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

Lecture Slides for ATPG: Reducing Test Generation Effort, the Check-Point Theorems and Issues with Multiple Stuck-Faults

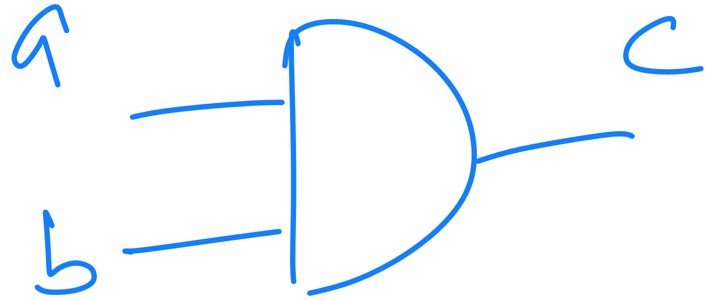
THE UNIVERSITY OF UTAH

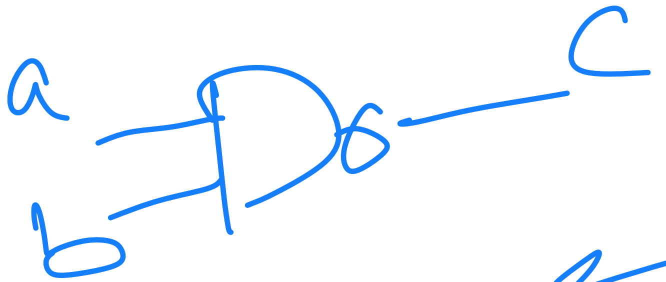
Priyank Kalla
Professor
Electrical & Computer Engineering

Fault Collapsing

- We have to derive s/0, s/1 tests for the entire circuit
    - Each net, including fanout stems and branches
- Primary goal is test detection
- Deriving a test for a fault at each net is too much work
- Thankfully, we can exploit the concept of fault equivalence and fault dominance to reduce the test generation effort
- Analyze fault equivalence and dominance “structurally”
- Equivalent faults cannot be distinguished, so test only one of these from the equivalence class
- Reduce the number of tests required to test the whole circuit

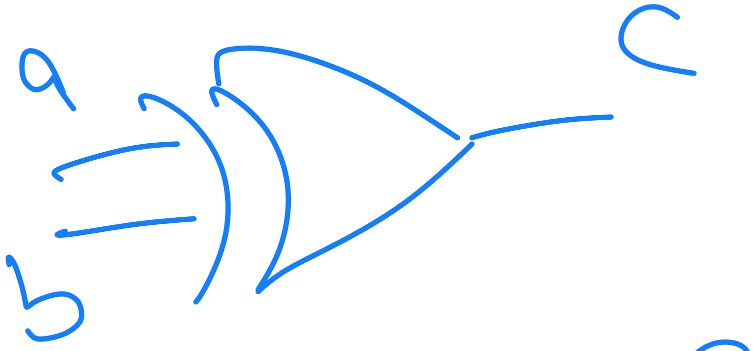
Structural Fault Equivalence





a/0 : <1 1>

c/1 : <1 1>

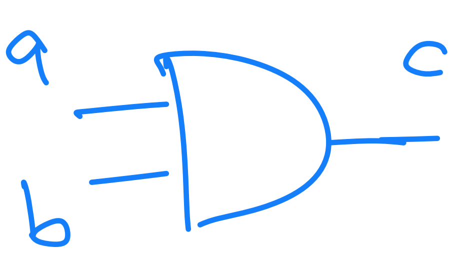


- For an AND gate: a/0 = b/0 = c/0 (equivalent faults)
  - Same test vector, and same fault effect
  - Test only one of these
- NAND gate: a/0 = b/0 = c/1
- OR gate: a/1 = b/1 = c/1
- XOR gate:
  - a/0 implies output c = b; a/1 implies c=b’
  - b/0 implies c = a; b/1 implies c = a’
  - c/0 implies c=0; c/1 implies c = 1
  - No fault equivalence at all

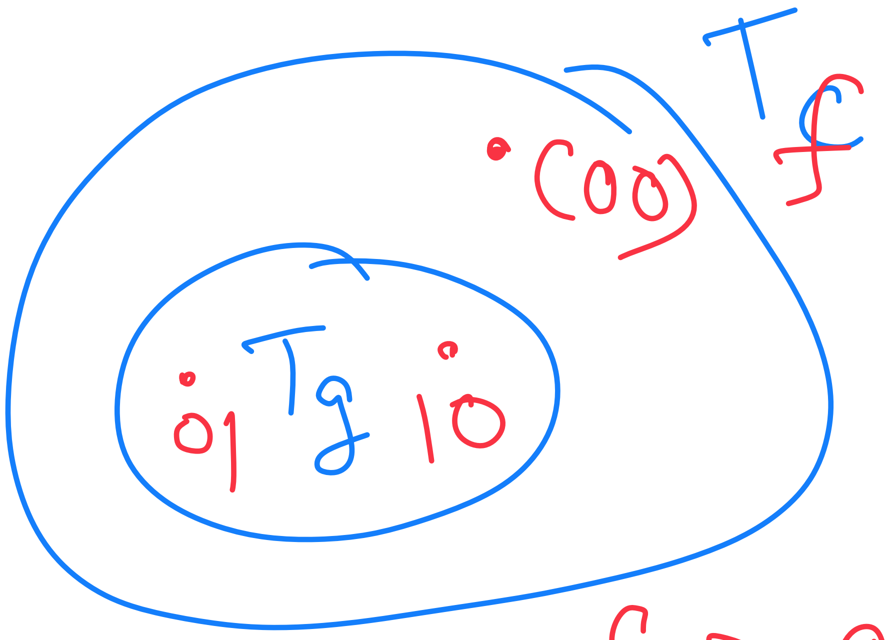
a/0 : a=0, c=b | b/0 : b=0, c=a

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



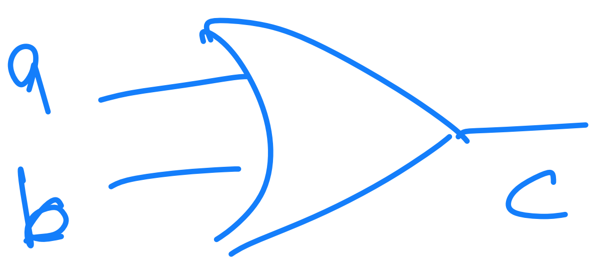
$c/1 \supseteq a/1, b/1$



$a/1 : \langle 01 \rangle \quad b/1 : \langle 10 \rangle$

$c/1 : \langle 00, 01, \underline{10} \rangle$

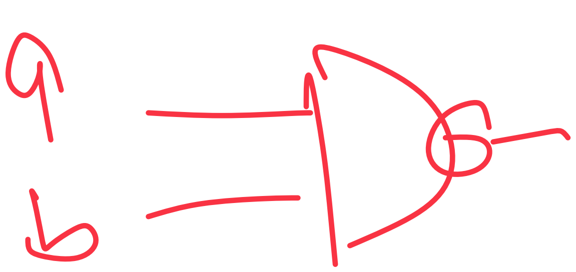
$f \supset g$



$c/0 \supset a/0, b/0$

Fault Dominance for NAND/NOR Gates

- 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



$a/0 = b/0 = c/1$

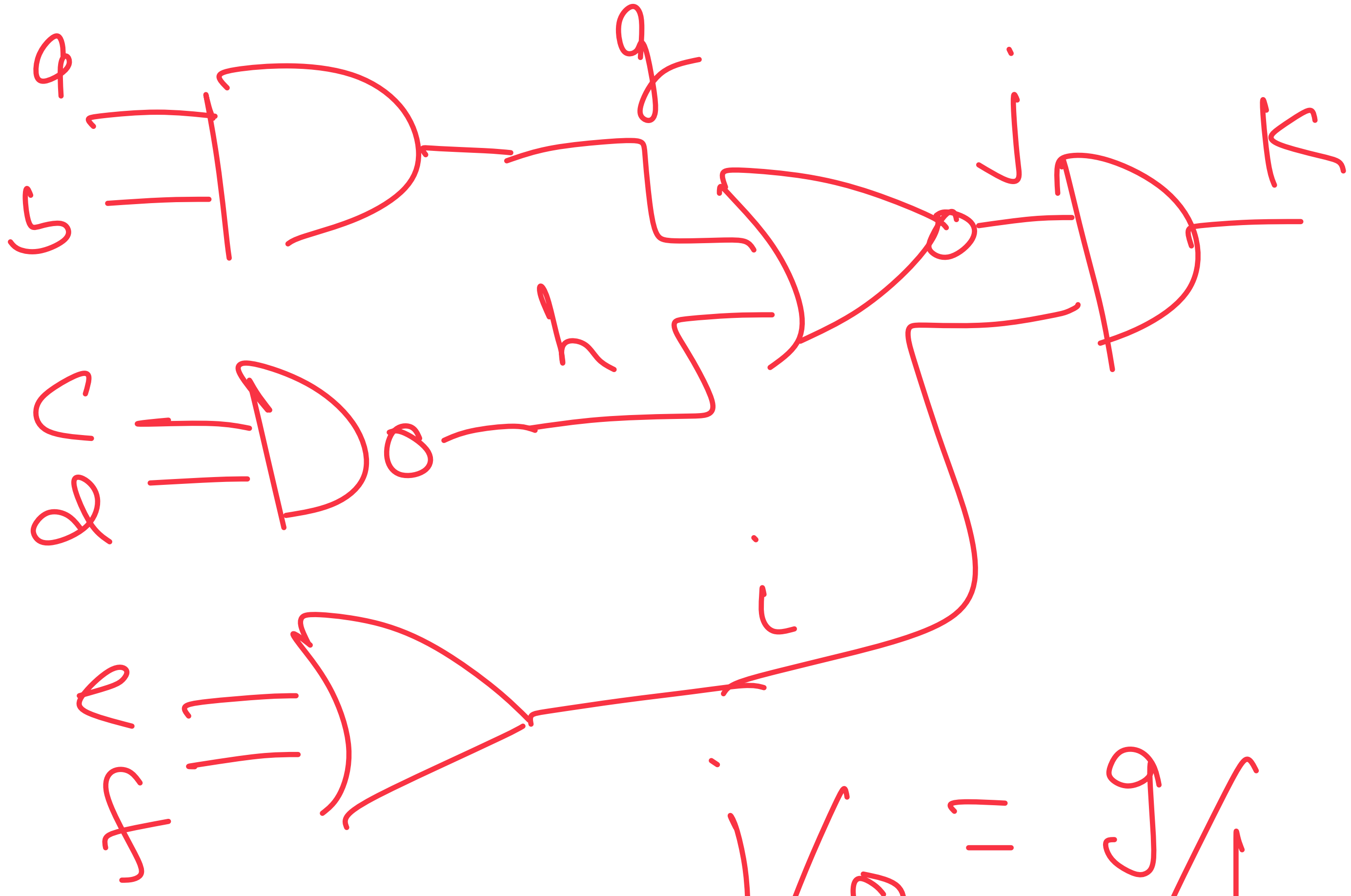
$a/1 = \langle 01 \rangle \qquad b/1 = \langle 10 \rangle$

$c/0 : \langle 00, 01, 10 \rangle$

$c/0 \supset a/1, b/1$

Fault Collapsing on a Fanout-Free Circuit

- If a test set T detects all s/0 and s/1 faults on the primary inputs (PIs) in a fanout-free circuit with AND/OR/INV gates, the T detects all s/0 and s/1 faults in the circuit!
  - Gate output fault is either equivalent to gate input faults, or it dominates gate input faults.
  - Tests for gate output faults need not be derived!



$g/0 = a/0$

$g/1 \supset a/1$

$h/1 = c/0$

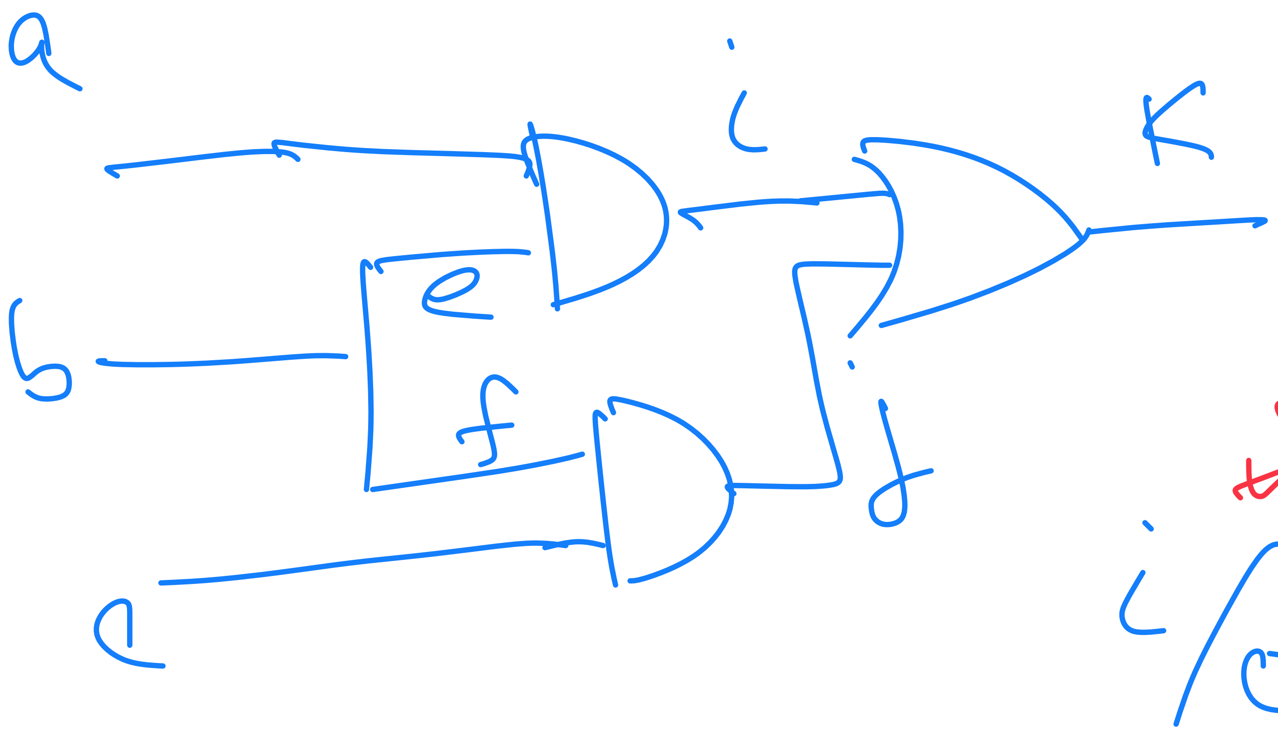
$h/0 \supset c/1$

$j/0 = g/1$

$j/1 \supset d/0$

But What about Fanouts? The Check Point Theorem!

- Check points of a circuit = PIs + fanout branches!
  - Fanout stems = gate outputs or PIs
  - If T detects all checkpoint faults, T detects all single-stuck faults in the circuit



✓ $e/0 \neq b/0$

✓ $e/1 \neq b/1$

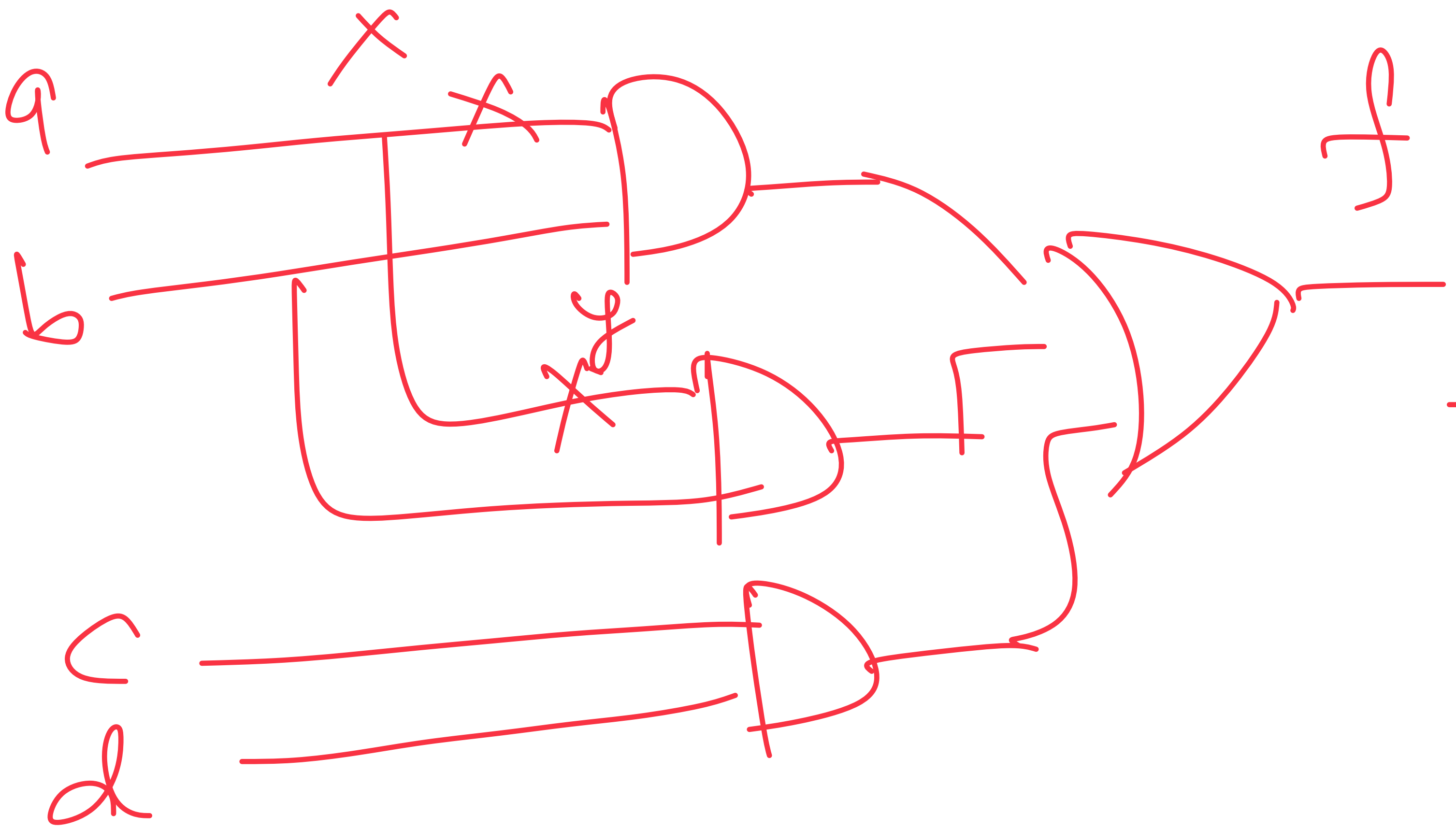
$i/0 = a/0$ ✓

$i/1 \supset a/1$ ✓

But $e/0 = a/0$: chkpts = sufficient not necessary.

Multiple-Stuck Faults

- Case 1: Let a SSF be undetectable. This implies redundancy. A multi-stuck-fault (MSF) under redundancy becomes detectable!



MSF $\{a/0, b/0\}$

SSF x/0 : No test

$a=1, b=1$ $f=1$

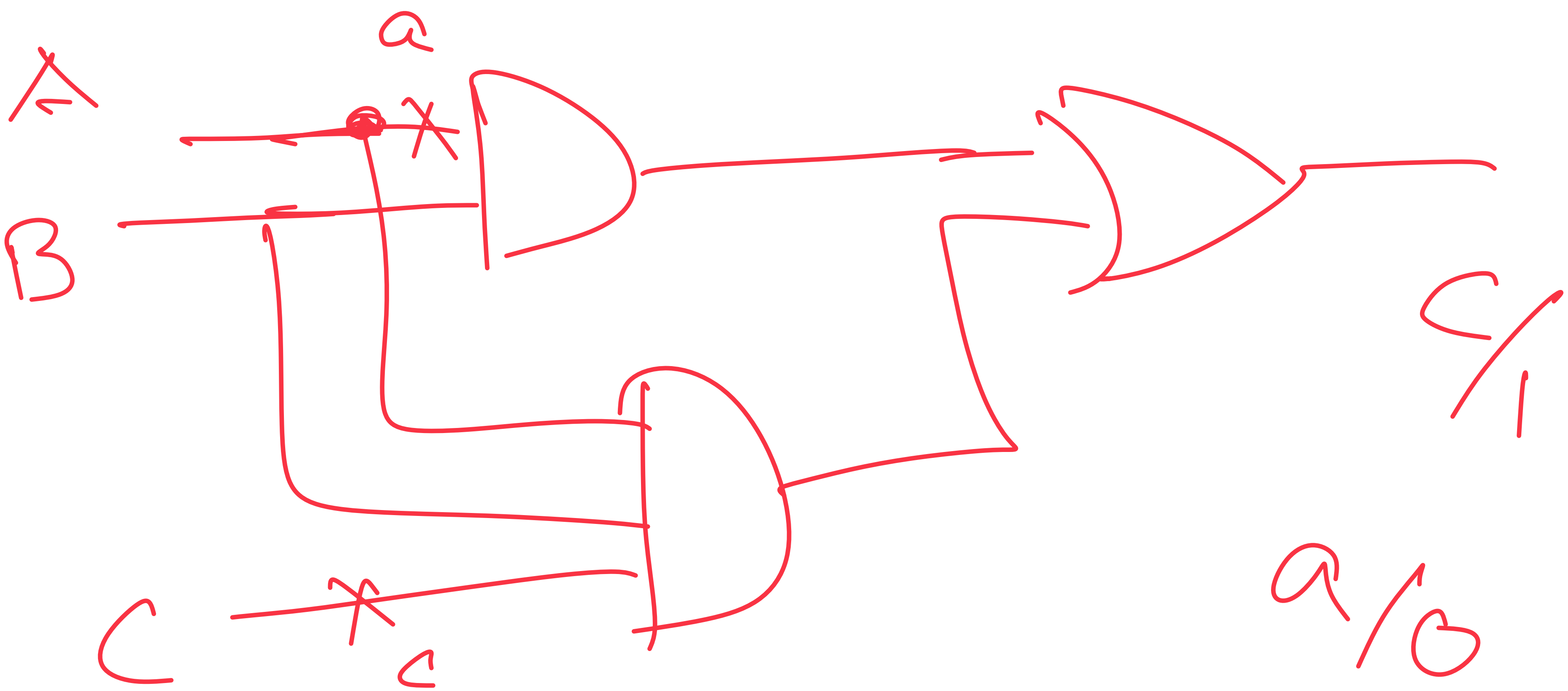
SSF y/0 No test

MSF: $x=D, y=D, \rightarrow f=D$ ✓

$b=1, c=0$

Multiple-Stuck Faults

- Case 2: Let a SSF f be detectable, and another SSF g be undetectable. Then the MSF (f, g) becomes undetectable.
- This is called “test invalidation” in the presence of redundancies. That’s why we prefer to to do SSF tests, under a frequent testing strategy.



c/1 X

a/0 A=1=B
C=0

{c/1, a/0}

No test

ATPG Fault Coverage

- Fault coverage = $F_{cov} = \dfrac{\text{\# of detectable faults}}{\text{\#of total faults}}$
- Fault efficiency of an ATPG tool
    - = $\dfrac{\text{total detectable faults} - \text{aborted faults}}{\text{total faults}}$
- In moderns ATPG tools, fault coverage is very high 95+%
- In the early days, ATPG algorithms D-algorithm, PODEM, FAN. Now a days, SAT solver based ATPG is very efficient
    - Miter model: Fault free (spec), faulty (with a stuck-line) implementation