# Outline: ATPG for stuck at faults

- Deterministic, Fault-Oriented ATG
  - **D-Algorithm**
  - **PODEM**
  - **FAN**
- Random ATG
  - **Weighted random**
  - **RAPS**
- Test Compaction



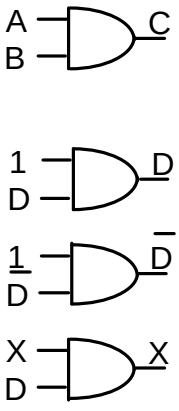
# D-Algorithm

- Roth (1966) proposed a D-algebra and a deterministic ATG algorithm.
  - **D: good value 1 / faulty value 0**
  - **D: good value 0 / faulty value 1**

**5-valued algebra:**

| $V_G / V_F$ | |
|---|---|
| 0/0 | 0 |
| 1/1 | 1 |
| 1/0 | D |
| 0/1 | D |
| -/X,X/- | X |

# D-Algorithm: 5-Valued Operations

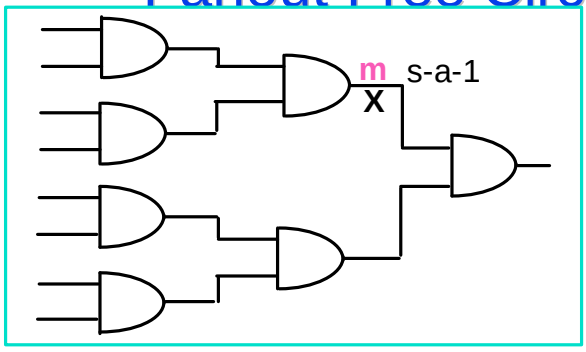



| A | B | C |
|---|---|---|
| 0 | - | 0 |
| - | 0 | 0 |
| 1 | 1 | 1 |
| 1 | X | X |
| X | 1 | X |
| D | 1 | D |
| 1 | $\bar{D}$ | $\bar{D}$ |
| $\bar{D}$ | 1 | $\bar{D}$ |
| D | D | D |
| D | X | X |
| X | D | D |

| AND | 0 | 1 | D | $\bar{D}$ | X |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | D | $\bar{D}$ | X |
| D | 0 | D | D | 0 | X |
| $\bar{D}$ | 0 | $\bar{D}$ | 0 | $\bar{D}$ | X |
| X | 0 | X | X | X | X |



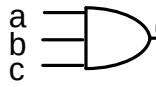
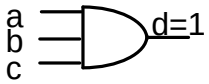
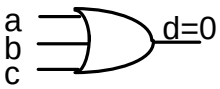
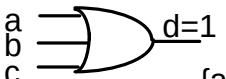
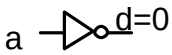
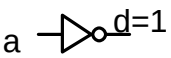
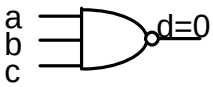
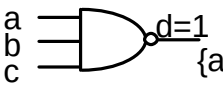
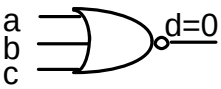
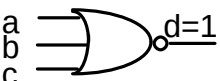
# Fanout-Free Circuit




no conflicts with implications or justifications

- Test for m s-a-1 in a fanout-free circuit
  - **begin**
    - set all lines to X;
    - Justify (m, 0); # activate the fault
    - Propagate (m, D);
  - **end**

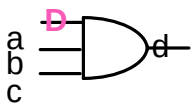
# Justify

| Justify d=0 | Justify d=1 |
|---|---|
|  d=0 select one i of {a,b,c} & justify i=0 |  d=1 justify (a=1) justify (b=1) justify (c=1) |
|  d=0 justify (a=0) justify (b=0) justify (c=0) |  d=1 select one i of {a,b,c} & justify i=1 |
|  d=0 justify (a=1) |  d=1 justify (a=0) |
|  d=0 justify (a=1) justify (b=1) justify (c=1) |  d=1 select one i of {a,b,c} & justify i=0 |
|  d=0 select one i of {a,b,c} & justify i=1 |  d=1 justify (a=0) justify (b=0) justify (c=0) |

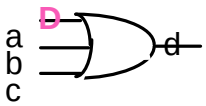


# Propagate (a, D)

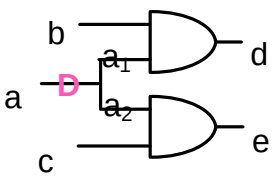
Propagate **D** on line **a**




justify (b=1) and
justify (c=1)

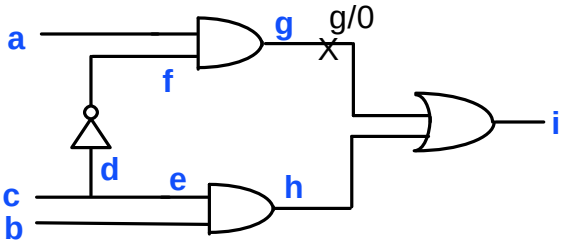



justify (b=0) and
justify (c=0)




select one path i ∈ {$a_1$,$a_2$}
Propagate (i, D)

# Test Generation

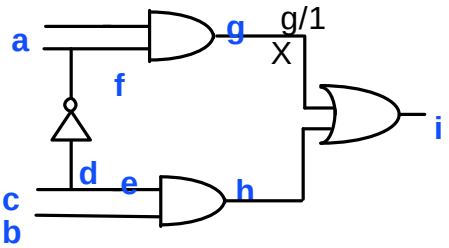



Set g=1
justify (g=1)
    justify (a=1), justify (f=1)
        justify (d=0), --> justify (c=0)
propagate (g, D)
    justify (h, 0)
        --> select one, justify (e=0) --> justify (c=0)



# Test Generation

- What if we make “wrong” selection (decision)?
- What if **justify (a=1)** fails?
- What if **propagate** fails?



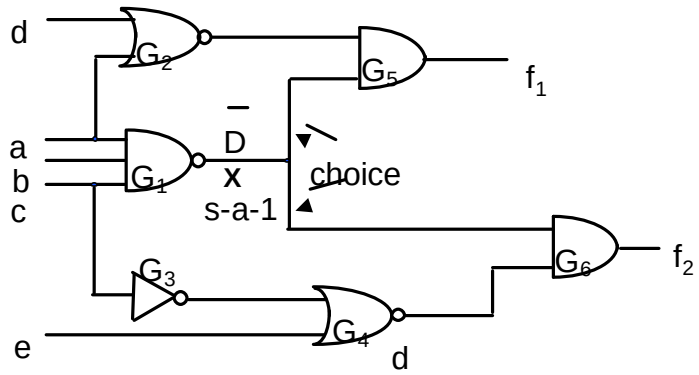

Set g=0
justify (g=0)
    select f --> justify (f=0)
        justify (d=1)
        justify (e=1)
propagate (g, D)
    justify (h, 0)
        select e
            justify (e=0)
            justify (c=0) X

# Backtracking

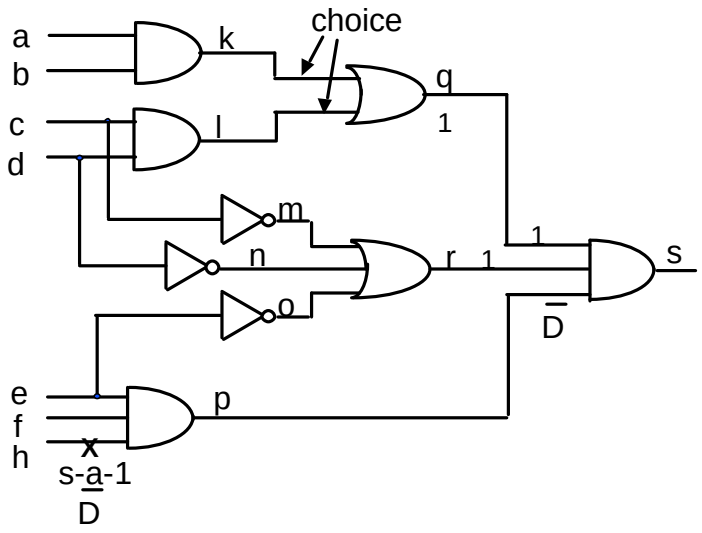
- Conflicts can occur in circuits with fanout and reconvergence.
  - **If a decision causes inconsistency, then we need to backtrack.**
  - **A backtracking strategy is simply a systematic exploration of all decisions (choices).**
  - **Conflict/inconsistency/contradiction**
    - An already-assigned value is different from the value implied by the last decision.
  - **Bounding conditions**
    - There is no D left in the circuit.
    - The fault is not excited.
    - Lookahead indicates that a D cannot propagate.



# Choice in D-Propagation



# Choice in Line Justification






# General Test Generation Algorithm

- Procedure Imply_and_check() implies and checks for inconsistency.
- Procedure Solve() is a generic branch-and-bound procedure.
  - **AND-OR search strategy.**