Combinational ATPG

- **Introduction**
- **Deterministic Test Pattern Generation**
  - **Boolean difference ***
  - **Path sensitization ****
  - **<u>D-Algorithm [Roth 1966] **</u>**
  - **PODEM [Goel 1981]****
  - **FAN [Fujiwara 1983]****
  - **SAT-based [Larrabee 1992]***
- **Acceleration Techniques**
- **Concluding Remarks**

*Boolean-based methods **path-based methods

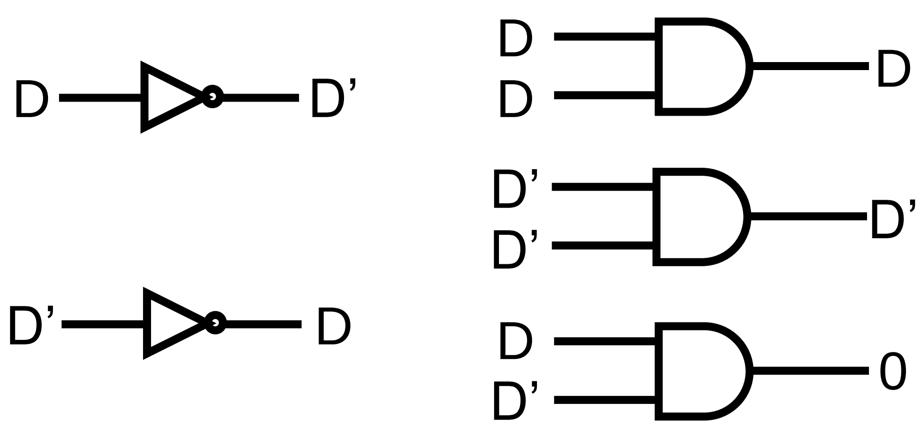
Our D-algorithm follows the original paper.
It is slightly different from WWW textbook.

D-Algorithm

- The D-algebra
- **An D-algorithm example**
- **Types of cubes**
- **Implication and Justification**
- **Flowchart of the D-algorithm**
- **Another example**
- **Problems with the D-Algorithm**

D-Algebra

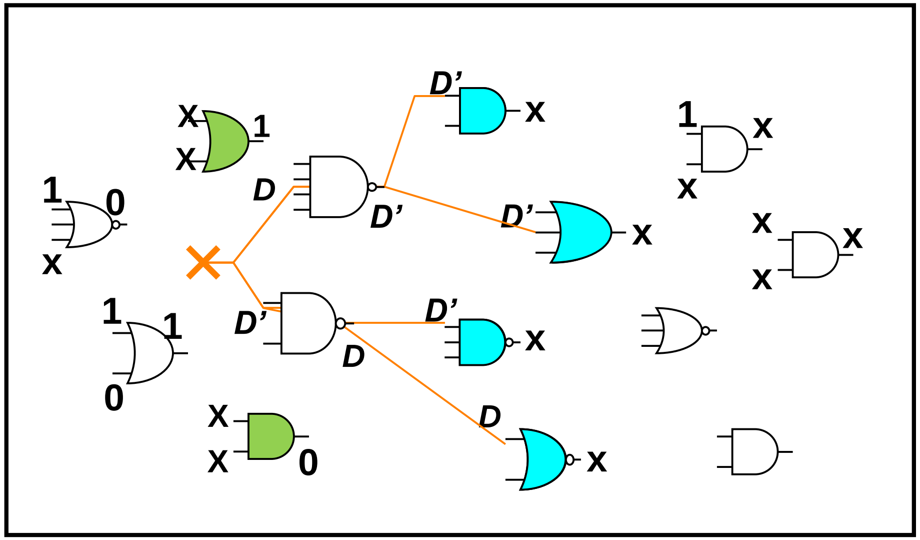
- ***Five-valued logic*: 1, 0, D, D’, X**
  - **Symbol D**
  - **D = 1/0**
    - **1 in fault-free circuit and 0 in the faulty circuit**
  - **D’ = $\overline{D}$ = 0/1**
    - **0 in fault-free circuit and 1 in the faulty circuit**
  - **x means “ not yet specified” in ATPG**



AND	0	1	D	$\overline{D}$	X
0	0	0	0	0	0
1	0	1	D	$\overline{D}$	X
D	0	D	D	0	X
$\overline{D}$	0	$\overline{D}$	0	$\overline{D}$	X
X	0	X	X	X	X

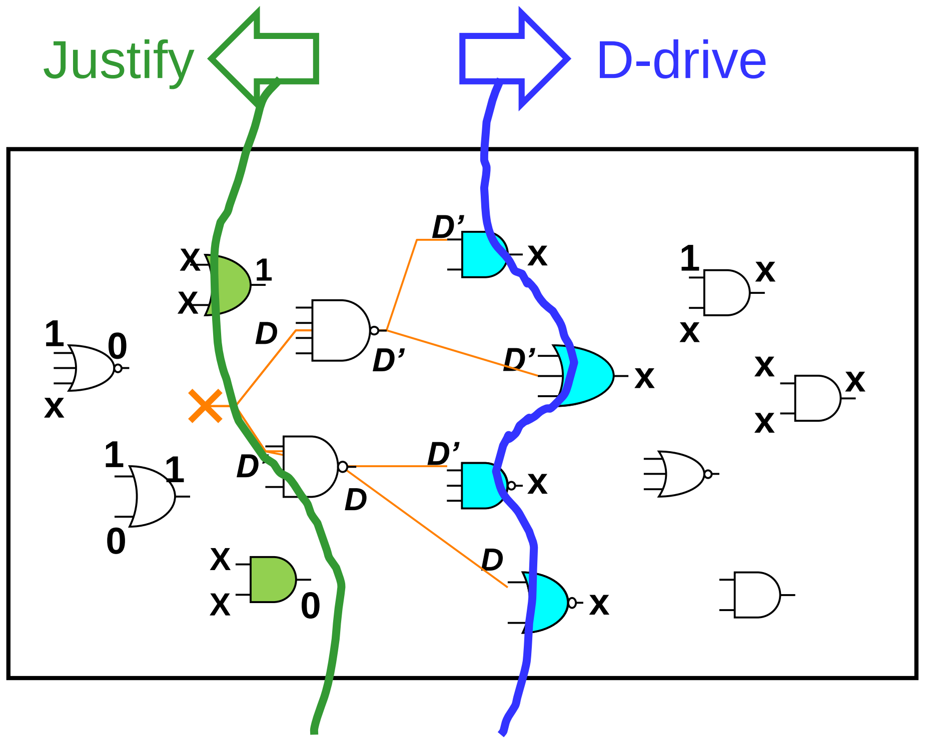
D-frontier , J-frontier

- ***D-frontier*** **: a set of gates whose output value is currently x,**
  - **but have one or more D (or D') at their inputs**
- ***J-frontier*****: a set of gates whose output value is assigned**
  - **But input values have not been decided yet**
- **Example**
  - **Blue gates are D-frontier; Green gates are J-frontier**



Idea of D-Algorithm

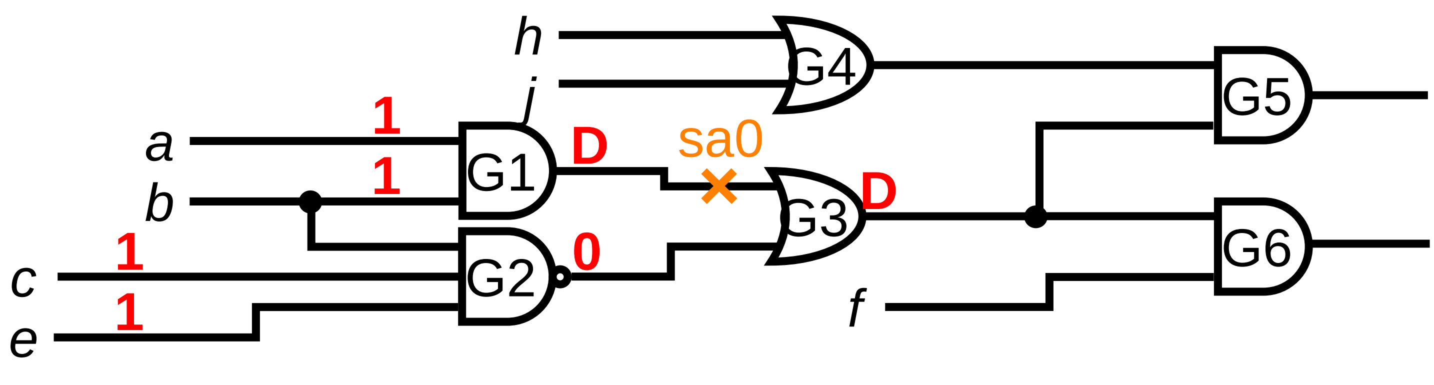
- 1. Create D-frontier (fault activation)
- 2. *Drive* D-frontier toward output (fault effect propagation)
- 3. *Justify* J-frontiers
- 4. *Backtrack* if any conflict occurs



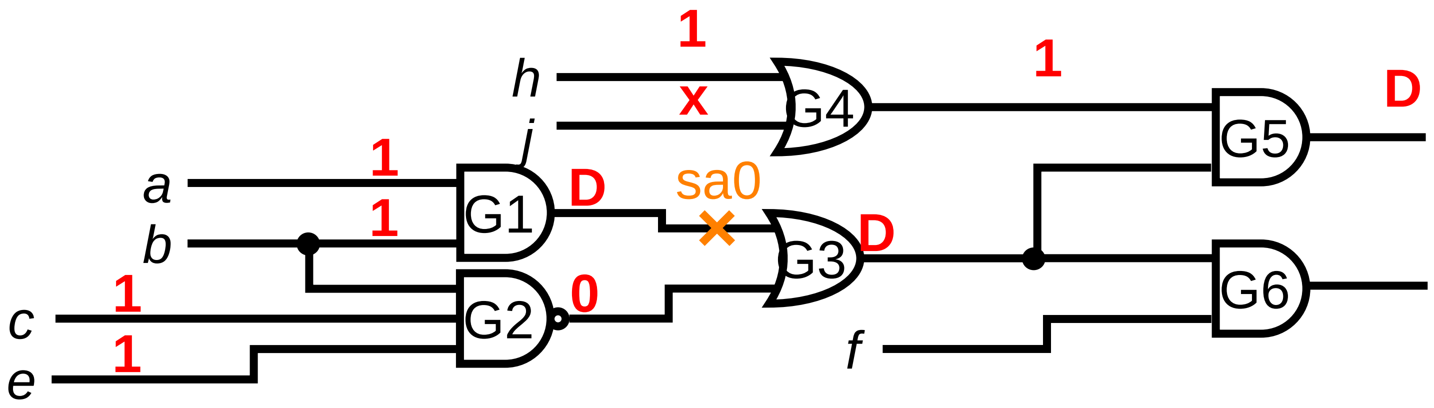
The D-Algorithm

- **The D-algebra**
- A D-algorithm example
- **Types of cubes**
- **Implication and Justification**
- **Flowchart of the D-algorithm**
- **Another example**
- **Problems with the D-Algorithm**

Example



	a	b	c	e	G_1	G_2	G_3	h	j	G_4	f	G_5	G_6	**Comments**
Initial test cube TC(0) = PDFC for G_1 stuck-at 0	**1**	**1**			**D**									**Fault Activation. Implication: nothing happen.**
Prop. D-cube of G_3, PD_{G3}					**D**	**0**	**D**							**D-frontier: {G3}**
$TC(1)=TC(0)\cap PD_{G3}$	**1**	**1**			**D**	**0**	**D**							**D-Drive through G3**
Singular Cover SC_{G2}		**1**	**1**	**1**		**0**								
$TC(2)=TC(1)\cap SC_{G2}$	**1**	**1**	**1**	**1**	**D**	**0**	**D**							**Backward implication**



	a	b	c	e	G_1	G_2	G_3	h	j	G_4	f	G_5	G_6	**Comments**
TC(2)	**1**	**1**	**1**	**1**	**D**	**0**	**D**							**D-frontier: $\{G_5,G_6\}$**
Propagation D-cube of G_5, PD_{G5}							**D**			**1**		**D**		**Choose path through G_5**
$TC(3)=TC(2)\cap PD_{G5}$	**1**	**1**	**1**	**1**	**D**	**0**	**D**			**$\underline{1}$**		**D**		**D-drive through G_5. D reach PO.**
Singular Cover SC_{G4}								**1**	**X**	**1**				**Justification J-frontier = $\{G_4\}$**
$TC(4)=TC(3)\cap SC_{G4}$	**1**	**1**	**1**	**1**	**D**	**0**	**D**	**1**	**X**	**1**	**X**	**D**	**X**	**Done.**

*Underlined numbers are unjustified lines

D-Algorithm

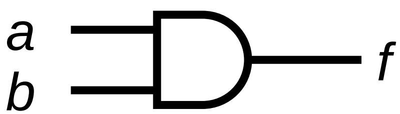
- **The D-algebra**
- **An D-algorithm example**
- Types of cubes
- **Implication and Justification**
- **Flowchart of the D-algorithm**
- **Another example**
- **Problems with the D-Algorithm**

Primitive D-Cubes for a Fault (PDCF)

- **Specify minimal input conditions**
  - **applied to gate input to produce error at gate output**
  - **Used in *fault activation* (more on this later)**
- **Example: AND gate output stuck-at faults:**
  - **Stuck-at-0 fault: 11D**
  - **Stuck-at-1 fault: 0xD' and x0D'**

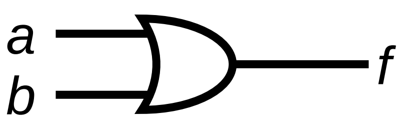
Singular Cover (SC)

- **Minimum gate input assignments for gate output =0 or =1**
  - **Used in line *justification* or *implication* (more on this later)**



Singular covers of AND

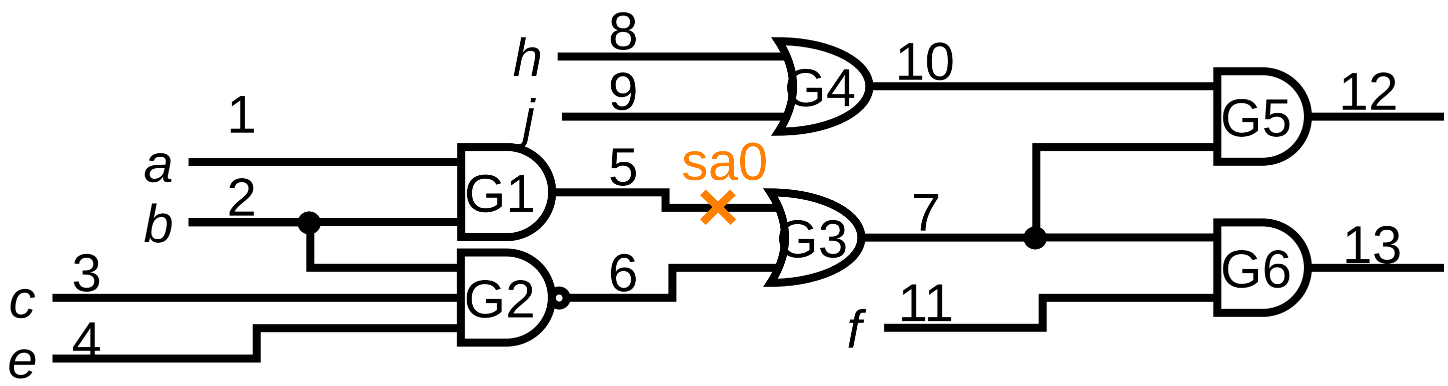
a	*b*	*f*
1	1	1
0	x	0
x	0	0



Singular covers of OR

a	*b*	*f*
0	0	0
1	x	1
x	1	1

Example



	a	b	c	e	G_1	G_2	G_3	h	j	G_4	f	G_5	G_6
Primitive D-cube for G1 sa0 fault	1	1			D								
Singular covers of G2		0	x	x		1							
		x	0	x		1							
		x	x	0		1							
		1	1	1		0							
Singular covers of G3					1	x	1						
					x	1	1						
					0	0	0						
Singular covers of G4								1	x	1			
								x	1	1			
								0	0	0			
Singular covers of G5							x			0		0	
							0			x		0	
							1			1		1	
Singular covers of G6							x				0		0
							0				x		0
							1				1		1

Propagation D-Cube (PDC)

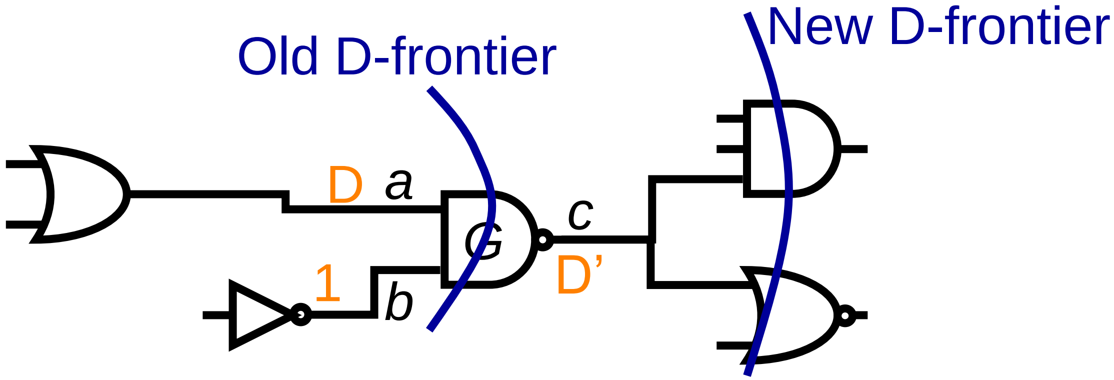
- **Minimum gate input assignments required**
  - **to propagate a D or D' from gate input(s) to gate output**
- **Used in *D-Drive* or *implication* (more later)**

A	*B*	*AB*
D	1	D
1	D	D
D	D	D
D'	1	D'
1	D'	D'
D'	D'	D'

A	*B*	(*A*+*B*)'
0	D	D'
D	0	D'
D	D	D'
0	D'	D
D'	0	D
D'	D'	D

D-drive

- D-drive selects an element in D-frontier
  - and attempts to propagate D or D' from gate input to gate output
  - Using propagation D-cube



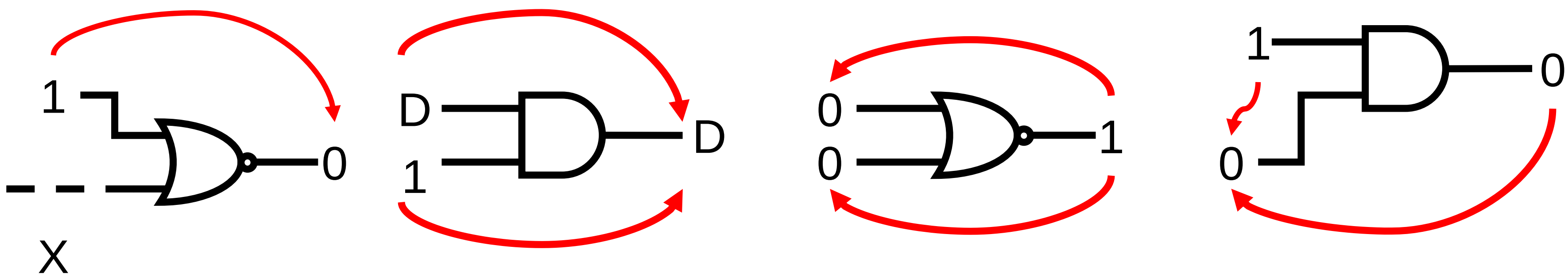
	…	a	b	c	…
Test cube before D-drive	…	D	X	X	
Propagation D cube of gate G	…	D	1	D'	…
Test cube after D-drive	…	D	1	D'	…

The D-Algorithm

- **The D-algebra**
- **An D-algorithm example**
- **Types of cubes**
- Implication and Justification
- **Flowchart of the D-algorithm**
- **Another example**
- **Problems with the D-Algorithm**

Implication

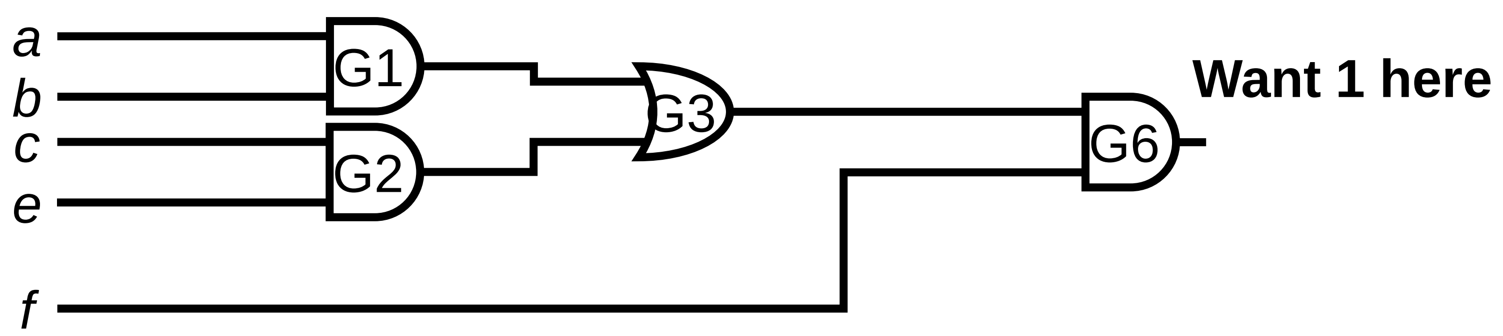
- ***Forward Implication***
  - **partially (or fully) specified input values *uniquely* determines the output values.**
- ***Backward Implication***
  - **knowing the output values (and some input values) can *uniquely* determine the un-specified input values.**
- **Examples**



- **Note 1: Implication means NO choice**
- **Note 2: Implication can be done any time a decision is made**

Justification

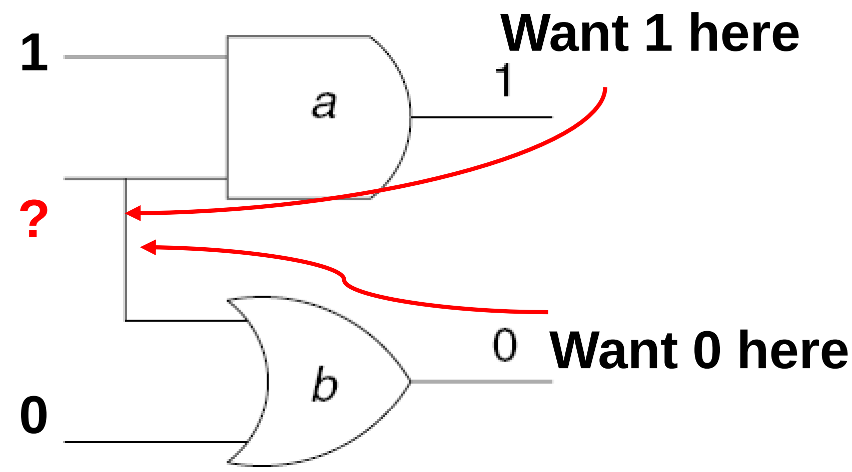
- **Definition: find a valid primary input assignment for desired values**
- **Justification is easy inside a *fanout-free* circuit**
- **No decision needed**
  - **Always finds an answer**
- **Example**



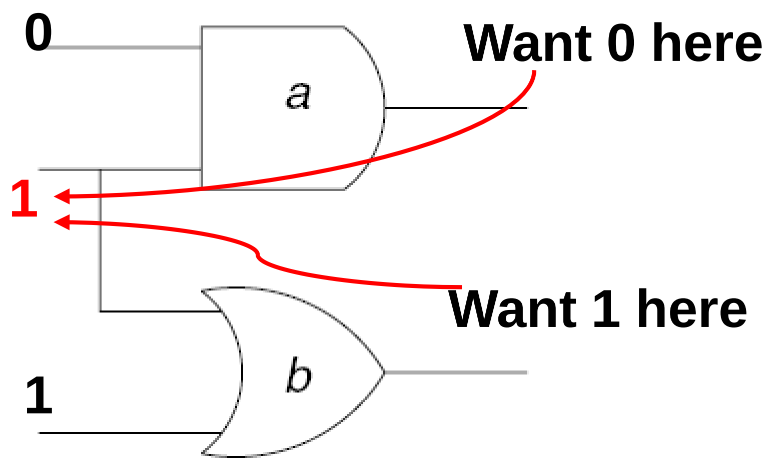
Justification (2)

- Justification may fail when there are fanout branches

Justification fails



Justification succeeds



The D-Algorithm

- **The D-algebra**
- **A D-algorithm example**
- **Types of cubes**
- **Implication and Justification**
- Flowchart of the D-algorithm
- **Another example**
- **Problems with the D-Algorithm**

Test Cube (TC)

- **A *Test cube* is a partially specified Boolean values for testing a fault**
- **In D algorithm, a test cube contains**
  - **not only primary inputs, but also internal nodes**
- **Notation**
  - **TC(*n*) = test cube at ATPG step *n***

- **NOTE: In PODEM, a test cube contains primary inputs only**
  - **More details later**

Intersection of Test Cubes

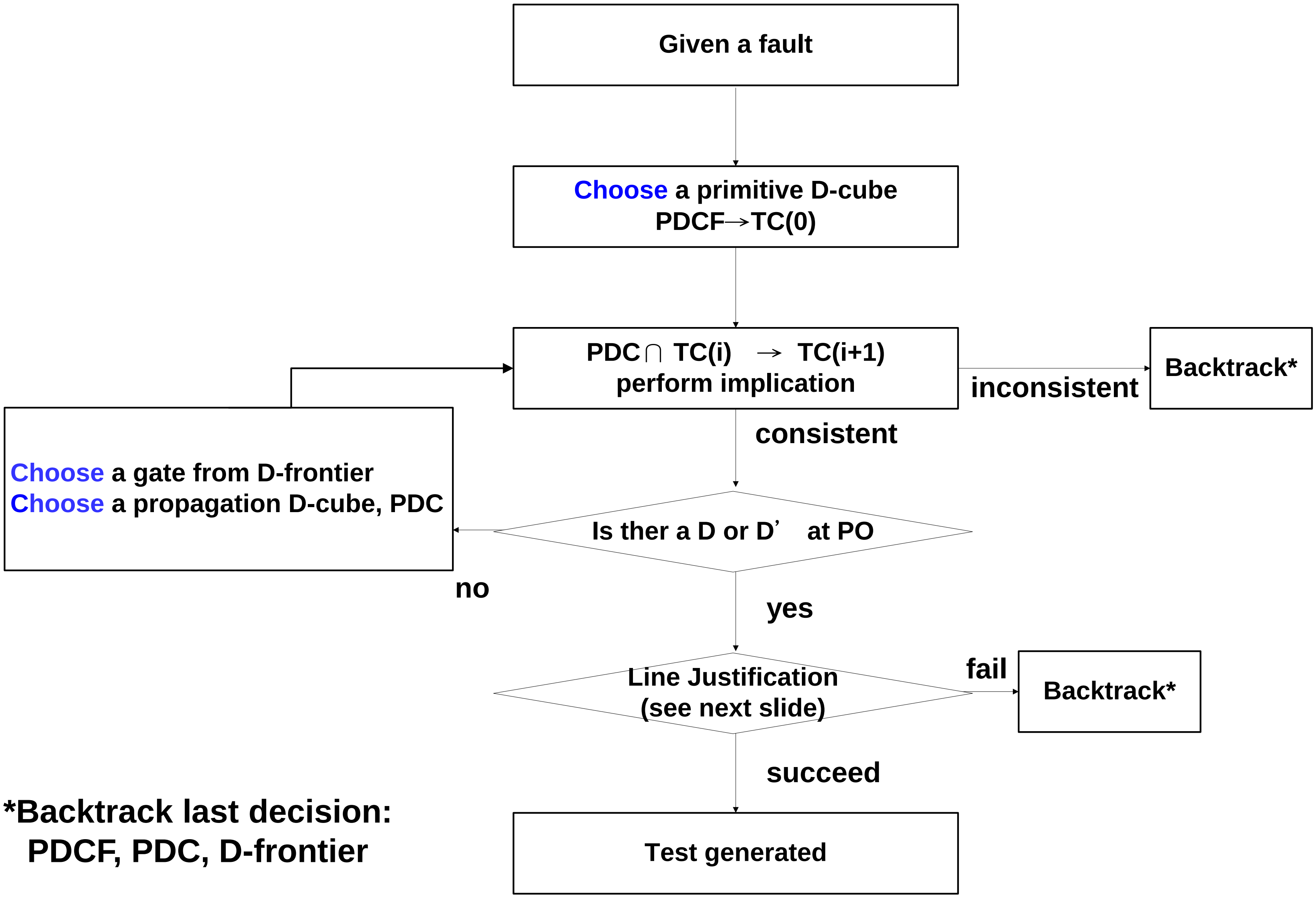
- Two bits has *intersection* if their logic values are not conflicting
- Two test cubes has intersection if there is no confliction in any bit
- Example :
  - TC(1) ∩ TC(2) = X0X1 ∩ 1XXX = 10X1
  - TC(1) ∩ TC(2) = 10X1 ∩ 0XXX = no intersection

Bit 1∩ Bit 2

bit2 \ bit1	0	1	X	D	D’
0	0		0		
1		1	1		
X	0	1	X	D	D’
D			D	D	
D’			D’		D’

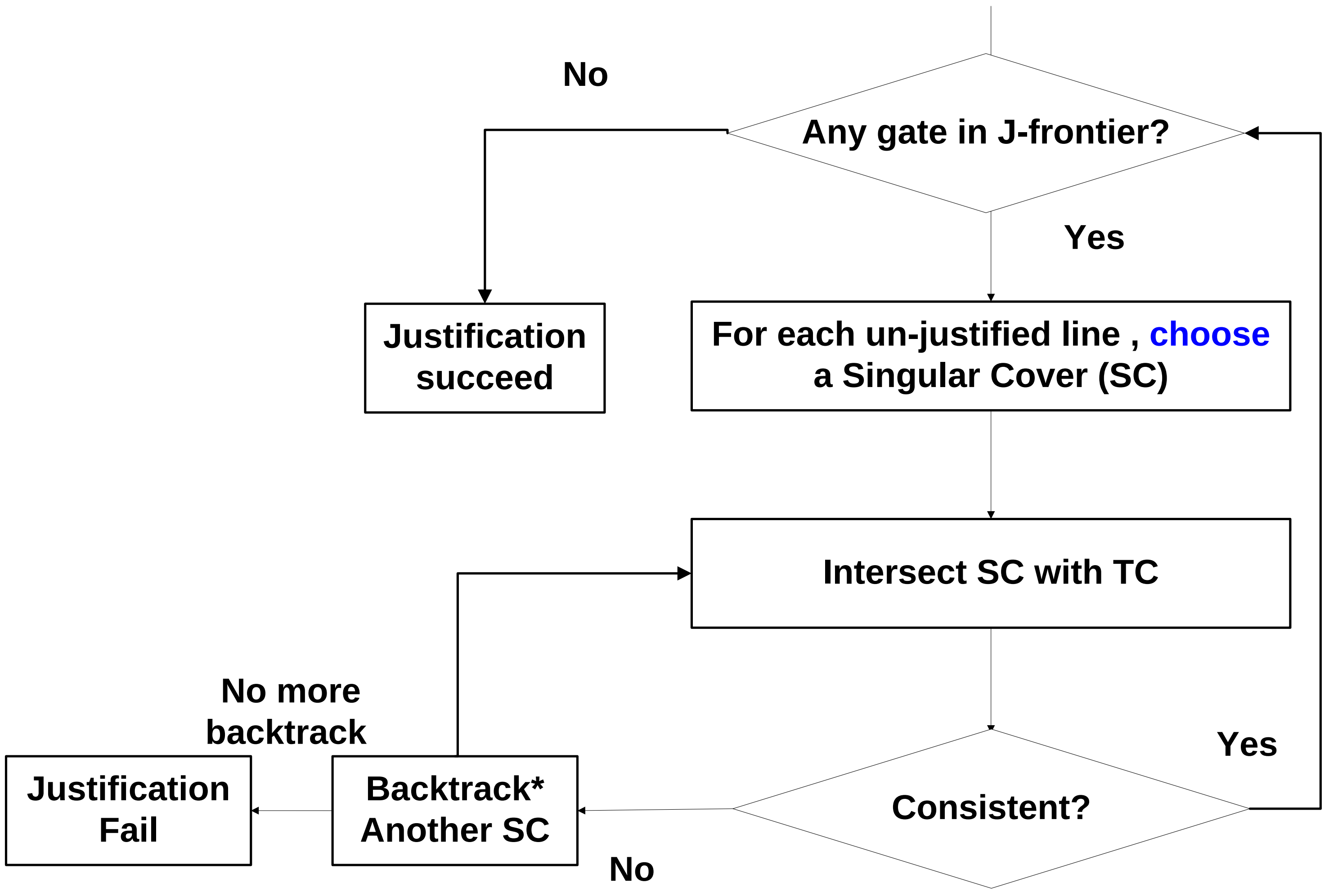
empty box = confliction = no intersection

Flowchart of the D-Algorithm



*Backtrack last decision:
PDCF, PDC, D-frontier

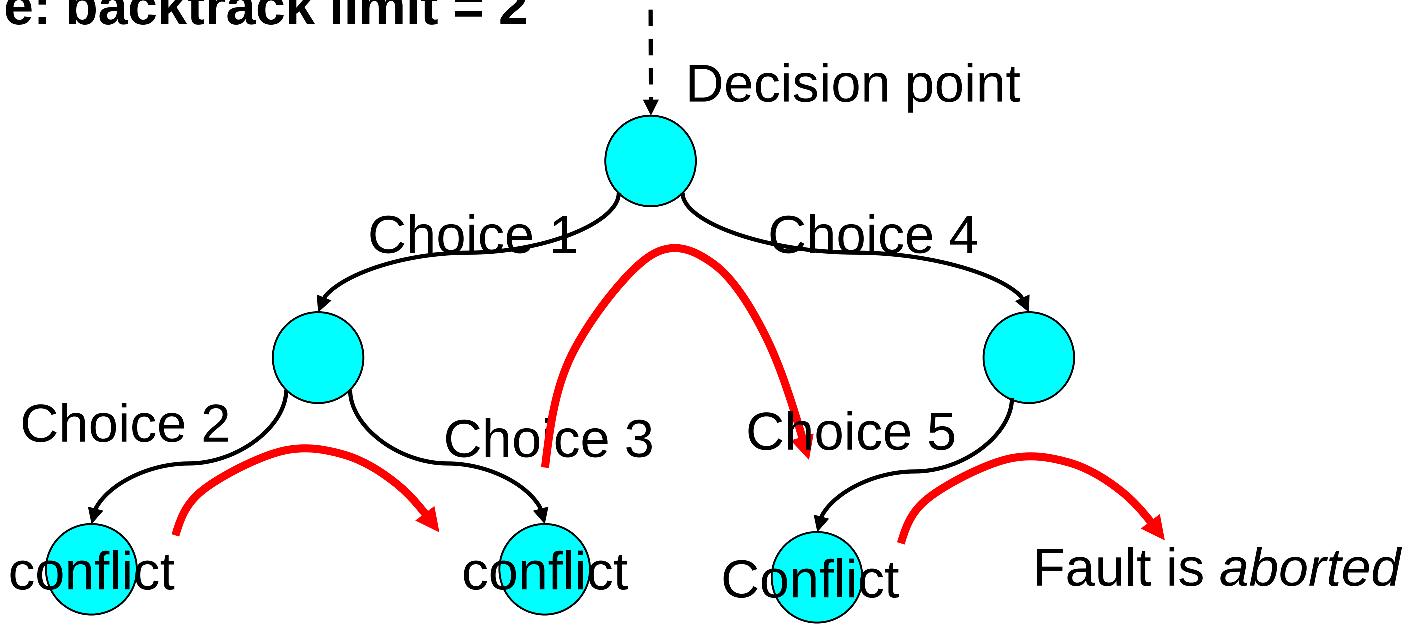
Line Justification



*backtrack last decision: SC

Backtrack

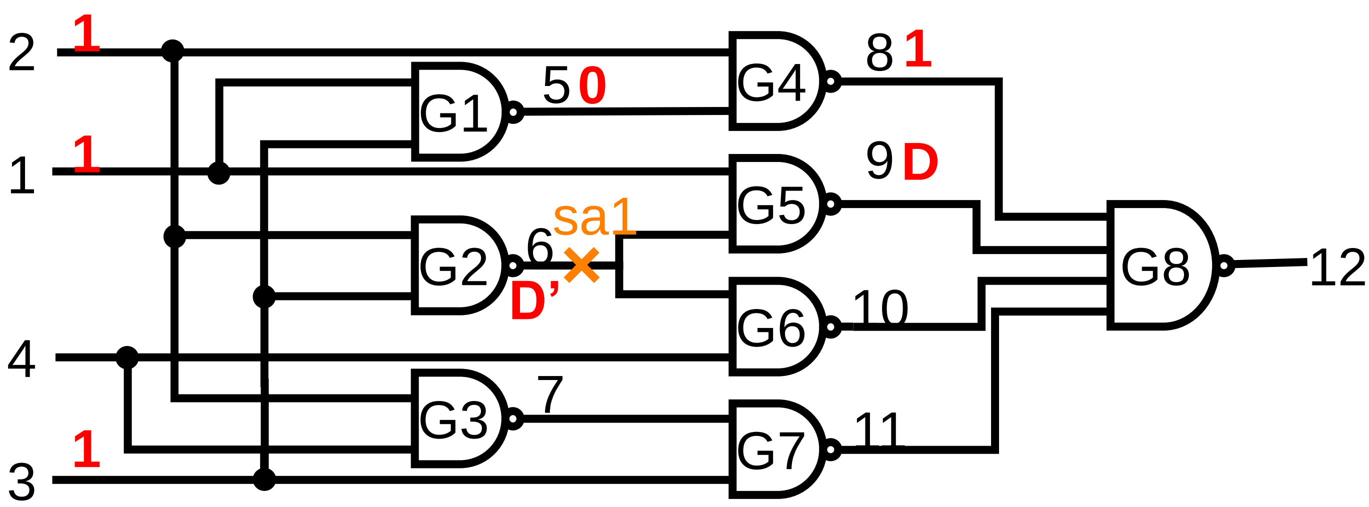
- When conflict, ***backtrack*** to last decision point and change choice
  - Choice can be: PDCF, PDC, SC, D-drive gate
- To avoid spending too much time on a fault
  - Use specify a ***backtrack limit***
    - maximum number of backtracks allowed for a single fault
  - Fault is ***aborted*** if backtrack limit is reached
- Example: backtrack limit = 2



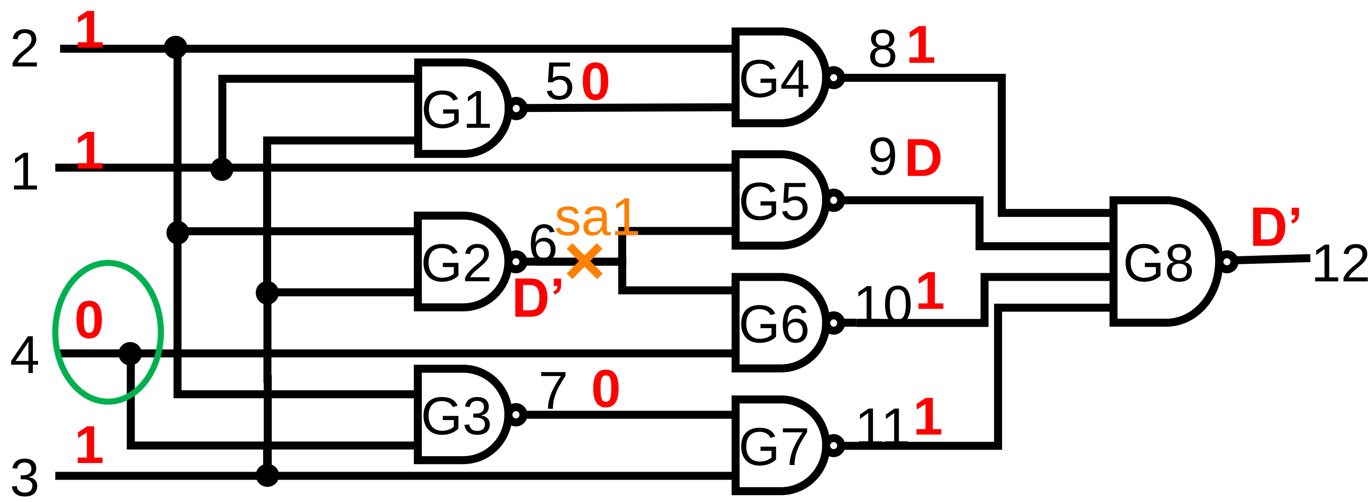
The D-Algorithm

- **The D-algebra**
- **A D-algorithm example**
- **Types of cubes**
- **Components of the D-algorithm**
- **Flowchart of the D-algorithm**
- Another example
- **Problems with the D-Algorithm**

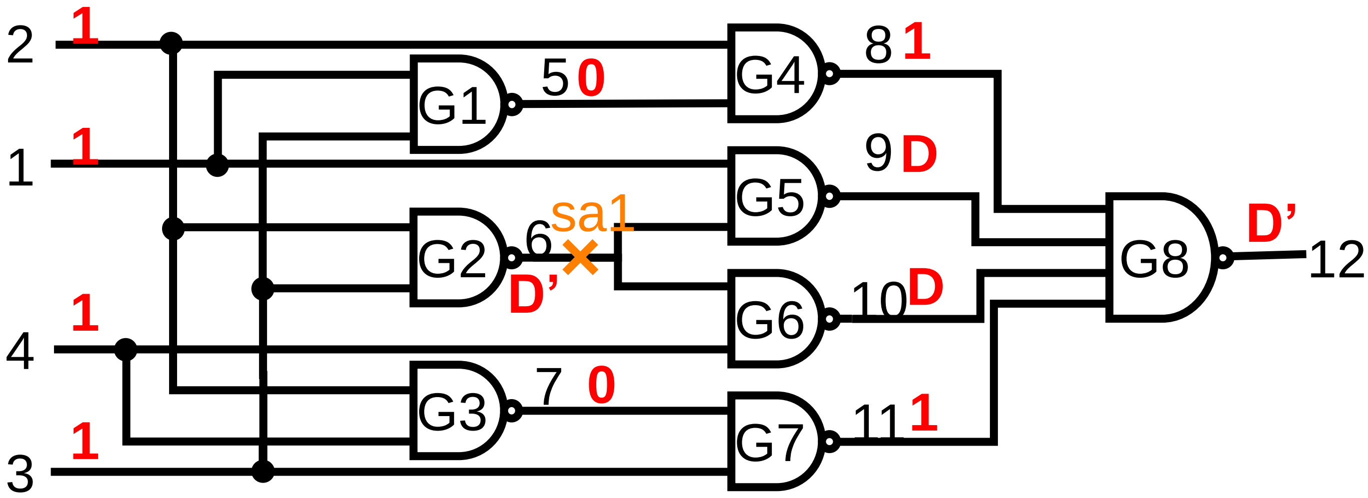
Another D-Algorithm Example



		1	2	3	4	5	6	7	8	9	10	11	12
1	**PDCF:TC(0) = PDFC**		**1**	**1**			**D'**						
2	**Implication: nothing happens**		**1**	**1**			**D'**						
3	**choose D-frontier G5: PDC** **TC(1)=TC(0)∩ PDC**	**1** **1**	 **1**	 **1**			**D'** **D'**			**D** **D**			
4	**Forward Implication: SC$_{G1}$** **TC(2)**	**1** **1**	 **1**	**1** **1**		**0** **0**	 **D'**			 **D**			
5	**Forward Implication: SC$_{G4}$** **TC(3)**	 **1**	**X** **1**	 **1**		**0** **0**	 **D'**		**1** **1**	 **D**			



		1	2	3	4	5	6	7	8	9	10	11	12
	TC(3)	**1**	**1**	**1**		**0**	**D’**		**1**	**D**			
6	**choose D-frontier G8: PDC**								**1**	**D**	**1**	**1**	**D’**
	TC(4)	**1**	**1**	**1**		**0**	**D’**		**1**	**D**	**1**	**1**	**D’**
7	**Backward Implication G_7**			**1**				**0**				**1**	
	TC(5)	**1**	**1**	**1**		**0**	**D’**	**0**	**1**	**D**	**1**	**1**	**D’**
8	**Backward Implication G_6: SC_{G6}**				**0**		**D’**				**1**		
	TC(6)	**1**	**1**	**1**	**0**	**0**	**D’**	**0**	**1**	**D**	**1**	**1**	**D’**
9	**Backward Implication G_3: SC_{G3}**		**1**		**1**			**0**					
	Fail. Backtrack step to 6												



		1	2	3	4	5	6	7	8	9	10	11	12
	Backtrack to step 6. TC(3)	**1**	**1**	**1**		**0**	**D'**		**1**	**D**			
10	**Choose D-frontier G6: PDC**				**1**		**D'**				**D**		
	TC(4)	**1**	**1**	**1**	**1**	**0**	**D'**		**1**	**D**	**D**		
11	**Forward Implication: SC_{G3}**		**1**		**1**			**0**					
	TC(5)	**1**	**1**	**1**	**1**	**0**	**D'**	**0**	**1**	**D**	**D**		
12	**Forward Implication: SC_{G7}**			**1**				**0**				**1**	
	TC(6)	**1**	**1**	**1**	**1**	**0**	**D'**	**0**	**1**	**D**	**D**	**1**	
13	**Forward Implication: SC_{G8}**								**1**	**D**	**D**	**1**	**D'**
	TC(7)	**1**	**1**	**1**	**1**	**0**	**D'**	**0**	**1**	**D**	**D**	**1**	**D'**
14	**No justification needed. Test generated.**	**1**	**1**	**1**	**1**	**0**	**D'**	**0**	**1**	**D**	**D**	**1**	**D'**

The D-Algorithm

- **The D-algebra**
- **A D-algorithm example**
- **Types of cubes**
- **Components of the D-algorithm**
- **Flowchart of the D-algorithm**
- **Another example**
- Plus & Minus of D-Algorithm

Plus/Minus of D-Algorithm

+ **D algorithm is *complete ATPG***
  - **Guarantee to generate a pattern for a testable fault**
– **Large search space**
  - **Assignment of values is allowed for internal signals**
  - **Backtracking could occur at each gate**
  - **Very large search space**

FFT

- **Q1. Why we do justification at the end of D-algorithm**
  - **Why not immediately after implication?**
- **Q2. Please explain why D-algorithm is complete ATPG**