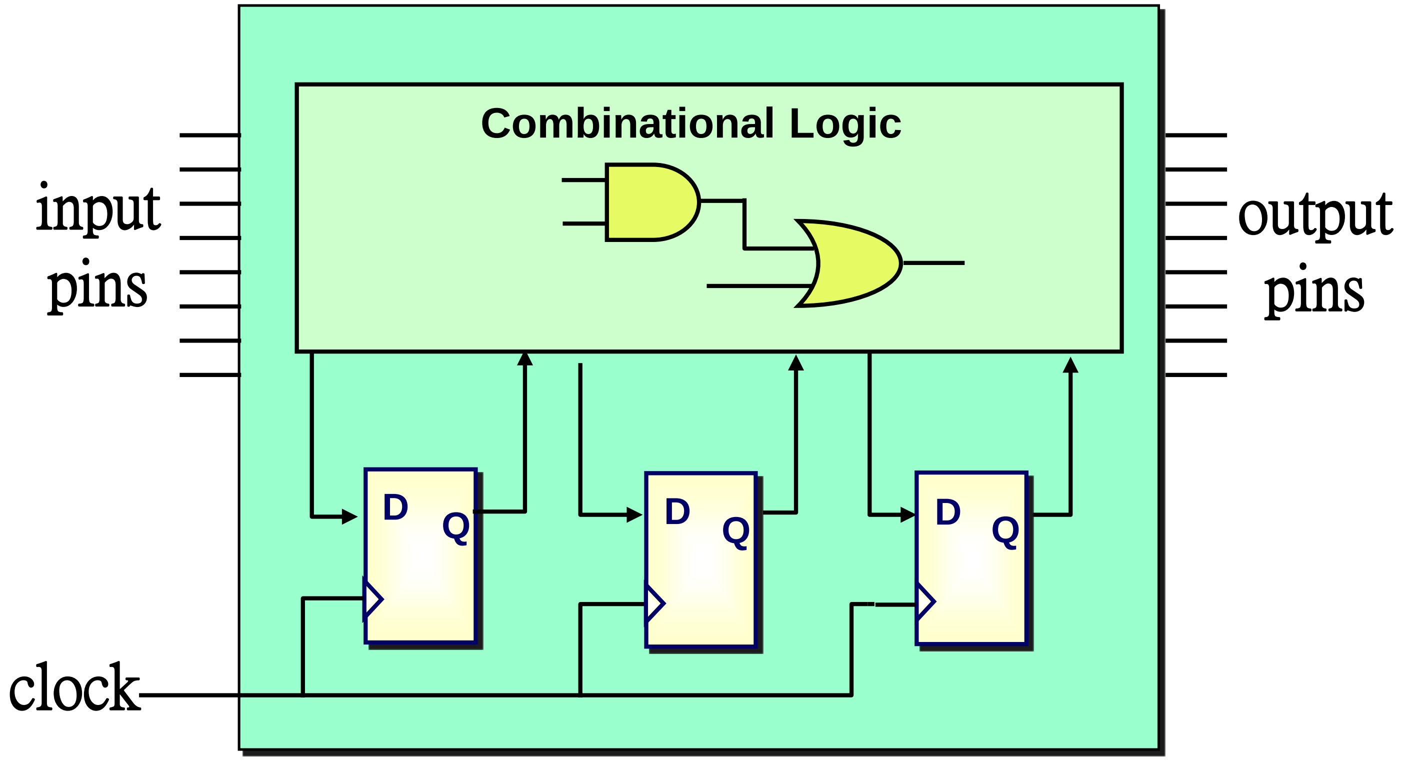
What is Scan ?

- Objective
  - To provide controllability and observability at internal state variables for testing
- Method
  - Add test mode control signal(s) to circuit
  - Connect flip-flops to form shift registers in test mode
  - Make inputs/outputs of the flip-flops in the shift register controllable and observable
- Types
  - Internal scan
    - Full scan, Partial scan, Random access
  - Boundary scan

The Scan Concept

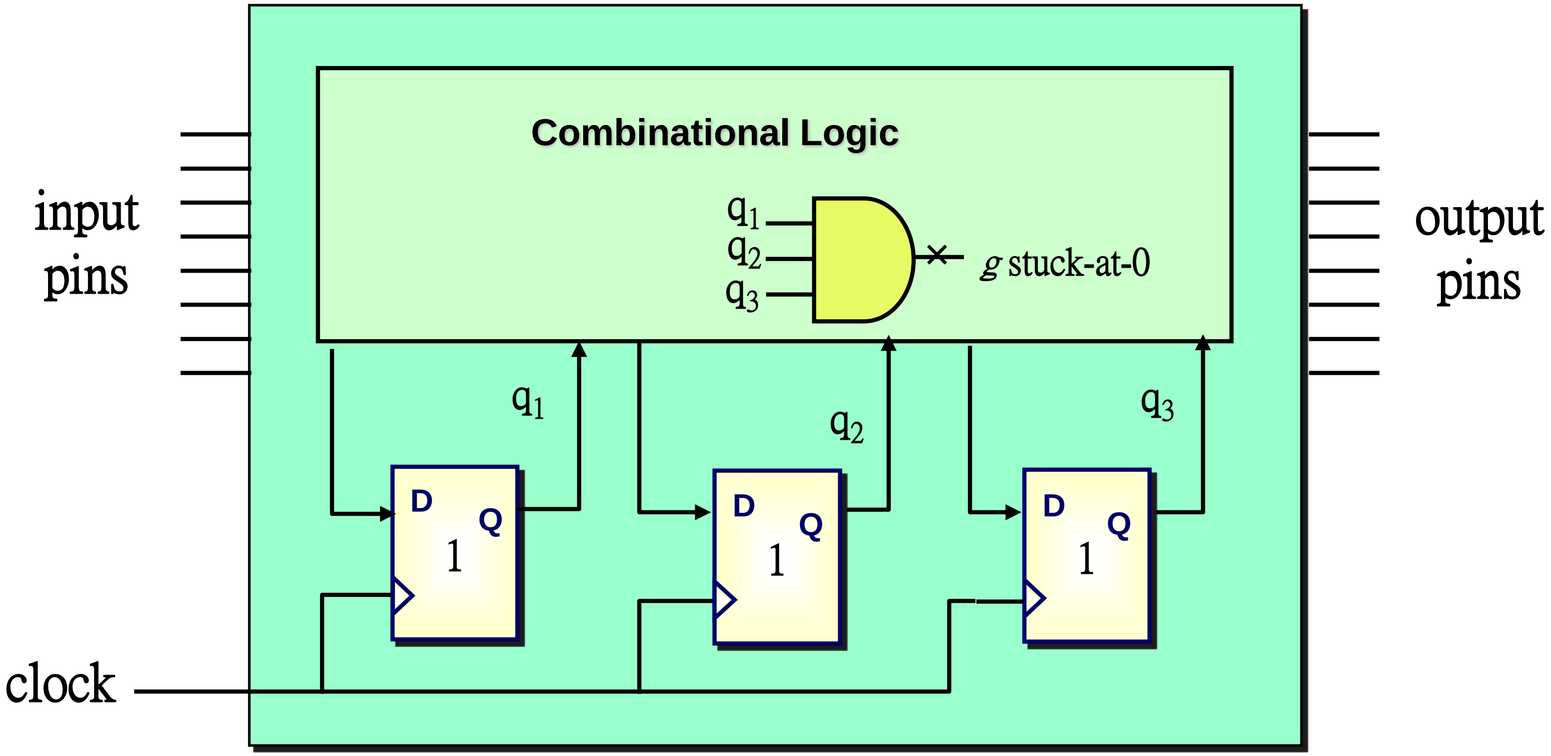


A Logic Design Before Scan Insertion



Sequential ATPG is extremely difficult:
due to the lack of controllability and observability at flip-flops.

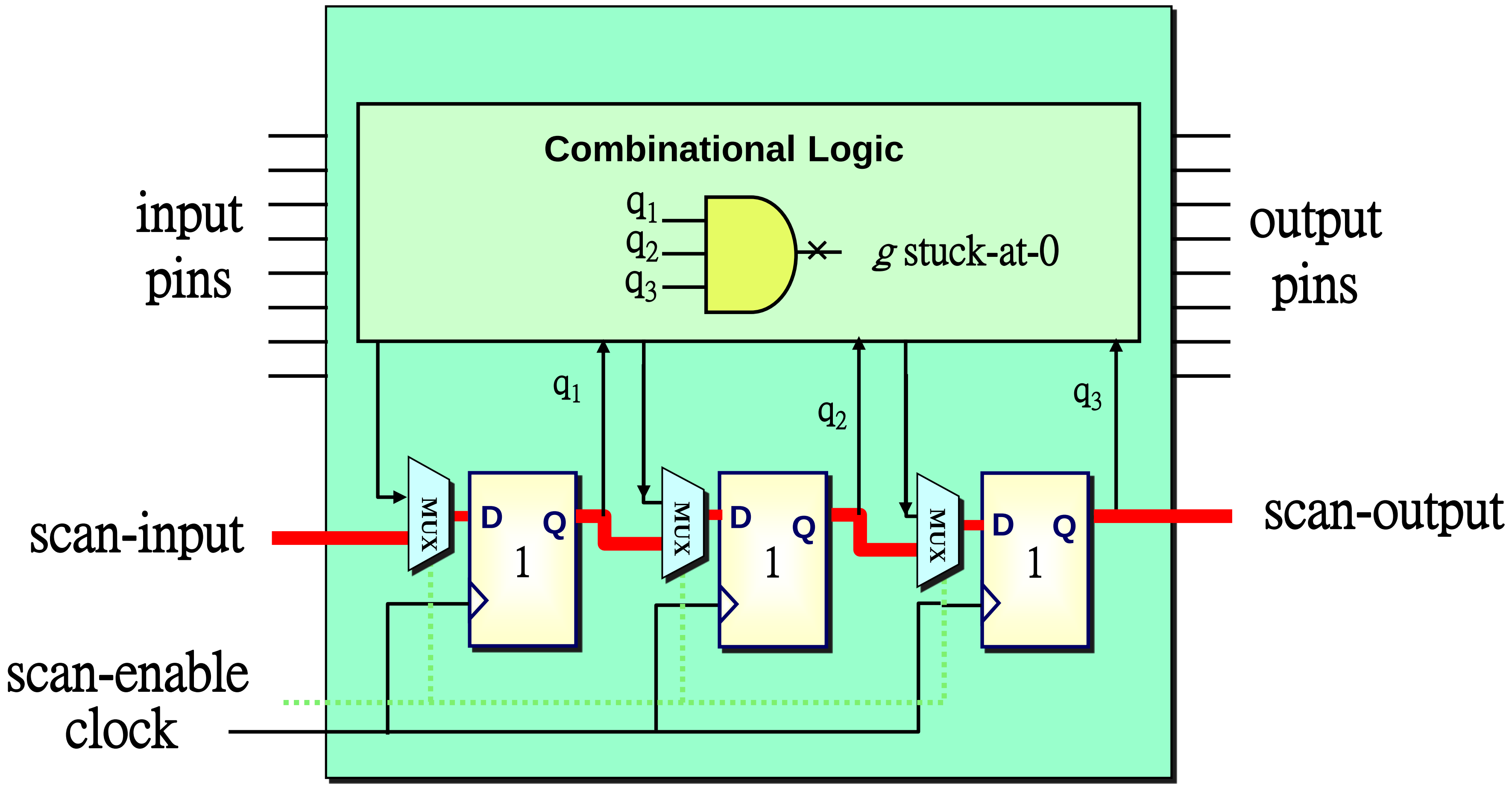
Example: A 3-stage Counter



It takes 8 clock cycles to set the flip-flops to be (1, 1, 1), for detecting the target fault *g* stuck-at-0 fault

(2^{20} cycles for a 20-stage counter !)

A Logic Design After Scan Insertion



Scan Chain provides an easy access to flip-flops

$\longrightarrow$ Pattern Generation is much easier !!

Some Problems with Full Scan

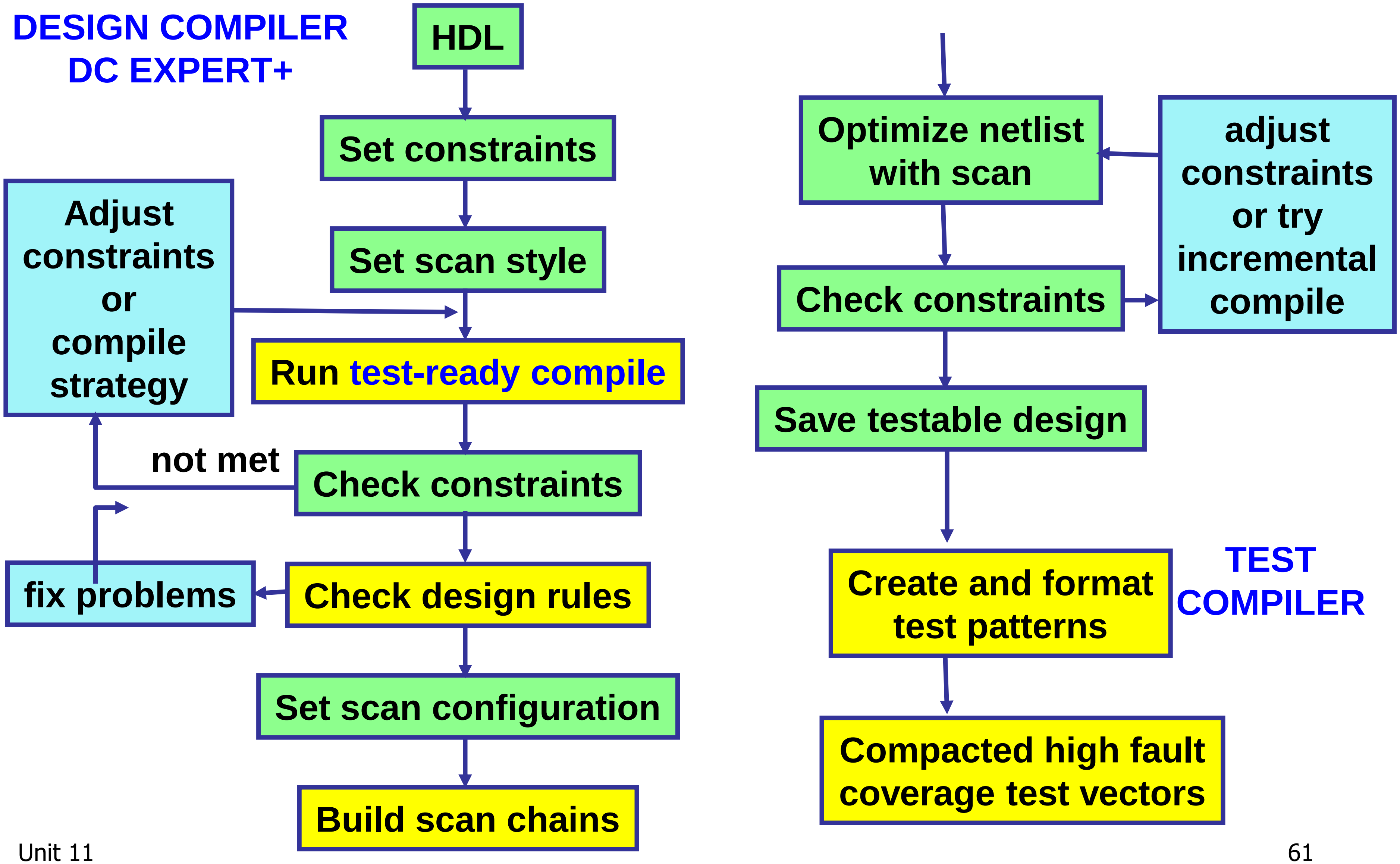
Major Commercial Test Tool Companies
Synopsys
Mentor-Graphics
SynTest (華騰科技)
LogicVision

- Problems
  - Area overhead
  - Possible performance degradation
  - High test application time
  - Power dissipation
- Features of Commercial Tools
  - Scan-rule violation check (e.g., DFT rule check)
  - Scan insertion (convert a FF to its scan version)
  - ATPG (both combinational and sequential)
  - Scan chain reordering after layout

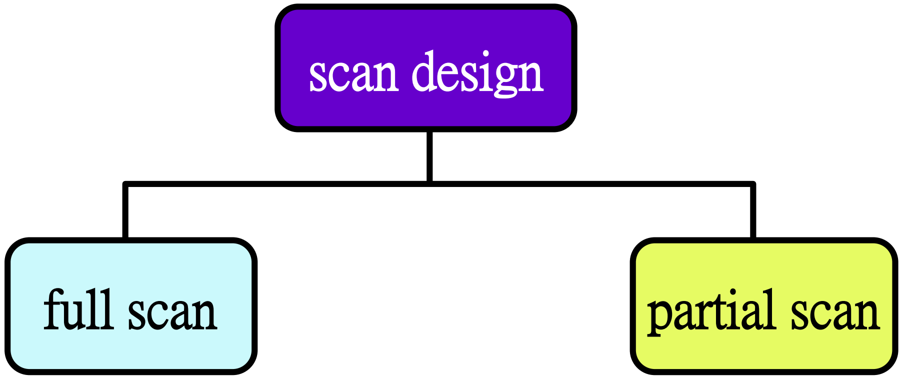
Performance Overhead

- The increase of delay along the normal data paths include:
  - Extra gate delay due to the multiplexer
  - Extra delay due to the capacitive loading of the scan-wiring at each flip-flop's output
- Timing-driven partial scan
  - Try to avoid scan flip-flops that belong to the timing critical paths
  - The flip-flop selection algorithm for partial scan can take this into consideration to reduce the timing impact of scan to the design

Typical Flat Design Flow



Full Scan vs. Partial Scan



every flip-flop is a scan-FF   NOT every flip-flop is a scan-FF

test time	longer	shorter
hardware overhead	more	less
fault coverage	~100%	unpredictable
ease-of-use	easier	harder

Summary

- Testing
  - Conducted after manufacturing
  - Must be considered during the design process
- Major Fault Models
  - Stuck-At, Bridging, Stuck-Open, Delay Fault, …
- Major Tools Needed
  - Design-For-Testability
    - By Scan Chain Insertion or Built-In Self-Test
  - Fault Simulation
  - Automatic Test Pattern Generation
- Other Applications in CAD
  - ATPG is a way of Boolean Reasoning, applicable to may logic-domain CAD problems