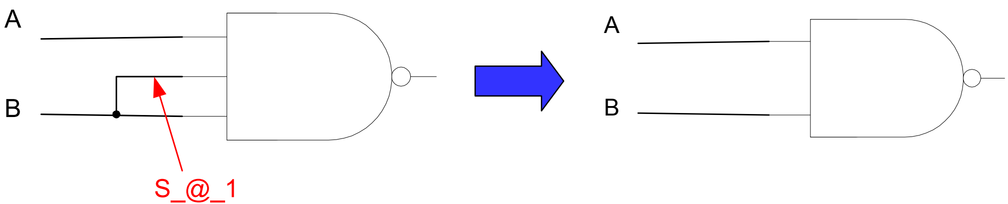
Combinational Circuit Testing

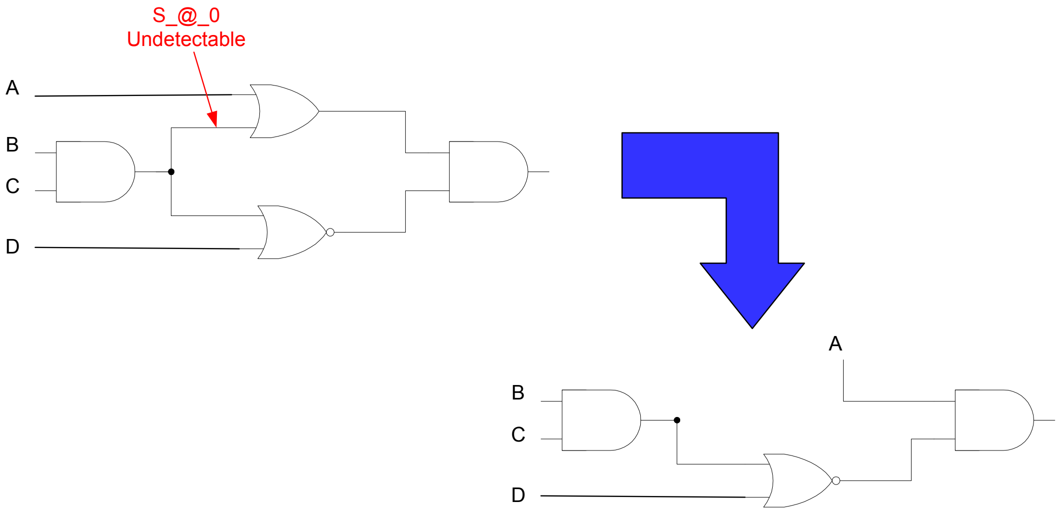
Basic Definitions

- 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.

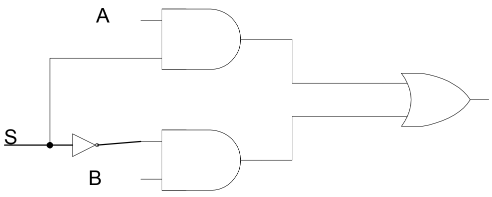


Combinational Circuit Testing

Another Example

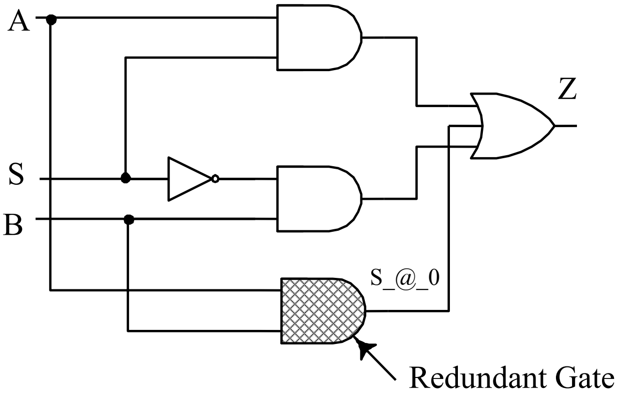


Redundancy



Hazardous Circuit

(a)



Hazard-Free Circuit

(b)

Combinational Circuit Testing

Basic Definitions (cont'd)

- A set of tests $T = \{t_1, t_2, \ldots, t_k\}$ is *complete* if it detects (covers) all detectable SSL faults in the circuit
- We can represent any test set $\{t_1, t_2, \ldots, t_k\}$ by the Boolean function whose minterms are $\{t_1, t_2, \ldots, t_k\}$.
- 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 :
  $\boxed{T_0 = x_i (F_i(1) \oplus F_i(0))}$, Likewise
- The Test Set to Detect $x_i/1$ corresponds to :
  $\boxed{T_1 = \overline{x}_i (F_i(1) \oplus F_i(0))}$

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 :

$$\boxed{T_0 = x_i \cdot (dF/dx_i)}$$

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

$$\boxed{T_1 = \overline{x}_i \cdot (dF/dx_i)}$$

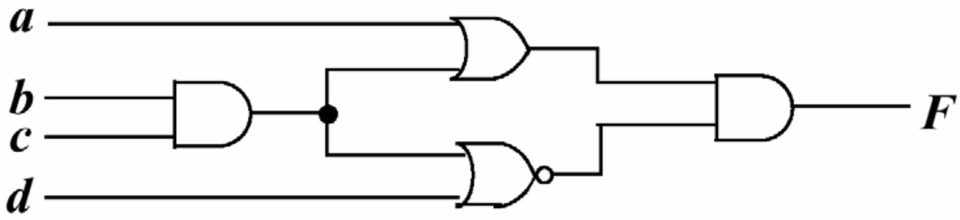
Notes:

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

$$\boxed{F_i(1) = F_i(0)} \quad OR \quad \boxed{dF/dx_i = 0}$$

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

Example



- Fault = a/1
- Fault-Free Fun= $F(a,b,c,d) = (a + bc)(b' + c')d'$
- **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. $d\overline{F}/dx_i = dF/dx\ i$

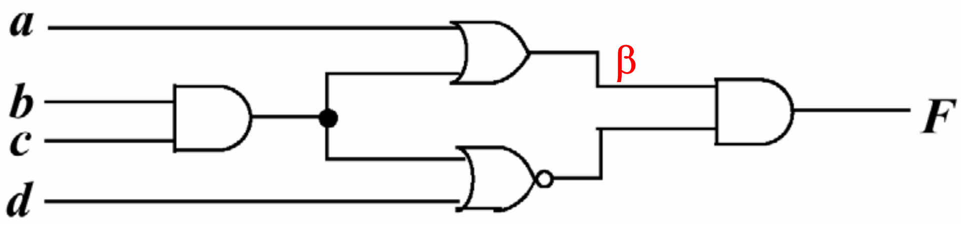
2. $d/dx_i[dF/dx_j] = d/dx_j[dF/dx_i]$

3. $d[F(X) \oplus G(X)]/dx_i = dF/dx_i \oplus dG/dx_i$

4. $d[F(X).G(X)]/dx_i = F.dG(X)/dx_i \oplus G.dF(X)/dx_i \oplus dF(X)/dx_i\ .dG(X)/dx_i$

5. $d[F(X) + G(X)]/dx_i = \overline{F}.dG/dx_i \oplus \overline{G}.dF/dx_i \oplus (dF/dx_i).(dG/dx_i)$

Example



- Fault = β/0
- **T** (β /0) = β(X).$dF(x, \beta)/d\beta$
  - ➢ $\beta(X) = a + bc$ $F(x, \beta) = (b' + c')d'\beta$
- $dF/d\beta = (b' + c')d'$
- 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

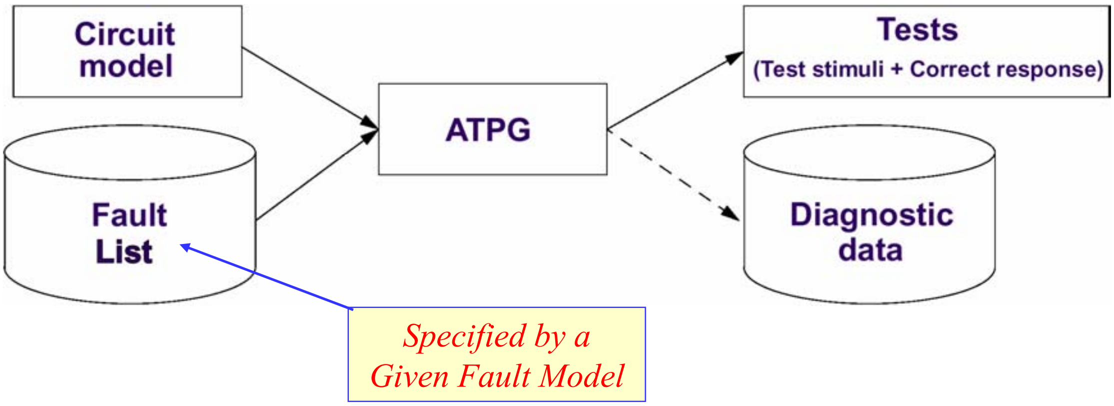
Testing

- **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



Circuit model

Fault List

ATPG

Tests

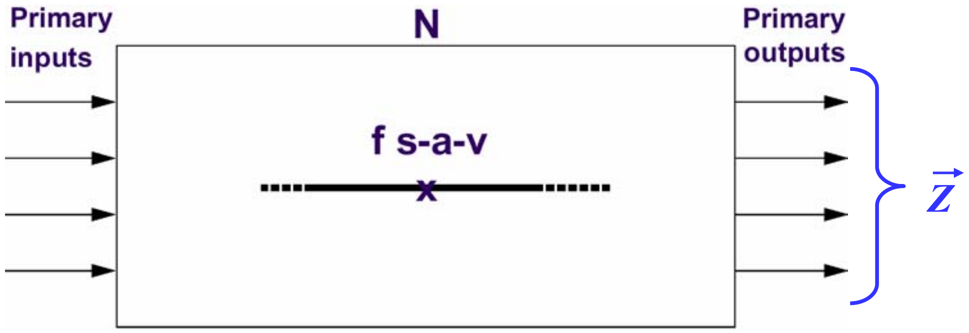
(Test stimuli + Correct response)

Diagnostic data

Specified by a Given Fault Model

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

Fault-Oriented ATPG



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

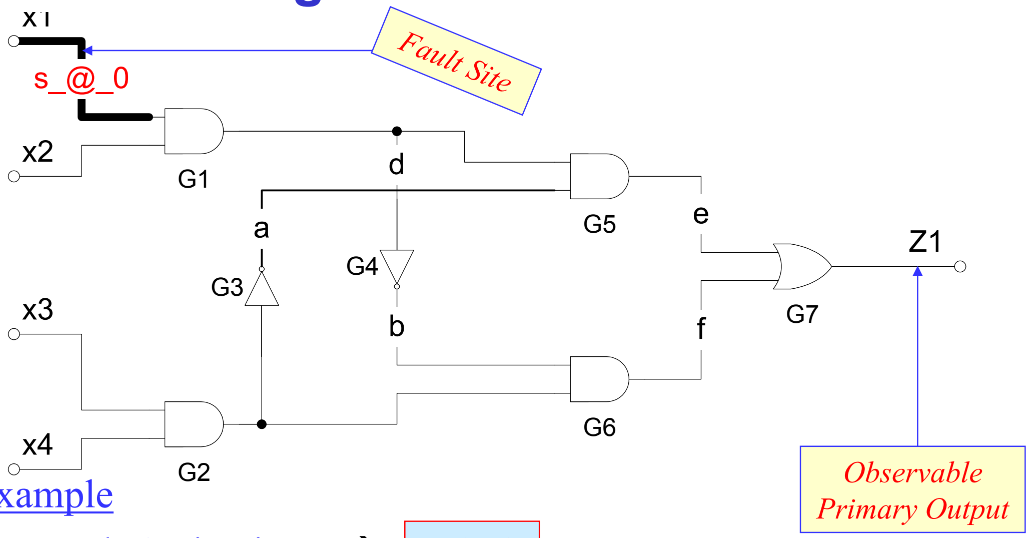
TV Generation

- Two Steps Are Necessary to Generate TV *t* which Detects Fault *f* :
  - Fault Activation/excitation/provoking: Effecting a Different Value $\overline{v}$ 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



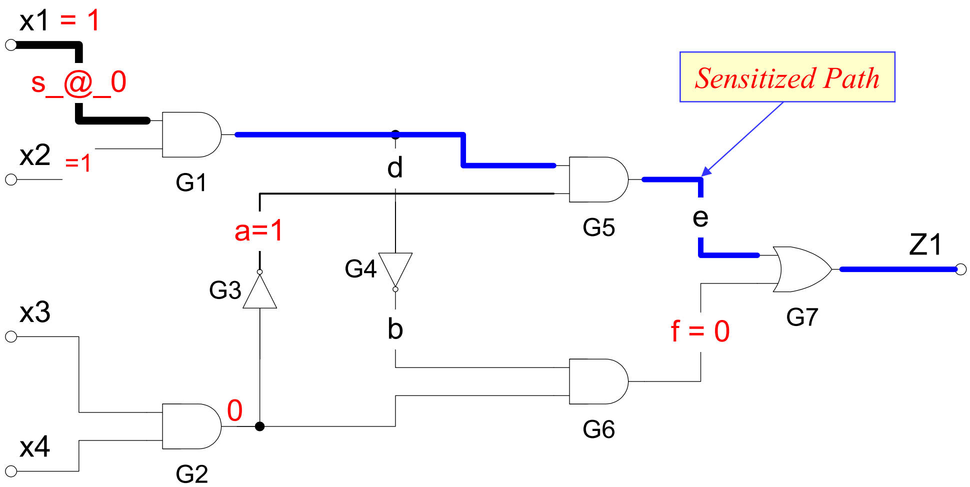
Fault Site

Observable Primary Output

Example

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

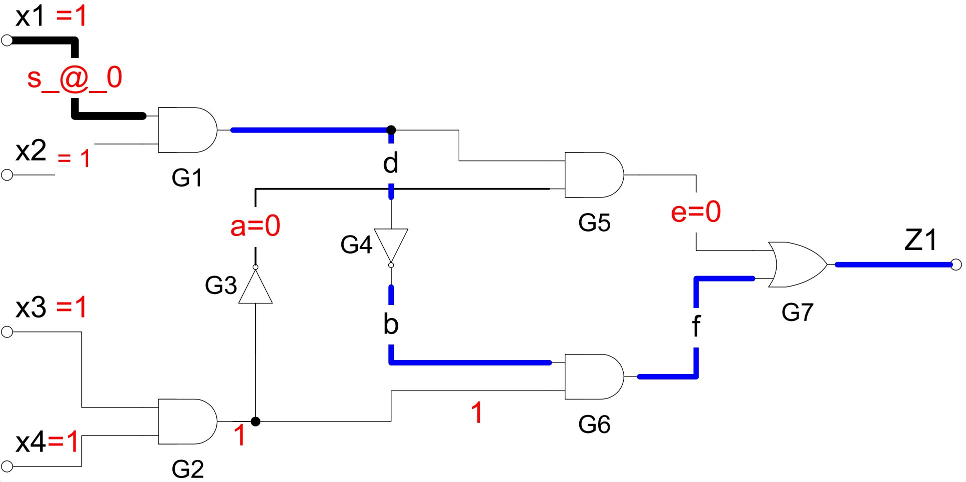
Single Path Sensitization *Option 1*



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

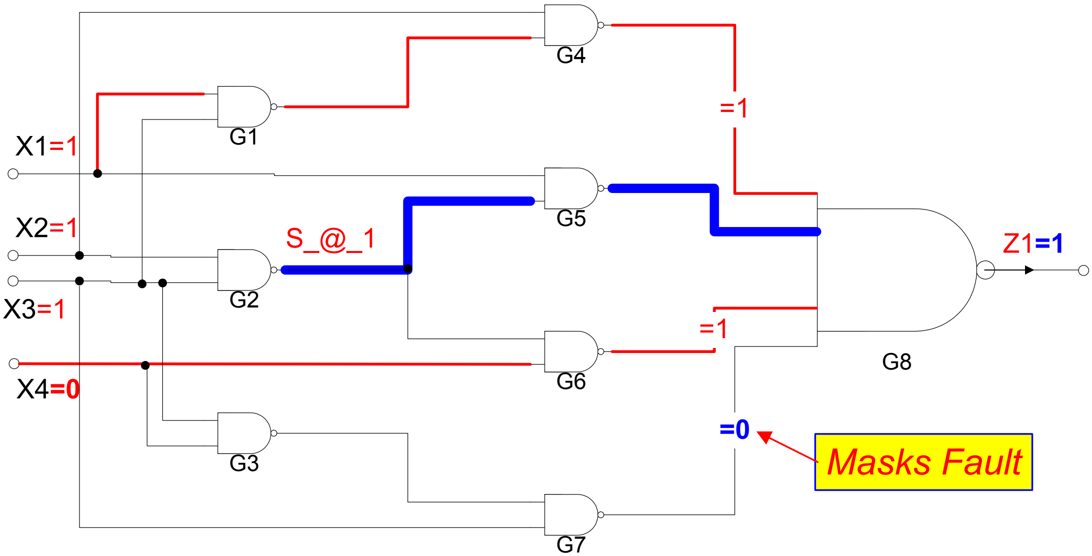
1. Fault Activation → X1=1
2. Fault Propagation : G1 – G4 – G6 - G7

→ *One Possible TV* → T = 1111 , 0

Multiple Path Sensitization

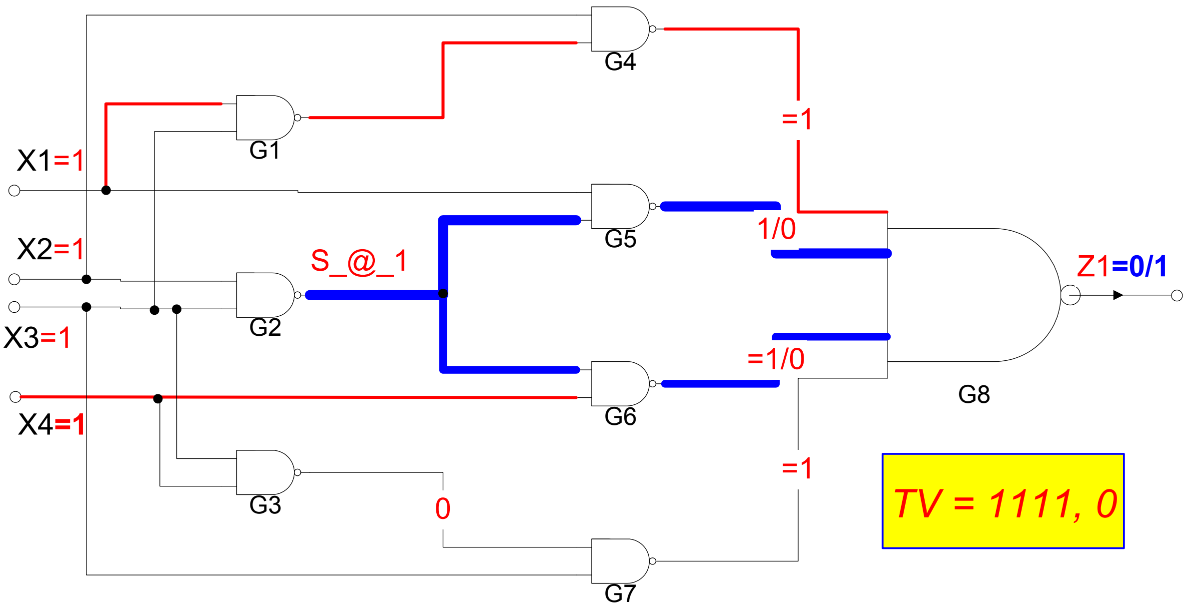
Major *Problem* with Single Path Sensitization

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



Masks Fault

Multiple Path Sensitization



TV = 1111, 0

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)
- **$\overline{D}$**: *A Line is Assigned a* **$\overline{D}$** *value if it has a value* ***0*** *in the* ***Fault-Free*** *Circuit but has a 1 Value in the Faulty Circuit.*
  (**$\overline{D}$** ≈ S_@_1)
- D / $\overline{D}$ Follow Rules of Boolean Algebra

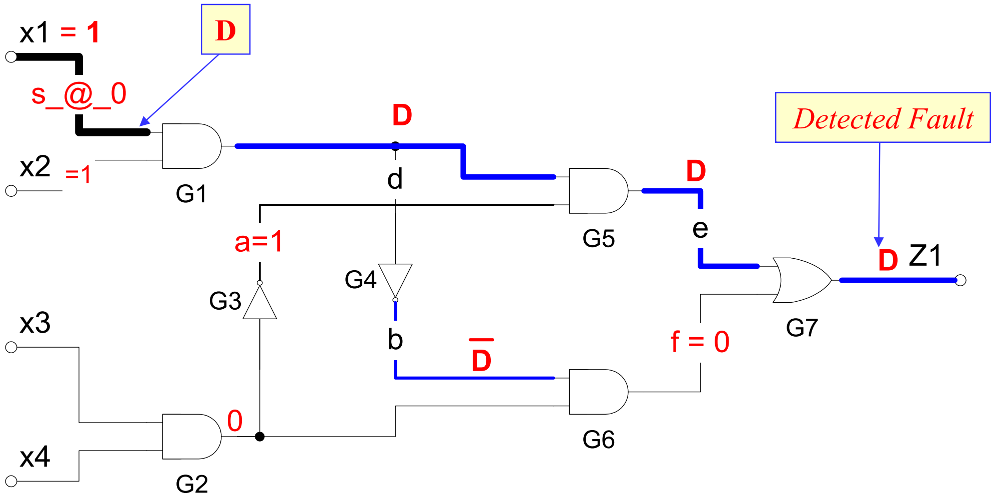
The D-Algorithm

$D + D = D$	**$\overline{D} + \overline{D} = \overline{D}$**
$D \,.\, D = D$	**$\overline{D} \,.\, \overline{D} = \overline{D}$**
$\overline{D} + D = 1$	**$D \,.\, \overline{D} = 0$**
$D \,.\, 1 = D$	**$\overline{D} \,.\, 1 = \overline{D}$**
$D \,.\, 0 = 0$	**$\overline{D} \,.\, 0 = 0$**
$D + 1 = 1$	**$\overline{D} + 1 = 1$**
$D + 0 = D$	**$\overline{D} + 0 = \overline{D}$**

•	**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

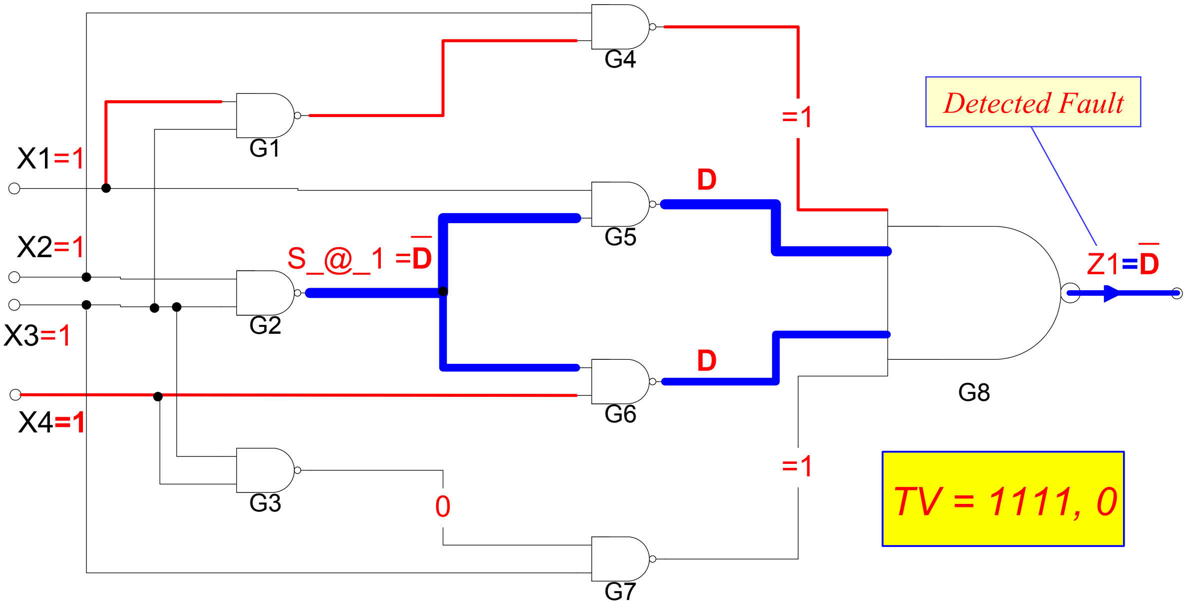
AND Operation of the 5-Valued D-Calculus

Example - Single Path Sensitization



Detected Fault

Example - Multiple Path Sensitization



Detected Fault

TV = 1111, 0

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 **F** **(α1)** &&
Prime Implicants of **$\overline{F}$** **(α0)**

Examples Singular Covers of 2-Input NAND Gate

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

PIs of **F** (α**1**)

Pis of **$\overline{F}$** (α**0**)

Definitions-- Singular Covers (SC)

(*Primitive Cubes*)

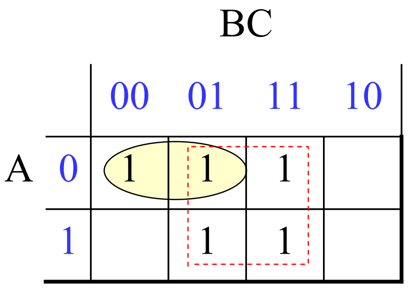
Example

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

PIs of **F** (α**1**)

PIs of **$\overline{F}$** (α **0**)

SCs of F



A \ BC	00	01	11	10
0	1	1	1	
1		1	1	

K-Map of F

Definitions

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

= Prime Implicants of the *Faulty* Function which Produce D / $\overline{D}$ 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 F
(α1)

PIs of $\overline{F}$
(α0)

A	B	F
D	1	$\overline{D}$
$\overline{D}$	1	D
1	D	$\overline{D}$
1	$\overline{D}$	D
D	D	$\overline{D}$
$\overline{D}$	$\overline{D}$	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 (α1, α0)
2. Obtain the SCs of the Faulty Function (β1, β0)
3. PDCF for this module Result from Intersecting α1 with β0 and α0 with β1.

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

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

$\alpha 1 \cap \beta 0$

$\alpha 0 \cap \beta 1$

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 $\overline{\mathbf{F}}$

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

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

Example PDCF of 2-Input NAND Gate

	A	B	F	
r1	0	X	1	PIs of F (α1)
r2	X	0	1	
r3	1	1	0	PIs of $\overline{F}$ (α0)

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

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

- ***The D-intersection*: of A and B, denoted A ∩ 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

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

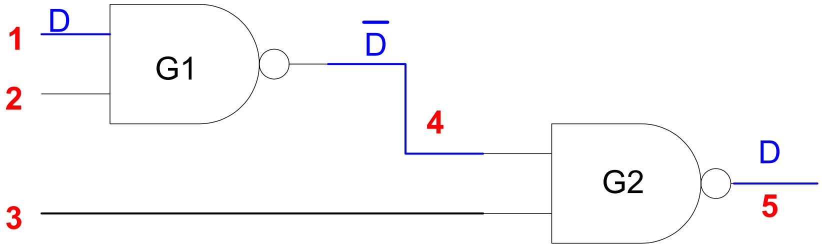
3. $A \cap B = \Phi$ "Empty Cube" IF $a_i \cap b_i = \Phi$ For Any i

D- Intersection

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

Example PDC of 2-Input NAND Gate →

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



$$\begin{array}{|c|c|c|c|c|}\hline 1&2&3&4&5\\\hline D&1&x&\bar{D}&x\\\hline\end{array} \cap \begin{array}{|c|c|c|c|c|}\hline 1&2&3&4&5\\\hline x&x&1&\bar{D}&D\\\hline\end{array} = \begin{array}{|c|c|c|c|c|}\hline 1&2&3&4&5\\\hline D&1&1&\bar{D}&D\\\hline\end{array}$$

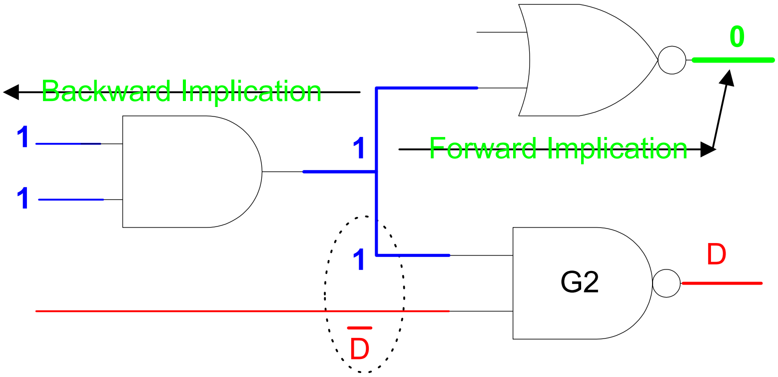
D- Algorithm

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** / **$\bar{D}$** 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 **$D/\overline{D}$** 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.

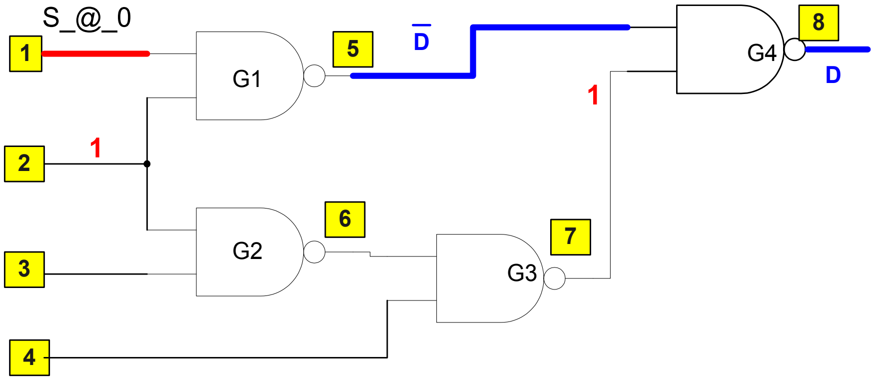
- *Line Justification* is Performed after $D/\overline{D}$ Appear at Some **PO**.

Line Justification-Example

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

J-Frontier

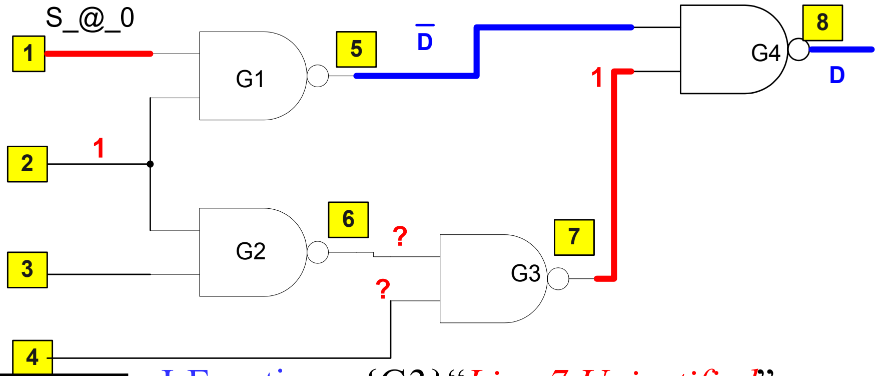
Is the Set of ALL Gates whose Output Lines are *Unjustified*



Line Justification

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

t^0 = PDFC[1]



1	2	3	4	5	6	7	8
D	1			$\bar{D}$			
D	1			$\bar{D}$		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	$\bar{D}$	0	1	D	J_F={G_2}
$t^3=t^2\cap Sc^2$	D	1	1	x	$\bar{D}$	0	1	D	J_F={Φ}

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