Introduction to Electronic Design Automation

Jie-Hong Roland Jiang
江介宏

Department of Electrical Engineering
National Taiwan University

Spring 2011

Testing

Slides are by Courtesy of Prof. S.-Y. Huang and C.-M. Li

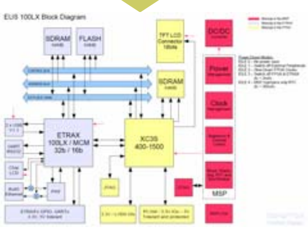
Testing

- **Recap**
  - Design verification
    - Is what I specified really what I wanted?
      - Property checking
  - Implementation verification
    - Is what I implemented really what I specified?
      - Equivalence checking
  - Manufacture verification
    - Is what I manufactured really what I implemented?
      - Testing; post manufacture verification
      - Quality control
        - Distinguish between good and bad chips

Design Flow

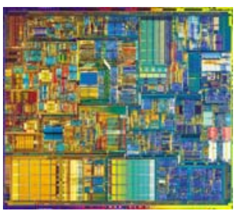
Idea

Architecture Design



Block diagram

Circuit & Layout Design



Layout

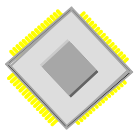
IC Fabrication

Wafer (hundreds of dies)

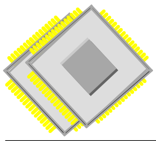
Sawing & Packaging

Final chips

Final Testing



Bad chips



Good chips

customers

Manufacturing Defects

- Processing faults
  - 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

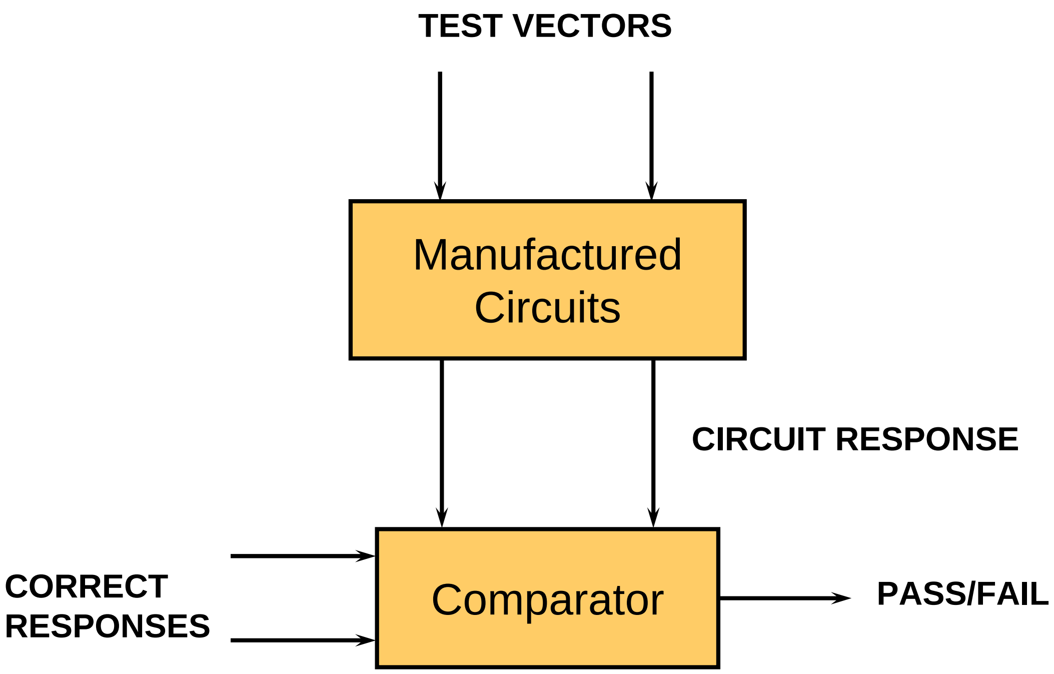
- Faults
  - A physical defect within a circuit or a system
  - May or may not cause a system failure
- Errors
  - Manifestation of a fault that results in incorrect circuit (system) outputs or states
  - Caused by faults
- Failures
  - Deviation of a circuit or system from its specified behavior
  - Fail to do what is supposed to do
  - Caused by errors
- Faults cause errors; errors cause failures

Testing and Diagnosis

- Testing
  - Exercise a system and analyze the response to ensure whether it behaves correctly after manufacturing

- Diagnosis
  - Locate the causes of misbehavior after the incorrectness is detected

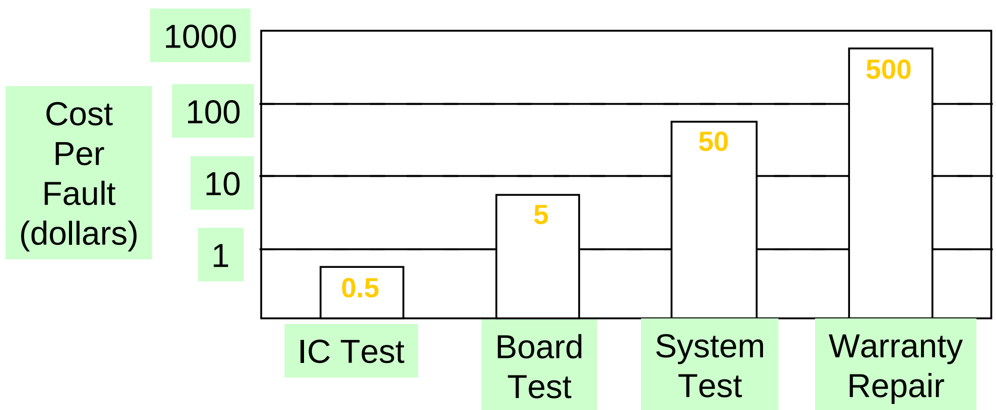
Scenario of Manufacturing Test



Test Systems

Purpose of Testing

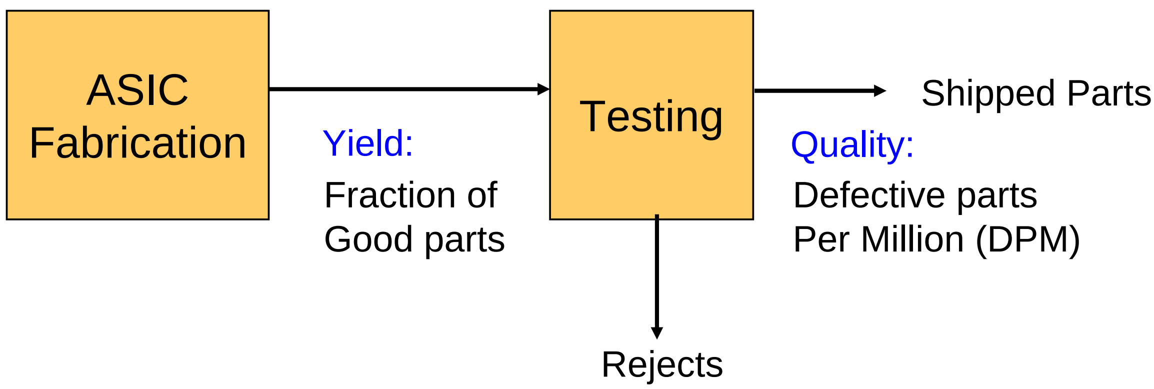
- Verify manufactured circuits
  - Improve system reliability
  - Reduce repair costs
    - Repair cost goes up by an order of magnitude each step away from the fab. line



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

Testing and Quality

- Quality of shipped part can be expressed as a function of the yield Y and test (fault) coverage T.



Fault Coverage

- Fault coverage T
  - Measure of the ability of a test set to detect a given set of faults that may occur on the Design Under Test (DUT)

$$T = \frac{\text{\# detected faults}}{\text{\# all possible faults}}$$

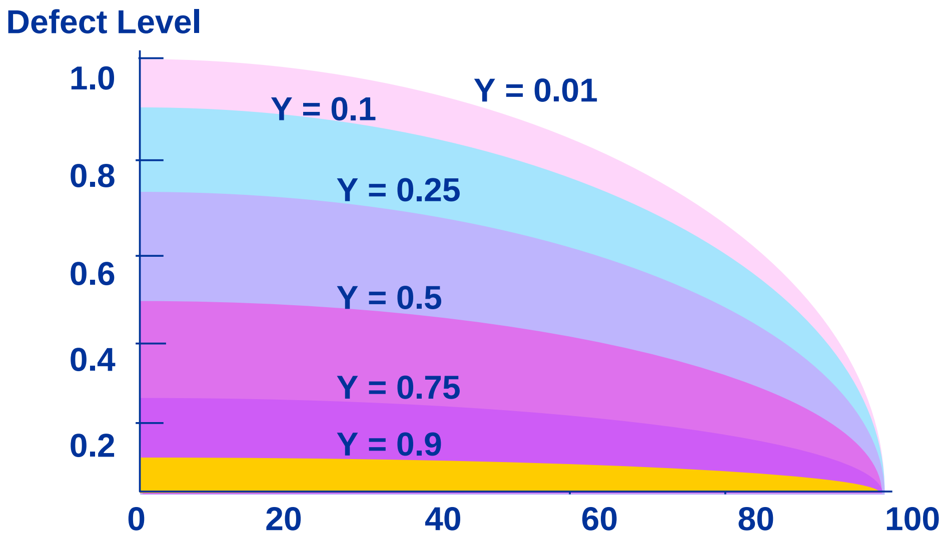
Defect Level

- A defect level is the fraction of the shipped parts that are defective

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

Y: yield
T: fault coverage

Defect Level vs. Fault Coverage



(Williams IBM 1980)

High fault coverage → Low defect level

DPM vs. Yield and Coverage

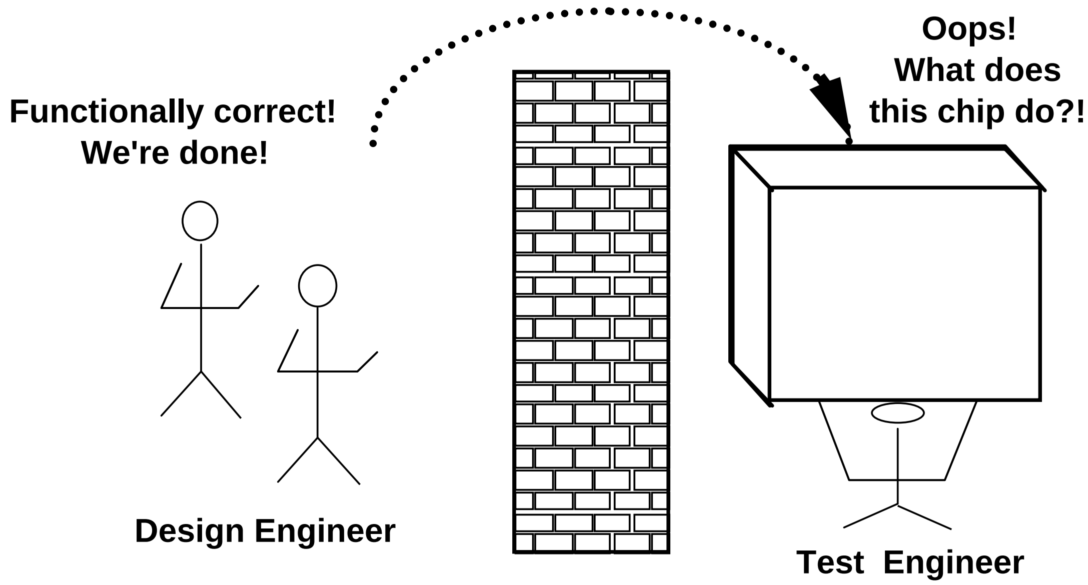
Yield	Fault Coverage	DPM
50%	**90%**	**67,000**
75%	**90%**	**28,000**
90%	**90%**	**10,000**
95%	**90%**	**5,000**
99%	**90%**	**1,000**
90%	**90%**	**10,000**
90%	**95%**	**5,000**
90%	**99%**	**1,000**
90%	**99.9%**	**100**

Why Testing Is Difficult ?

- Test time explodes exponentially in exhaustive testing of VLSI
  - For a combinational circuit with 50 inputs, need 2^{50} = 1.126 x 10^{15} test patterns.
  - Assume one test per 10^{-7}sec, it takes 1.125x10^{8}sec = 3.57years.
  - 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!



Outline

- Fault Modeling
- Fault Simulation
- Automatic Test Pattern Generation
- Design for Testability

Functional vs. Structural Testing

- I/O functional testing is inadequate for manufacturing
  - Need fault models
- Exhaustive testing is daunting
  - Need abstraction and smart algorithms
  - Structural testing is more effective

Why Fault Model ?

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

Fault Modeling vs. Physical Defects

- 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 vs. Physical Defects (cont'd)

- Electrical effects
  - Shorts (bridging faults)
  - Opens
  - Transistor stuck-on/open
  - Resistive shorts/opens
  - Change in threshold voltages

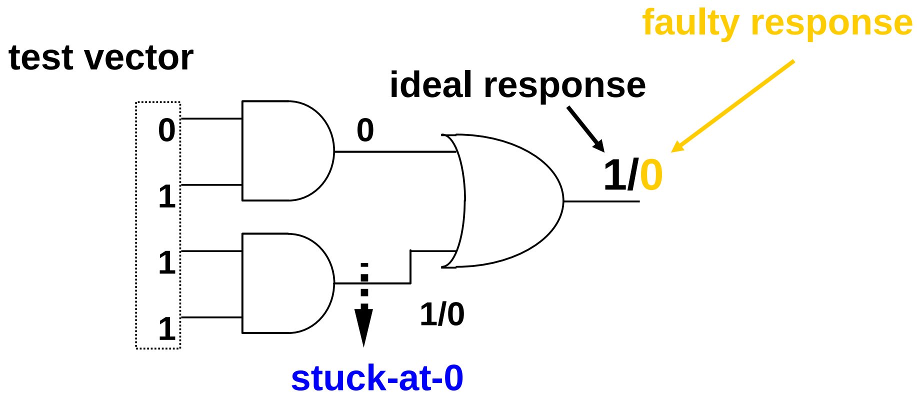
- Logical effects
  - Logical stuck-at-0/1
  - Slower transition (delay faults)
  - AND-bridging, OR-bridging

Typical Fault Types

- **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 wire is faulty
  - Fault can be at an input or output of a gate
  - Faulty wire permanently sticks at 0 or 1



Multiple Stuck-At Faults

- Several stuck-at faults occur at the same time
  - Common 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 design
- Detection capability of un-modeled defects
  - Empirically, many un-modeled defects can also be detected accidentally under the single stuck-at fault model
- Cover a large percentage of multiple stuck-at faults

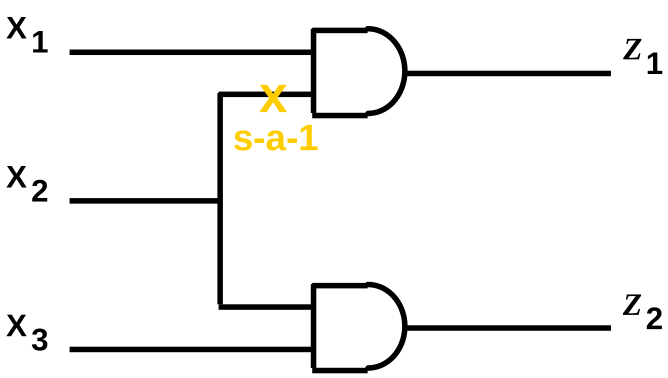
Why Logical Fault Modeling ?

- 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
- Wide applications
  - The derived tests may be used for physical faults whose effect on circuit behavior is not completely understood or too complex to be analyzed
- Popularity
  - 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 (i.e. $z(t) \neq z_f(t)$)

Example



$z_1=x_1x_2$ $\quad$ $z_2=x_2x_3$

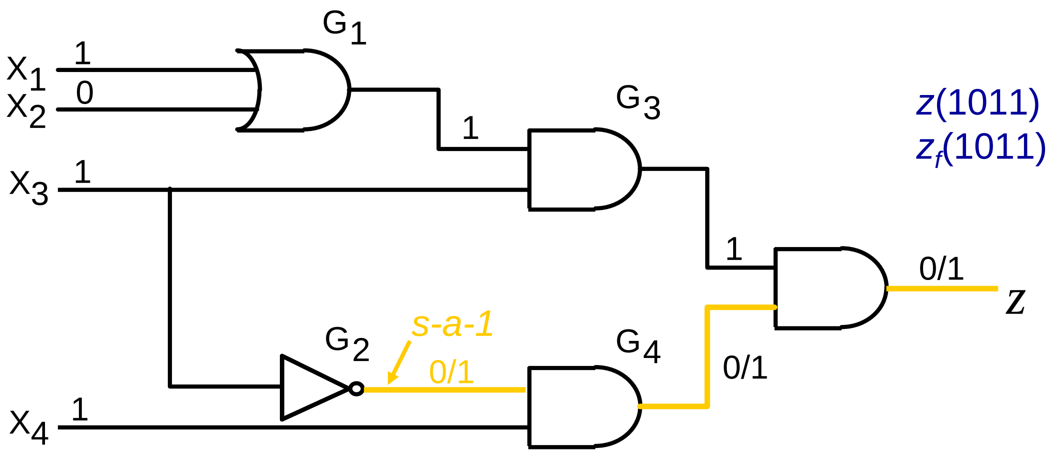
$z_{1f}=x_1$ $\quad$ $z_{2f}=x_2x_3$

Test (x1,x2,x3) = (100) detects f because $z_1(100)=0$ and $z_{1f}(100)=1$

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 *z* by making all the wires along at least one path between the fault site and *z* have different *v* and v_f values
- Sensitized wire
  - A wire 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 wires is called a sensitized path

Fault Sensitization



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

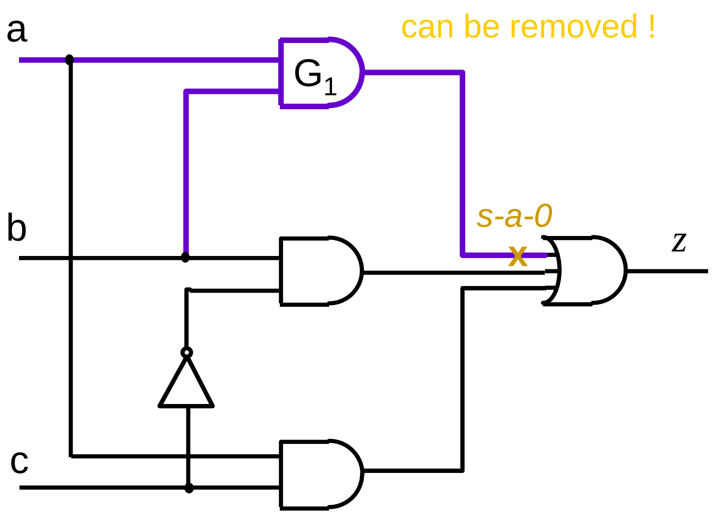
Input vector 1011 detects the fault *f* (G_2 stuck-at-1)
v/v_f: v = signal value in the fault free circuit
v_f = signal value in the faulty circuit

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 some primary output

Undetectable Fault

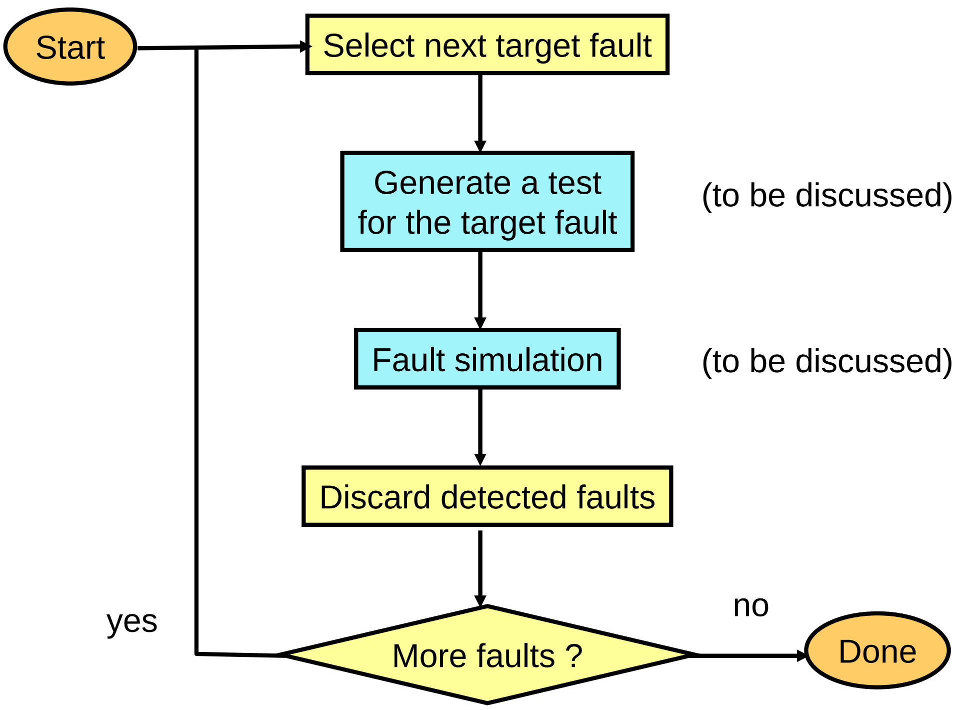
- The stuck-at-0 fault at G_1 output is undetectable
  - Undetectable faults do not change the function of the circuit
  - The related circuit can be deleted to simplify the circuit



Test Set

- Complete detection test set
  - A set of tests that detects any detectable fault in a designated set of faults
- Quality of a test set
  - is measured by fault coverage
- Fault coverage
  - Fraction of the faults detected by a test set
  - can be determined by fault simulation
  - >95% is typically required under the single stuck-at fault model
  - >99.9% required in the ICs manufactured by IBM

Typical Test Generation Flow

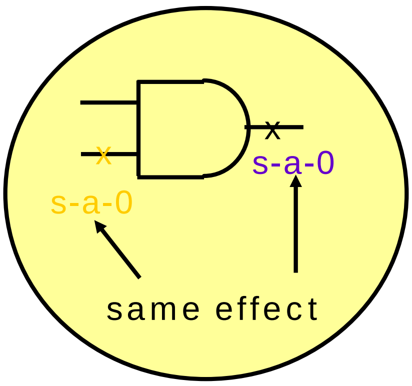


Fault Equivalence

- **Distinguishing test**
  - A test t distinguishes faults α and β if $z_\alpha(t) \neq z_\beta(t)$ for some PO function z
- **Equivalent faults**
  - Two faults α and β are said to be equivalent in a circuit iff the function under α is equal to the function under β for every input assignment (sequence) of the circuit.
  - That is, no test can distinguish α and β, i.e., test-set(α) = test-set(β)

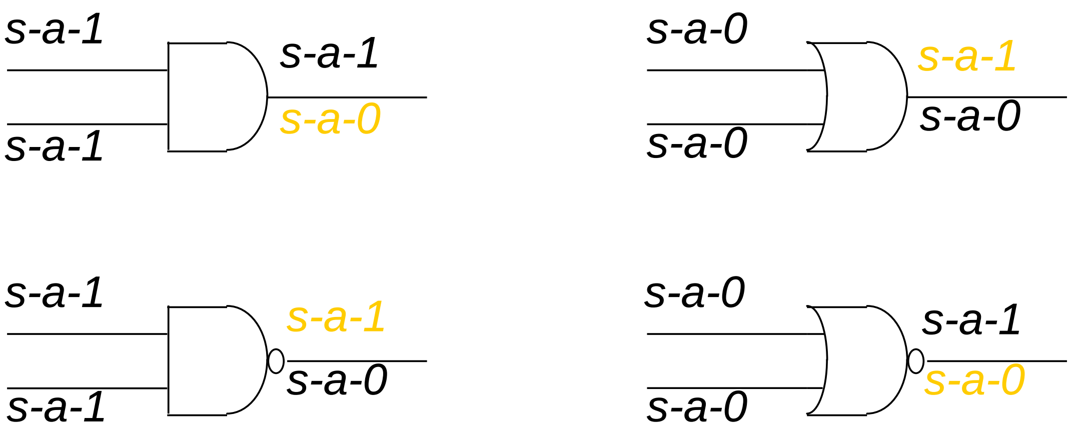
Fault Equivalence

- AND gate:
  - all *s-a-0* faults are equivalent
- OR gate:
  - all *s-a-1* faults are equivalent
- NAND gate:
  - all the input *s-a-0* faults and the output *s-a-1* faults are equivalent
- NOR gate:
  - all input *s-a-1* faults and the output *s-a-0* faults are equivalent
- Inverter:
  - input *s-a-1* and output *s-a-0* are equivalent
  - input *s-a-0* and output *s-a-1* are equivalent



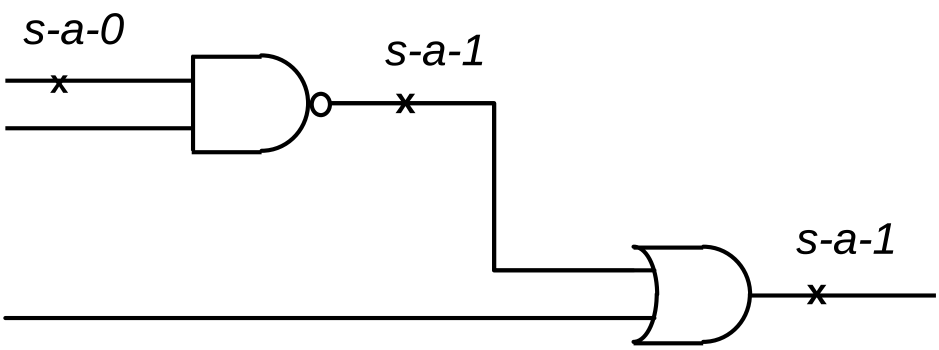
Equivalence Fault Collapsing

- *n+2*, instead of *2(n+1)*, single stuck-at faults need to be considered for *n*-input AND (or OR) gates



Equivalent Fault Group

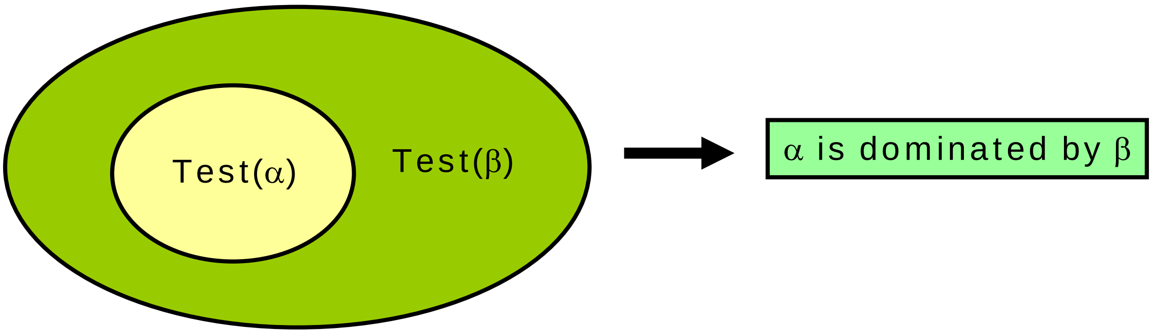
- In a combinational circuit
  - Many faults may form an equivalence group
  - These equivalent faults can be found in a reversed topological order from POs to PIs



Three faults shown are equivalent !

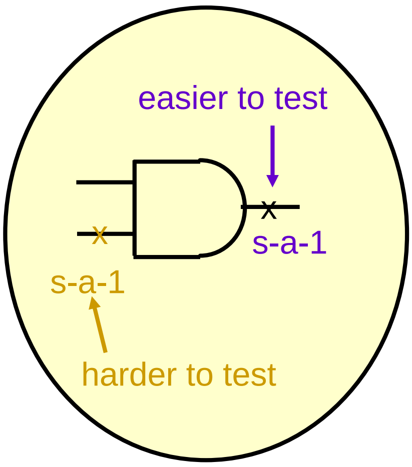
Fault Dominance

- Dominance relation
  - A fault β is said to *dominate* another fault α in an irredundant circuit iff every test (sequence) for α is also a test (sequence) for β.
  - I.e., test-set(α) ⊆ test-set(β)
  - No need to consider fault β for fault detection



Fault Dominance

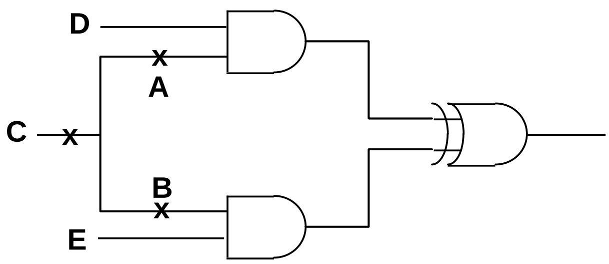
- AND gate
  - Output *s-a-1* dominates any input *s-a-1*
- NAND gate
  - Output *s-a-0* dominates any input *s-a-1*
- OR gate
  - Output *s-a-0* dominates any input *s-a-0*
- NOR gate
  - Output *s-a-1* dominates any input *s-a-0*



- Dominance fault collapsing
  - Reducing the set of faults to be analyzed based on the dominance relation

Stem vs. Branch Faults

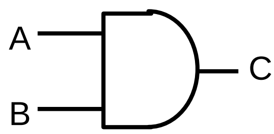
- Detect A s-a-1:
  $z(t) \oplus z_f(t) = (\mathbf{CD} \oplus \mathbf{CE}) \oplus (\mathbf{D} \oplus \mathbf{CE}) = \mathbf{D} \oplus \mathbf{CD} = \mathbf{1}$
  $\Rightarrow (\mathbf{C} = \mathbf{0},\ \mathbf{D} = \mathbf{1})$
- Detect C s-a-1:
  $z(t) \oplus z_f(t) = (\mathbf{CD} \oplus \mathbf{CE}) \oplus (\mathbf{D} \oplus \mathbf{E}) = \mathbf{1}$
  $\Rightarrow (\mathbf{C} = \mathbf{0},\ \mathbf{D} = \mathbf{1})$ **or** $(\mathbf{C} = \mathbf{0},\ \mathbf{E} = \mathbf{1})$
- Hence, C s-a-1 dominates A s-a-1
- Similarly
  - C s-a-1 dominates B s-a-1
  - C s-a-0 dominates A s-a-0
  - C s-a-0 dominates B s-a-0
- In general, there might be no equivalence or dominance relations between stem and branch faults



C: stem of a multiple fanout
A, B: branches

Analysis of a Single Gate

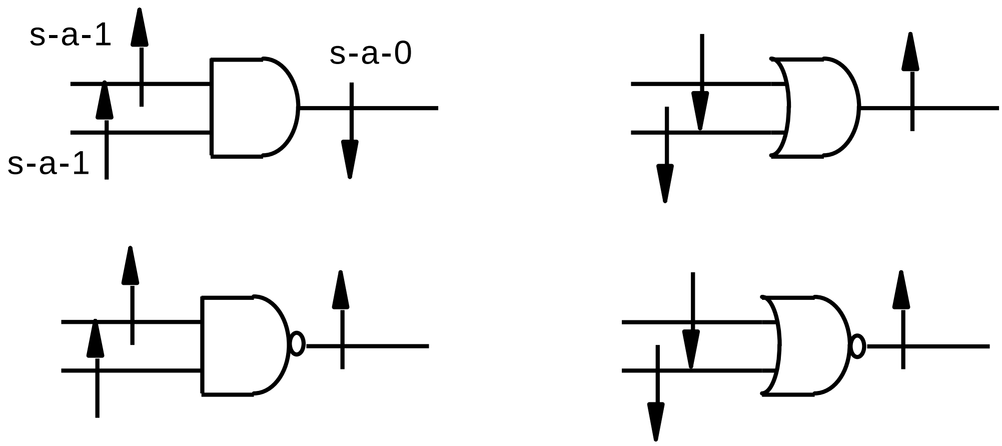
- Fault Equivalence Class
  - (A s-a-0, B s-a-0, C s-a-0)
- Fault Dominance Relations
  - (C s-a-1 > A s-a-1) and (C s-a-1 > B s-a-1)
- Faults that can be ignored:
  - A s-a-0, B s-a-0, and C s-a-1



AB	C	A sa1	B sa1	C sa1	A sa0	B sa0	C sa0
00	0			1			
01	0	1		1			
10	0		1	1			
11	1				0	0	0

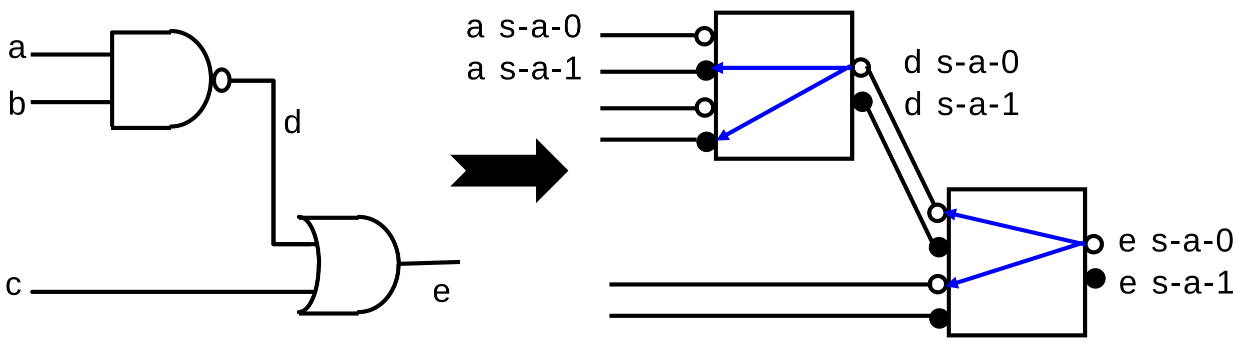
Fault Collapsing

- Collapse faults by fault equivalence and dominance
  - For an *n*-input gate, we only need to consider *n+1* faults in test generation

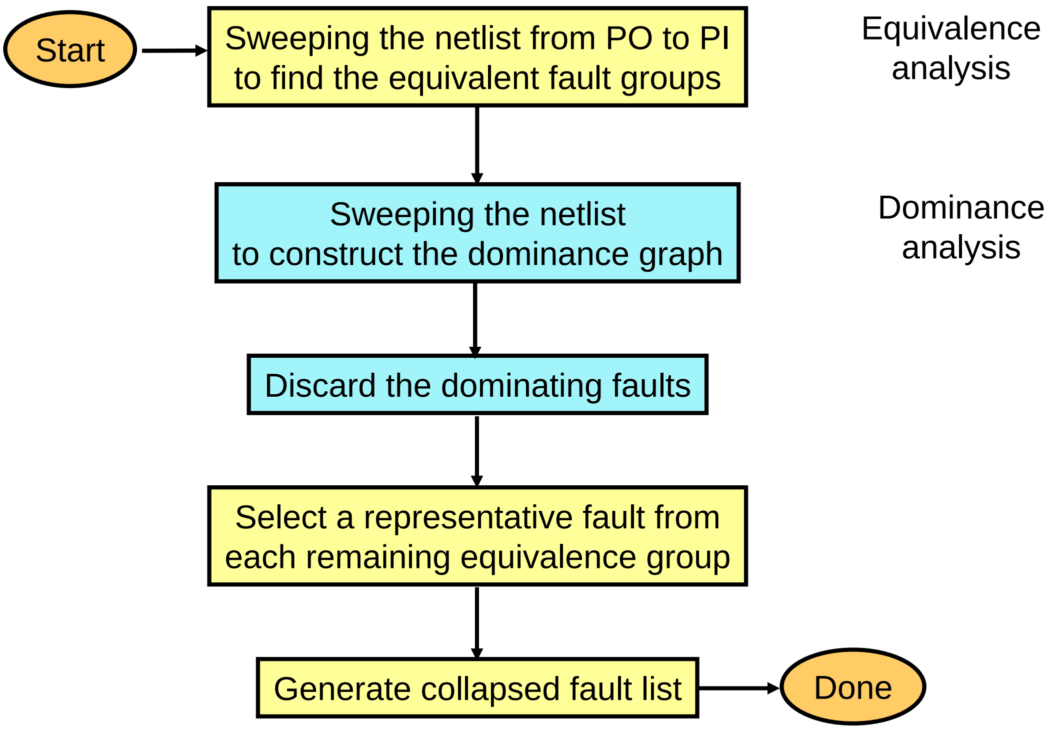


Dominance Graph

- Rule
  - When fault α dominates fault β, then an arrow is pointing from α to β
- Application
  - Find out the transitive dominance relations among faults



Fault Collapsing Flow



Prime Fault

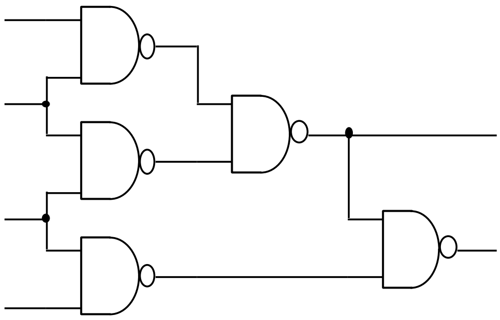
- α is a prime fault if every fault that is dominated by α is also equivalent to α

- Representative Set of Prime Fault (RSPF)
  - A set that consists of exactly one prime fault from each equivalence class of prime faults
  - True minimal RSPF is difficult to find

Why Fault Collapsing ?

- Save memory and CPU time
- Ease testing generation and fault simulation

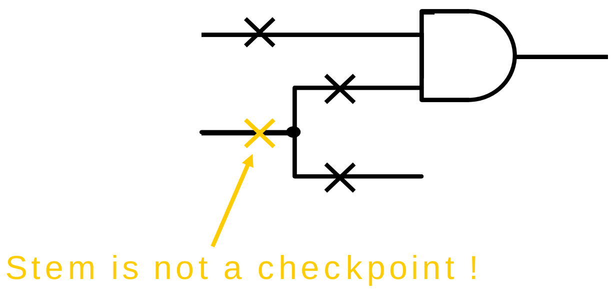
- Exercise



* 30 total faults → 12 prime faults

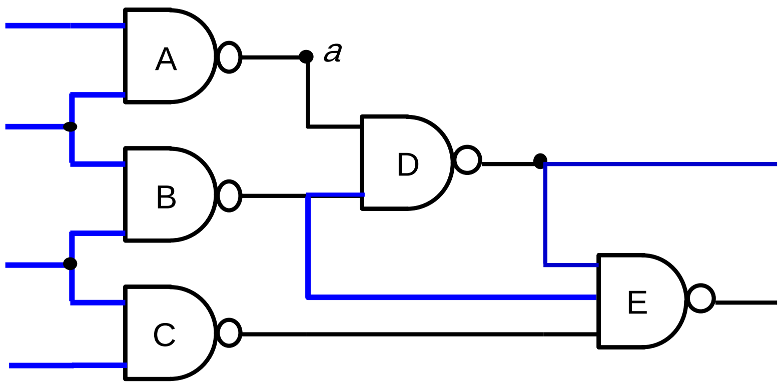
Checkpoint Theorem

- Checkpoints for test generation
  - A test set detects every fault on the primary inputs and fanout branches is complete
    - I.e., this test set detects all other faults, too
  - Therefore, primary inputs and fanout branches form a *sufficient* set of checkpoints in test generation
    - In fanout-free combinational circuits (i.e., every gate has only one fanout), primary inputs are the checkpoints



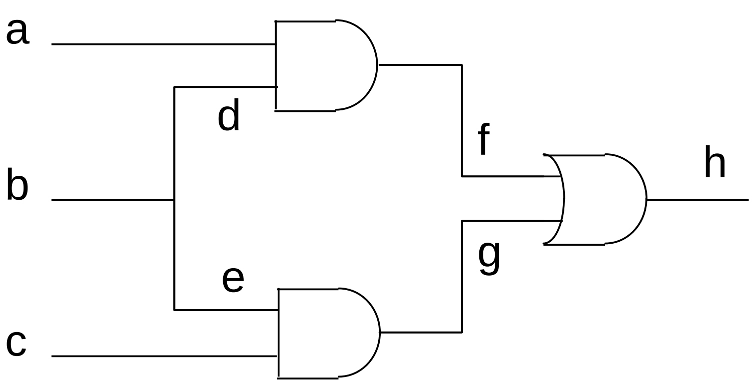
Why Inputs + Branches Are Enough ?

- Example
  - Checkpoints are marked in blue
  - Sweeping the circuit from PI to PO to examine every gate, e.g., based on an order of (A->B->C->D->E)
  - For each gate, output faults are detected if every input fault is detected



Fault Collapsing + Checkpoint

- Example:
  - 10 checkpoint faults
  - a s-a-0 <=> d s-a-0 , c s-a-0 <=> e s-a-0
    b s-a-0 > d s-a-0 , b s-a-1 > d s-a-1
  - 6 faults are enough



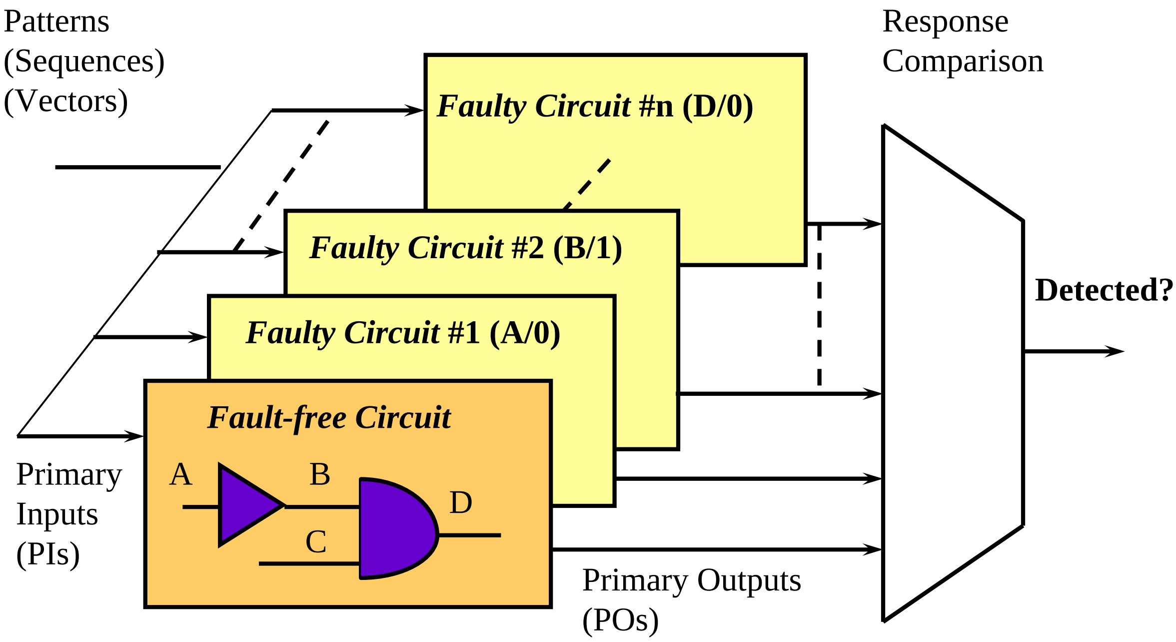
Outline

- Fault Modeling
- Fault Simulation
- Automatic Test Pattern Generation
- Design for Testability

Why Fault Simulation ?

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

Conceptual Fault Simulation

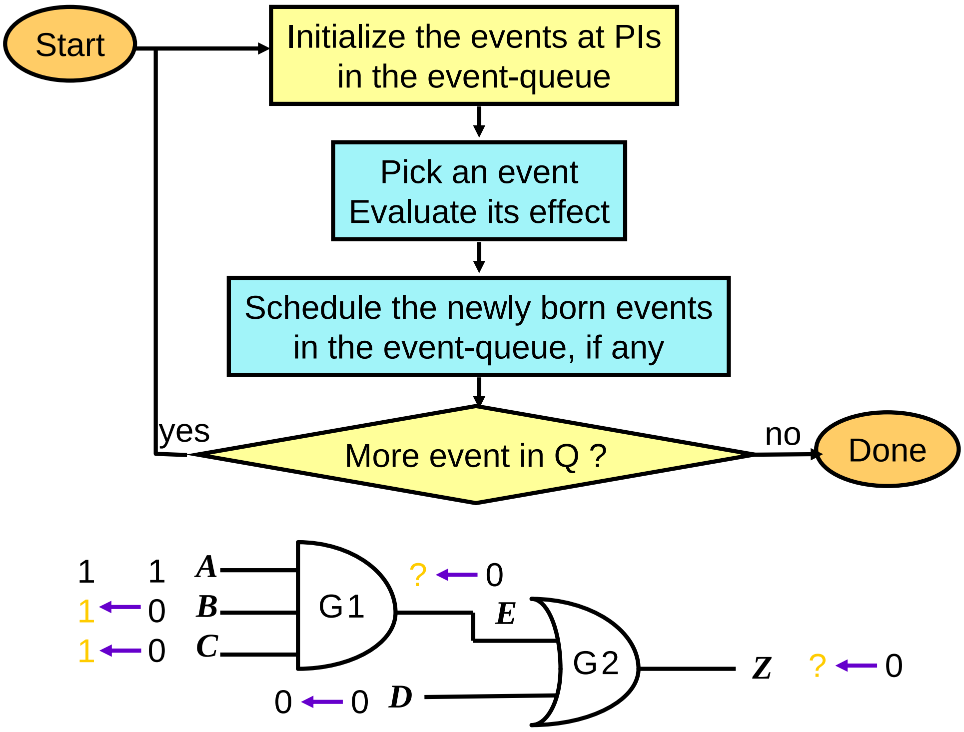


Logic simulation on both good (fault-free) and faulty circuits

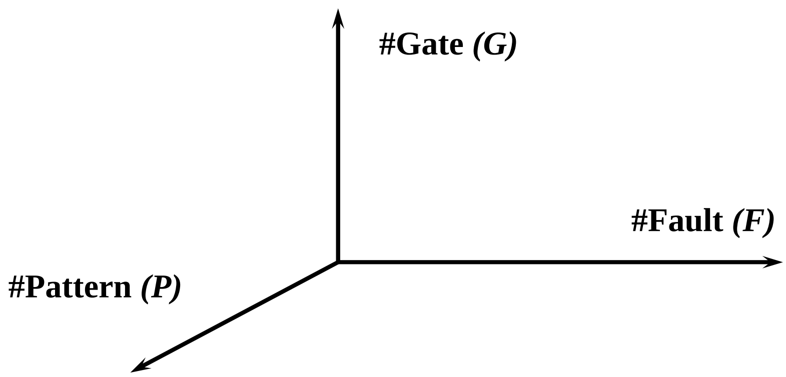
Some Basics for Logic Simulation

- In fault simulation, our main concern is functional faults; gate delays are assumed to be zero unless delay faults are considered
- Logic values can be either {0, 1} (for two-value simulation) or {0, 1, X} (for three-value simulation)
- Two simulation mechanisms:
  - Compiled-code valuation:
    - A circuit is translated into a program and all gates are executed for each pattern (may have redundant computation)
  - Event-driven valuation:
    - Simulating a vector is viewed as a sequence of value-change events propagating from PIs to POs
    - Only those logic gates affected by the events are re-evaluated

Event-Driven Simulation



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 it is usually much lower than ATPG
- The complexity can be greatly reduced using
  - fault collapsing and other advanced techniques