DIGITAL SYSTEM TESTING
COE -545

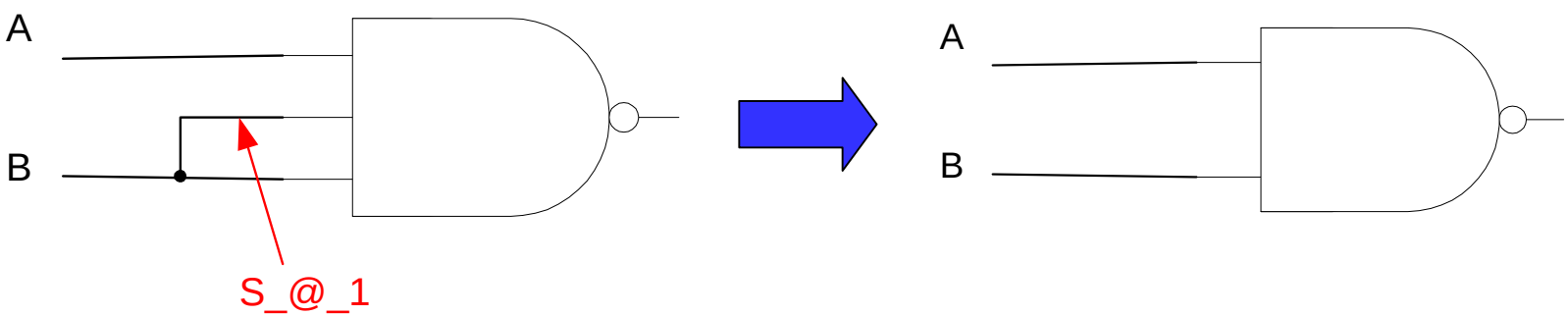
Lecture – 05

Test Generation (Boolean Difference)

Combinational Circuit Testing

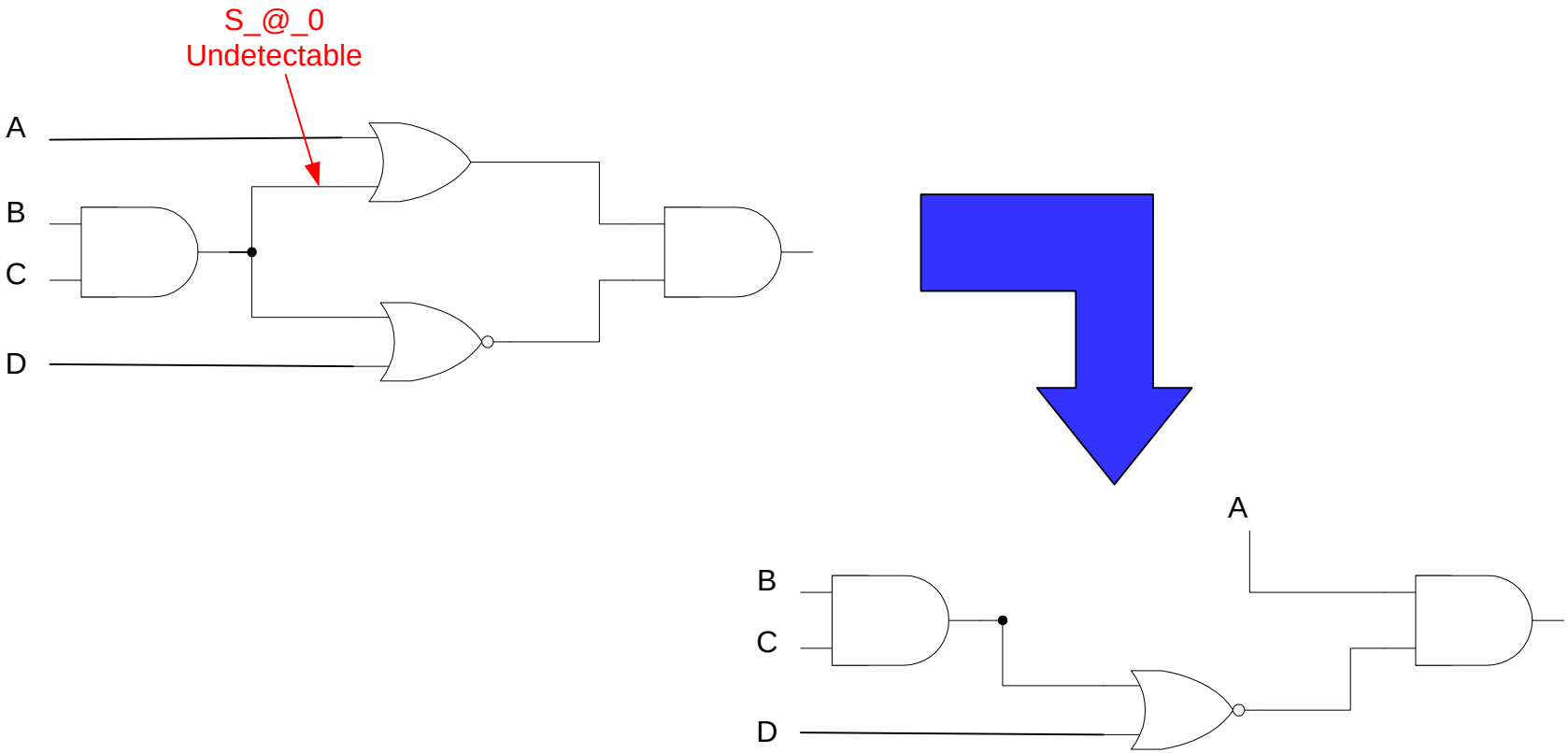
Basic Definitions

- Let fault *f* change output Z(X) of a circuit *C* to $Z_f(X)$.
- A TV *t* detects *f* if $Z(t) \neq Z_f(t) \rightarrow Z(t) \oplus Z_f(t) = 1$
- Fault *f* is *Undetectable* or *Redundant* if $Z(t) = Z_f(t)$ $\forall t$.
- If the fault *line x s-a-d* is undetectable, then *x and circuits feeding x* can be *removed* from the circuit.

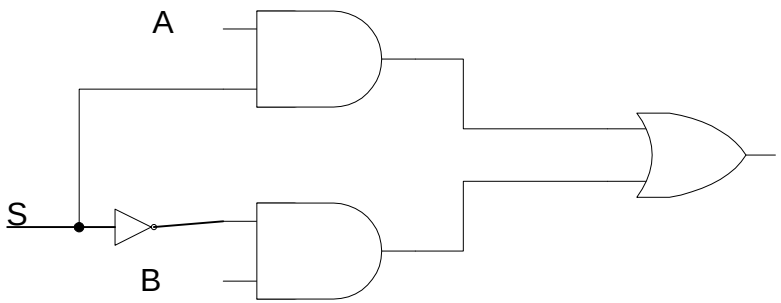


Combinational Circuit Testing

Another Example

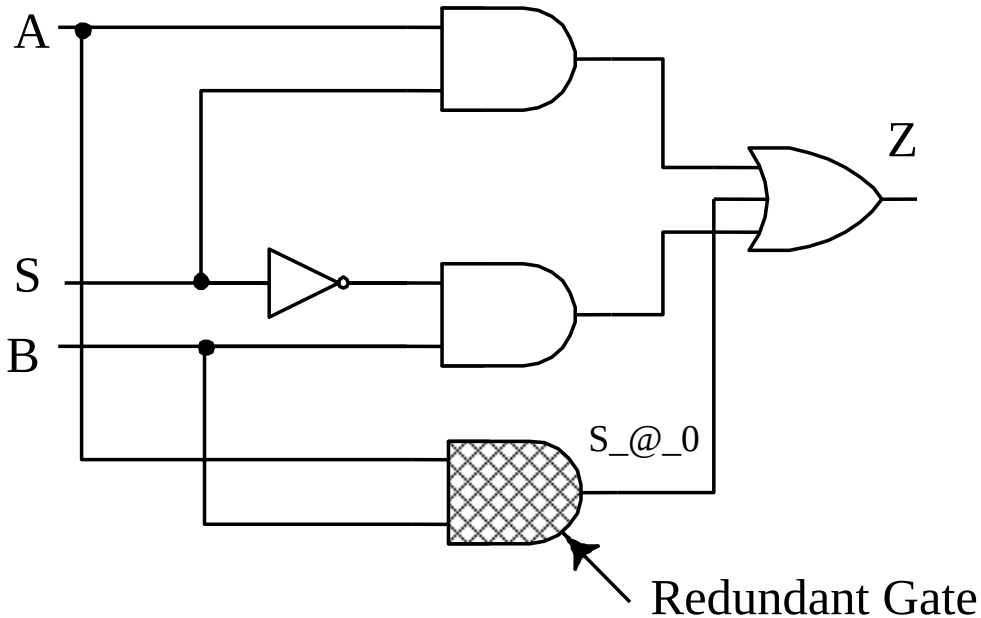


Redundancy



Hazardous Circuit

(a)



Hazard-Free Circuit

(b)

Combinational Circuit Testing

Basic Definitions (cont'd)

- A set of tests $T = \{t_1, t_2, ..., t_k\}$ is *complete* if it detects (covers) all detectable SSL faults in the circuit
- We can represent any test set $\{t_1, t_2, ..., t_k\}$ by the Boolean function whose minterms are $\{t_1, t_2, ..., t_k\}$.
- Example, the function $Z(a,b,c,d) = a'bd' + abcd$ denotes the 3-member test set {0100, 0110, 1111}.

- The set of all TVs for fault f is expressed by the Boolean function $(Z(x) \oplus Z_f(x))$

Example

- Fault-Free Fun= F(a,b,c) = $(a + b)(b + c)$
- Fault = b/0
- *Faulty Fun* = $F_\alpha = ac$
- TVs Detecting this Fault Should Satisfy: $(F \oplus F_\alpha) = 1$
- *Thus*, $(a + b)(b + c) \oplus ac = B(A' + C')$
- This represents 2 possible Tests
- TV set for a/1 = {01x, x10} → {010, 011, 110}

Boolean Difference Method

- Fault-Free Fun= $F\ (x_1, x_2, \ldots, x_n)$
- Assuming a SSL Fault $x_i/1$
- $F_\alpha = F(x_i = 1) = F\ (x_1, x_2, \ldots, 1, \ldots, x_n) = F_i\ (1)$
- Assuming a SSL Fault $x_i/0$
- $F_\alpha = F(x_i = 0) = F\ (x_1, x_2, \ldots, 0, \ldots, x_n) = F_i\ (0)$
- According to Shannon's Expansion Theorem:

  $F\ (x_1, x_2, \ldots, x_n) = x_i\ F_i\ (1) + \overline{x}_i\ F_i\ (0)$
- Thus, the Test Set to Detect $x_i/0$ corresponds to :

  $T_0 = x_i\ (F_i\ (1) \oplus F_i\ (0))$, Likewise
- The Test Set to Detect $x_i/1$ corresponds to :

  $T_1 = \overline{x}_i\ (F_i\ (1) \oplus F_i\ (0))$

Boolean Difference Method

- The Expression ($F_i\ (1) \oplus F_i\ (0)$) is Termed the Boolean Difference of F wrt x_i , or (dF/dx_i)
- Thus, the Test Set to Detect $x_i/0$ corresponds to :

$$T_0 = x_i \,.\, (dF/dx_i)$$

The Test Set to Detect $x_i/1$ corresponds to :

$$T_1 = \overline{x}_i \,.\, (dF/dx_i)$$

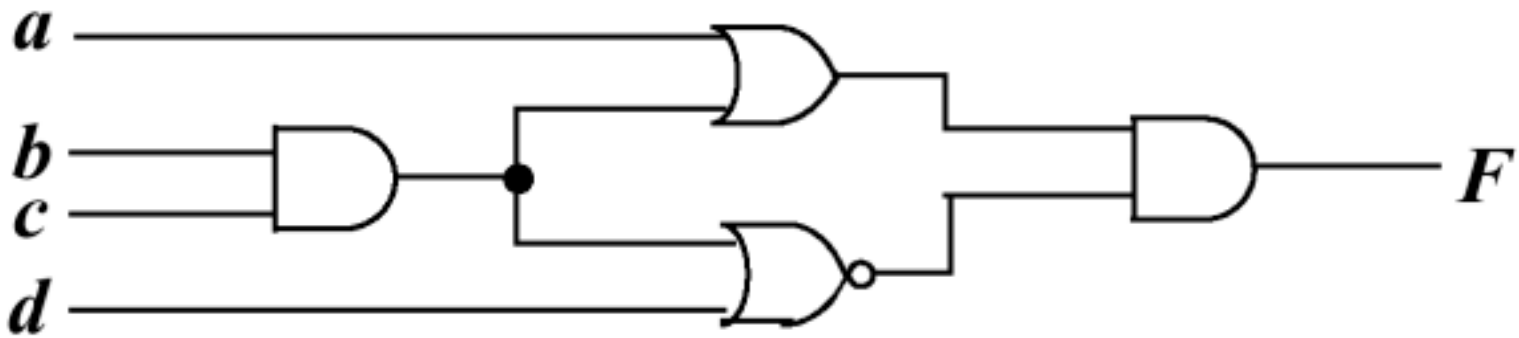
Notes:

- The x_i`s are Primary Inputs
- A Fault on some PI (x_i) is Undetectable iff

$F_i\ (1) = F_i\ (0)$ *OR* $dF/dx_i = 0$

Which Means that **F is INDEPENDENT OF x_i**

Example



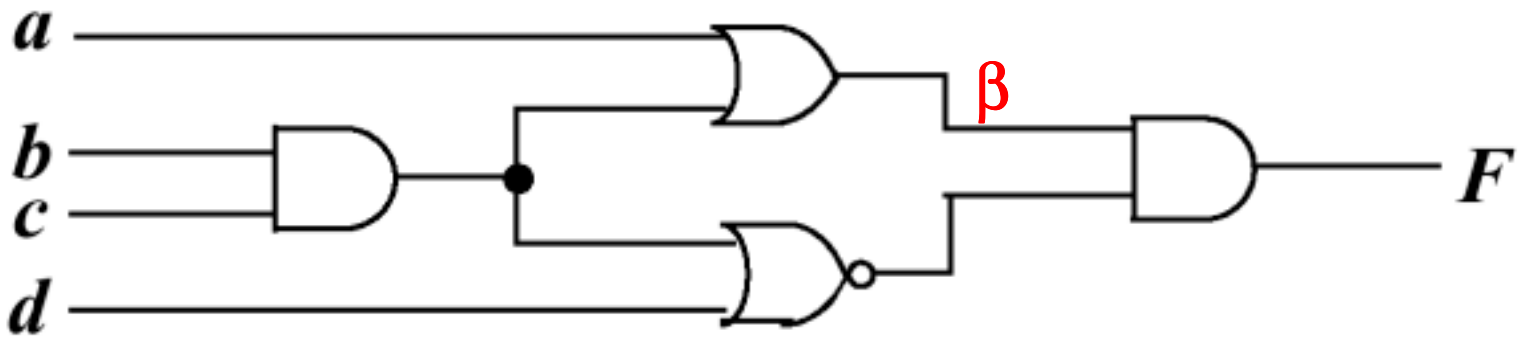
- Fault = a/1
- Fault-Free Fun= F(a,b,c,d) = $(a + b\text{c})(b' + c')d'$
- **T**(a/1) = a'.dF/da
- dF/da = [1$(b' + c')$d' $\oplus$ bc$(b' + c')d'$]
  = $b'd' + c'd'$
- So **T**(a/1) = $a'b'd' + a'c'd'$
- Test set for a/1 = {00x0, 0x00} = {0000, 0010, 0100}

Boolean Difference Method

- Boolean Difference (BD) has the following Properties:

1. $d\overline{F}/dx_i = d\mathrm{F}/dx\ i$
2. $\mathrm{d}/dx_i[d\mathrm{F}/dx_j\ ] = \mathrm{d}/dx_j\ [d\mathrm{F}/dx_i\ ]$
3. $\mathrm{d}[\mathrm{F(X)} \oplus \mathrm{G(X)}]/dx_i = d\mathrm{F}/dx_i \oplus d\mathrm{G}/dx_i$
4. $\mathrm{d}[\mathrm{F(X).G(X)}]/dx_i = \mathrm{F}.d\mathrm{G(X)}/dx_i \oplus \mathrm{G}.d\mathrm{F(X)}/dx_i \oplus d\mathrm{F(X)}/dx_i\ .d\mathrm{G(X)}/dx_i$
5. $\mathrm{d}[\mathrm{F(X)} + \mathrm{G(X)}]/dx_i = \overline{F}.d\mathrm{G}/dx_i \oplus \overline{G}.d\mathrm{F}/dx_i \oplus (d\mathrm{F}/dx_i).(d\mathrm{G}/dx_i)$

Example



- Fault = β/0
- **T (**β /0) = β(X).dF(x, β)/dβ
  - β(X) = $a + bc$ F(x, β) = $(b' + c')d'$ β
- dF/dβ = $(b' + c')d'$
- So **T(**β /0) = $ab'd' + ac'd'$
- Test set for β/0 = {1000,1010,1100}

Test Generation Schemes

- BD method is more Theoretical → Too Complex to Use for Practical Circuits
- Need a More Practical Approach
- **Deterministic Test Generation**
  - **Fault- Oriented ATG**
  - **Fault Independent ATG**
- **Random Test Generation**
  - **Combined Deterministic and Random Test Generation**
- **ATG Systems**
- **Conclusions**

Classification & Problems

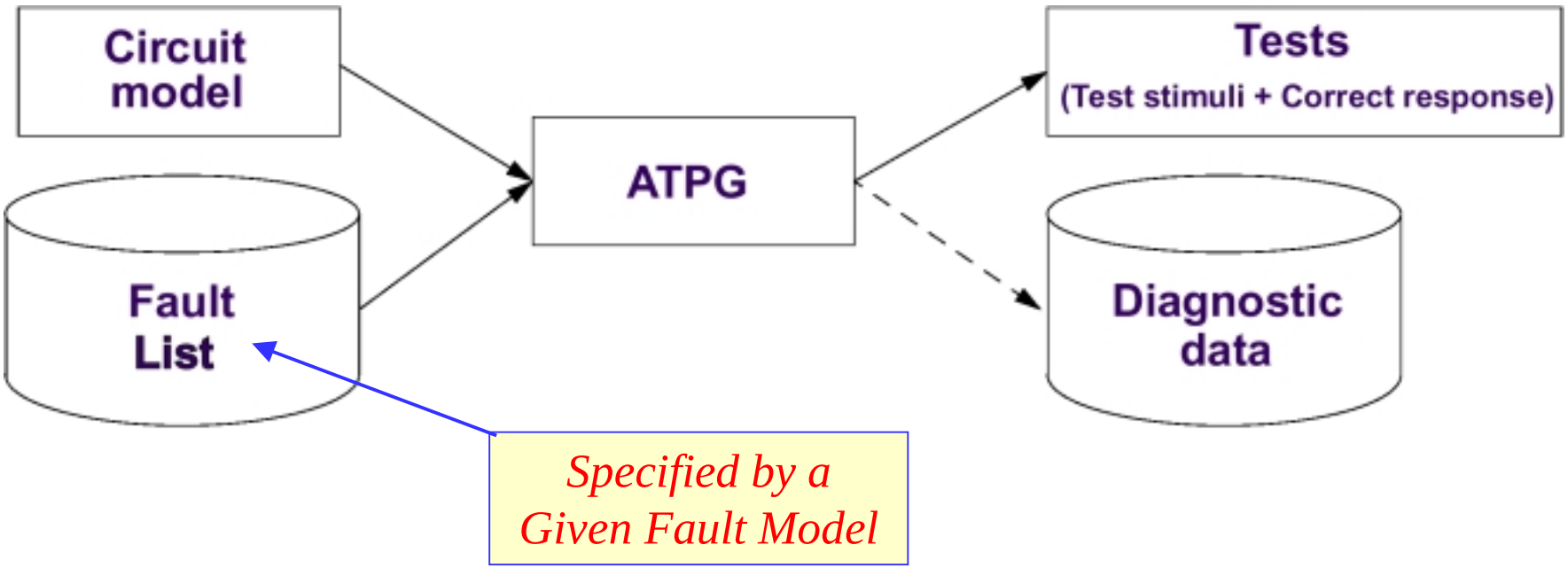
Testing

- Off- Line
- Edge- Pin
- Stored- Pattern
- Full Comparison of the Output Results

General Problems for TG:

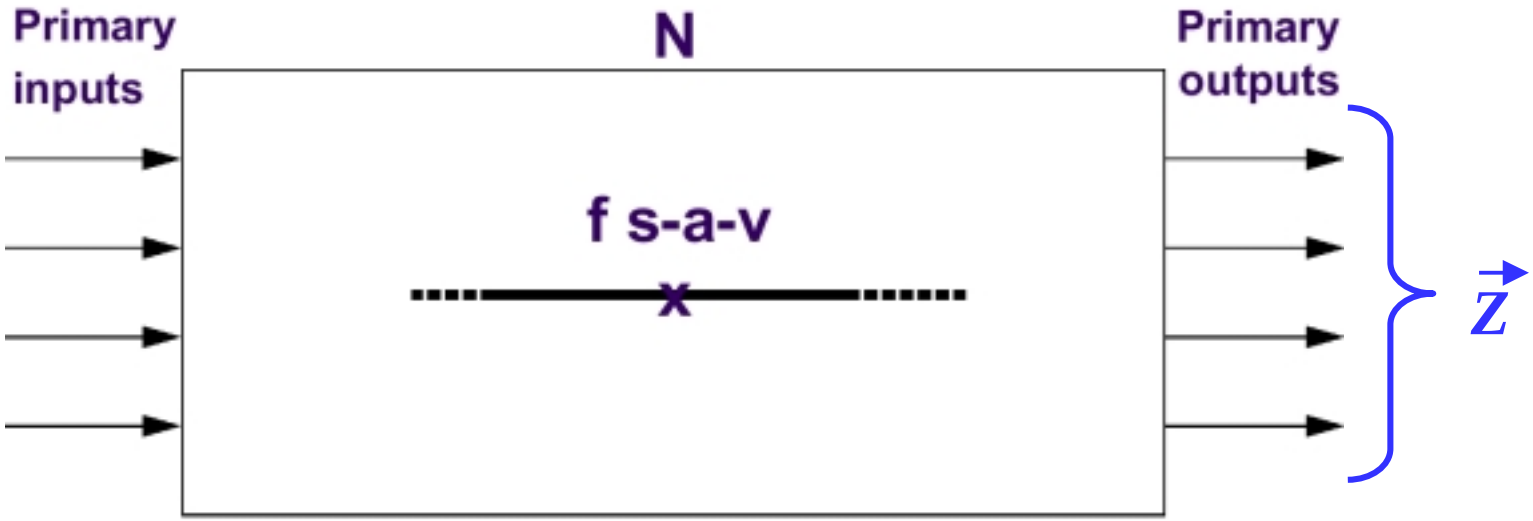
- Cost of Generating the Test
- Quality of the Generated Test
- Cost of Applying the Test

Deterministic Test Generation



- **Manual / Automatic**
  - **Automatic Test Pattern Generation (ATPG)**
- **Deterministic ATPG Can Be Fault- Oriented / Fault-Independent**

Fault-Oriented ATPG



- *Targets* a Particular Fault $f \in \{ \textit{Fault List} \}$
  - Finds a TV t which *Detects* f
    - t *Detects* f <=> $Z(t) \neq Z_f(t)$ <=> $Z(t) \oplus Z_f(t) = 1$

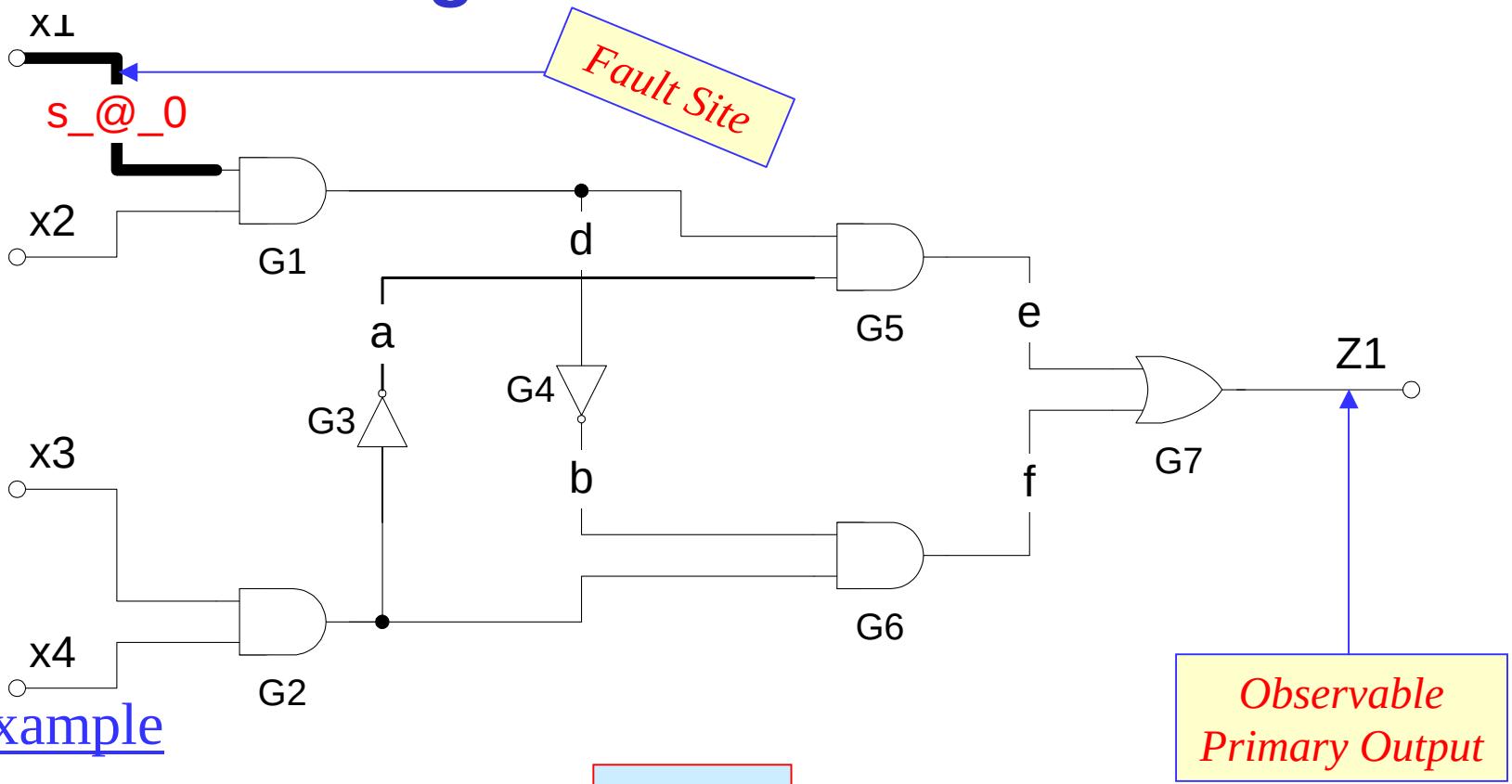
TV Generation

- Two Steps Are Necessary to Generate TV t which Detects Fault f:
  - Fault Activation/excitation/provoking: Effecting a Different Value $\bar{v}$ at the Faulty Line x When x Is Stuck at v.
  - Fault Propagation: Propagating Error to an Observable PO

Definitions

- A *Line* whose Value under TV t Changes in the Presence of Fault f is Said to Be *Sensitized* to the Fault f by the Test t
- A Path Composed of Sensitized Lines Is Called a *Sensitized Path*

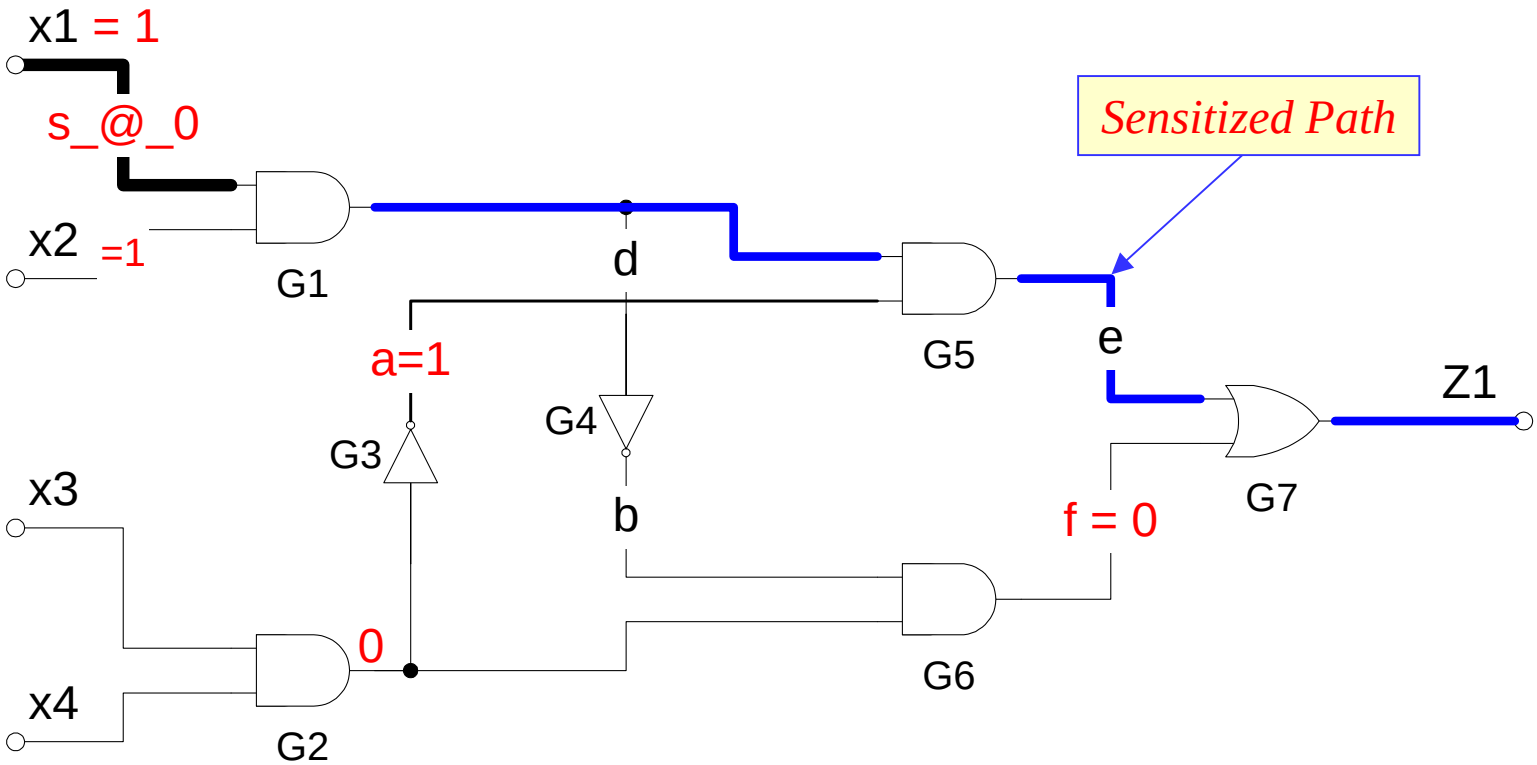
Single Path Sensitization



Example

1. Fault Activation → **X1=1**
2. Fault Propagation → 2 Possible Options (Paths)
   - Option 1: G1 – G5 – G7
   - Option 2: G1 – G4 – G6 –G7

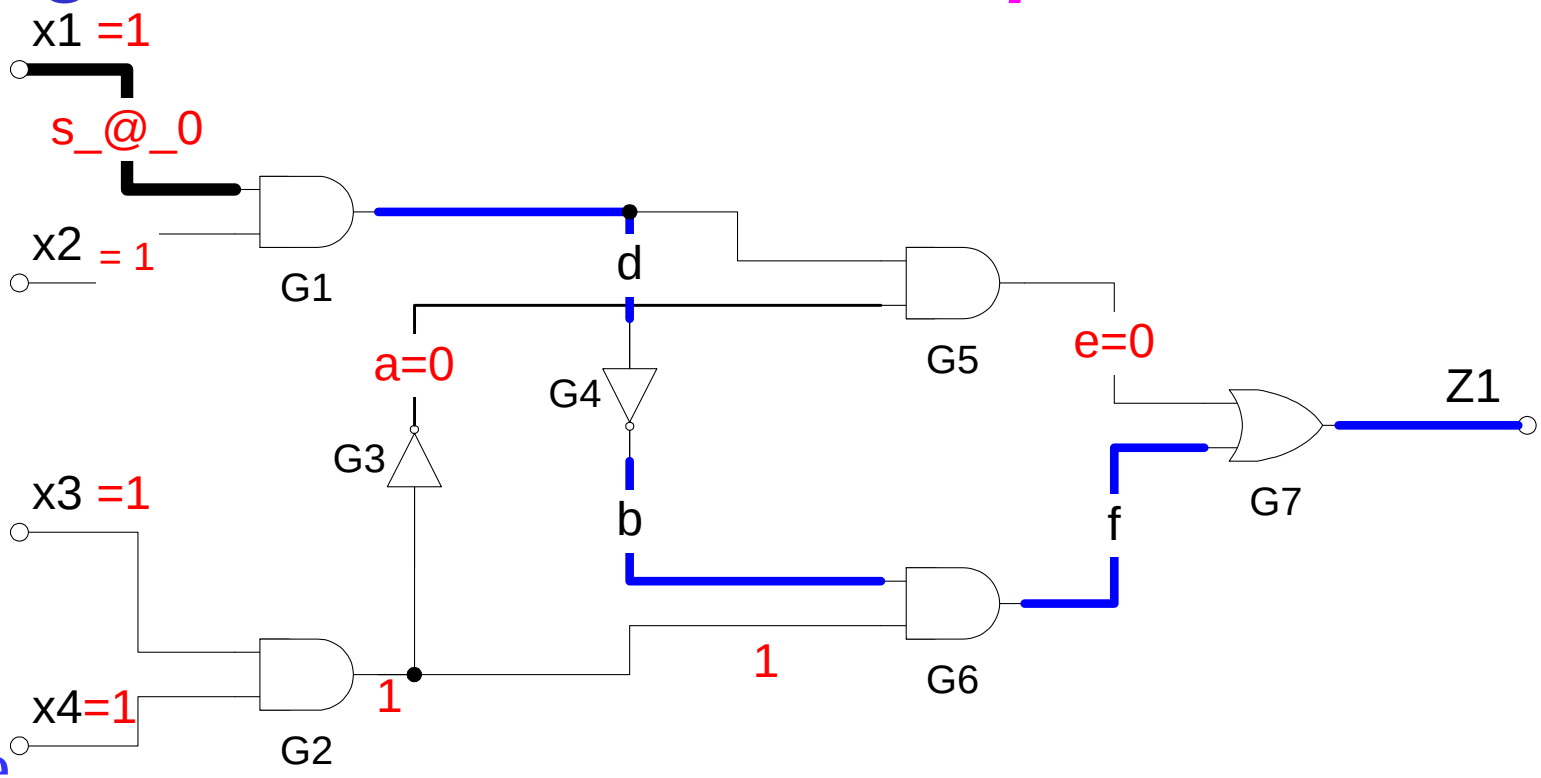
Single Path Sensitization *Option 1*



Example

1. Fault Activation → X1=1
2. Fault Propagation : G1 – G5 – G7 → *Two Possible TVs*
   - T1 = 110X , 1
   - T2 = 11X0 , 1

Single Path Sensitization *Option 2*



Example

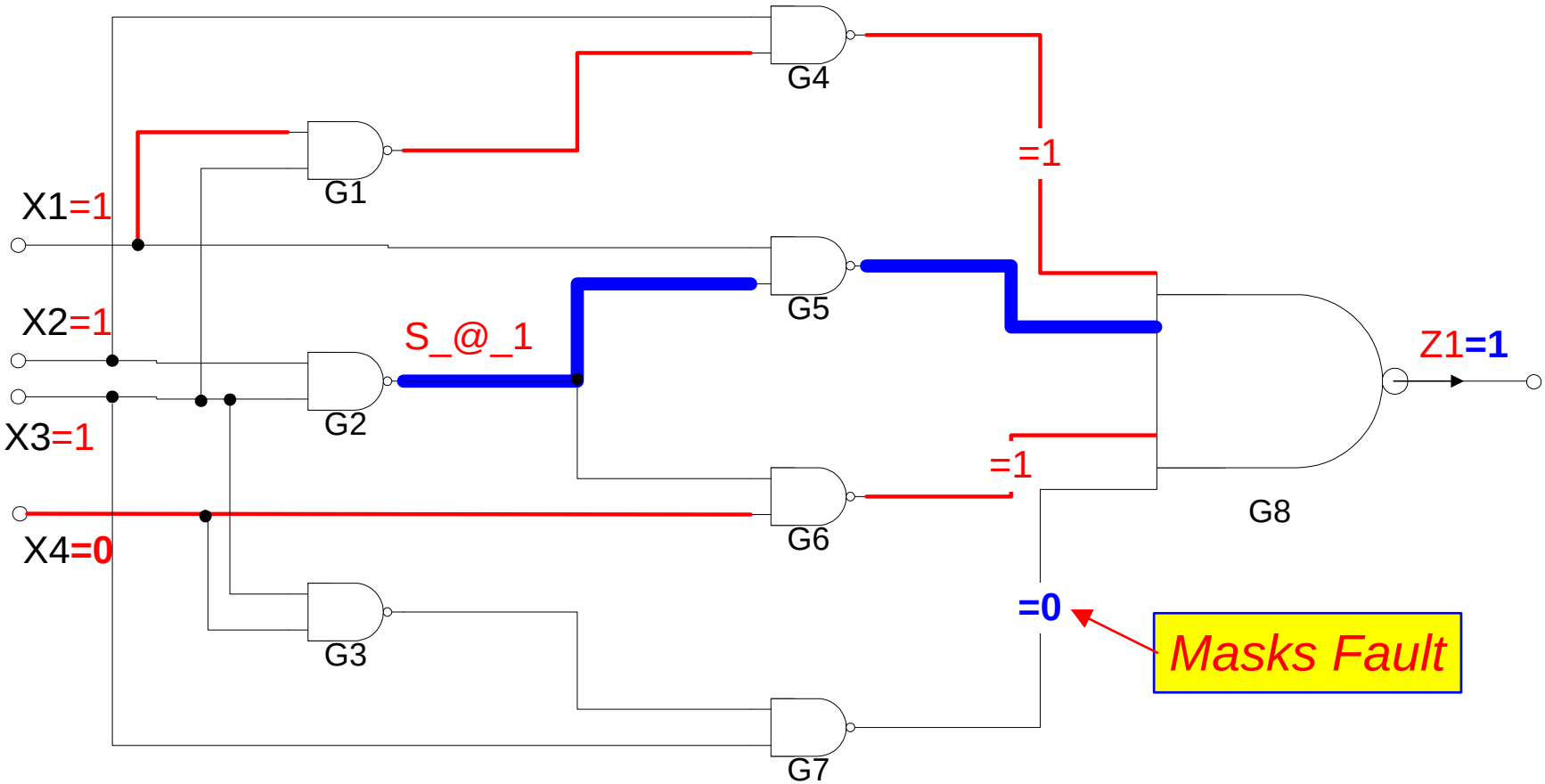
1. Fault Activation → X1=1
2. Fault Propagation : G1 – G4 – G6 - G7

→ *One Possible TV* → T = 1111 , 0

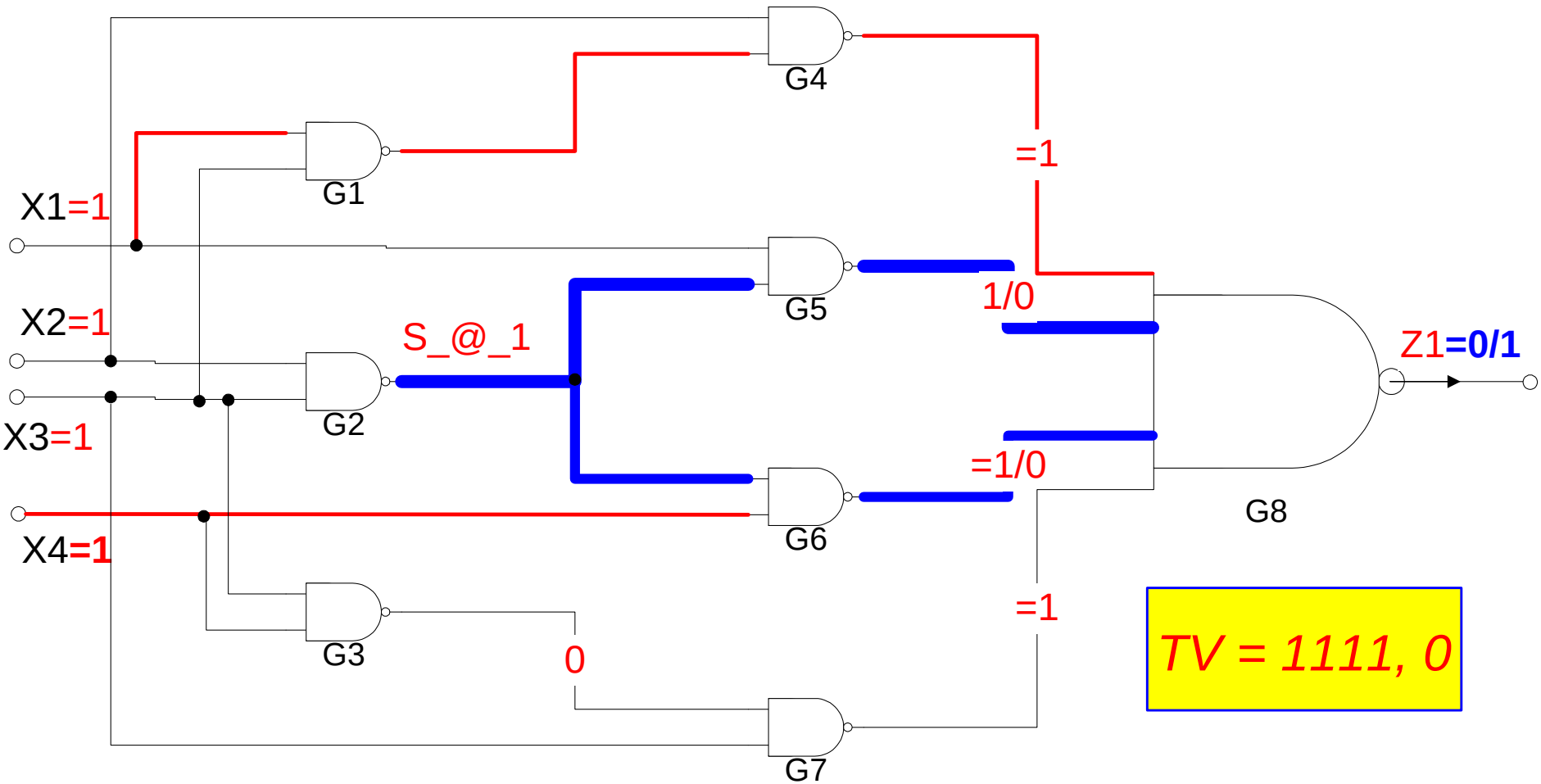
Multiple Path Sensitization

Major *Problem* with Single Path Sensitization

- A TV May not be Possible to Generate for Some *Testable* Faults if Only One Path is Sensitized at a Time



Multiple Path Sensitization



The D-Algorithm

- An Algorithmic Approach which Generates a TV for a Given Fault if One Exists
- Multiple Path Sensitization
- 5-Valued Logic { 0, 1, X, D, $\overline{D}$}
- **D**: *A Line is Assigned a* **D** *value if it has a value* ***1*** *in the* ***Fault-Free*** *Circuit but has a 0 Value in the Faulty Circuit.*
  (**D** ≈ S_@_0)
- $\overline{\mathbf{D}}$: *A Line is Assigned a* $\overline{\mathbf{D}}$ *value if it has a value* ***0*** *in the* ***Fault-Free*** *Circuit but has a 1 Value in the Faulty Circuit.*
  ($\overline{\mathbf{D}}$ ≈ S_@_1)
- D / $\overline{D}$ Follow Rules of Boolean Algebra

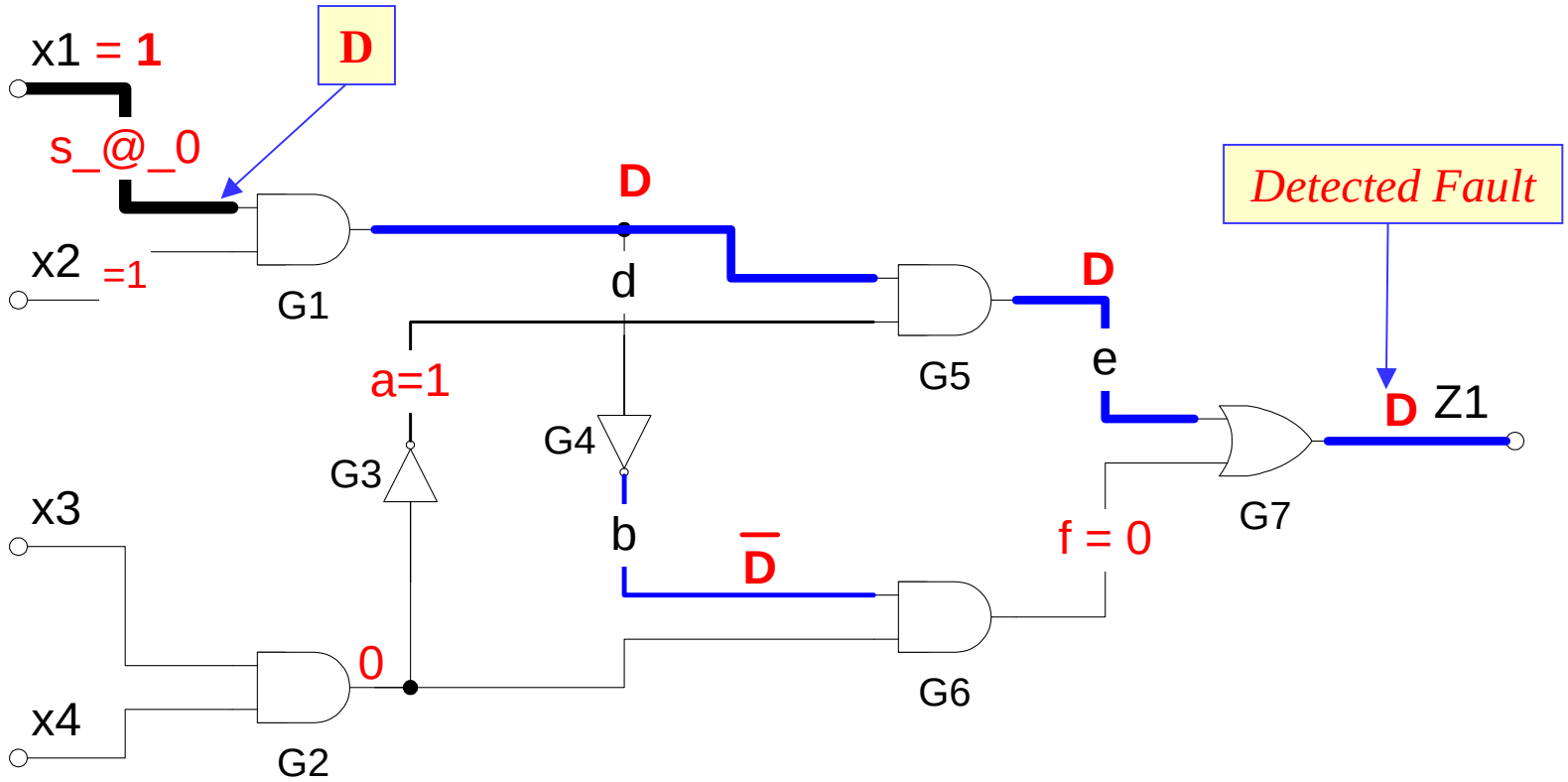
The D-Algorithm

$D + D = D$	$\overline{D} + \overline{D} = \overline{D}$
$D \cdot D = D$	$\overline{D} \cdot \overline{D} = \overline{D}$
$\overline{D} + D = 1$	$D \cdot \overline{D} = 0$
$D \cdot 1 = D$	$\overline{D} \cdot 1 = \overline{D}$
$D \cdot 0 = 0$	$\overline{D} \cdot 0 = 0$
$D + 1 = 1$	$\overline{D} + 1 = 1$
$D + 0 = D$	$\overline{D} + 0 = \overline{D}$

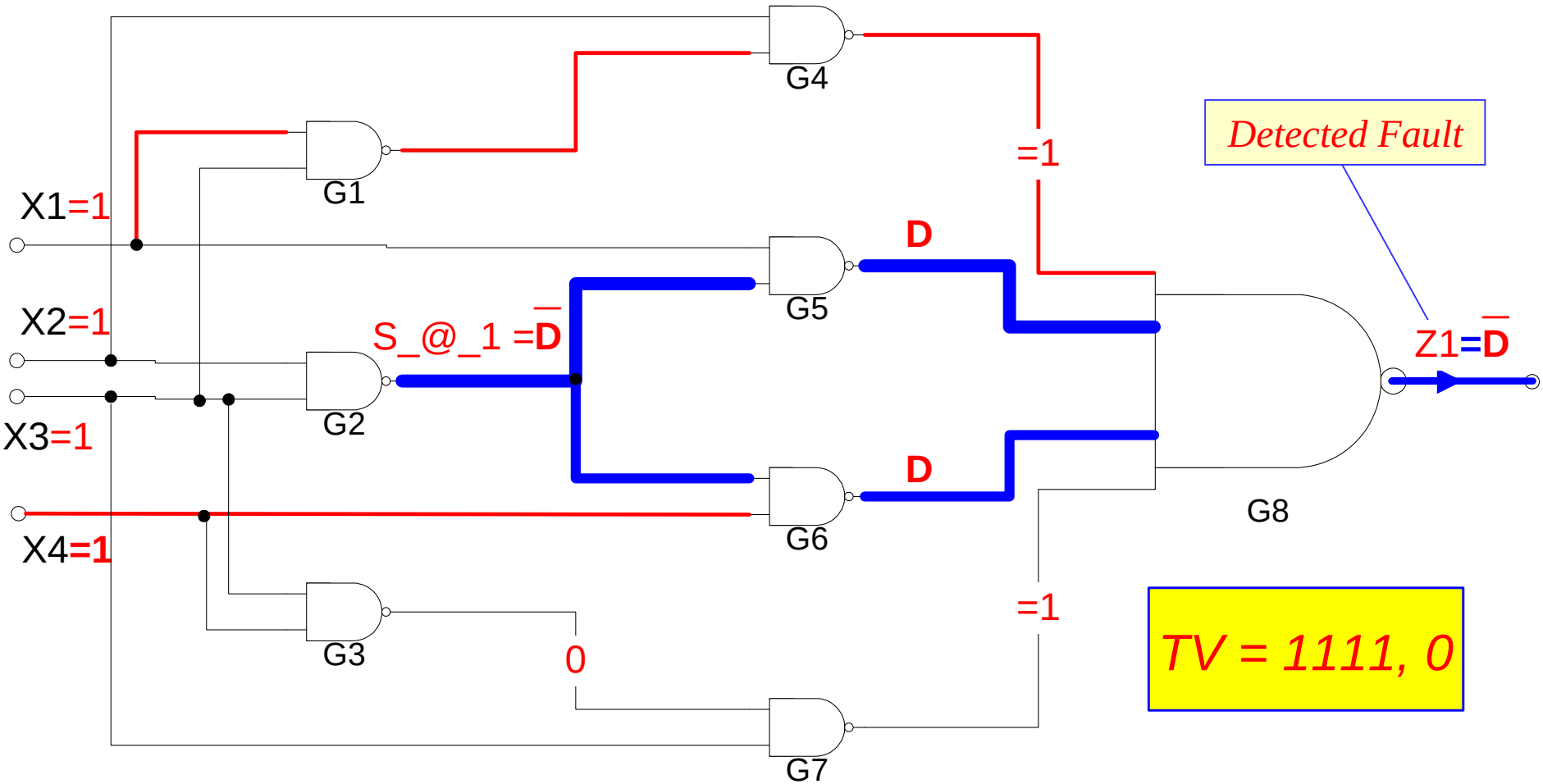
•	0	1	D	$\overline{D}$	x
0	0	0	0	0	0
1	0	1	D	$\overline{D}$	x
D	0	D	D	0	x
$\overline{D}$	0	$\overline{D}$	0	$\overline{D}$	x
x	0	X	x	x	x

AND Operation of the 5-Valued D-Calculus

Example - Single Path Sensitization



Example - Multiple Path Sensitization

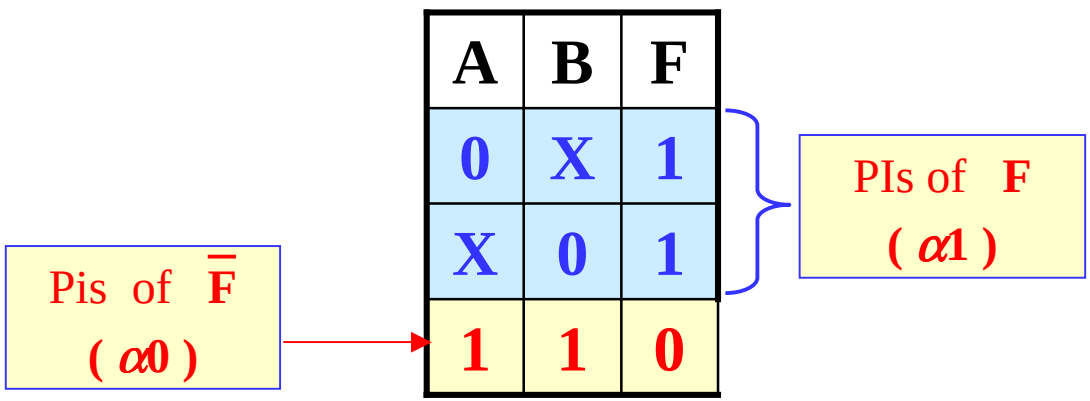


Definitions

1. Singular Covers (SC) of Some Function F (*Primitive Cubes of F*) : *Minimal Set of Logic Signal Assignments Showing Essential Prime Implicants*

   = Prime Implicants of $\overline{F}$ ($\alpha 1$) && Prime Implicants of F ($\alpha 0$)

Examples Singular Covers of 2-Input NAND Gate



Definitions-- Singular Covers (SC)

(*Primitive Cubes*)

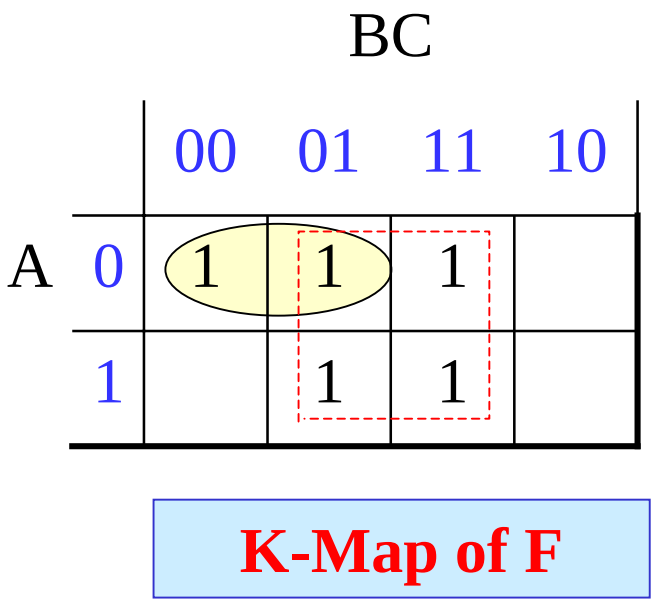
Example

A	B	C	F
X	X	1	1
0	0	X	1
X	1	0	0
1	X	0	0

PIs of **F** ($\alpha 1$)

PIs of $\overline{\mathbf{F}}$ ($\alpha 0$)

SCs of F



K-Map of F

Definitions

2. Primitive D-Cubes of a Fault (PDCF)

= Prime Implicants of the *Faulty* Function which Produce D / $\bar{D}$ Output

{I/P. Stimuli Required to Activate Faulty Condition at a Gate/Module Output}

Examples PDCF of 2-Input NAND Gate

A	B	F
0	X	1
X	0	1
1	1	0

PIs of $\bar{F}$ ($\alpha 0$)

PIs of F ($\alpha 1$)

A	B	F
D	1	$\bar{D}$
$\bar{D}$	1	D
1	D	$\bar{D}$
1	$\bar{D}$	D
D	D	$\bar{D}$
$\bar{D}$	$\bar{D}$	D

Definitions -- *Primitive D-Cubes of a Fault (PDCF)*

Computing PDCF of a General Module (Function)

1. Obtain the SCs of the Fault-Free Function ($\alpha 1$, $\alpha 0$)
2. Obtain the SCs of the Faulty Function ($\beta 1$, $\beta 0$)
3. PDCF for this module Result from Intersecting $\alpha 1$ with $\beta 0$ and $\alpha 0$ with $\beta 1$.

Thus $PDCF = \{ \alpha 1 \cap \beta 0 \ , \ \alpha 0 \cap \beta 1 \ \}$

Definitions-- -- *Primitive D-Cubes of a Fault (PDCF)* -- Example

Example

A	B	C	F
X	X	1	1
0	0	X	1
X	1	0	0
1	X	0	0

PIs of F ($\alpha 1$)

PIs of $\overline{F}$ ($\alpha 0$)

A	B	C	F
X	X	1	1
0	1	X	1
X	0	0	0
1	X	0	0

PIs of F ($\beta 1$)

PIs of $\overline{F}$ ($\beta 0$)

PDCF

A	B	C	F
0	0	0	D
0	1	0	D

$\alpha 0 \cap \beta 1$

$\alpha 1 \cap \beta 0$

Definitions

3. Propagation D-Cube of a Gate/Module (PDC)

= Prime Implicants of the Function which Allow a $D/\overline{D}$ Values on Inputs to Propagate to the Output

- Let { $\alpha 0$, $\alpha 1$ } be the Singular Covers of the Function F, Where;
  - $\alpha 1$ = Be the Prime Implicants of **F**
  - $\alpha 0$ = Be the Prime Implicants of $\overline{\mathbf{F}}$

$$PDC = \{ \alpha 1 \cap \alpha 0 \}$$

Definitions-- *Propagation D-Cube (PDC)*

Example PDCF of 2-Input NAND Gate

	A	B	F
r1	0	X	1
r2	X	0	1
r3	1	1	0

PIs of $\overline{F}$ ($\alpha 0$)

PIs of F ($\alpha 1$)

Singular Covers

	A	B	F
r2 ∩ *r3*	1	D	$\overline{D}$
	1	$\overline{D}$	D
r1 ∩ *r3*	D	1	$\overline{D}$
	$\overline{D}$	1	D
Multiple Faults on Inputs	D	D	$\overline{D}$
	$\overline{D}$	$\overline{D}$	D

Propagation D-Cubes

Intersection of D-Cubes

- Let A = $(a_1, a_2, a_3, \ldots, a_n)$, and
- Let B = $(b_1, b_2, b_3, \ldots, b_n)$ be 2 D-Cubes, where

$$a_i, b_i \in \{ 0, 1, x, D, \overline{D}\}$$

- ***The D-intersection*: of A and B, denoted $A \cap B$ is given by $\{a_i \cap b_i \mid a_i \in A, b_i \in B\}$, where :**

1. $x \cap a_i = a_i$
2. If ($a_i \neq x$ and $b_i \neq x$) Then

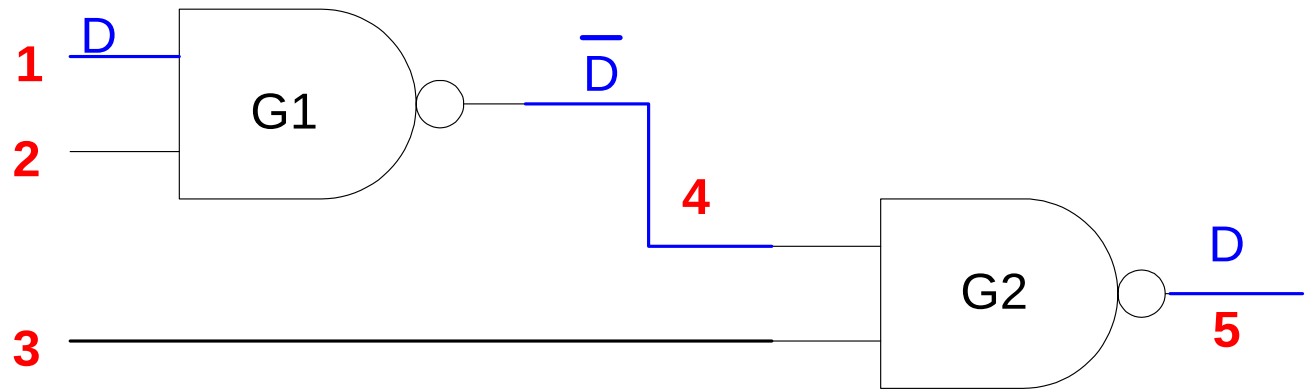
$$a_i \cap b_i = \begin{cases} a_i & IF \quad a_i = b_i \\ \Phi & IF \quad a_i \neq b_i \end{cases}$$

3. $A \cap B = \Phi$ "Empty Cube" IF $a_i \cap b_i = \Phi$ For Any i

D- Intersection

- D-Intersection is Used to Generate Sensitized Paths (**D-Drive**)

Example PDC of 2-Input NAND Gate →



A	B	F
1	D	$\overline{D}$
1	$\overline{D}$	D
D	1	$\overline{D}$
$\overline{D}$	1	D
D	D	$\overline{D}$
$\overline{D}$	$\overline{D}$	D

1	2	3	4	5
D	1	*x*	$\overline{D}$	*x*

$\cap$

1	2	3	4	5
x	*x*	1	$\overline{D}$	D

=

1	2	3	4	5
D	1	1	$\overline{D}$	D

D- Algorithm

1. Initialize Test Cube to all *x*'s (*x, x, x, … x*)
2. Fault Provoking → Select a Primitive D-Cube of the Fault (PDCF) → Usually a *Choice Step*
3. Path Sensitization From the Faulty Line To a **PO**
   - ➢ Successive X-ion of Test Cube with the Propagation D-Cubes of Successor Gates (*D-Frontier*) Till a **PO** Gets a **D / $\overline{\mathbf{D}}$** Value (*Choice Step*) → This Step is known as *D-Drive*
   - ➢ This Step Requires *2 Major Procedures*

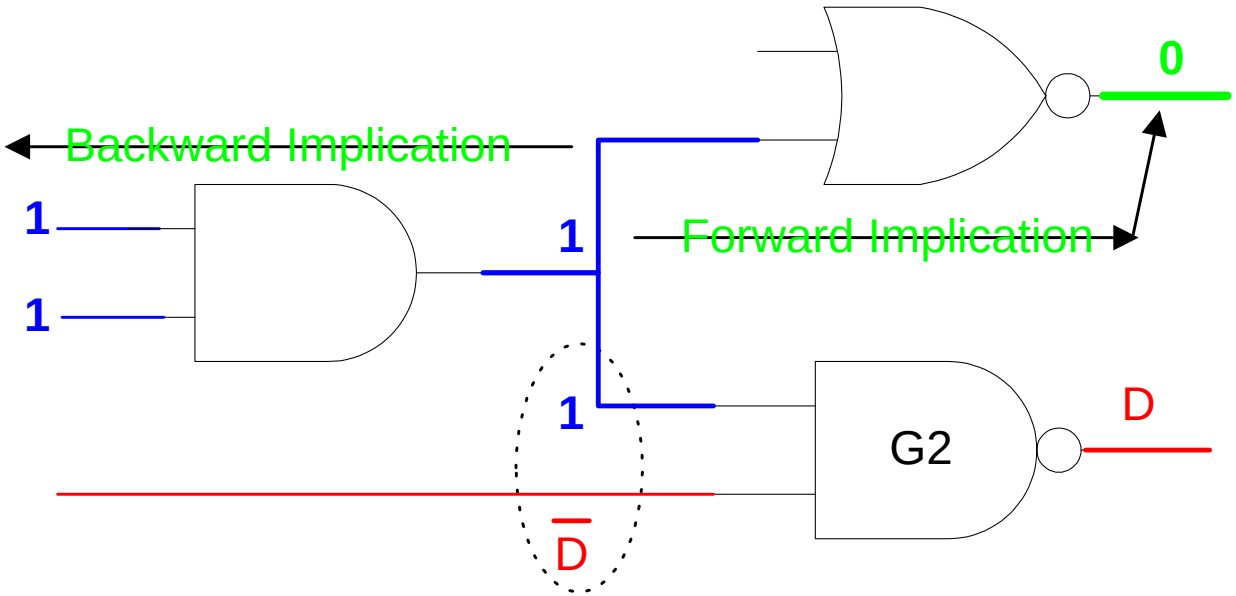
   a. Implication (Forward and Backward), and

   b. Line Justification (***Consistency Checks***)→ May Lead to *Backtracks* if Inconsistencies are Detected.

Implication

- Assignments Made Due to Choices, e.g.a Propagation D-Cube, Usually Uniquely Imply Other Signal Values {Forward & Backward}

Example



Chosen Propagation Cube

D-Frontier

- The D-Frontier Consists of **ALL** Gates whose Current O/P Value is "***x***" but have one or more **D/$\overline{\mathbf{D}}$** on their inputs
- One of the D-Frontier Gates is Usually Chosen to Propagate The Fault → "D-Drive"

Line Justification

Is a Backward *Implication Step* Where The Gate I/P Values are Selected *To Justify* the Specified Gate Output. This Step is Repeated Till the Relevant **PIs** are Defined.

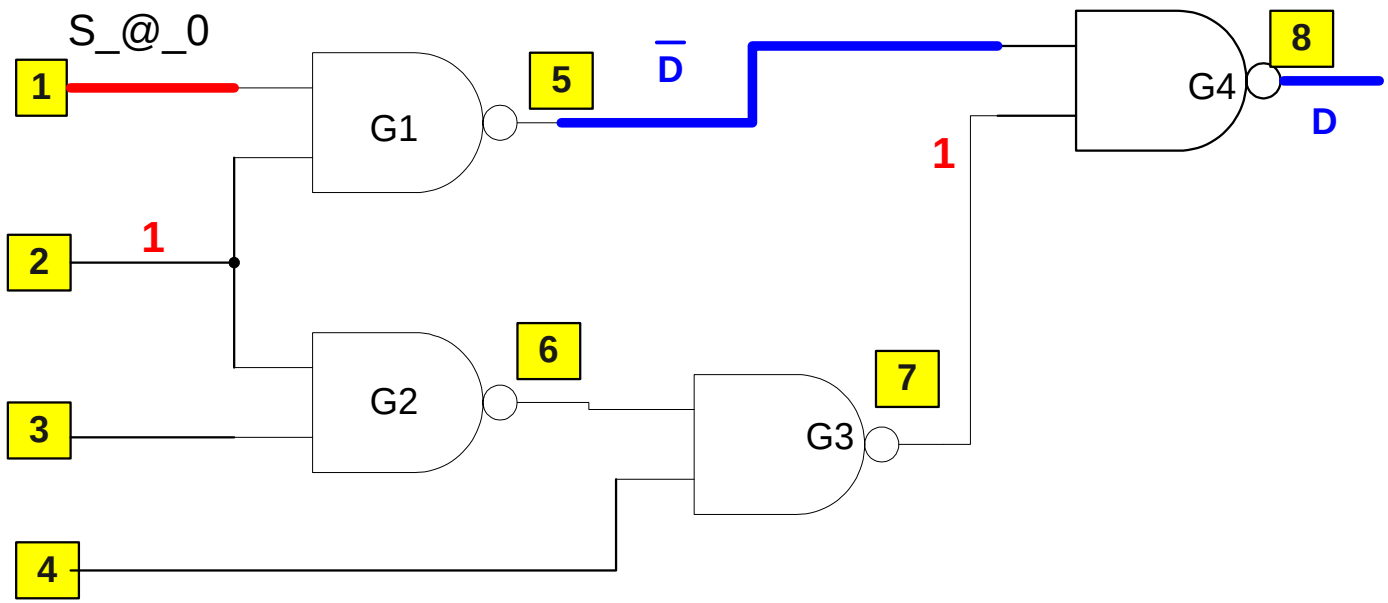
- *Line Justification* is Performed after D/$\overline{\text{D}}$ Appear at Some **PO**.

Line Justification-Example

- An Unjustified Line is a Defined Gate Output which is not Implied By The Gate Inputs

J-Frontier

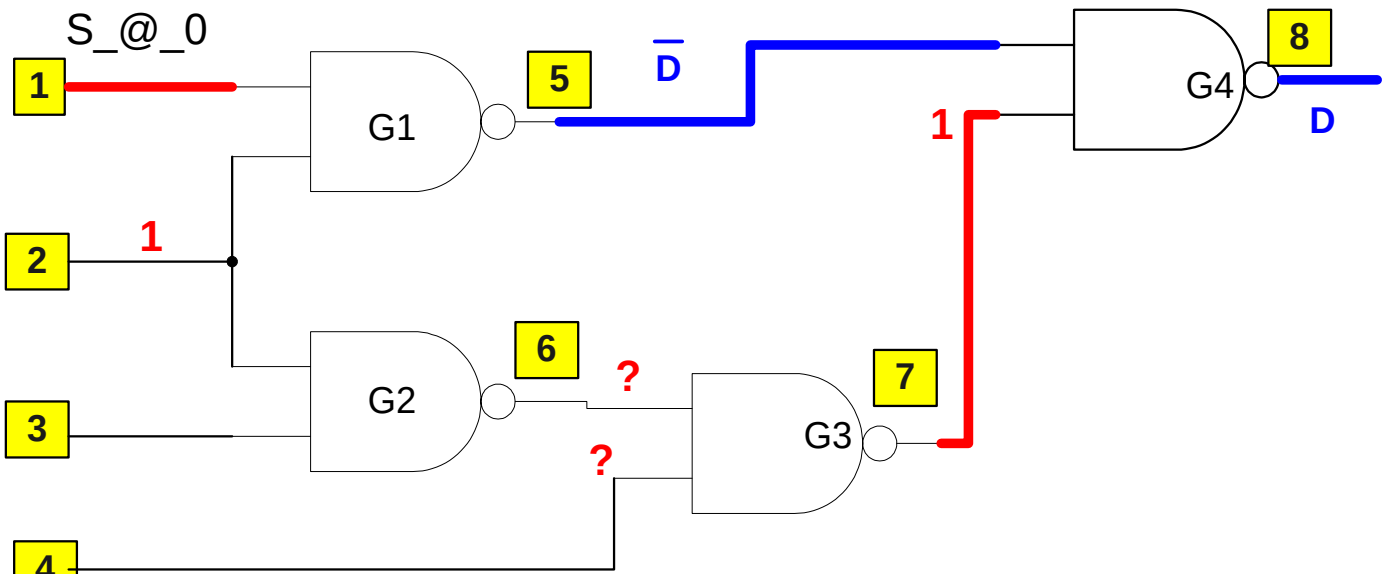
Is the Set of ALL Gates whose Output Lines are *Unjustified*



Line Justification

Example (*Line Justification*) → Line 1 S_@_0

$t^0 = PDFC^1$



1	2	3	4	5	6	7	8
D	1			$\overline{D}$			
D	1			$\overline{D}$		1	D

D-Drive

J-Frontier = {G3}"*Line 7 Unjustified*"

Either Line 4 = 0, OR

Line 6=0

	1	2	3	4	5	6	7	8	
$t^2=t^1 \cap Sc^3$	D	1	*x*	*x*	$\overline{D}$	0	1	D	$J\_F=\{G_2\}$
$t^3=t^2 \cap Sc^2$	D	1	1	*x*	$\overline{D}$	0	1	D	$J\_F=\{\Phi\}$

NOTES on D-ALGORITHM

1. Exhaustive Search of All Possible Choices
2. Guaranteed to Find a Solution (TV) *IF One Exists*.
3. Stops As Soon As A Solution Is Found
4. Being Exhaustive, the Worst Case Complexity is Exponential in the # of Gates
5. The Best Case Behavior Occurs When a Solution IS Generated *WITHOUT ANY BACKTRACKING*:
   - Always Correct Decisions Are Made
   - Solution Is Obtained Only Through Forward & Backward Implication.

NOTES on D-ALGORITHM

6. THUS, the # of Backtracks Determines the Complexity of the Algorithm (Should be Minimized).

7. An Upper Limit is Placed on the # of Backtracks Beyond Which a Fault Is Declared *UNTESTABLE*!

8. To Minimize the # of Backtracks, It is Advisable to Discover Inconsistencies as Early as Possible Through *Forward & Backward Implications for Each Change in the Test Cube*

OutLine of D-Algorithm

1. Model Fault with appropriate *Primitive D-cube of Fault* (PDF)
2. Select *Propagation D-cubes* to Propagate fault effect to a circuit output (*D-drive* procedure)
3. Select *singular cover* cubes to justify internal circuit signals (*Line Justification / Consistency* procedure)

- Put Signal Assignments in *Test Cube*
- Regrettably, Cubes Are Selected Very *Arbitrarily* by D-ALG