ECE/CS 5745/6745
Testing and Verification of Digital Circuits

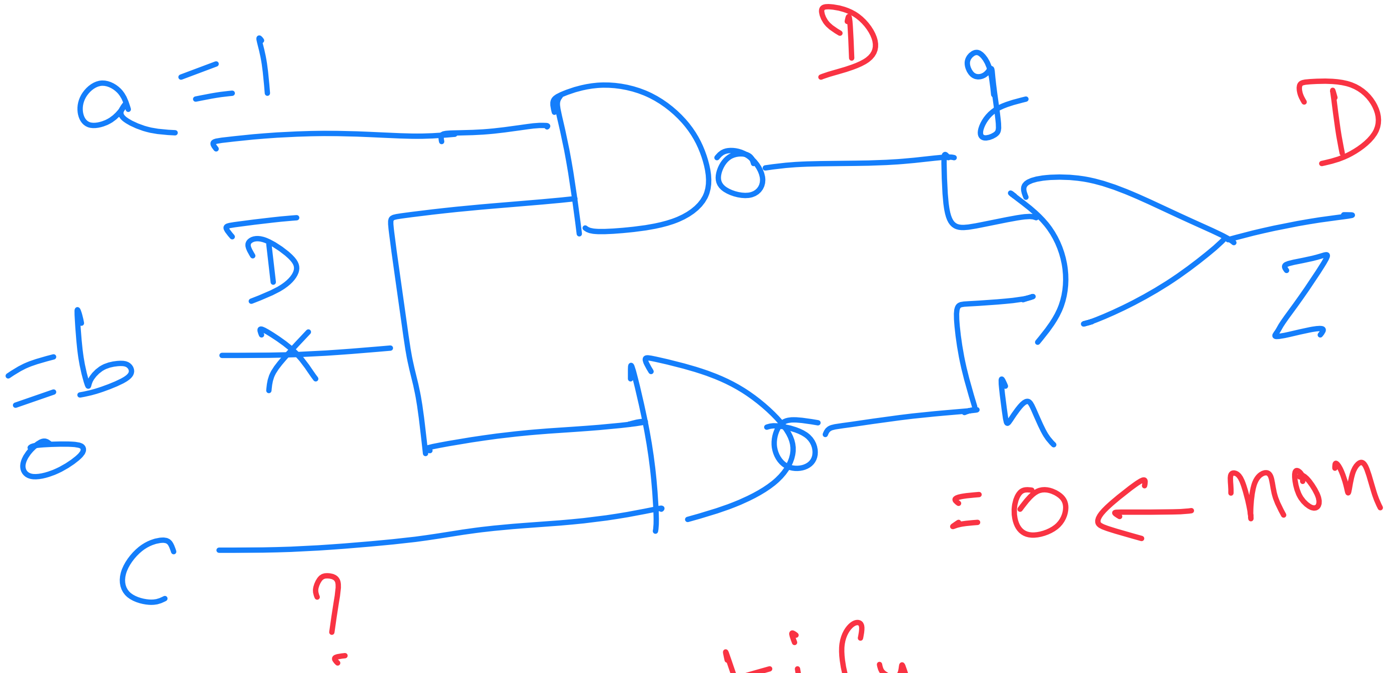
Lecture Slides for ATPG, Redundancy & Test Generation,
Reducing Test Generation Effort, and the Check-Point Theorems

Priyank Kalla
Professor
Electrical & Computer Engineering

ATPG: Path Sensitization

- First Try Single Path Sensitization
  - D-calculus: v/v_f = fault-free/faulty value
  - D=1/0, $\overline{D} = 0/1, \;\; 0 = 0/0, \;\; 1 = 1/1$

$T_{b/1}$



=0 ← non-controling value

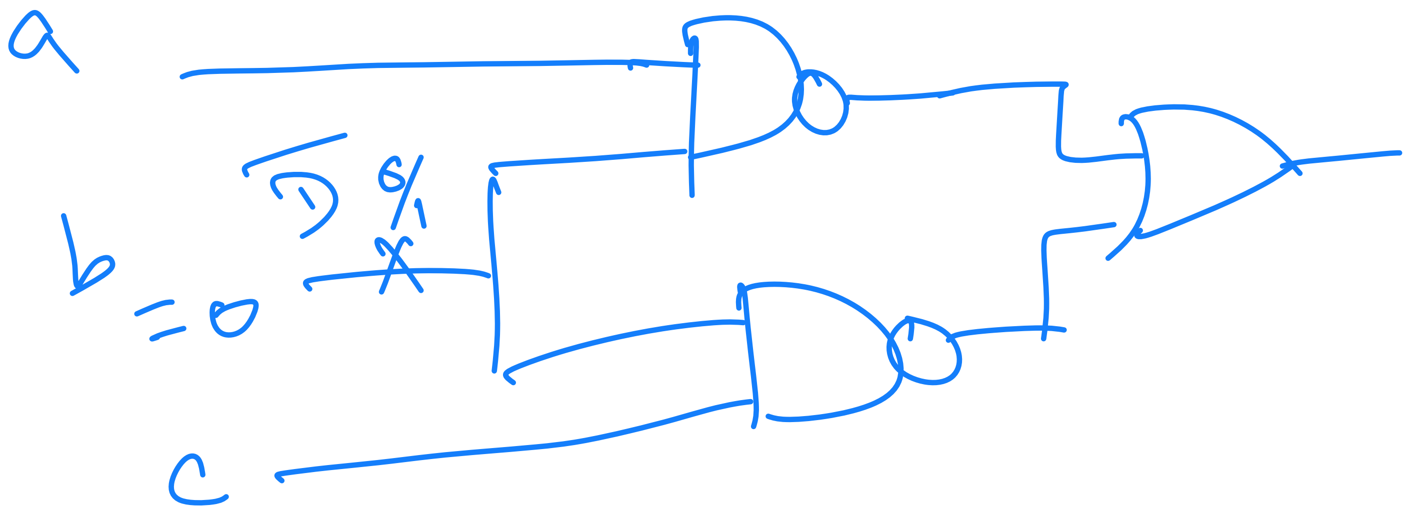
Cannot justify.

b → g → Z

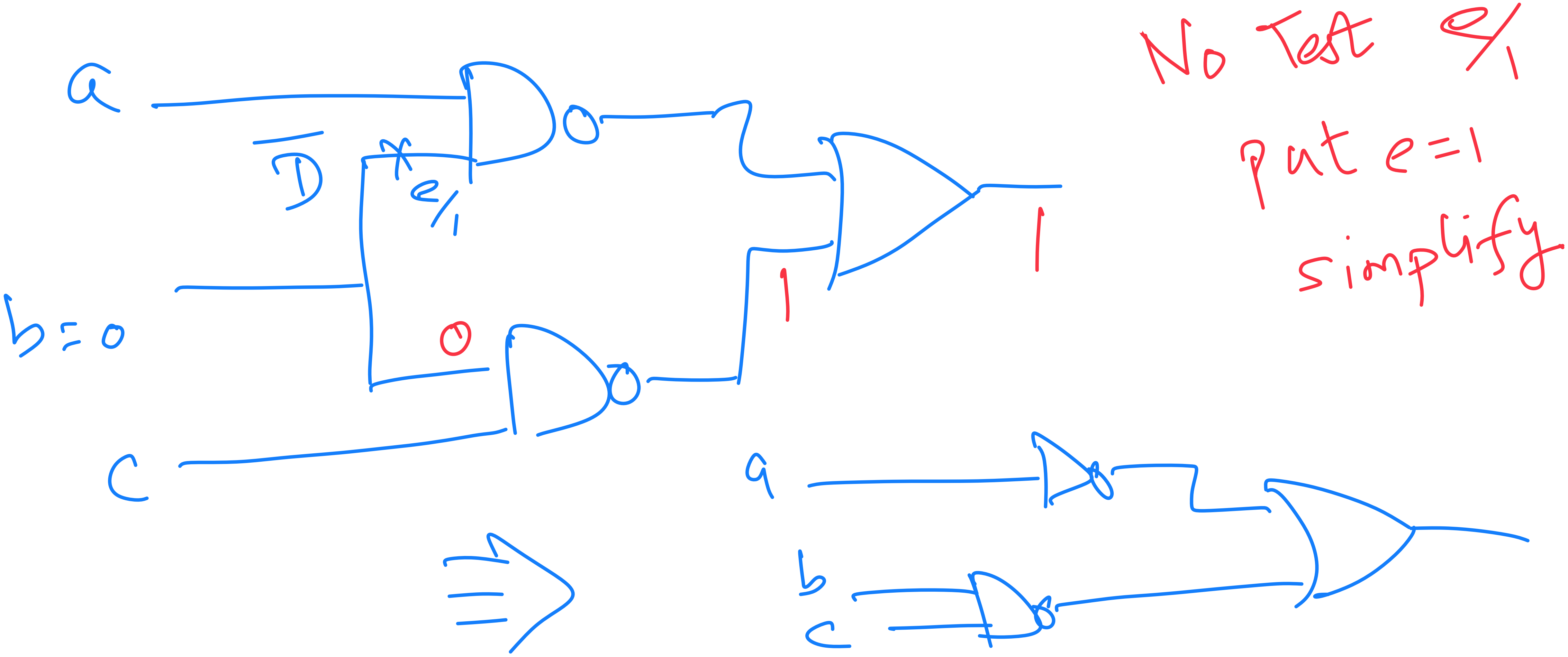
b → h → Z
same problem

ATPG: Path Sensitization

- Now Try Multi Path Sensitization
  - D-calculus: v/v_f = fault-free/faulty value
  - D=1/0, $\overline{D} = 0/1, \;\; 0 = 0/0, \;\; 1 = 1/1$



ATPG: No Test = Redundancy



No Test e/1

put e=1

simplify

- Sometimes single-path sensitization works
- Sometimes multi-path sensitization works
- Sometimes, test does not exist = redundancy
- Redundancy removal via s-a-f propagation

Testing using Boolean Difference

- Let x be a PI, Z the PO. Prove that Test for x/1 $T_{x/1} = \bar{x} \cdot \frac{\delta Z}{\delta x} = \bar{x} \cdot [Z_x \oplus Z_{x'}]$
- Fault-free function Z, faulty function $Z_f = Z(x = 1) = Z_x$
- $T_{x/1} = \underline{Z \oplus Z_x}$ = miter between faulty and fault-free circuit

Shannon's Expansion: $Z = x Z_x + \bar{x} Z_{\bar{x}}$

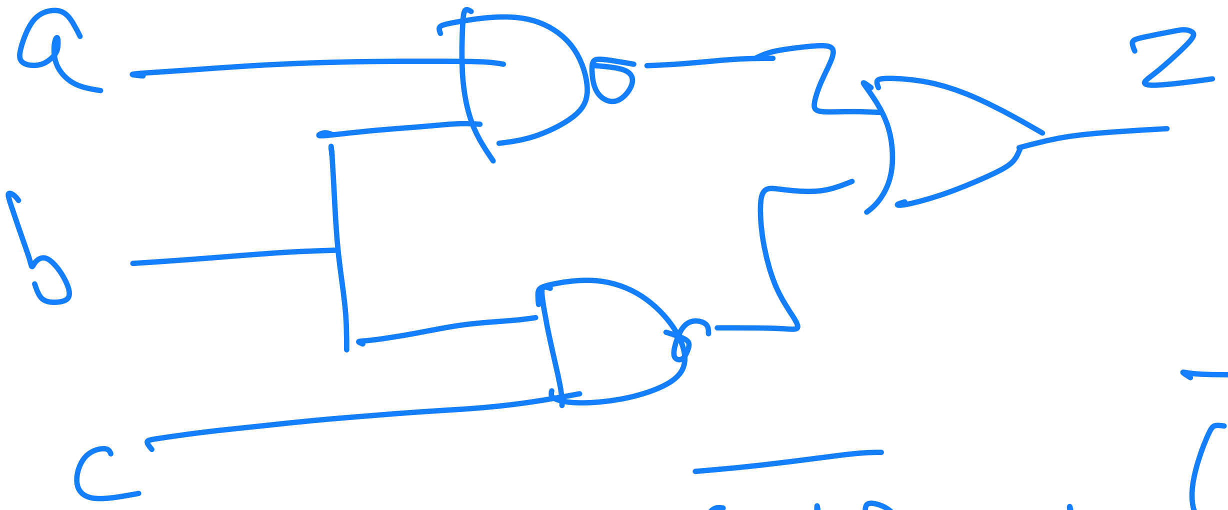
$$T_{x/1} = \underline{x Z_x} \oplus \bar{x} Z_{\bar{x}} \oplus \underline{Z_x}$$

$$= Z_x (x \oplus 1) \oplus \bar{x} Z_{\bar{x}}$$

$$= \bar{x} Z_x \oplus \bar{x} Z_{\bar{x}}$$

$$= \bar{x} \left[Z_x \oplus Z_{\bar{x}} \right] = \bar{x} \frac{\partial Z}{\partial x}$$

$$T_{x/0} = x \cdot \partial Z / \partial x$$



$T_{b/0}: b \cdot \frac{\partial z}{\partial b}$

$$Z = \overline{(ab)} + \overline{(bc)}$$

$$\frac{\partial z}{\partial b} = Z_b \oplus Z_{\bar{b}} = \left[\overline{(a \cdot 1)} + \overline{(1 \cdot c)}\right] \oplus \left[\bar{0} + \bar{0}\right]$$

$$= (\bar{a} + \bar{c}) \oplus 1 = \overline{(\bar{a} + \bar{c})} = ac$$

$T_{b/0}: b \cdot ac$

$T_{b/1}: \bar{b} \cdot ac$

Testability of Two-Level Logic Circuits

- Two level logic = sum of product forms
- Implementation on Programmable Logic Array (PLA)
- Exact minimization using K-maps
  - Make the cover prime and irredundant
  - Minimize area, minimize delay, also fully testable circuit!

$$f = ab\bar{c} + a\bar{b}c + \bar{a}bc + abc$$

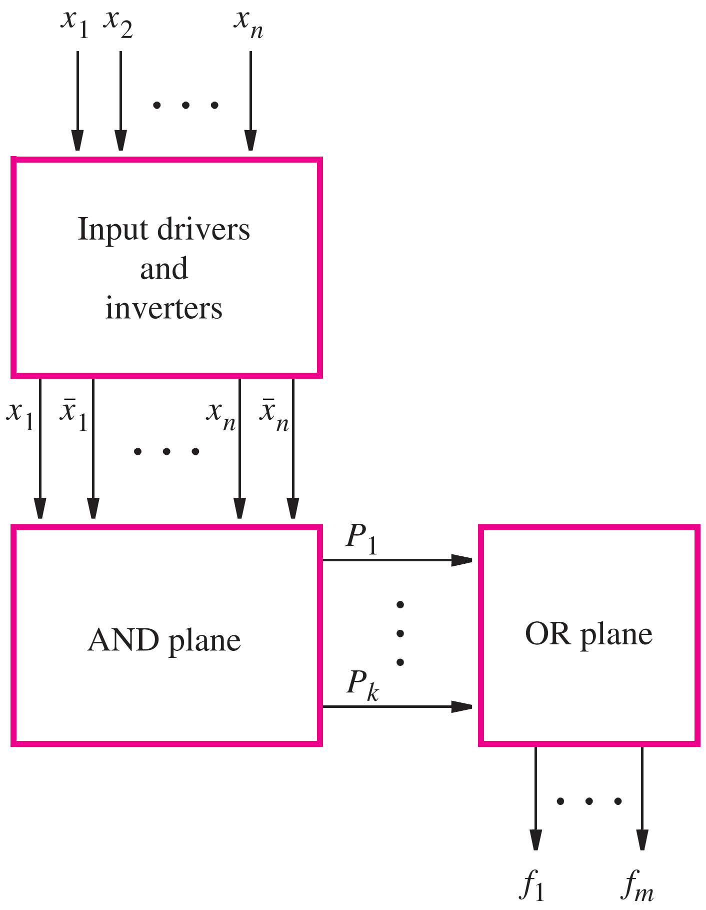
$$= ab + ac + bc$$

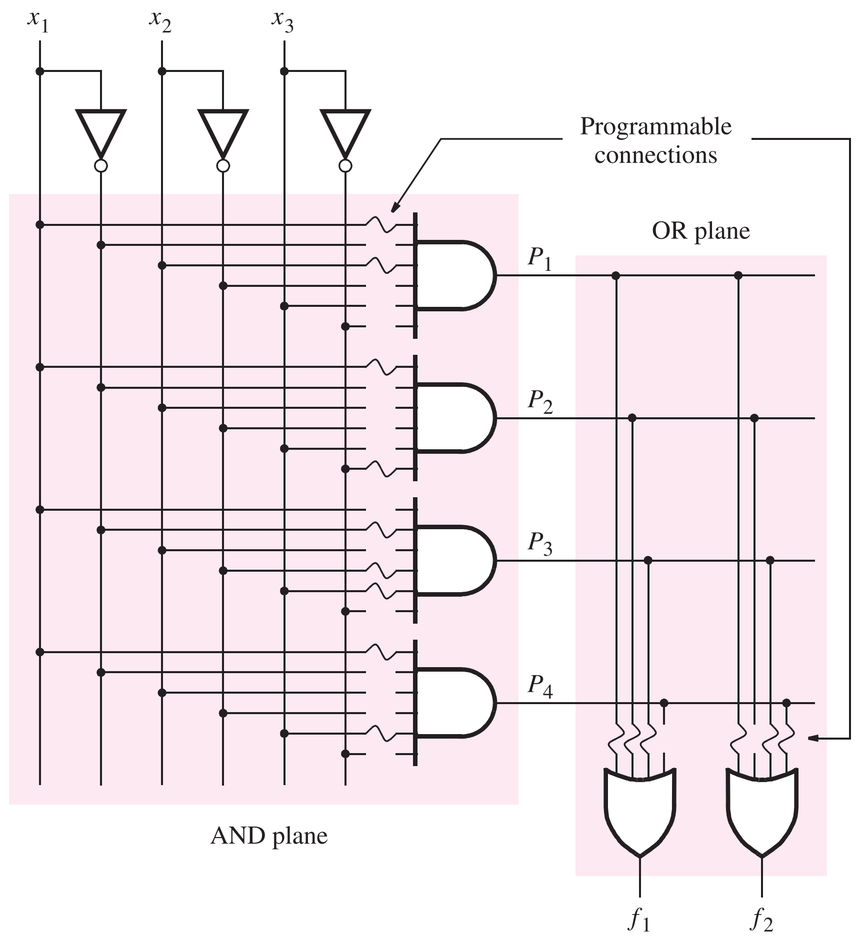
a \ bc	00	01	11	10
0			1	
1		1	1	1

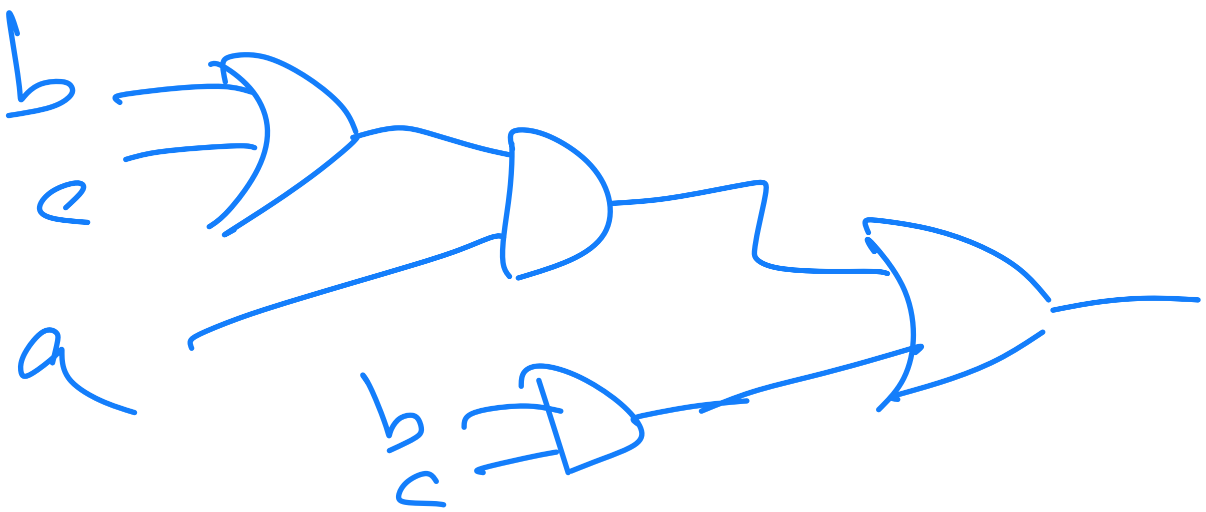
$= a(b+c) + bc$ = factored form

= multi-level ckt.

Structure of PLAs





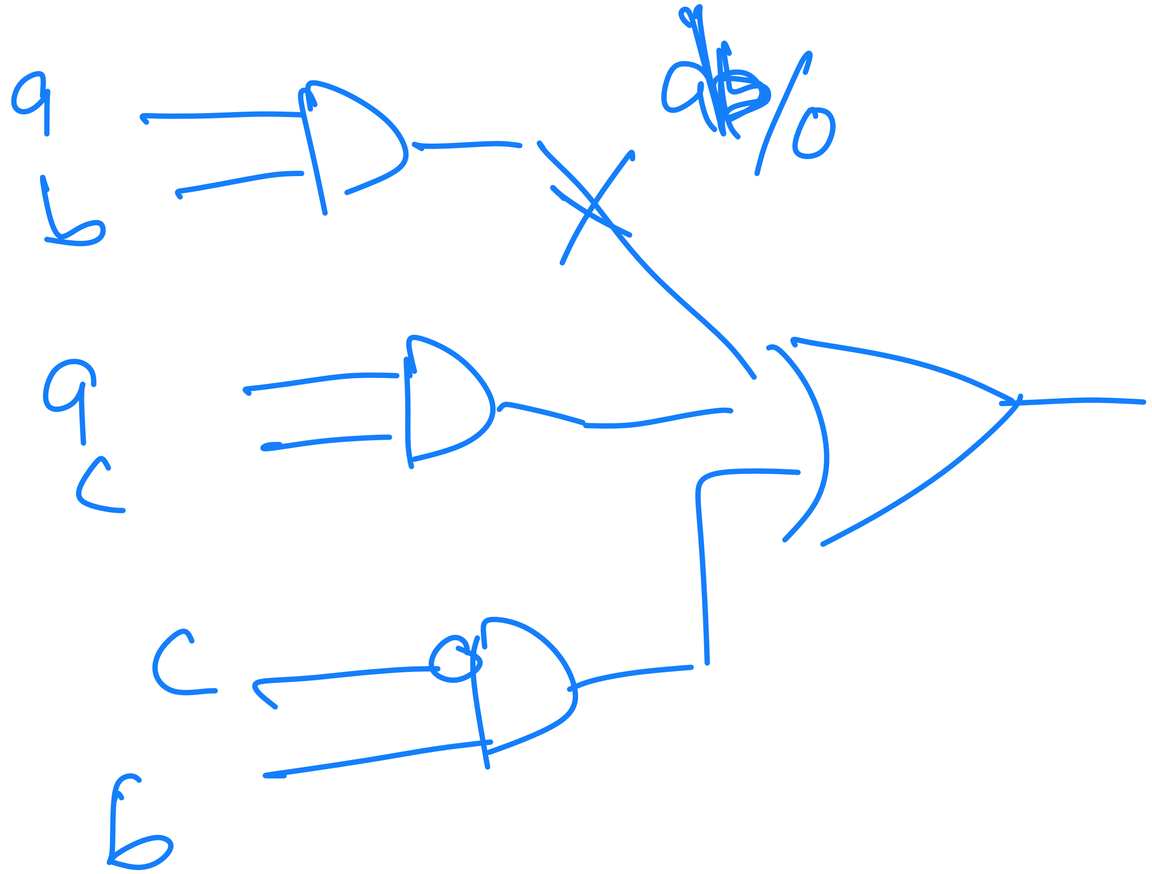


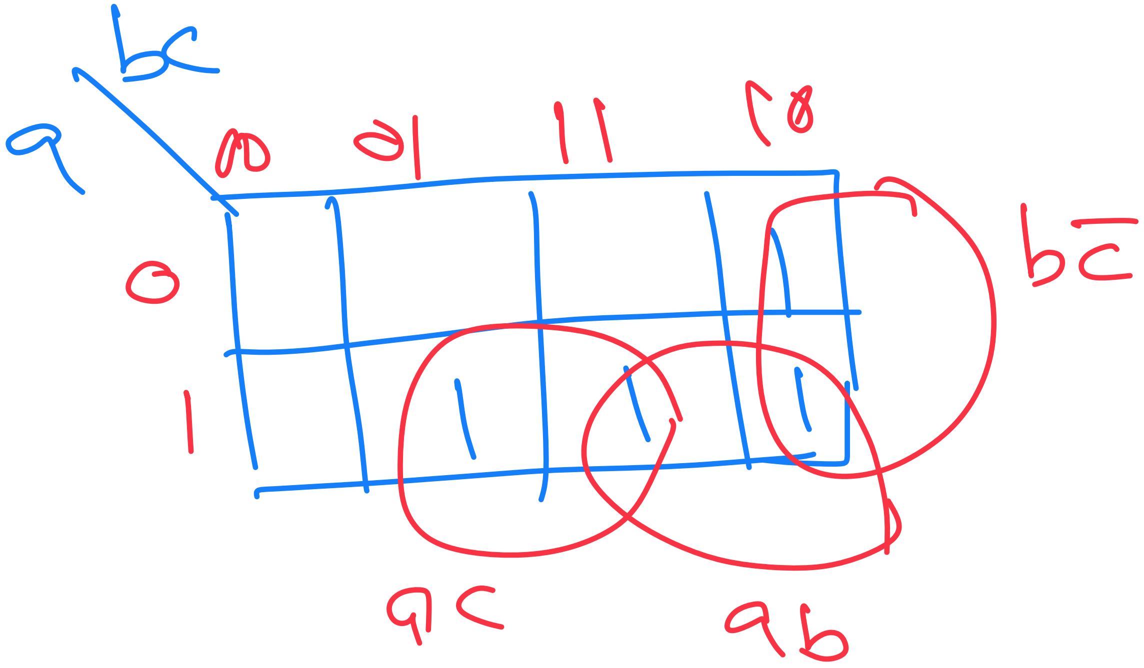
$f = M.L. \neq T.L.$

ATPG: Redundant Cover

- Redundant cube = untestable s-a-0 fault at AND-gate output

$f = ab + ac + b\bar{c}$

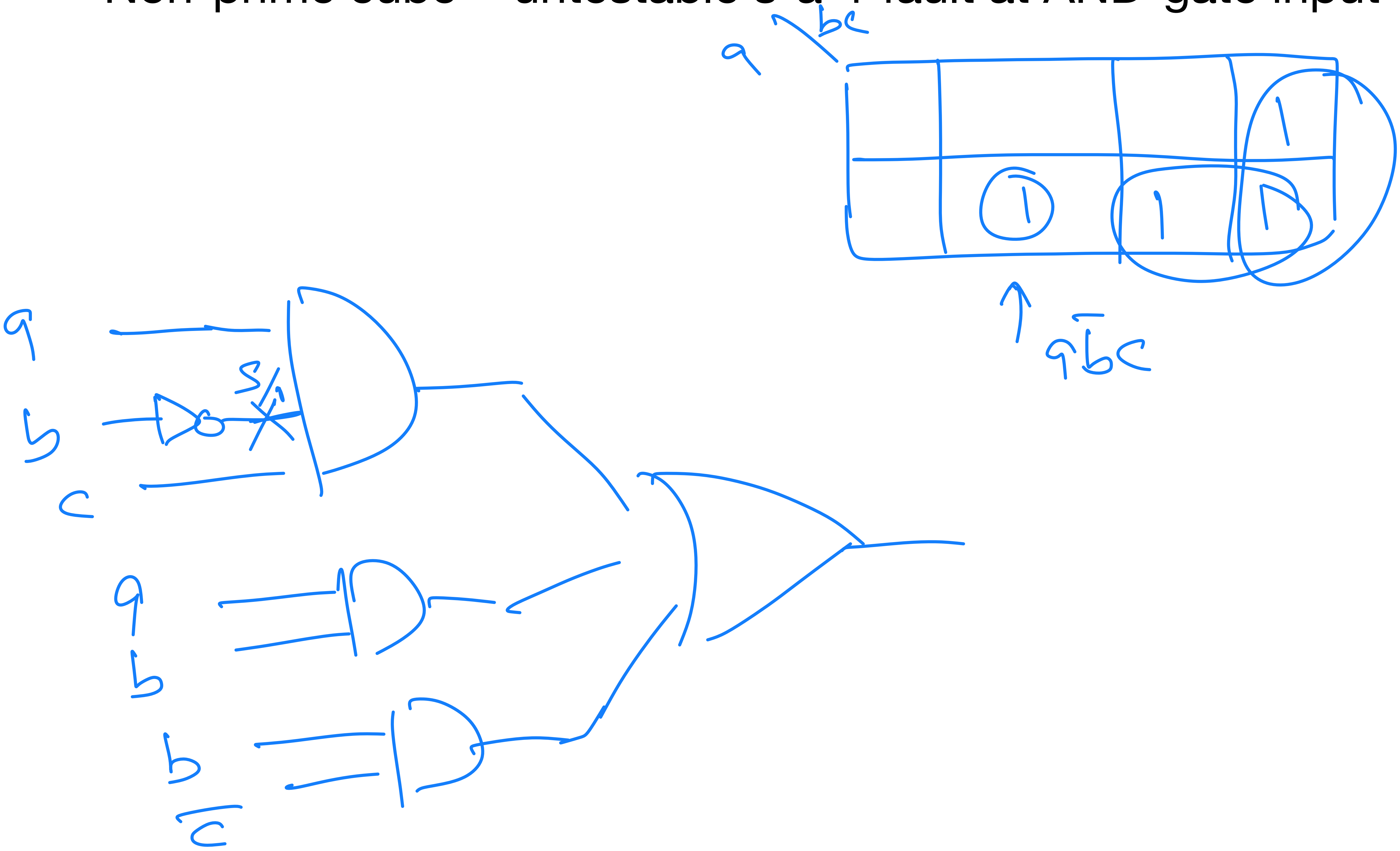




a \ bc	00	01	11	10
0				
1		1	1	1

ATPG: Non-prime Cover

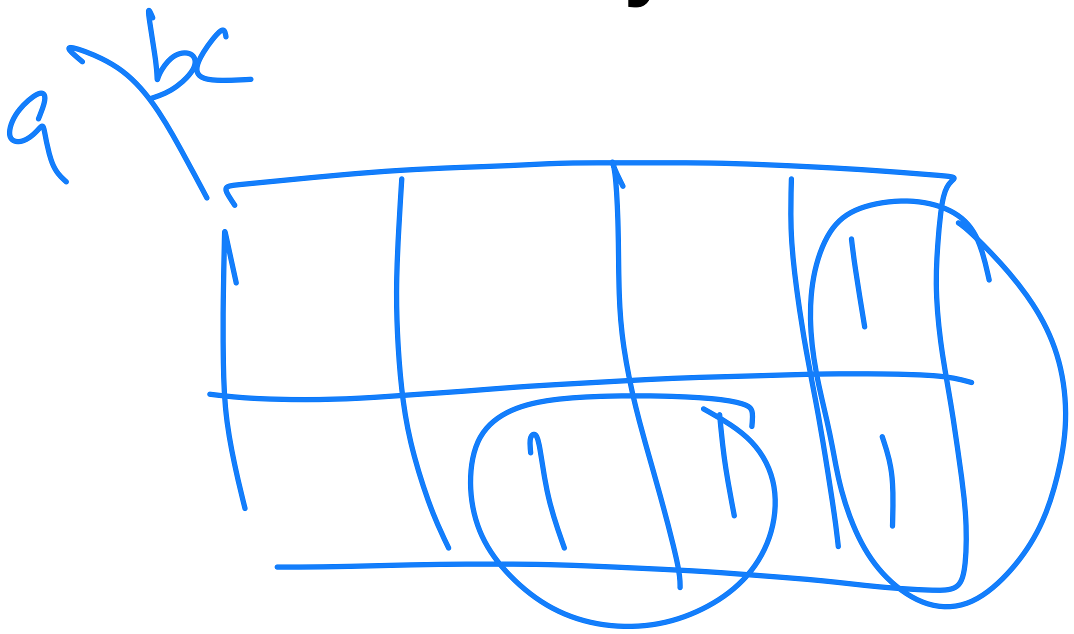
- Non-prime cube = untestable s-a-1 fault at AND-gate input

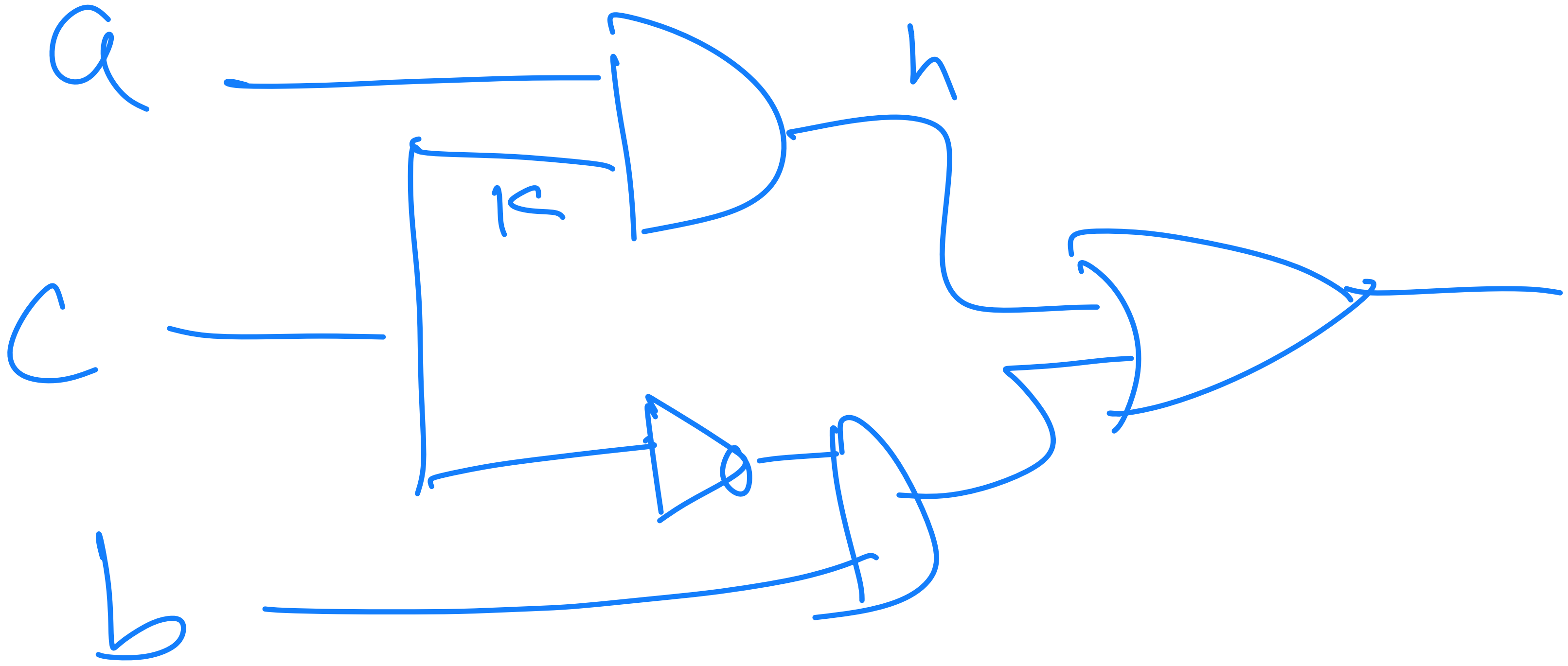


Prime & Irredundant 2-level Circuit = Fully Testable!

$f = ab + ac + b\bar{c}$

$= ac + b\bar{c}$



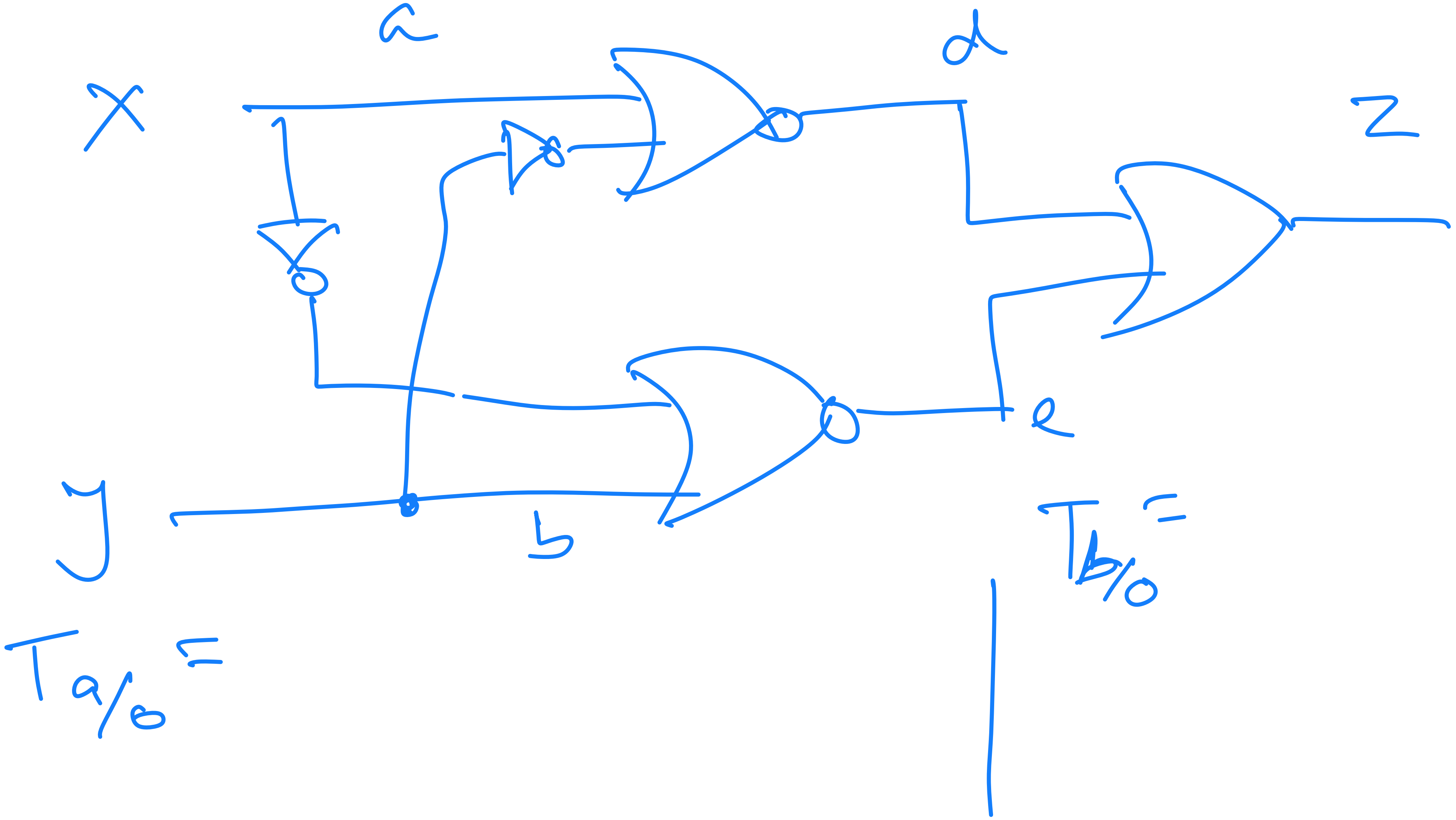


$T_{k/1}$

$T_{h/0}$

Fault Equivalence

- Two faults are functionally equivalent if their effect cannot be distinguished
- Detect only 1 out of ALL equivalent faults
- Infeasible to identify all functionally equivalent faults, so we try a simple problem of structural equivalence

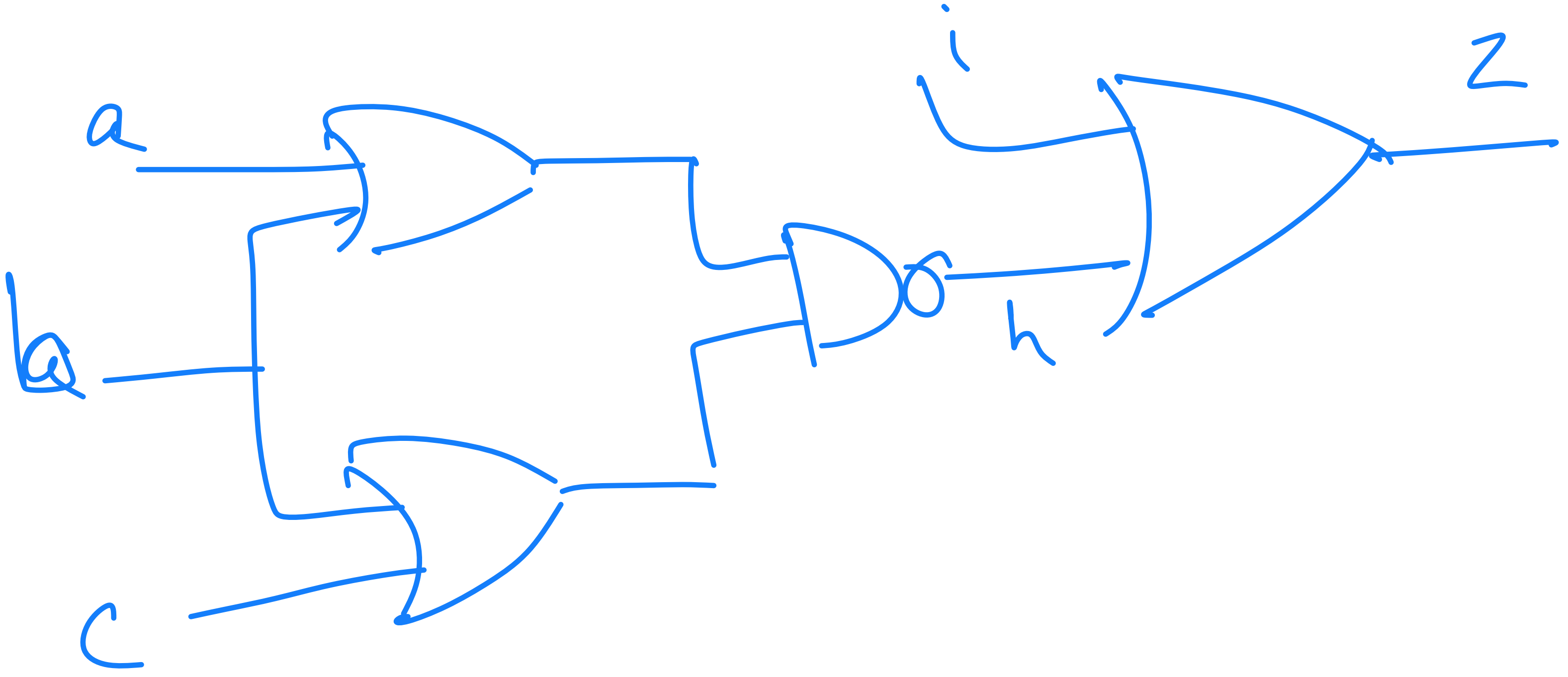


$T_{a/0} =$

$T_{b/0} =$

Fault Equivalence

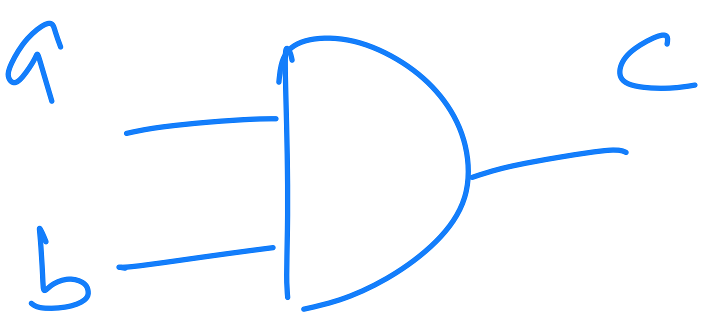
- Two faults are functionally equivalent if their effect cannot be distinguished
- Detect only 1 out of ALL equivalent faults
- Infeasible to identify all functionally equivalent faults, so we try a simple problem of structural equivalence



$T_{b/1}$

$T_{h/0}$

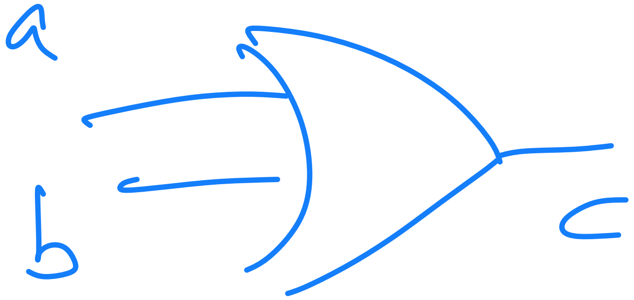
Structural Fault Equivalence



a/0 b/0 c/0

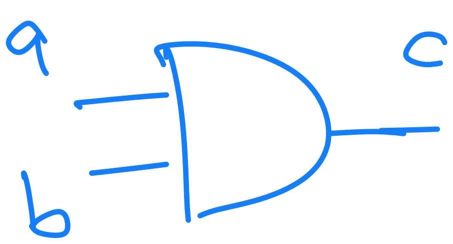
a/1 b/1

c/1

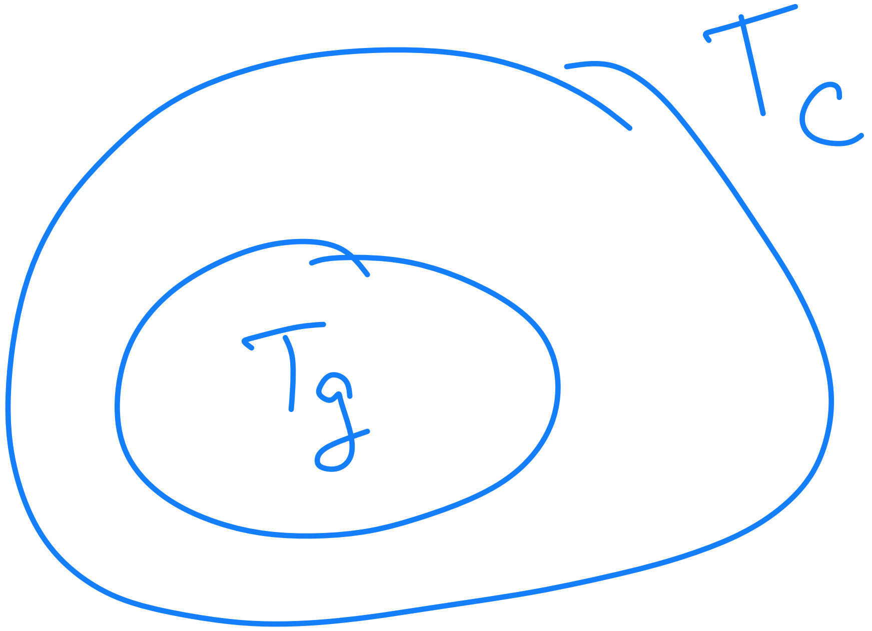


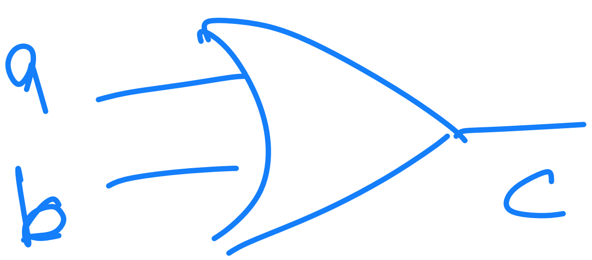
Fault Dominance

- Let T_g be the set of all tests that detect fault g
- Fault f dominates fault g if $Z_f(t) = Z_g(t) \quad \forall t \in T_g$
- Clearly $T_f \supseteq T_g$.
- If the goal is fault detection (and not fault distinguishing/diagnosis), then T_f is not needed, T_g suffices to detect fault f



$c/1 \supseteq a/1, \; b/1$





$c/0 \supseteq a/0, \; b/0$