

Fault Modeling

- **Some Definitions**
- **Why Modeling Faults**
- **Various Fault Models**
- **Fault Detection**
- **Fault Collapsing**

Some Real Defects in Chips

- **Processing Faults**
 - missing contact windows
 - parasitic transistors
 - oxide breakdown
- **Material Defects**
 - bulk defects (cracks, crystal imperfections)
 - surface impurities (ion migration)
- **Time-Dependent Failures**
 - dielectric breakdown
 - Electromigration (open, short)
- **Packaging Failures**
 - contact degradation
 - seal leaks

Faults, Errors and Failures

- **Fault: A physical defect within a circuit or a system**
 - May or may not cause a system failure
- **Error: Manifestation of a fault that results in incorrect circuit (system) outputs or states**
 - Caused by faults
- **Failure: Deviation of a circuit or system from its specified behavior**
 - Fails to do what it should do
 - Caused by an error
- **Fault ---> Error ---> Failure**

最嚴重

Why Model Faults ?

- **Fault model identifies target faults**
 - Model faults most likely to occur
- **Fault model limits the scope of test generation**
 - Create tests only for the modeled faults
- **Fault model makes effectiveness measurable by experiments**
 - Fault coverage can be computed for specific test patterns to reflect its effectiveness
- **Fault model makes analysis possible**
 - Associate specific defects with specific test patterns

只有 fault 才能夠分析

2.為何建立故障模型

- 確定目標故障，針對最可能發生的故障建模。
- 減少測試生成的範圍，提升測試效率。
- 提供可量化的故障覆蓋率，衡量測試有效性。
- 使缺陷與測試模式的分析更加精準。

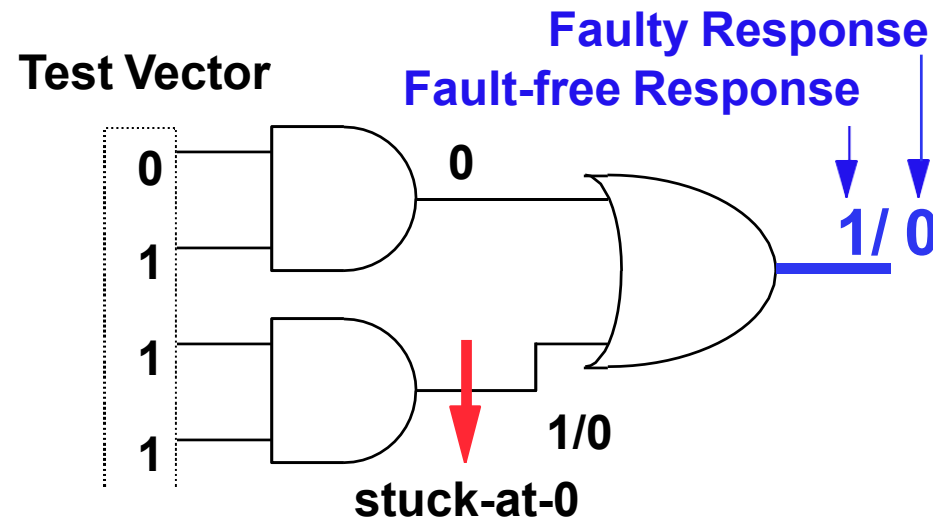
Fault Models

這模組可以偵測這麼多種fault

- **Stuck-At Faults**
- **Bridging Faults**
- **Transistor Stuck-On/Open Faults**
- **Functional Faults**
- **Memory Faults**
- **PLA Faults**
- **Delay Faults**
- **State Transition Faults**

卡住的fault

Single Stuck-At Faults



- Assumptions:
- Only one line is faulty.
 - Faulty line permanently set to 0 or 1.
 - Fault can be at an input or output of a gate.

Multiple Stuck-At Faults

- **Several stuck-at faults occur at the same time**
 - Important in high density circuits
- **For a circuit with k lines**
 - there are $2k$ single stuck-at faults
 - there are $3^k - 1$ multiple stuck-at faults

有 k 個電線
就有 $2k$ 個單線卡住
就有 $3^k - 1$ 個多重錯誤

為何要用這種 單個 卡住的 模型呢

Why Single Stuck-At Fault Model?

因為他減少了複雜度

- **Complexity is greatly reduced.**
Many different physical defects may be modeled by the same logical single stuck-at fault.
因為他可以是獨立的測試物件
- **Single stuck-at fault is technology independent.**
Can be applied to TTL, ECL, CMOS, etc.
- **Single stuck-at fault is design style independent.**
Gate Arrays, Standard Cell, Custom VLSI
他的設計 也可以是獨立的
- **Even when single stuck-at fault does not accurately model some physical defects, the tests derived for logic faults are still valid for most defects.**
- **Single stuck-at tests cover a large percentage of multiple stuck-at faults.**

橋接 錯誤

就是 短路

甚至不用接在一起

只要夠靠近 就可能有問題

Bridging Faults

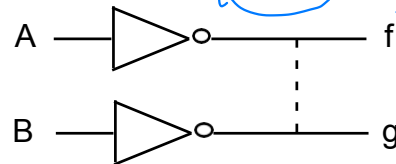
0 和 1 橋接了 短路了 兩個都會是0

因為接地 的排水地方 太大了

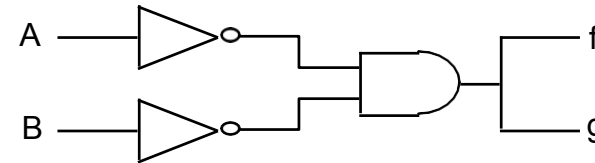
- **Two or more normally distinct points (lines) are shorted together**

- Logic effect depends on technology

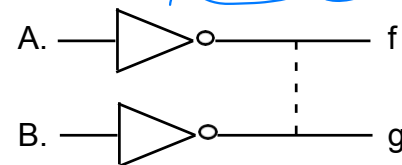
- **Wired-AND for TTL**



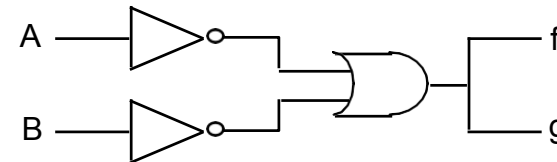
就是
⇒



- **Wired-OR for ECL**

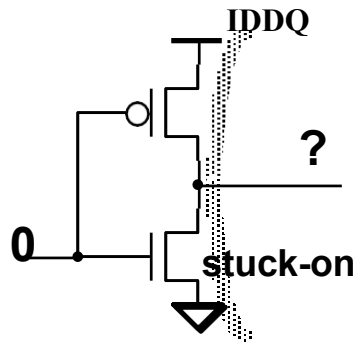
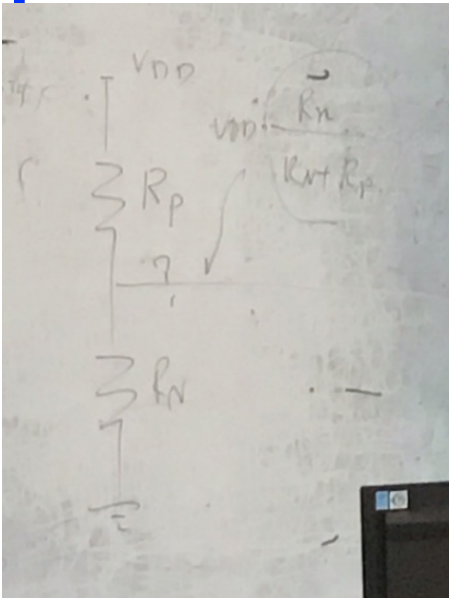


就是
⇒



- **CMOS ?**

CMOS Transistor Stuck-ON



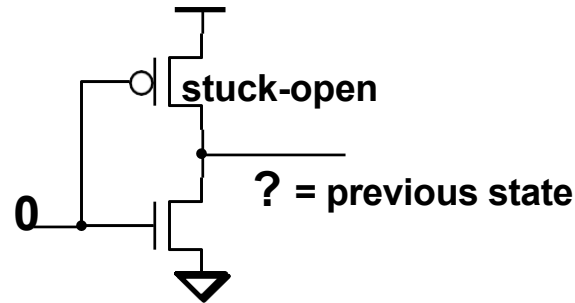
電晶體的卡住會有不明確的邏輯電壓

- **Transistor stuck-on may cause ambiguous logic level.**
 - depends on the relative impedances of the pull-up & pull-down networks
- **When input is low, both P and N transistors are conducting causing increased quiescent current, called IDDQ fault.**

CMOS Transistor Stuck-OPEN

這樣子 是一個 NOT

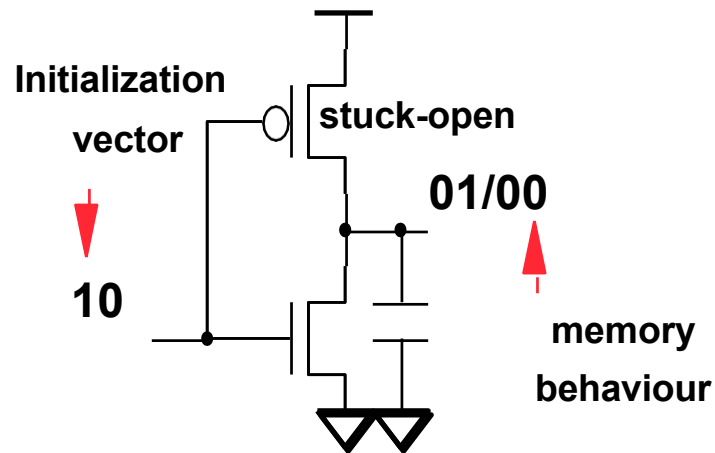
卡在 開的 這電路永遠都接不上



- Transistor stuck-open may cause output floating.

電晶體 這樣子 上面的 卡住 會造成
輸出是浮接
也就代表 輸出 一直會是 前一個狀態

CMOS Transistor Stuck-OPEN (Cont.)



- Can turn the circuit into a sequential one
- Stuck-open faults require two-vector tests

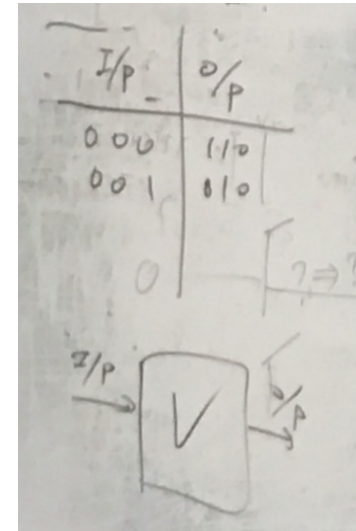
功能 錯誤

Functional Faults

- Fault effects modeled at a higher level than logic for function modules, such as

Decoders
Multiplexers
Adders
Counters
RAMs
ROMs

這些 功能元件 有錯誤
就是



像這個 解碼器 的 功能錯誤
就是 又可能像著下面一樣

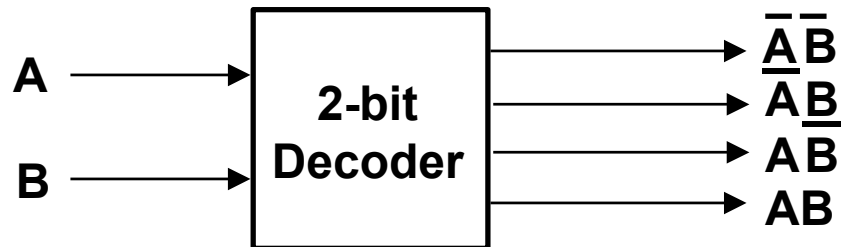
Functional Faults of Decoder

原本要輸出 L_i 的 卻輸出 L_j

$f(L_i/L_j)$: Instead of line L_i , Line L_j is selected

$f(L_i/L_i+L_j)$: In addition to L_i , L_j is selected f 卻輸出 L_i or L_j

$(L_i/0)$: None of the lines are selected 卻輸出 0



Memory Faults

參數 錯誤

- **Parametric Faults**

- Output Levels
- Power Consumption
- Noise Margin
- Data Retention Time

- **Functional Faults**

- Stuck Faults in Address Register, Data Register, and Address Decoder
- Cell Stuck Faults
- Adjacent Cell Coupling Faults
- Pattern-Sensitive Faults

相鄰 的記憶體 也可能有 耦合錯誤

Memory Faults (Cont.)

- **Pattern-sensitive faults:** the presence of a faulty signal depends on the signal values of the nearby points
 - Most common in DRAMs

0	0	0
0	d	b
0	a	0

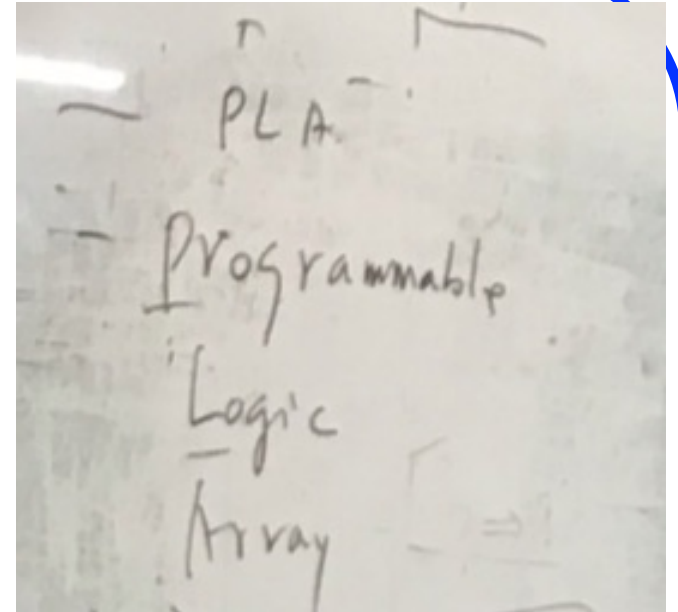
$a=b=0 \longrightarrow d=0$

$a=b=1 \longrightarrow d=1$

d會因為 a b 都是0 而被改變成0
1也同理

- **Adjacent cell coupling faults**
 - Pattern sensitivity between a pair of cells

PLA Faults

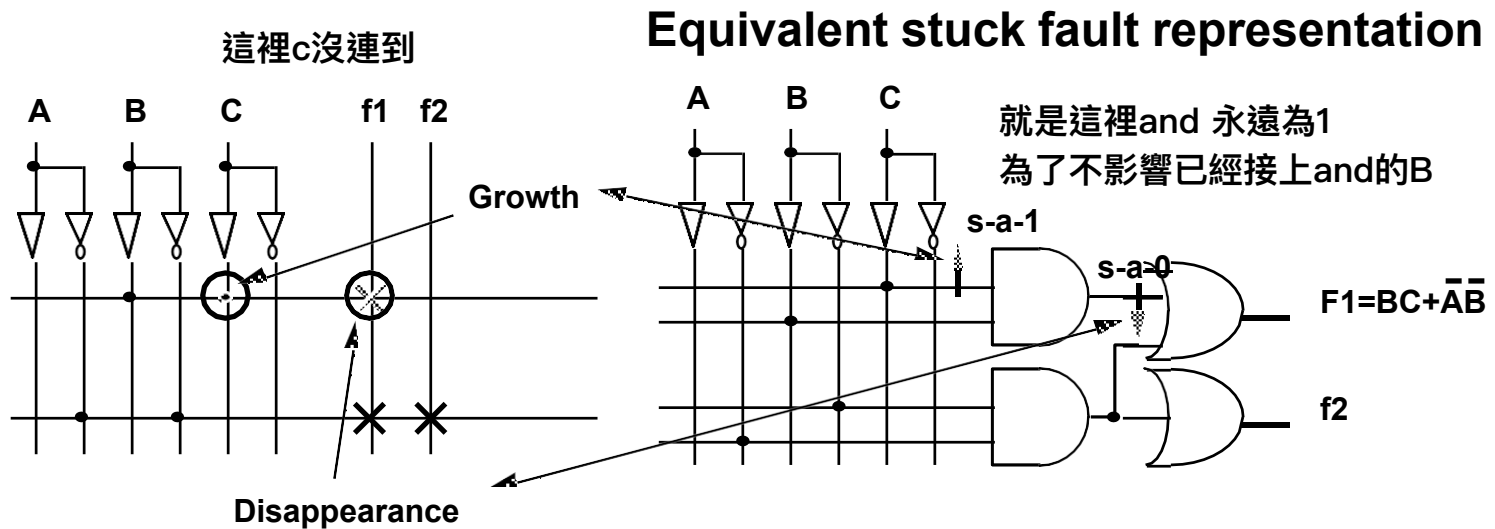


- Stuck Faults
- Crosspoint Faults
 - Extra/Missing Transistors
- Bridging Faults
- Break Faults

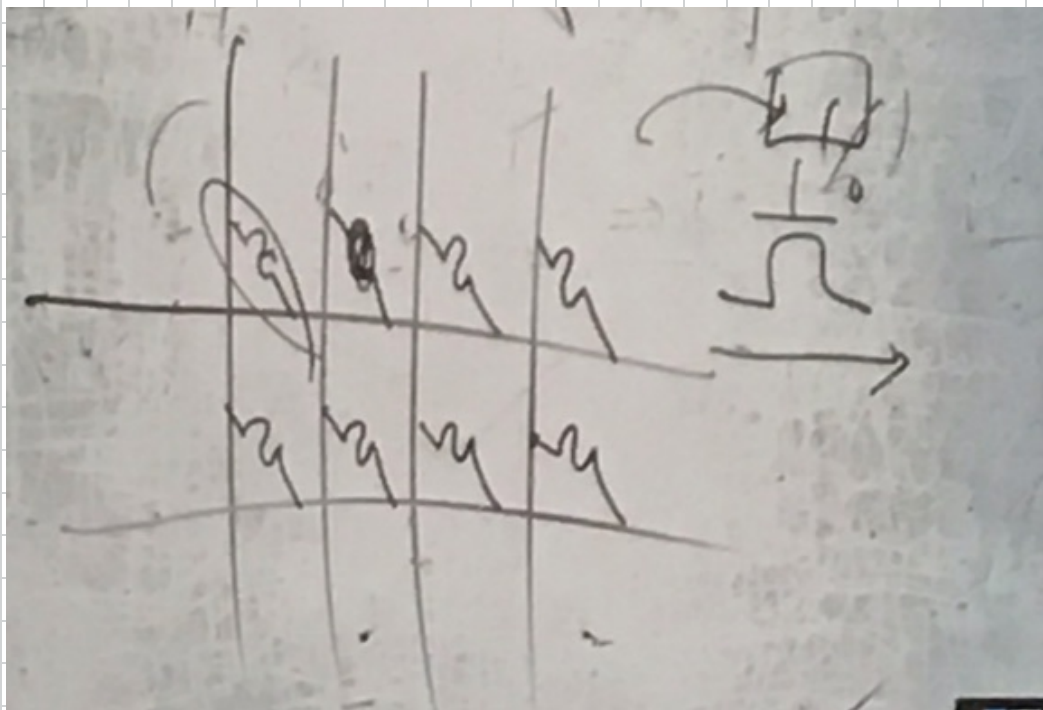
可程式規劃 邏輯 陣列

Missing Crosspoint Faults in PLA

- Missing crosspoint in AND-array
 - Growth fault
- Missing crosspoint in OR-array
 - Disappearance fault



每個 交接處



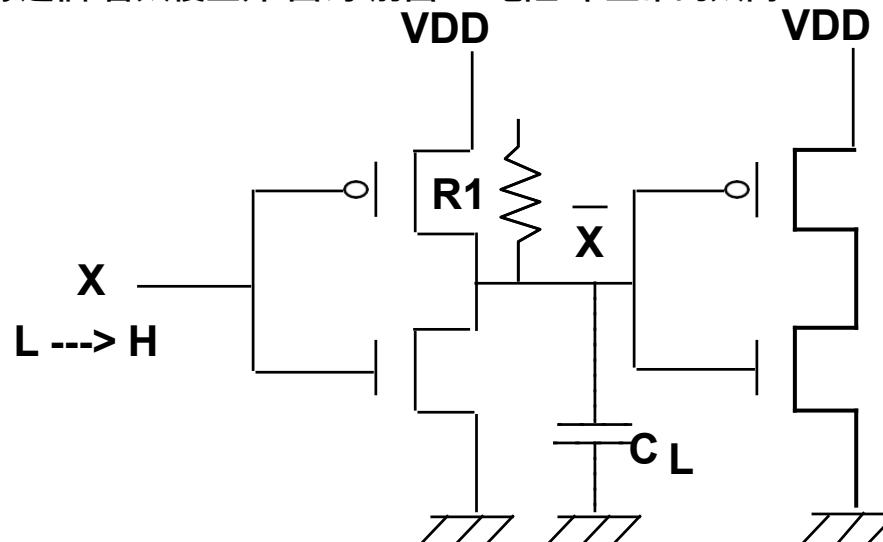
邏輯閘的 延遲 錯誤

Gate-Delay-Fault

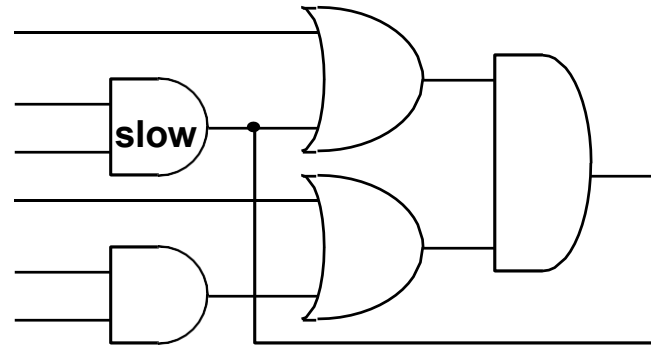
電壓 太慢上升 太慢下降

- **Slow to rise, slow to fall**
 - \bar{x} is slow to rise when channel resistance R1 is abnormally high

像這個 會太慢上升 因為 前面R1 電阻 不正常的太高



Gate-Delay-Fault

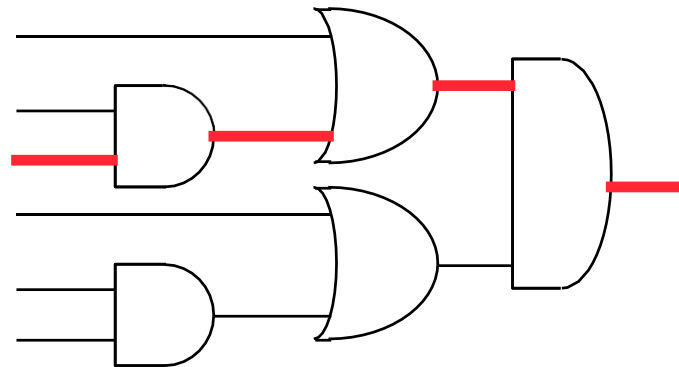


- Disadvantage:
Delay faults resulting from the sum of several small incremental delay defects may not be detected.

在能操作的 clock 的時間內 data 傳輸不過去

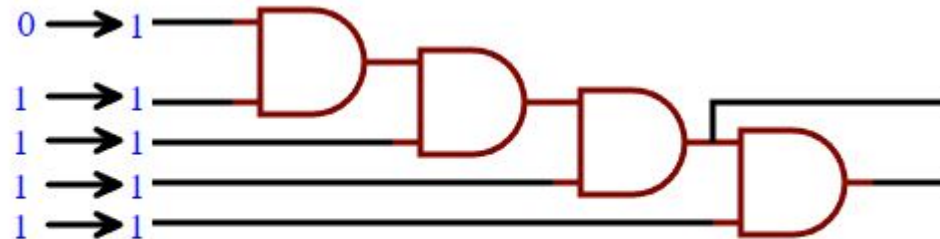
Path-Delay-Fault

- Propagation delay of the path exceeds the clock interval.
- The number of paths grows exponentially with the number of gates.



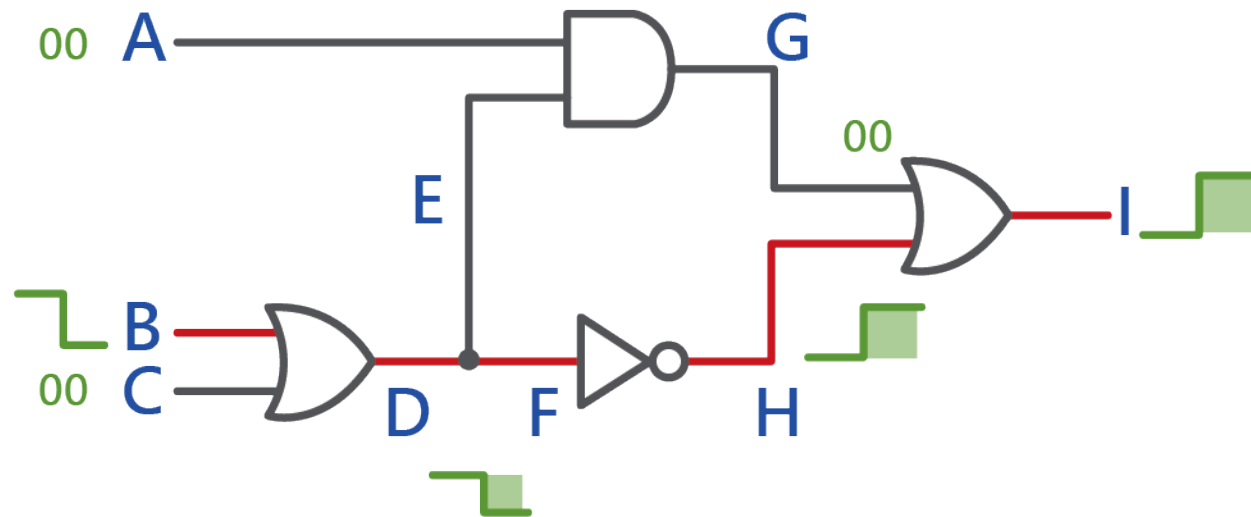
Path-Delay-Fault

- Assume clock period = 15ns and good gate delay = 3ns
Its path delay fault is :
 - $5 + 3.5 + 4 = 12.5 < 15 \rightarrow \text{pass}$
 - $5 + 3.5 + 4 + 5 = 17.5 > 15 \rightarrow \text{fail}$



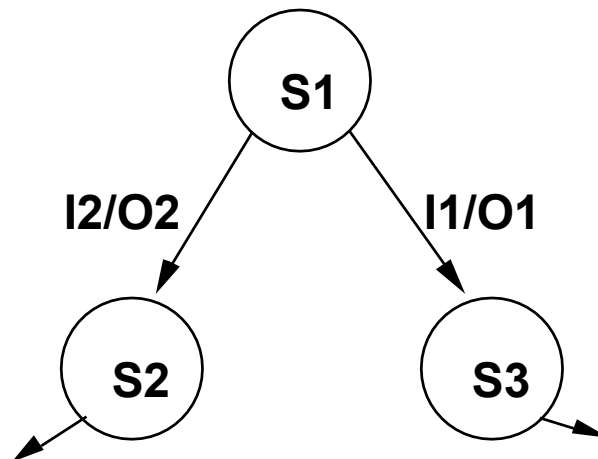
Path Delay Fault

- Two polarity (rising, falling) for each path
- Example: 這電路 需要測 五個路徑
 - 5 paths (AGI, BDEGI, BDFHI, CDEGI, CDFHI)
 - 10 Delay faults 每個都要去測試 0變1 和 1變0
 - Two-pattern for testing falling case of BDFHI



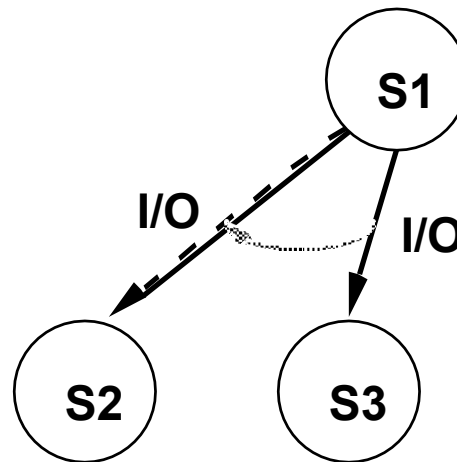
State Transition Graph

- Each state transition is associated with a 4-tuple: (source state, input, output, destination state)



Single State Transition Fault Model

- A fault causes a single state transition to a wrong destination state.

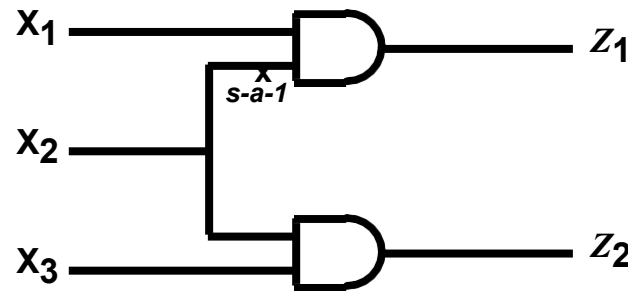


Fault Detection

XOR這裡可以偵測出 f 錯誤

- A test (vector) t detects a fault f iff $z(t) \oplus z_f(t) = 1$
 - t detects $f \iff z_f(t) \neq z(t)$

- Example



$$z_1 = x_1 x_2$$

$$z_2 = x_2 x_3$$

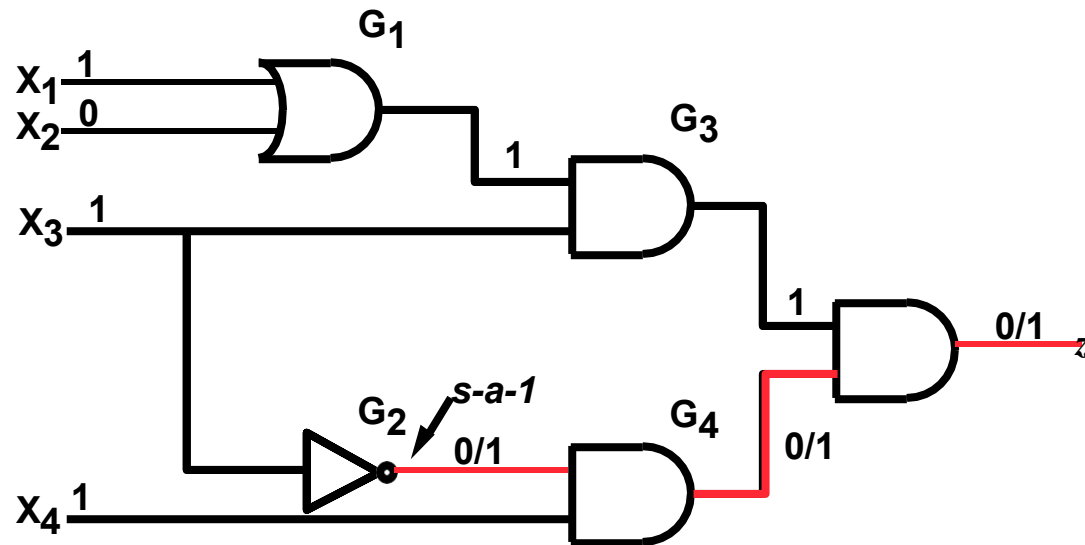
$$z_{1f} = x_1$$

$$z_{2f} = x_2 x_3$$

兩個有不一樣的 才會輸出1

The test 001 detects f because $z_1(001)=0$ while $z_{1f}(001)=1$

Sensitization



$$z(1011)=0$$

$$z_f(1011)=1$$

1011 detects the fault f (G_2 stuck-at 1)

v/v_f : v = signal value in the fault free circuit

v_f = signal value in the faulty circuit

Sensitization

1. 敏化測試 (Sensitization)

- A test t that detects a fault f
 - Activates f (or generate a fault effect) by creating different v and v_f values at the site of the fault
 - Propagates the error to a primary output w by making all the lines along at least one path between the fault site and w have different v and v_f values
- A line whose value in the test changes in the presence of the fault f is said to be sensitized to the fault f by the test
- A path composed of sensitized lines is called a sensitized path

- 測試向量須激活故障，並將其影響傳遞至主要輸出。
- 定義「敏化路徑」，故障效應需通過該路徑到達輸出。

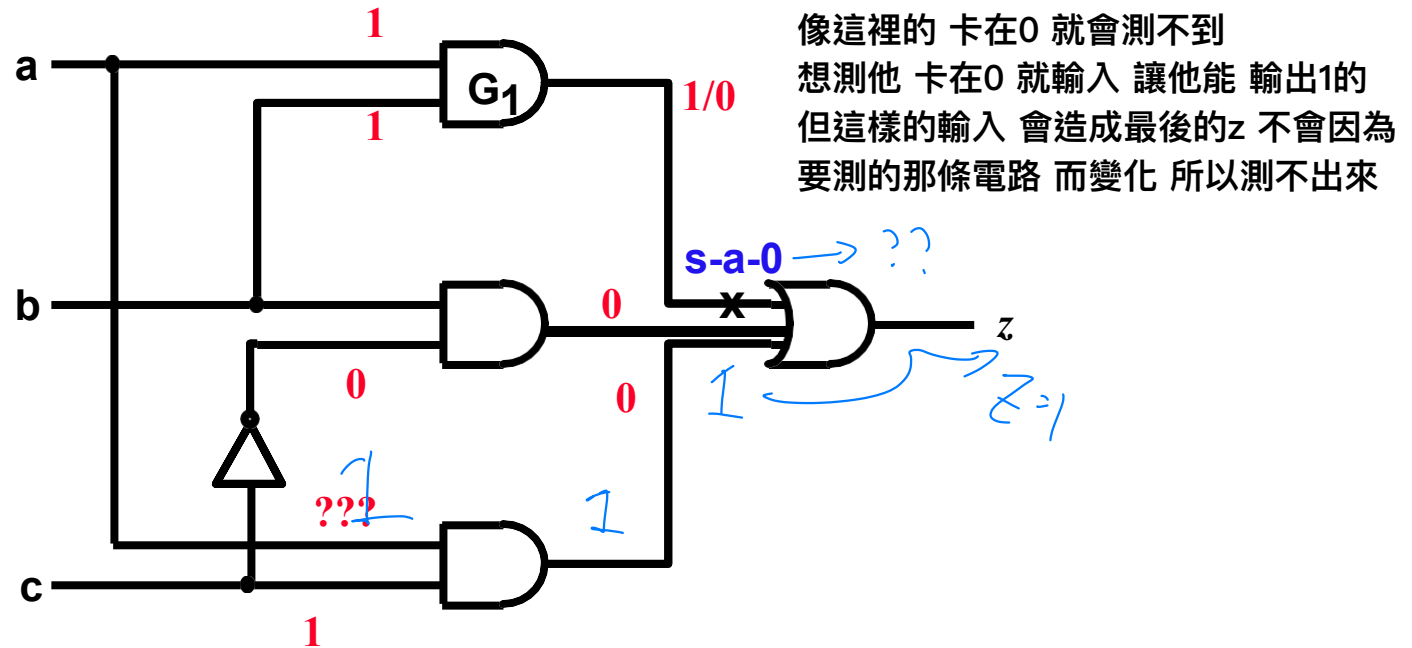
Detectability

- A fault f is said to be detectable if there exists a test t that detects f ; otherwise, f is an undetectable fault
- For an undetectable fault f

$$z_f(x) = z(x)$$

- No test can simultaneously activate f and create a sensitized path to a primary output

Undetectable Fault



- G_1 output stuck-at-0 fault is undetectable
 - Undetectable faults do not change the function of the circuit
 - The related circuit can be deleted to simplify the circuit

Test Set

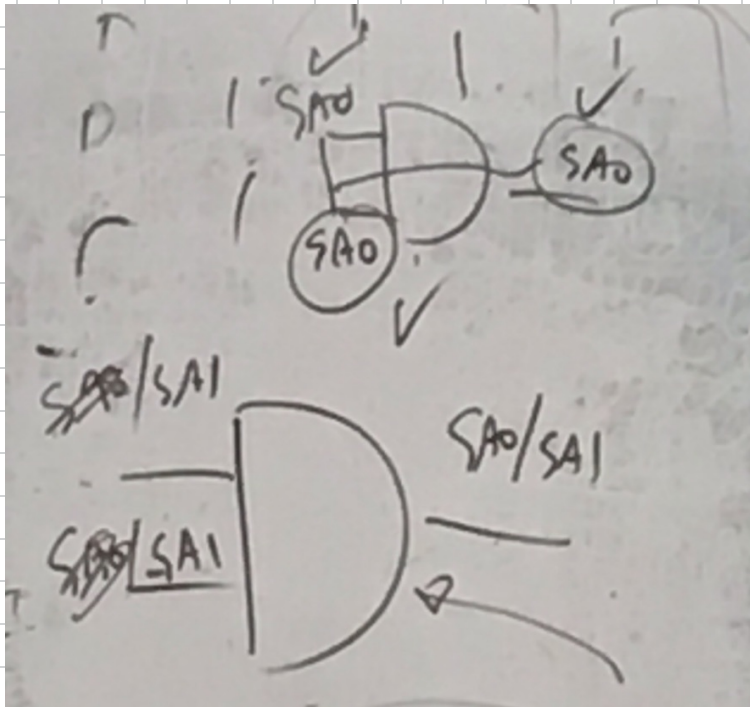
能偵測到的 就叫做test set

- **Complete detection test set:** A set of tests that detect any detectable faults in a class of faults
- The quality of a test set is measured by fault coverage
- **Fault coverage:** Fraction of faults that are detected by a test set
- The fault coverage can be determined by fault simulation
 - >95% is typically required for single stuck-at fault model in a complex system such as a CPU

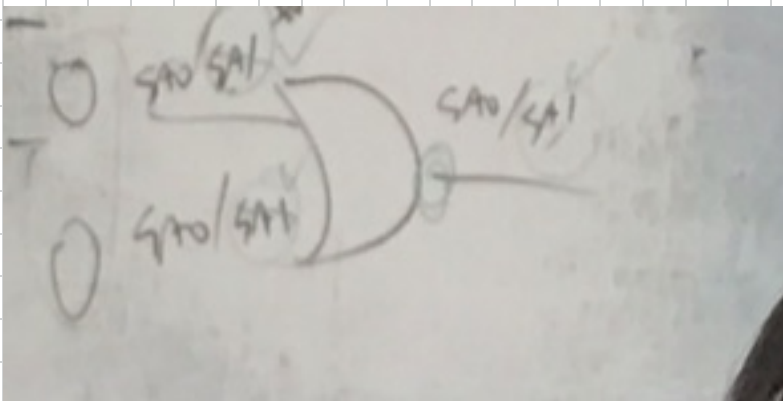
Fault Equivalence

同一個輸入 是一樣的 可以測試多個不同的錯誤
這樣就不用再多測一次

- **A test t distinguishes between faults α and β if $z_{\alpha}(t) \neq z_{\beta}(t)$**
- **Two faults, α & β are said to be equivalent in a circuit , iff the function under α is equal to the function under β for any input combination (sequence) of the circuit.**
 - $z_{\alpha}(t) = z_{\beta}(t)$ for all t
 - No test can distinguish between α and β
 - Any test which detects one of them detects all of them



像是 and 每條線 的卡在0 測試可以在同一個輸入完成



OR的卡在 0 同理 相同

Fault Equivalence

- AND gate: all *s-a-0* faults are equivalent
- OR gate: all *s-a-1* faults are equivalent
- NAND gate: all the input *s-a-0* faults and the output *s-a-1* faults are equivalent
- NOR gate: all input *s-a-1* faults and the output *s-a-0* faults are equivalent
- Inverter: input *s-a-1* and output *s-a-0* are equivalent
input *s-a-0* and output *s-a-1* are equivalent

Fault Equivalent Case for AND gate

A/0 C/0 B/0 are fault equivalent

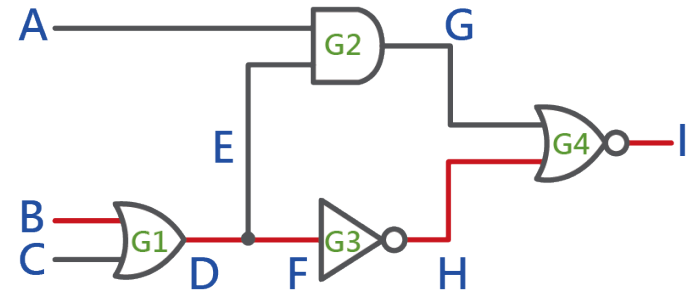
也可以從 真值表 來看 誰的卡在哪一樣

Input		Output						
A	B	good	A/0	C/0	B/0	A/1	C/1	B/1
0	0	0	0	0	0	0	<u>1</u>	0
0	1	0	0	0	0	<u>1</u>	<u>1</u>	0
1	0	0	0	0	0	0	<u>1</u>	<u>1</u>
1	1	1	<u>0</u>	<u>0</u>	<u>0</u>	1	1	1

Fault Equivalence

- SSF on fanout stem is not equivalent to SSF on fanout branch
- D is fanout stem; E and F are fanout branch

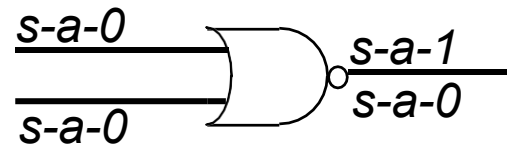
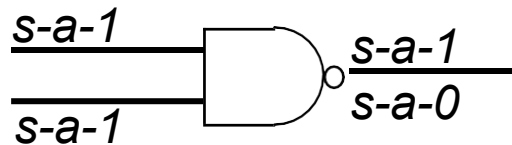
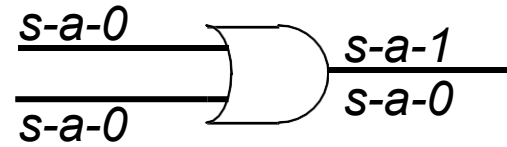
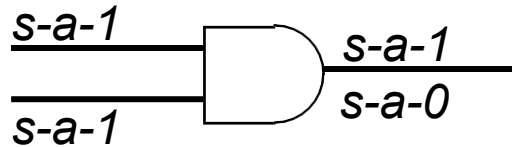
像這裡 DEF 三段電路 是不一樣的



Input			Output						
A	B	C	good	D/0	F/0	E/0	D/1	F/1	E/1
0	0	0	0	0	0	0	<u>1</u>	<u>1</u>	0
0	0	1	1	<u>0</u>	<u>0</u>	1	1	1	1
0	1	0	1	<u>0</u>	<u>0</u>	1	1	1	1
0	1	1	1	<u>0</u>	<u>0</u>	1	1	1	1
1	0	0	0	0	0	0	0	<u>1</u>	0
1	0	1	0	0	0	<u>1</u>	0	0	0
1	1	0	0	0	0	<u>1</u>	0	0	0
1	1	1	0	0	0	<u>1</u>	0	0	0

Equivalence Fault Collapsing

- $n+2$ instead of $2n+2$ faults need to be considered for an n -input gate.

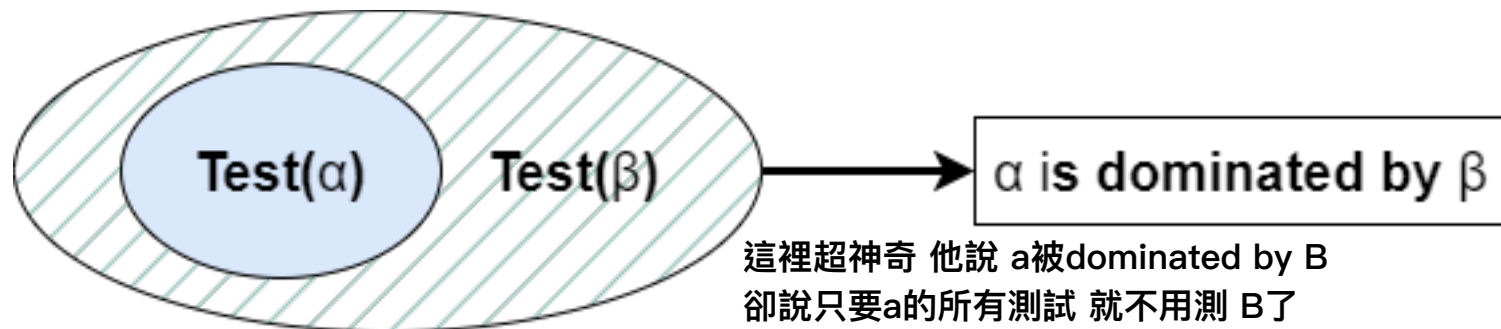


Fault Dominance

- A fault β is said to *dominate* another fault α in an irredundant circuit, iff every test (sequence) for α is also a test (sequence) for β .

$$T_\alpha * T_\beta$$

- No need to consider fault β for fault detection



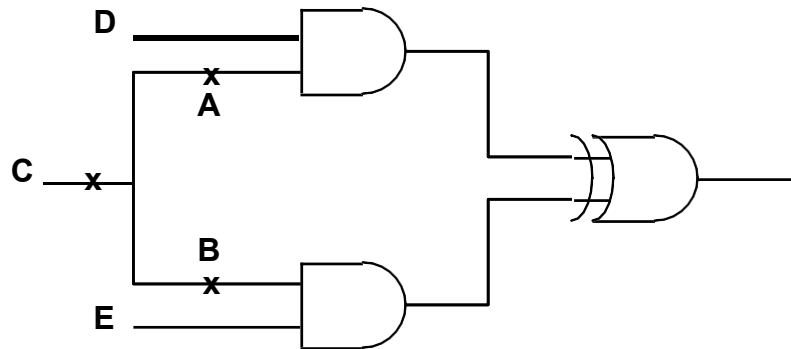
這裡超神奇 他說 α 被 dominated by β
卻說只要 α 的所有測試 就不用測 β 了
是在說 fault equivalence 的觀念
只需要更

Fault Dominance

很奇特 這裡主動的dominates 是 被統治的意思

- **AND gate: Output $s-a-1$ dominates any input $s-a-1$**
- **NAND gate: Output $s-a-0$ dominates any input $s-a-1$**
- **OR gate: Output $s-a-0$ dominates any input $s-a-0$**
- **NOR gate: Output $s-a-1$ dominates any input $s-a-0$**
- **Dominance fault collapsing: The reduction of the set of faults to be analyzed based on dominance relation**

Fault Dominance



- **Detect A sa1:**

$$z(t) \oplus z_f(t) = (CD \oplus CE) \oplus (D \oplus CE) = D \oplus CD = 1$$

$$\Rightarrow (C = 0, D = 1)$$

- **Detect C sa1:**

$$z(t) \oplus z_f(t) = (CD \oplus CE) \oplus (D \oplus E) = 1$$

$$\Rightarrow (C = 0, D = 1) \text{ or } (C = 0, E = 1)$$

$$C \text{ sa1} \rightarrow A \text{ sa1}$$

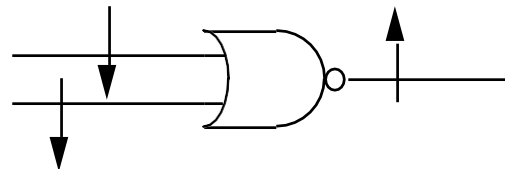
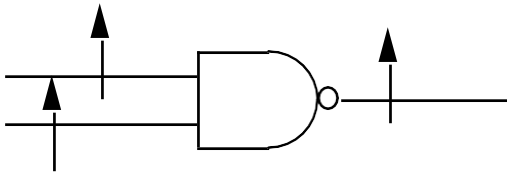
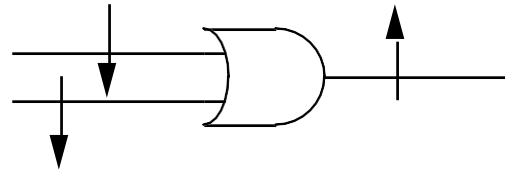
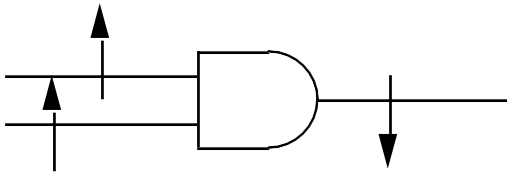
- **Similarly** $C \text{ sa1} \rightarrow B \text{ sa1}$

$$C \text{ sa0} \rightarrow A \text{ sa0}$$

$$C \text{ sa0} \rightarrow B \text{ sa0}$$

Fault Collapsing

- For each n -input gate, we only need to consider $n+1$ faults



Fault Equivalence

多推幾次 記一下 感覺比較快

- AND gate: all *s-a-0* faults are equivalent
- OR gate: all *s-a-1* faults are equivalent
- NAND gate: all the input *s-a-0* faults and the output *s-a-1* faults are equivalent
- NOR gate: all input *s-a-1* faults and the output *s-a-0* faults are equivalent
- Inverter: input *s-a-1* and output *s-a-0* are equivalent
input *s-a-0* and output *s-a-1* are equivalent

Fault Dominance

很奇特 這裡主動的dominates 是 被統治的意思

- AND gate: Output *s-a-1* dominates any input *s-a-1*
- NAND gate: Output *s-a-0* dominates any input *s-a-1*
- OR gate: Output *s-a-0* dominates any input *s-a-0*
- NOR gate: Output *s-a-1* dominates any input *s-a-0*
- Dominance fault collapsing: The reduction of the set of faults to be analyzed based on dominance relation

Fault Collapsing Example

And 卡在0 會等價

Or 卡在 1 會等價

Fault Equivalent :

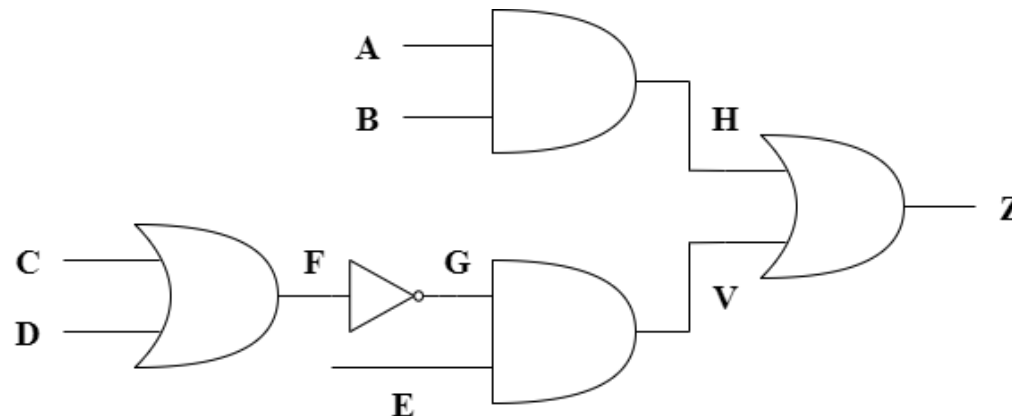
1. { A/0, B/0, H/0}
2. { C/1, D/1, F/1, G/0}
3. { E/0, G/0, V/0}
4. { H/1, V/1, Z/1}
5. { F/0, G/1}

Fault Dominance:

6. A/1 \rightarrow H/1
7. C/0 \rightarrow F/0
8. V/0 \rightarrow Z/0
9. B/1 \rightarrow H/1
10. D/0 \rightarrow F/0
11. E/1 \rightarrow V/1

這樣原本20個錯誤 就只需要測試7個

{A/0, A/1, B/1, C/0, C/1, D/0, E/1}



equivalent

and $sa_{0, \psi} = sa_{0, \lambda}$

or $sa_{1, \psi} = sa_{1, \lambda}$

not $sa_{\underset{0}{1}, \psi} = sa_{\underset{1}{0}, \lambda}$

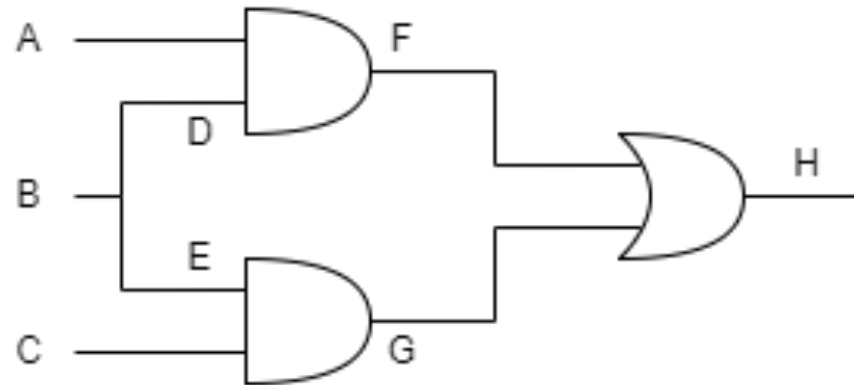
A

dominance

$$sa_{1, \lambda} \rightarrow sa_{1, \psi}$$

$$sa_{0, \lambda} \rightarrow sa_{0, \psi}$$

Minimum Test Vector Example



Minimum Test Vector Example

Fault free 沒錯誤 正常的狀況

A	B	C	FF	A ₀	A ₁	B ₀	B ₁	C ₀	C ₁	D ₀	D ₁	E ₀	E ₁	F ₀	F ₁	G ₀	G ₁	H ₀	H ₁
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1
0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	1	0	1	0	1
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	1
0	1	1	1	1	1	0	1	0	1	0	1	0	1	1	1	0	1	0	1
1	0	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	1	0	1
1	0	1	0	0	0	0	1	0	0	0	1	0	1	0	1	0	1	0	1
1	1	0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1	0	1
1	1	1	1	1	1	0	1	1	1	0	1	0	1	1	1	1	1	0	1

這些就是 有出某種錯誤時 最後的輸出

可以在跟正常情況 去做xor 就可以看到底有沒有出狀況

Minimum Test Vector Example (cont')

這些就是已經 和 正常狀況 xor 的結果








































A	B	C	A ₀ '	A ₁ '	B ₀ '	B ₁ '	C ₀ '	C ₁ '	D ₀ '	D ₁ '	E ₀ '	E ₁ '	F ₀ '	F ₁ '	G ₀ '	G ₁ '	H ₀ '	H ₁ '
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1
0	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	1
0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1	0	1	0
1	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	1	0	1
1	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0	1	0	1
1	1	0	1	0	1	0	0	0	1	0	1	0	1	0	1	0	1	0
1	1	1	0	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0

只有這個輸入 才能測到A0錯誤

(110, 010, 011) + (001 or 100 or 101)

or or

Minimum Test Vector Example (cont')

A	B	C	 A ₀ '	 A ₁ '	 B ₀ '	 B ₁ '	 C ₀ '	 C ₁ '	 D ₀ '	 D ₁ '	 E ₀ '	 E ₁ '	 F ₀ '	 F ₁ '	 G ₀ '	 G ₁ '	 H ₀ '	 H ₁ '
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1
 0	 0	1	0	0	0	 1	0	0	0	 1	0	 1	0	 1	0	 1	0	 1
0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	1
0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1	0	1	0
 1	 0	0	0	0	0	 1	0	0	0	 1	0	 1	0	 1	0	 1	0	 1
 1	 0	1	0	0	0	 1	0	0	0	 1	0	 1	0	 1	0	 1	0	 1
1	1	0	1	0	1	0	0	0	1	0	1	0	1	0	1	0	1	0
1	1	1	0	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0

(110, 010, 011) + (001 or 100 or 101)

A₀' = A₀ XOR FF