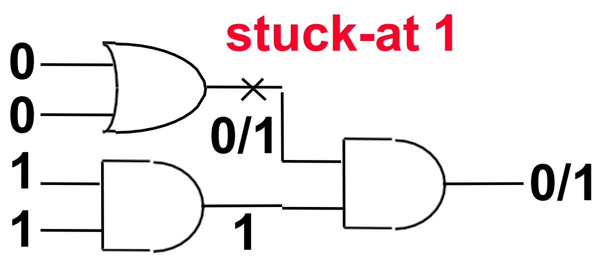
Combinational Test Generation

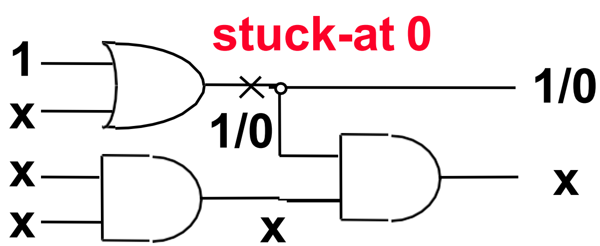
- **Test Generation (TG) Methods**
  - (1) From truth table (2) Using Boolean equation (3) Using Boolean difference (4) From circuit structure
- **TG from Circuit Structure**
  - Common Concepts
  - ATPG Algorithms : D-Algorithm

A Test Pattern

- A test pattern



- A test pattern with don't cares



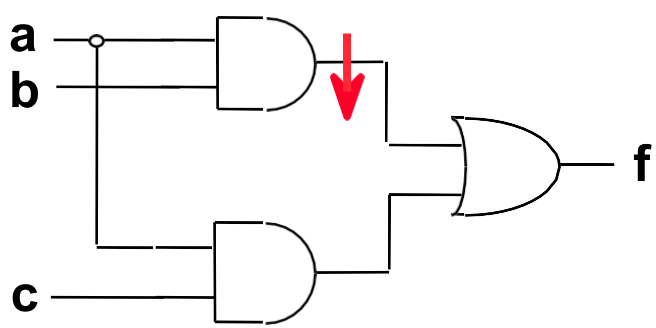
(Good value and faulty value are different at a PO.)

- Test generation: generates a test for a target fault.

Test Generation Methods

(From Truth Table)

Ex: How to generate tests for the stuck-at 0 fault (fault α)?



	abc	f	f_α
	000	0	0
	001	0	0
	010	0	0
	011	0	0
	100	0	0
	101	1	1
√	110	1	0
	111	1	1

Impractical !!

Test Generation Methods
(Using Boolean Equation)

Since $f = ab+ac$, $f_\alpha = ac$ =>

T_α = the set of all tests for fault α

$= \text{ON\_set}(f) * \text{OFF\_set}(f_\alpha) + \text{OFF\_set}(f) * \text{ON\_set}(f_\alpha)$

$= \{(a,b,c) \mid (ab+ac)(ac)' + (ab+ac)'(ac) = 1\}$

$= \{(a,b,c) \mid abc'=1\}$

$= \{ (110) \}$.

High complexity !!

Since it needs to compute the faulty function for each fault.

- *ON_set(f): All input combinations that make f have value 1.*
  OFF_set(f): All input combinations that make f have value 0.

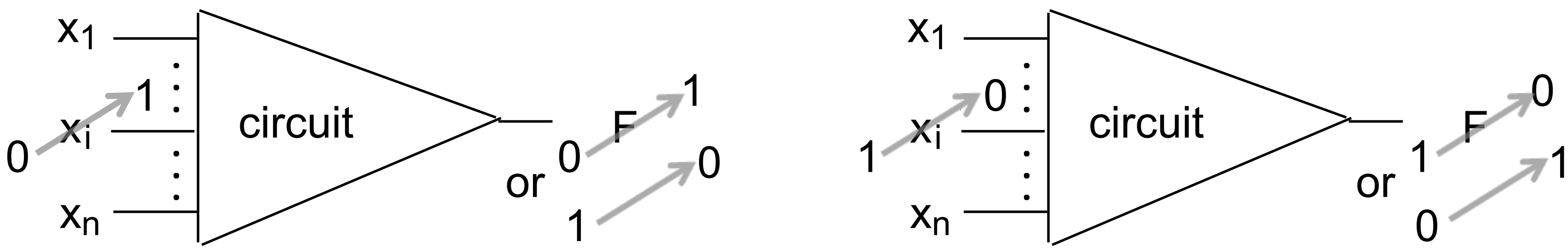
Boolean Difference

- **Physical Meaning of Boolean Difference**
  - For a logic function $F(X)=F(x_1, ..., x_i, ..., x_n)$, find all the input combinations that make the change of value in x_i also cause the change of value in F.
- **Logic Operation of Boolean Difference**
  - The Boolean difference of F(X) w.r.t. input x_i is

$$\frac{dF(X)}{dxi} = F_i(0) \oplus F_i(1) = \overline{F_i(0)} \bullet F_i(1) + F_i(0) \bullet \overline{F_i(1)},$$

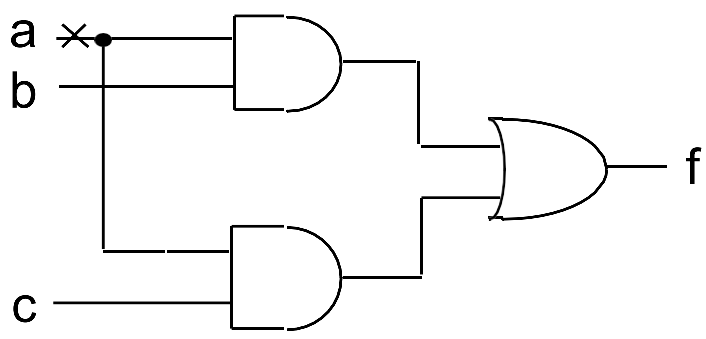
where $F_i(0) = F(x_1, ..., 0, ..., x_n)$ and $F_i(1) = F(x_1, ..., 1, ..., x_n)$.

- **Relationship between TG and Boolean Difference**



Applying Boolean Difference to Test Generation (1/2)

Case 1: Faults are present at PIs.

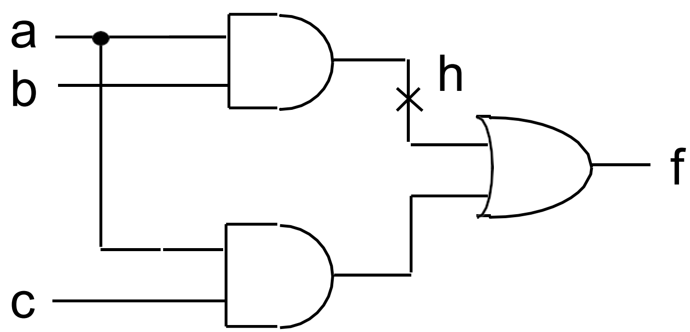


f = ab+ac => $\frac{df}{da} = f_a(0) \oplus f_a(1) = 1 \bullet (b+c)+0 = b+c$

The set of all tests for line *a* s-a-1 is {(a,b,c) | a'$*$(b+c)=1} = {(01x), (0x1)}.
The set of all tests for line *a* s-a-0 is {(a,b,c) | a$*$(b+c)=1} = {(11x), (1x1)}.

Applying Boolean Difference to Test Generation (2/2)

Case 2: Faults are present at internal lines.



f = h+ac, h = ab => $\frac{df}{dh} = f_h(0) \oplus f_h(1) = \overline{ac} \bullet 1 + ac \bullet \overline{1} = \overline{a} + \overline{c}$

The set of all tests for line *h* s-a-1 is

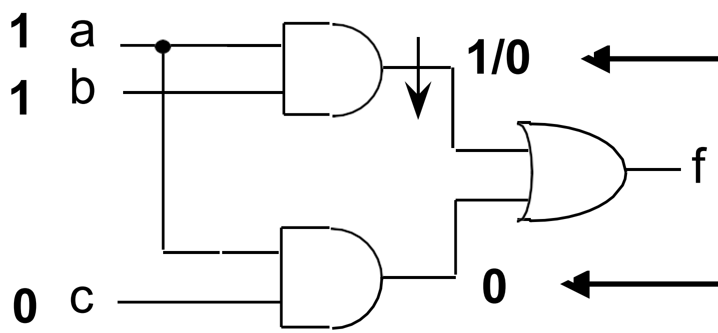
{ (a,b,c)|h'∗(a'+c')=1 } = { (a,b,c)|(a'+b')∗(a'+c')=1 } = { (0xx), (x00) }.

The set of all tests for line *h* s-a-0 is

{(a,b,c)|h∗(a'+c')=1} = {(110)}.

Test Generation Methods

(From Circuit Structure)



- **Two basic goals:**
- **Fault activation(FA)**
  Fault propagation(FP)

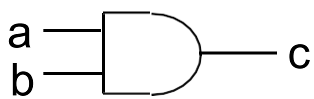
=> Line justification (LJ)

where 1/0 means that the good value is 1 and the faulty value is 0 and is denoted as D. Similarly, 0/1 is denoted as D'.
D and D' are called fault effects (FE).

Common Concepts for Structural TG

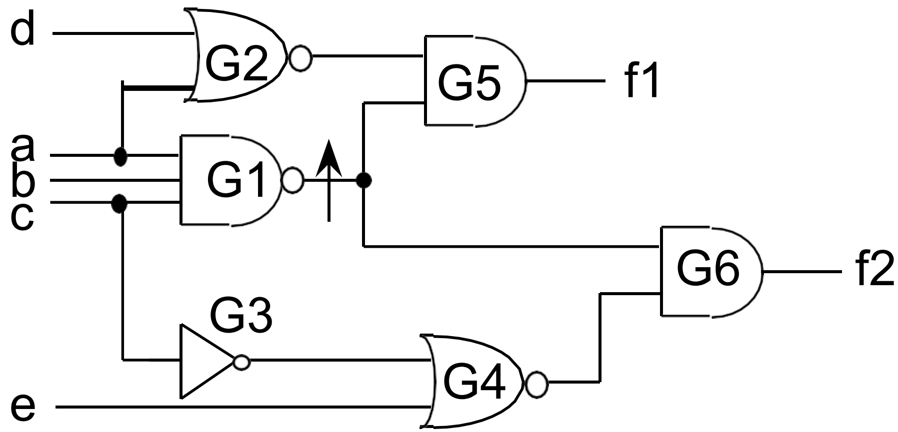
- The FA problem => a LJ problem.
- The FP problem =>
  (1) Select a FP path to a PO => decisions.
  (2) Once the path is selected => a set of LJ problems.
- The LJ problems => decisions or implications .

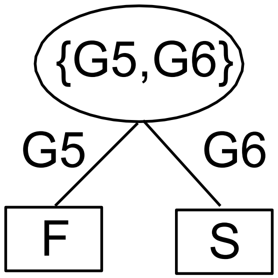
  ex:



  To justify c=1 => a=1 and b=1. (implication)
  To justify c=0 => a=0 or b=0. (need make decisions)
- Incorrect decision => Backtracking => Another decision.
- Once the fault effect is propagated to a PO and all line values to be justified are justified, the test is generated. Otherwise, the decision process must be continued repeatedly until all possible decisions have been tried.

Ex: Decisions When Fault Propagation





The corresponding decision tree

FA => a=1, b=1, c=1 => G1= D', G3=0; FP => through G5 or G6.

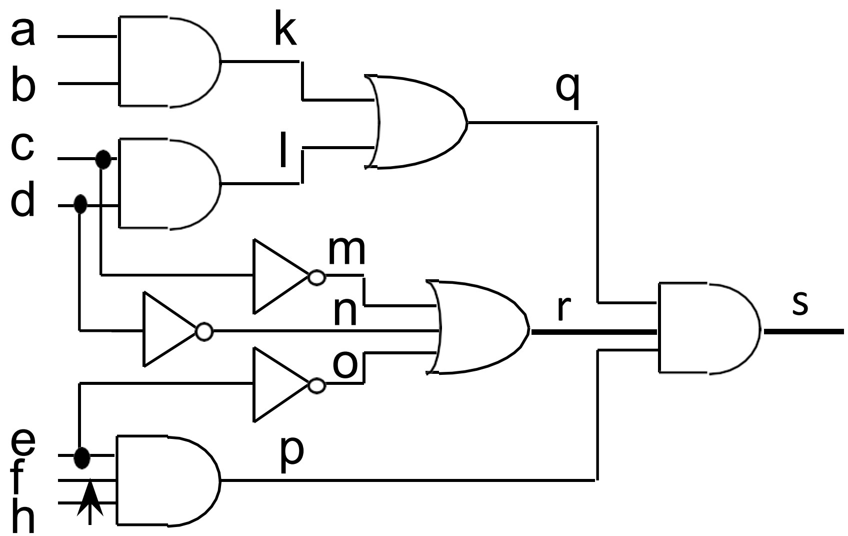
Decision: through G5 => G2=1 => d=0, a=0. => inconsistency => backtracking!!

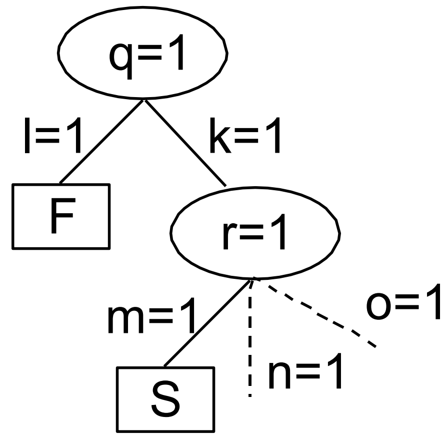
Decision: through G6 => G4=1 => e=0. => done!!

The resulting test is 111x0.

D-frontier*: *The set of all gates whose output value is currently x but have one or more fault signals on their inputs. Ex: Initially, the D-frontier of this example is {G5, G6}.

Ex: Decisions When Line Justification





The corresponding decision tree

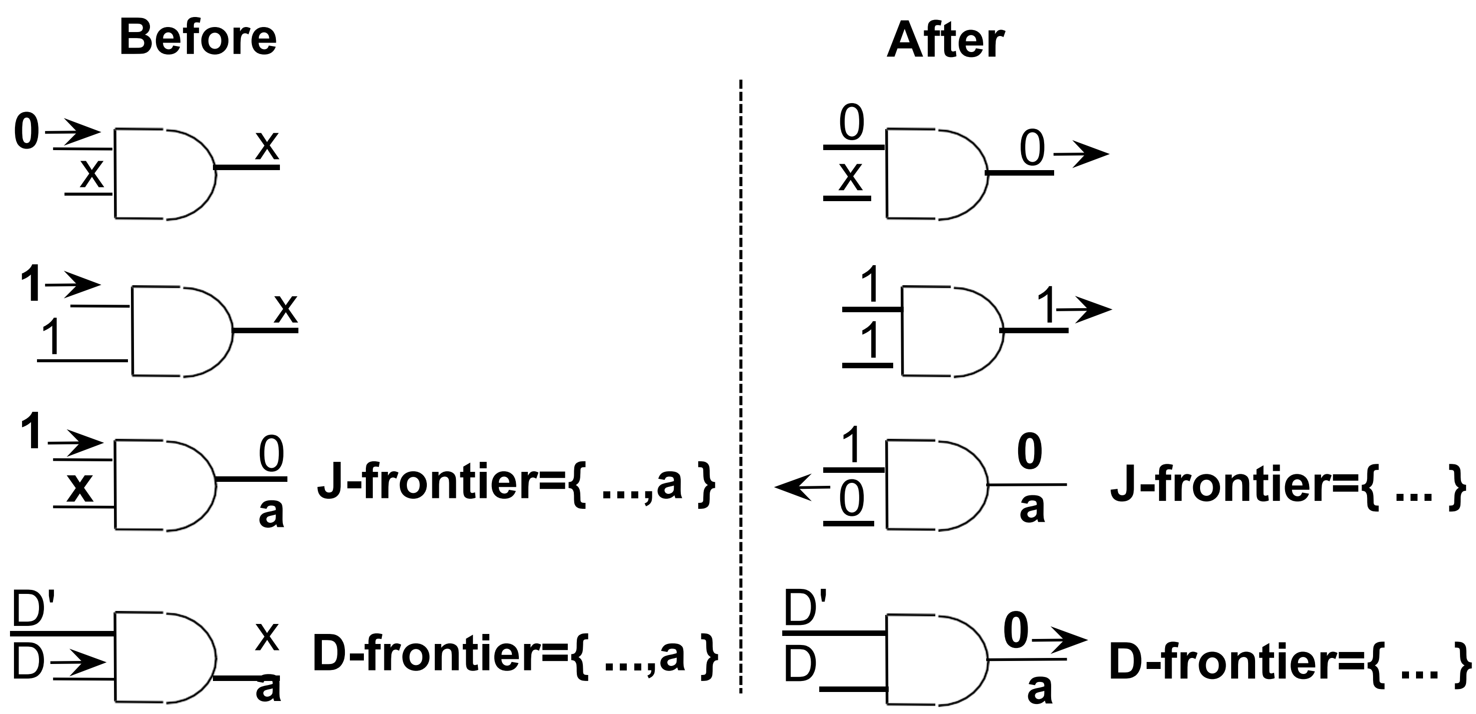
FA => h=D'; FP => e=1, f=1 (=> o=0); FP => q=1, r=1.
To justify q=1 => l=1 or k=1.
Decision: l=1 => c=1, d=1 => m=0, n=0 => r=0. => inconsistency => backtracking!!
Decision: k=1 => a=1, b=1.
To justify r=1 => m=1 or n=1 (=> c=0 or d=0). => done!!

***J-frontier**: The set of all gates whose output value is known but is not implied by its input values. Ex: Initially, the J-frontier of the example is {q=1, r=1}.*

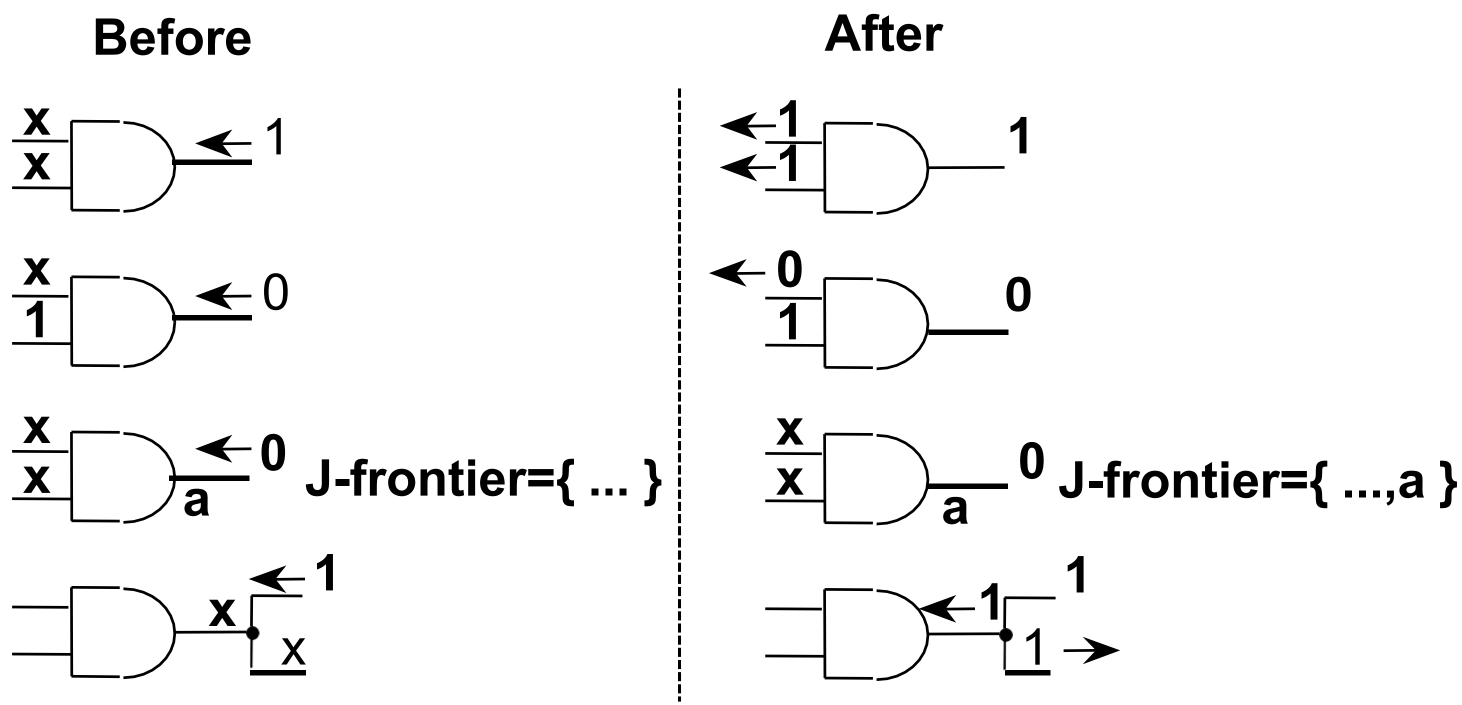
Implications

- Implication: computation of the values that can be uniquely determined.
  - Local implication: propagation of values from one line to its immediate successors or predecessors.
  - Global implication: the propagation involving a larger area of the circuit and reconvergent fanout.
- Maximum implication principle: perform as many implications as possible.
- Maximum implications help us to either reduce the number of problems that need decisions or to reach an inconsistency sooner.

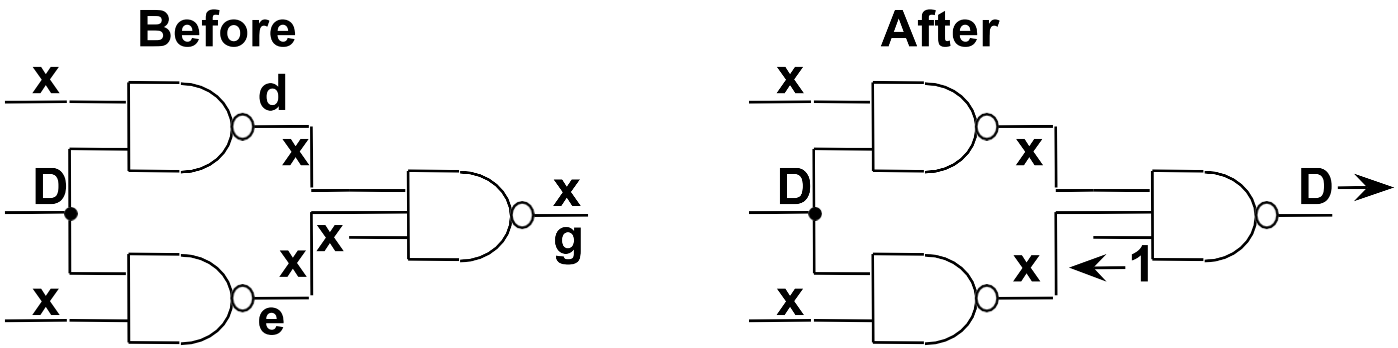
Local Implications (Forward)



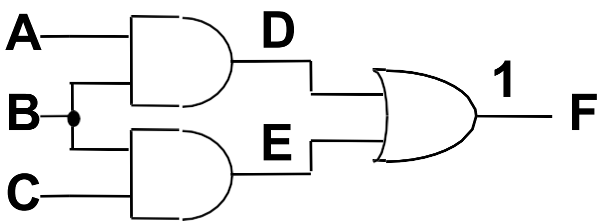
Local Implications (Backward)



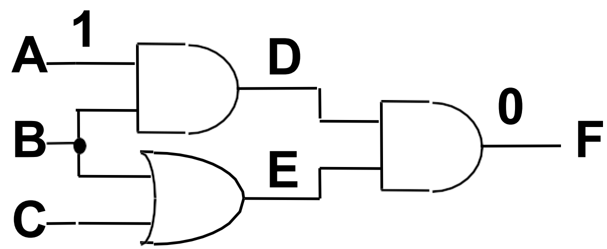
Global Implications



(1) Future unique D-drive.



(2) F=1 implies B=1.
(Static learning)

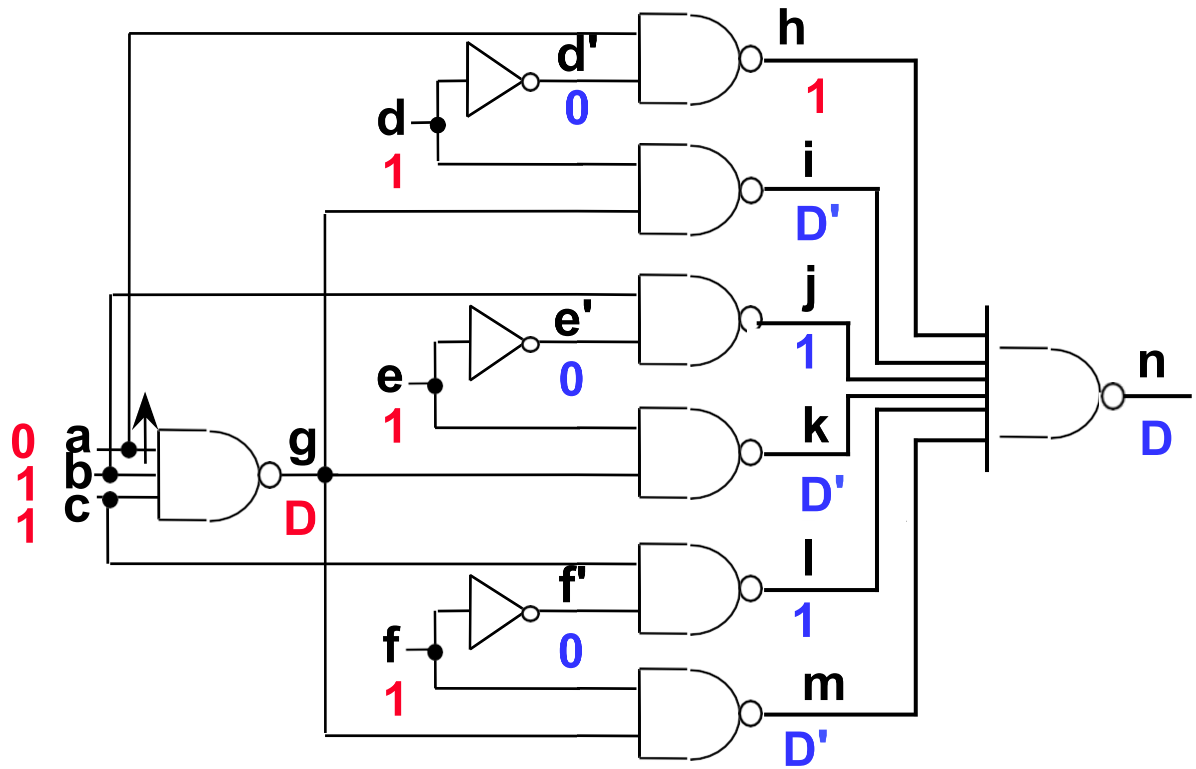


(3) F=0 implies B=0 when A=1.
(Dynamic learning)

(2), (3) are based on contraposition law: (A=>B) <=> (!B => !A).

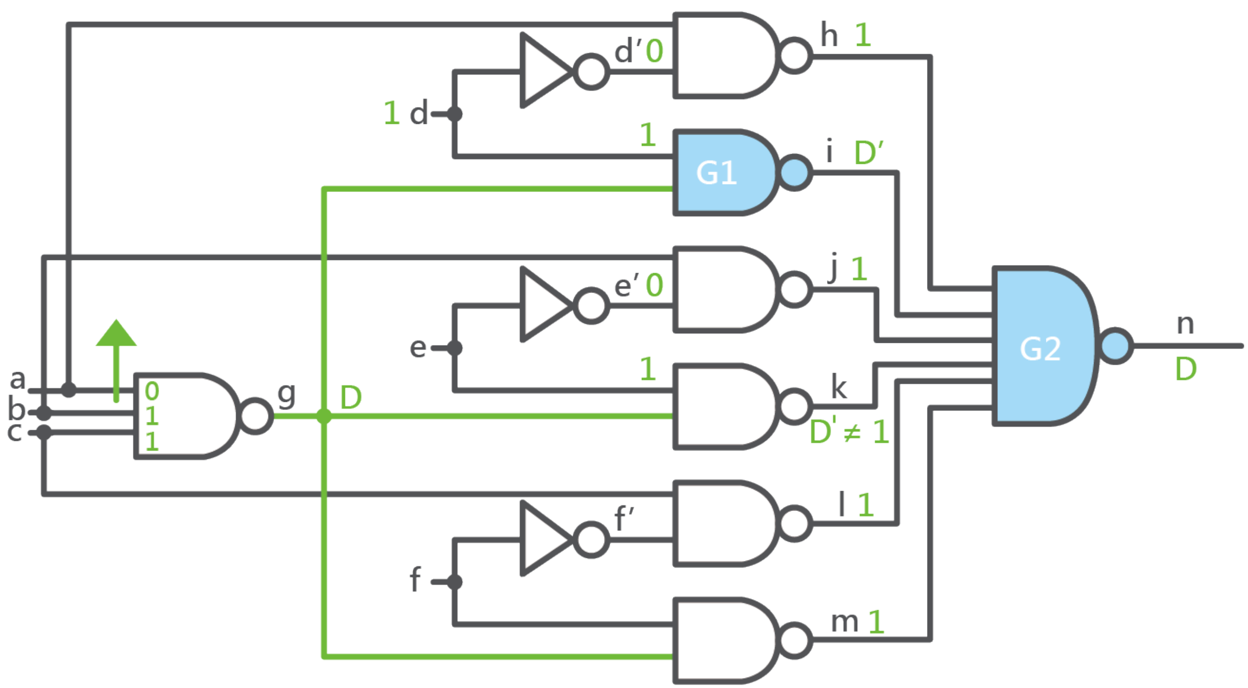
D-Algorithm: Example

- Logic values = {0, 1, D, D', x}.



D-Algorithm: Example

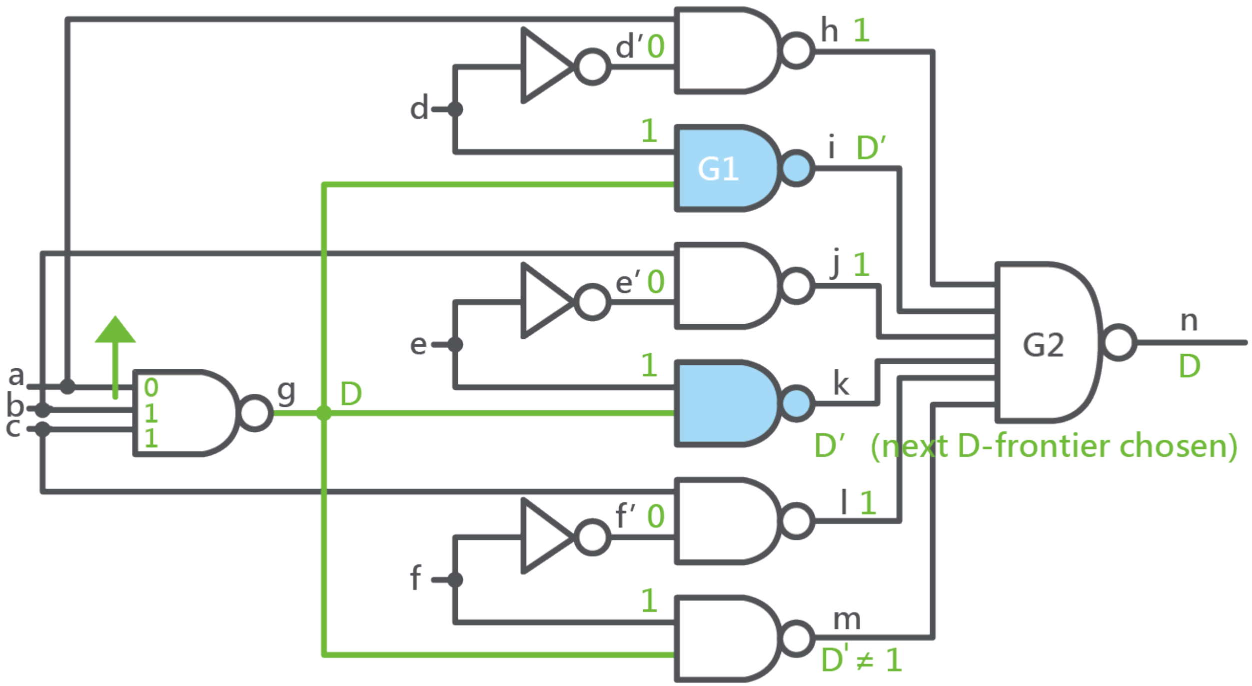
- **Logic values = {0, 1, D, D', x}.**



1. Propagate fault effect through G1 → Set d to 1
2. Propagate fault effect through G2 → Set j,k,l,m to 1
3. Conflict occured at k → Backtrack

D-Algorithm: Example

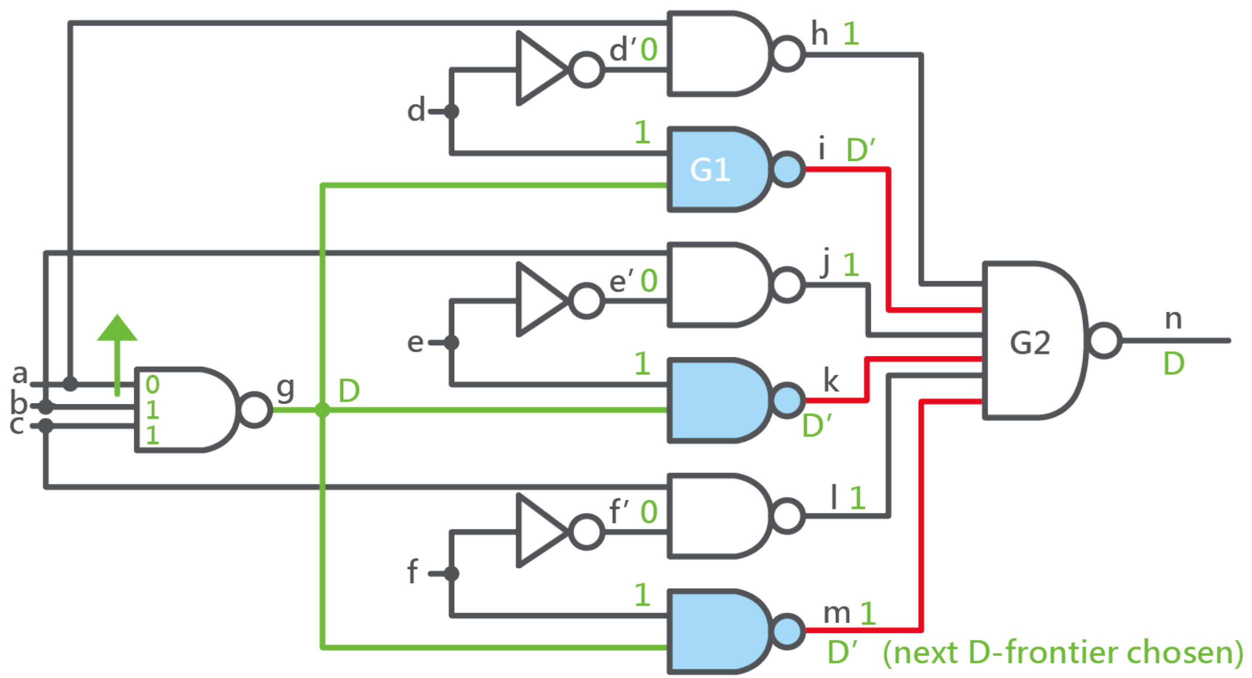
- **Logic values = {0, 1, D, D', x}.**



1. Propagate fault effect through G2 → Set j,l,m to 1
2. Conflict occured at m → Backtrack

D-Algorithm: Example

- **Logic values = {0, 1, D, D', x}.**

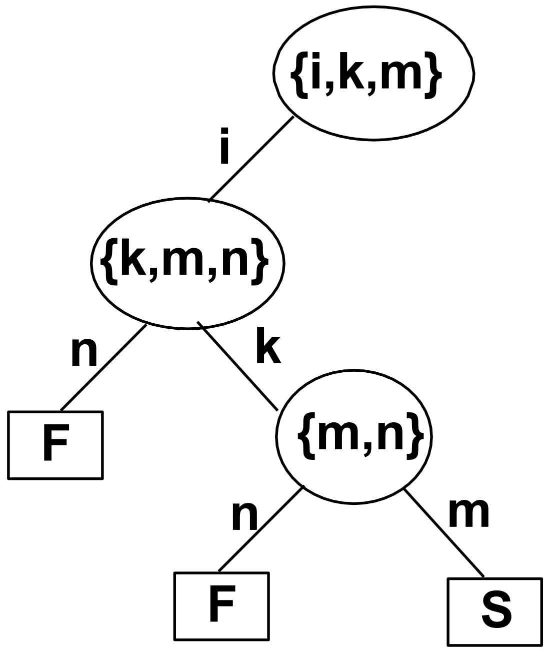


1. Propagate fault effect through G2 → Set j,l to 1
2. Fault propagation and line justification finish

D-Algorithm: Value Computation

Decision	Implication	Comments
	a = 0 h = 1 b = 1 c = 1 g = D	Active the fault Unique D-drive
d = 1	i = D’ d’ = 0	Propagate via i
j = 1 k = 1 l = 1 m = 1	n = D e’ = 0 e = 1 k = D’	Propagate via n **Contradiction**
e = 1	k = D’ e’ = 0 j = 1	Propagate via k
l = 1 m = 1	n = D f’ = 0 f = 1 m = D’	Propagate via n **Contradiction**
f = 1	m = D’ f’ = 0 l = 1 n = D	Propagate via m

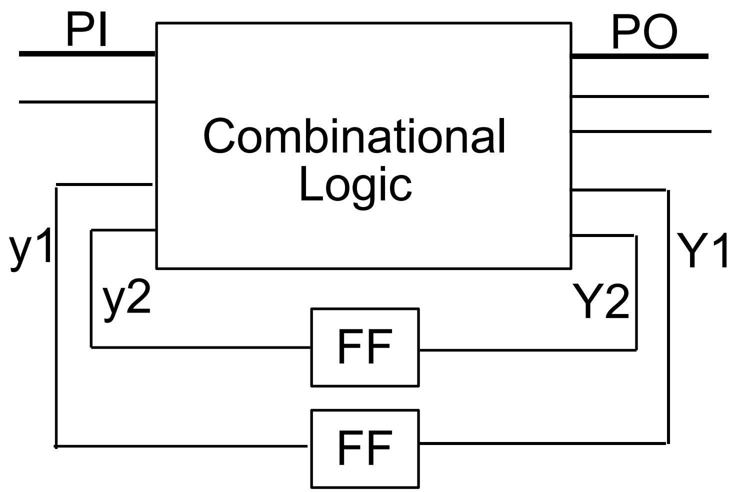
D-Algorithm: Decision Tree

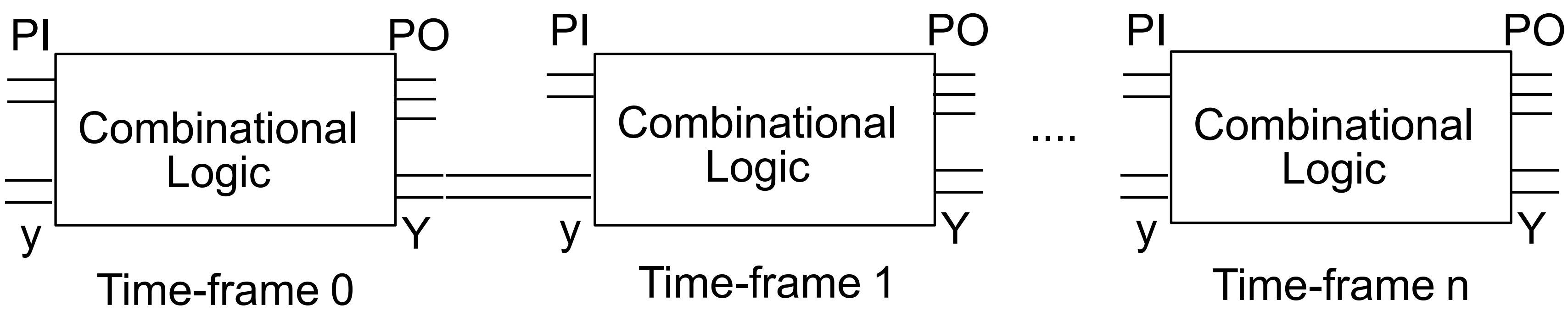


Two times of backtracking!!

- Decision node: the associated D-frontier.
  branch: the decision taken, i.e., the gate selected from the D-frontier.
- The D-algorithm first tried to propagate the fault solely through i, then through both i and k, and eventually succeeded when all three paths were simultaneously sensitized.

Iterative Logic Array (ILA) Model for Sequential Circuits



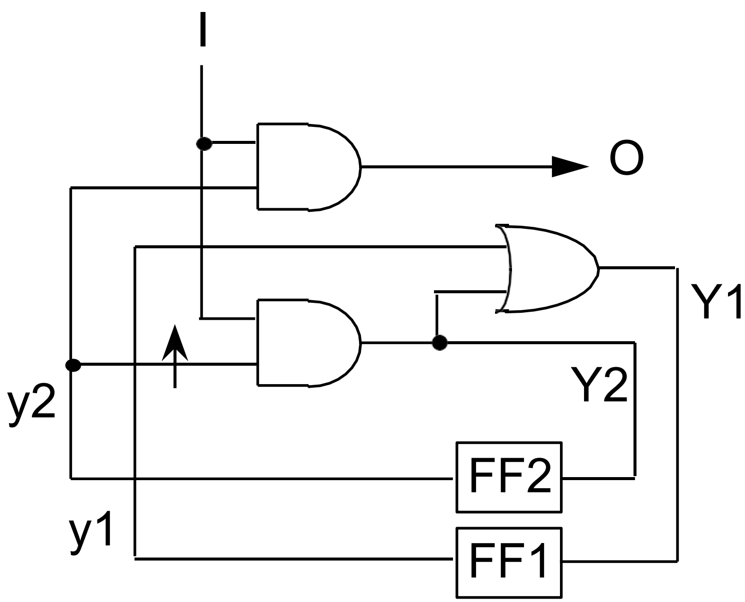


Sequential Test Generation

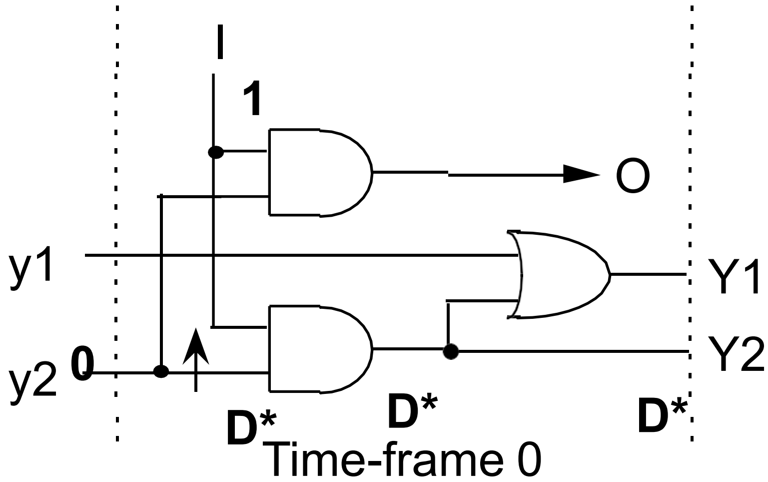
Extended D-Algorithm

1. Pick up a target fault f.
2. Create a copy of a combinational logic, set it time-frame 0.
3. Generate a test for f using D-algorithm for time-frame 0.
4. When the fault effect is propagate to the DFFs, continue fault propagation in the next time-frame.
5. When there are values required in the DFFs, continue the justification in the previous time-frame.

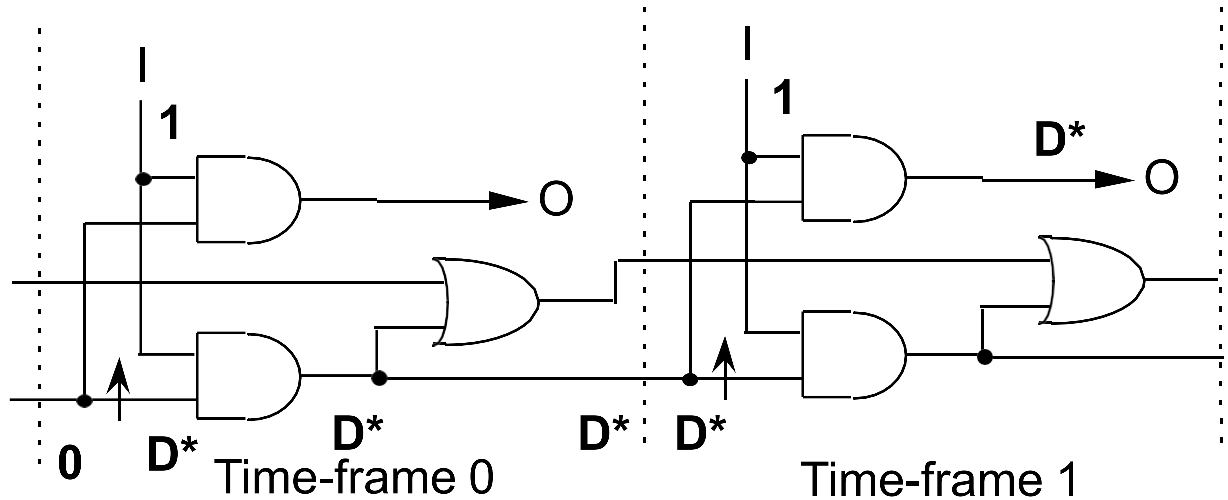
Example for Extended D-Algorithm



Example: Step 1



Example: Step 2



Example: Step 3