- PPT on blog

- in website book

FLAMINGO Date Page

RTL level simulators

Cycle Based

- only Boundary nodes
- No delay information
- does not detect glitches and setup/hold time violations
- faster and less memory usage

Event Based

- each internal node
- need scheduling and functions may be evaluated multiple times
- slower and more memory usage

Verification Matrices

code coverage : % of total code executed by given test case
functional coverage : % of total function " " " "

Bug Tracking sls (BTS) :

when bug is found by verification engineer, it is reported into BTS:

stages:
- open → when its filled
- close → when everything else works fine with new fix
- verified → when designer confirms that its bug
- fixed → when it is removed from design

Regression and Revision Control:

- return to normal state
  Ex: New feature + fix bug
- when multiple user accessing the same data, data may loss
  Ex: trying to write the same file simultaneously

System design flow:

Requirements

↓

functional design — Behavioral simulation

↓

RTL design — RTL simul. validation

↓

Logic — logic simulation verification/fault

↓

Circuit — Timing simulation, Circuit Analysis

↓

Physical — Design Rule check

↓

Description for manufacture

Sources of error

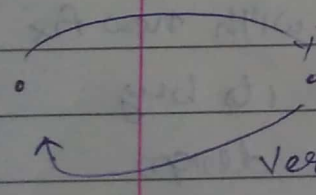
- error in specification
  - Unspecified func
  - Unrealized feature
  - conflicting requirements
- error in implementation
  - human error in interpreting design func

(How reduced this)

By Automation

Posig - Voice Redundancy (ASIC)

- Reconvergence model



- RTL coding from specifications
- Insertion of a scan chain
- synthesizing a RTL code into gate level
- gate level netlist into layout

Test Bench: It checks whether the RTL implementation meets the design spec or not.

functions:

- To gen the stimulus for simulation
- To apply this stimulus to the module under test and collect o/p response.
- To compare the o/p response with expected golden values

Verification Methods/Techniques:

- Simulation (functional & Timing)
  (Behavioral, RTL, gate level [Pre & Post], switch level, transistor level)
- formal verification (functional)
  Binary decision Diagrams, equivalence checking, model checking
- Static timing analysis (timing)

Black Box

- less time consumption and efforts
- independent of implementation
- verification process parallel with design process
- difficult to locate the source of problem
- " " set interesting state
- lack of visibility and controllability

formal verification

- mathematically proving that the modification has not change functionality.

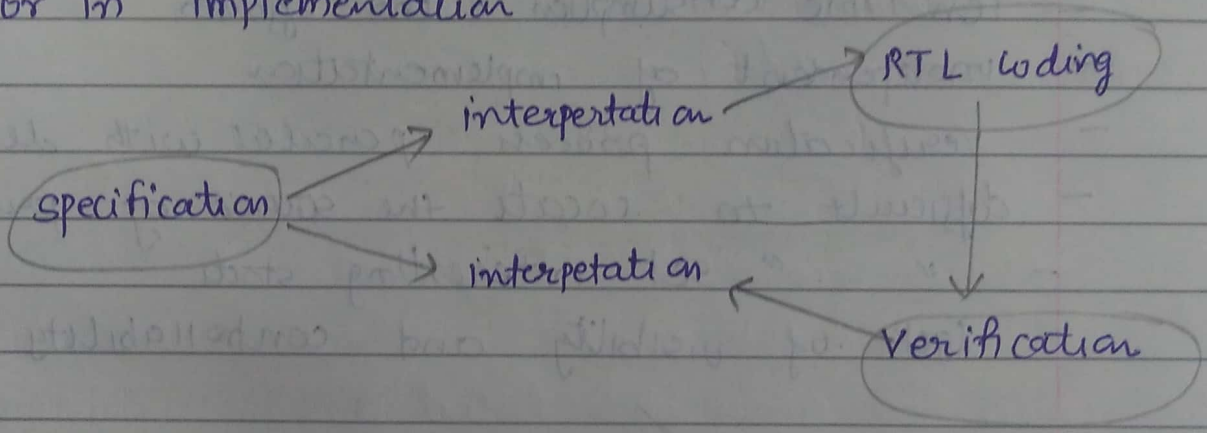
- Adding clock gating circuitry for power reduction
- restructuring critical Paths, inserting clock tree
  - does not require simulation vectors
    Equivalence and model checking → methods
- You only need to provide a fun correct or "golden design (reference design) and modified version of the design (implementation)

Date
Page 14-July

- Behavioural into RTL is called " synthesis"
- Gate level netlist (where gate delay estimation is done)
- Transistor level (propagation delay)
- Physical layout (manufacturability of design is checked with the help of Design Rule)
- Source of errors :-

- error in specification
  No model for checking as it is ~~top~~

- error in Implementation
-



- Verification Methods :

(I) functional verification
(II) formal " " (Equivalence, Model checking)
(III) Semiformal " " (Assertion based

(I) functional verification
- Black box approach → can not look into the

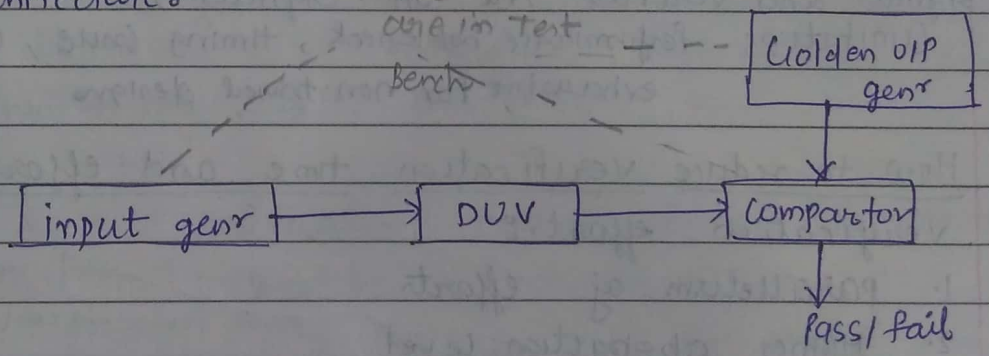
design. without knowledge of internal implementation
- Through available interfaces only, no internal state access

* Testing as user
  - white box approach → internal fully known, * Testing as Developer
  - Grey box approach → internals relevent to testing known, * testing as user with access to internals

Test Bench:

DUT → design under test
DUV → " " verification

architecture:



① random i/p gen, ② repetive i/p gen ~~for~~
- random with some constraint i/p gen
- ③ Directed i/p gen (narrow focused coverage)

limitation:
- large no. of simulation vectors are needed to provide confidence.
- slowing down Performance
- Once Behavioral design is verified, there are many requirements for small non-func modification in RTL.

$5000/month → 3 months
×70
350000

FLAMINGO Date 19-July Page

Clock gating circuit →

critical Path → in ckt which gives the max delay in the combinational ckt.

Verification Tools

EDA tool

① Linting
- Syntax check (static error → errors w/o i/p provided)

② Simulator (makes a computing model for ckt, executes (creating virtual environment) the model for a set of i/p signals and verifies the o/p signals.)

24-July Limitation: Performance bottleneck, timing issue, can't exhaustive for non-trivial designs

How to reduce verification time and efforts?

reduce verification efforts:
1. parallelism of efforts (By doing parallel, required more power → cost ↑ res)
2. Higher abstraction level
3. Automation (Test bench)

some new concepts:
1. Design for verification (Design in such a way that easily verify.) (adding some extra code for stored the intermediate data, inputs) display, counter,

Extra hardware is added just to debug, not consider in fabrication.
(synthesis off ——— synthesis on)

2. Verification of reusable Design
(IP codes → used at diff^r Platform)

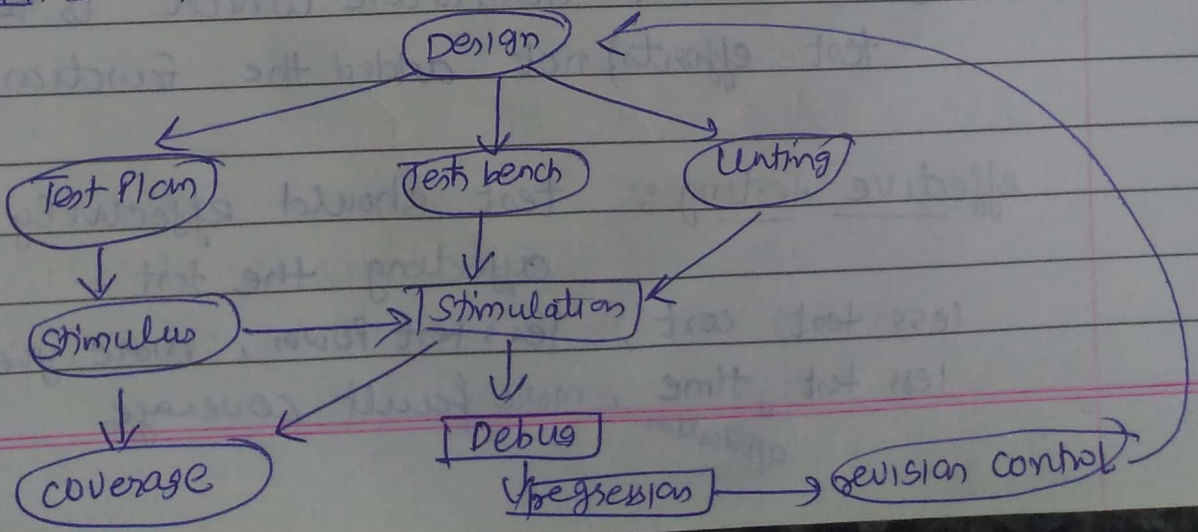
3. Verification Reuse
(RVM → reuse verification methodology)
Test bench is for adder, can be used ~~for~~ by which writing the test bench for the multiplier.

Off-line testing → when functional is stopped and then testing

On-line testing →

Verification → on design (functionality)
Testing → on device

Simulation Based functional Verification flow:



26-July

Testing

Testing Philosophy

Students — chips
course syllabus — specifications

Testing ⟶ on device

⇒

- Test Planning and test application
  (before fab) (after fab)

(I)- Test development / test Planning
How you test the system, before the fabrication it asked.

(II)- Test application / Manufacturing test
(after the fabrication, we have the Physical hardware on our hand.)

cost for testing :

1)- Test development (one time cost)
2)- Test application
3)- DFT (Design for test)
(some extra hardware added which is to reduce test efforts) not added the functionality)

effective testing : test should effectively w/o affecting the test.
less test cost, less test Power, more yield, less test time, more fault coverage
application

ATE → automatic test equipment

- functionality and (component) structural testing
  (To reduce the time, we will goes to the component testing)
  - ↓ Black box approach
  - ↓ white box

Causes of the Defects in CKt:

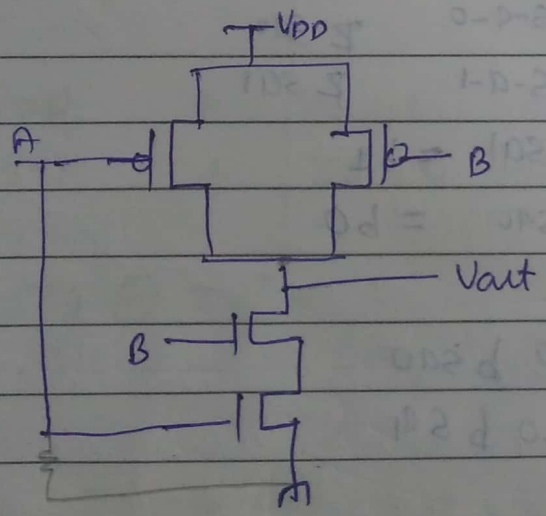
① Defects
(at the time of fabrication of IC as example ions dopping
(improper physical fabrication → defect)

② Errors

③ faults

for example:

$A \cdot \overline{B} = Y$



Defect: A is shorted to Gnd
fault: A Stucked at logic-0
error: at output we are getting
C=1 due to A=1, B=1
but correct o/p is C=0

FLAMINGO
Date 31-July
Page

- error is not permanent
for ex, no error occurs if atleast one i/p is 0.

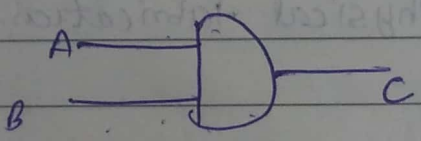
faults :- gates & ~~net~~/wire (interconnects → in Digital Design
- stuck at logic-0 and stuck at logic-1.

No. of multiple faults = $(3K+1)^n - 1$

$n \rightarrow$ no of wires/Ports

$K \rightarrow$ fault Possible

for example :



here $n = 3$

$K = 2$ (stuck at logic-1 & logic-0)

SSa (single stuck at)
- a s-a-0 ≡ sa0
- a s-a-1 ≡ sa1
- b sa1 = b1
- b sa0 = b0

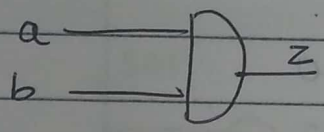
msa (multiple stuck)
- a sa0 b sa0
- a sa0 b sa1

no of single faults = $K \times n$
= 2×3
= 6 faults)

assume — 1 fault at a time
No bridge faults
no. of faults checked = $(K)^n$ (2^n)

2-Algo.

for checking the
if a is stuck at 1



to check this we applied $a=0, b=1$ then $z=0$
if don't get the o/p $z=0$ → then
a sa1.

- To examine the fault test vector is $a=0, b=1$
  z_g → genr o/p z_f → fault o/p

$z_g = 0$, t $z_f = 1$
$z_g = 1$, t $z_f = 0$

$z_g \oplus z_F = 1$

a s-a-1:

$z_g = ab$
$z_f = b$
$ab \oplus b = 1$
$ab \cdot b' + (\overline{ab}) \cdot b = 1$
$\bar{a} b = 1$
So; $a=0$ & $b=1$ → Test vector

z s-a-0 :

a=1, b=1 $z_g = a \cdot b$

↑ $z_f = \bar{a}\,\bar{b}$

Test vector

z s-a-1 :

a=0, b=1 & a=0, b=0 & a=1, b=0

b s-a-1 :

a=1, b=0

b s-a-0 :

a=1, b=1

a_1	01
b_1	10
a_0	11
b_0	11
z_0	11
z_1	00, 10, 01

$\{a_0, b_0\} \rightarrow z_0$

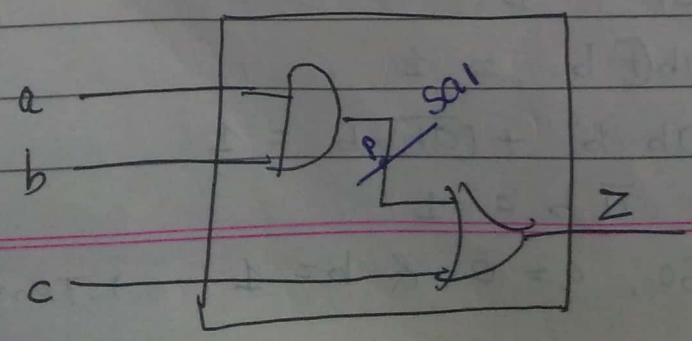
$\{a_0, b_1\} \rightarrow z_0$

$\{a_0, z_0\} \rightarrow z_0$

$\{a_0, z_1\} \rightarrow z_1$

If we only consider the single stuck faults, then multiple stuck faults are automatically included. or have been replaced with single stuck.

Ex:

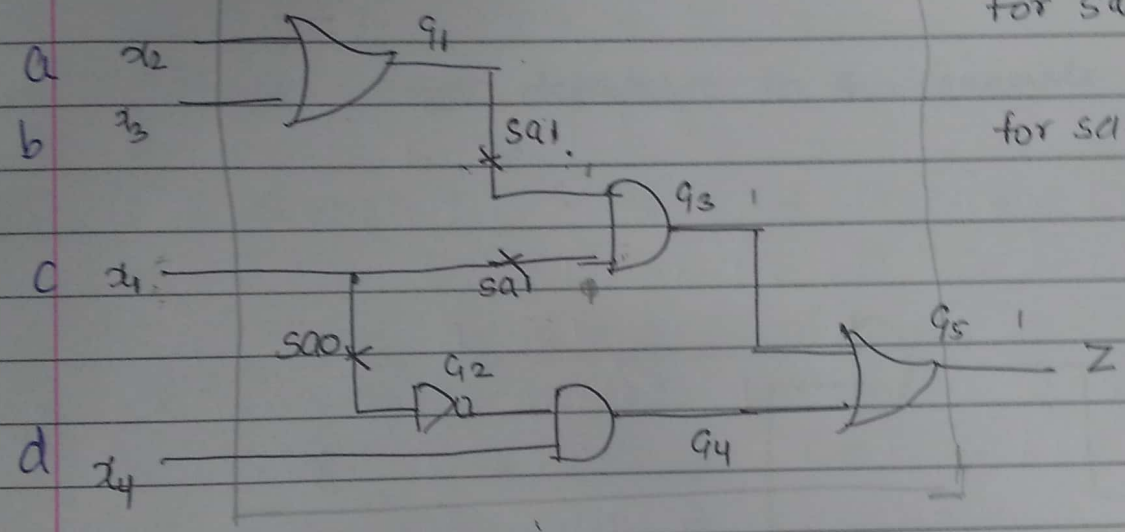


	abc
s-a-1 at f	0 00
	1 0 0
	0 1 0

$z_g = a \cdot b + c$

$z_f = 1$

Ex:-



for sa1 : 001x

for sa1 : 1x00

x100

for sa0 :

Ex:- s-a-1 :

x_1 x_2 x_3 x_4

0 0

$z_g = (x_2 + x_3) \cdot x_1 + \bar{x_1} \cdot x_4$

$z_f = 1 \cdot x_1 + \bar{x_1} \cdot x_4$

$z_g \oplus z_f = 1$

$(x_1 x_2 + x_1 x_3 + \bar{x_1} x_4) \oplus (x_1 + \bar{x_1} x_4) = 1$

$\overline{(x_1 x_2 + x_1 x_3 + \bar{x_1} x_4)}(x_1 + x_1' x_4) + \overline{(x_1 + \bar{x_1} x_4)}(x_1 x_2 + x_1 x_3 + \bar{x_1} x_4)$

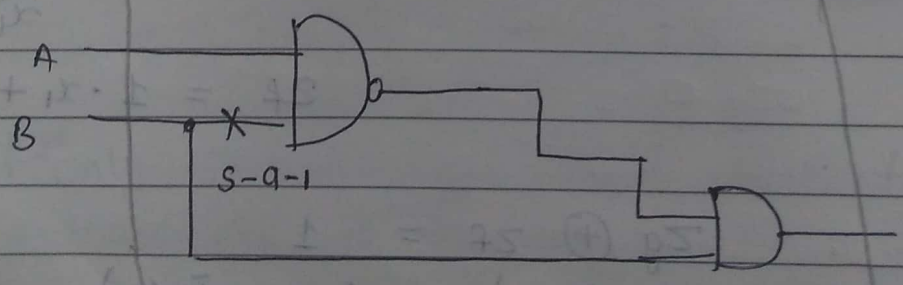
$\bar{x_1}\bar{x_2} \cdot \bar{x_1}\bar{x_3} \cdot \bar{\bar{x_1}} x_4$

When sa-0: $z_g = (a+b)\cdot c + \bar{c}\cdot d$

$z_f = (a+b)\cdot c + d$

A circuit with redundant fault

Ex8



Sol^n

$$z_g = \overline{A\cdot B}\cdot(\overline{A\cdot\bar{B}}\cdot B)$$

$$= \overline{A\bar{B}}\cdot B$$

$$= (\bar{A}+\bar{\bar{B}})\cdot B$$

$$= \bar{A}B$$

$z_f = \overline{T \cdot B} = 0$ $\quad \bar{A} \cdot B$

$z_g \oplus z_f = 1$

$(\bar{A} B) \oplus B = 1$

$\overline{(\bar{A} \cdot B)} B + \bar{B}(\bar{A} \cdot B) = 1$

$(A + \bar{B}) B = 1$

$A B = 1$

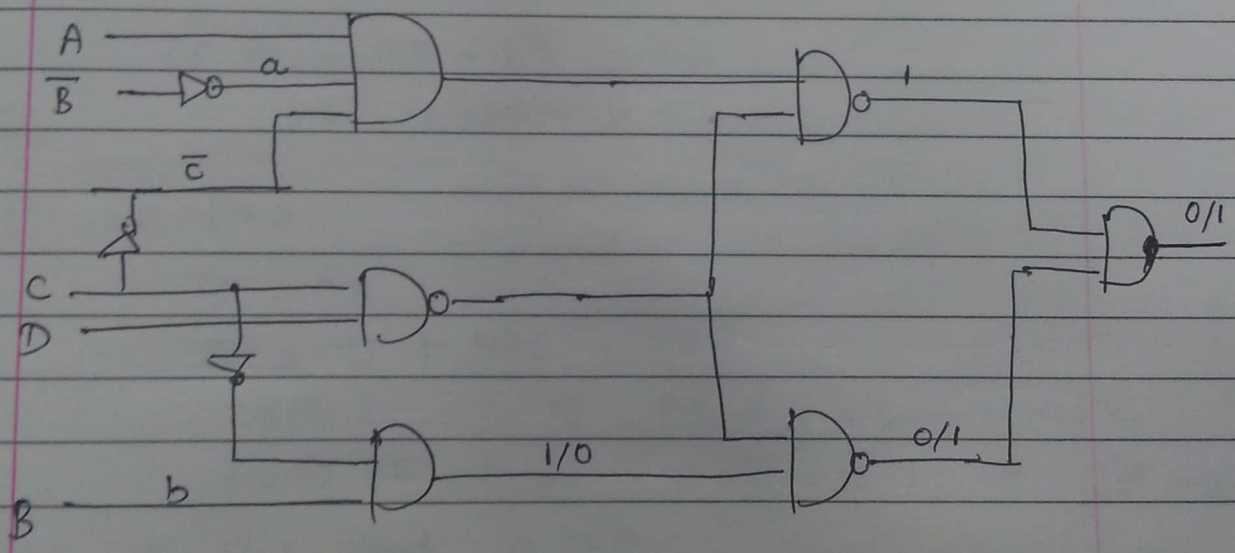
$z_g \oplus z_f = 1$

$(\bar{A} \cdot B) \oplus (\bar{A} \cdot B) = 1$

Not possible

fault is not detectable in this example.

Ex: a, s-a-1



when a sa1

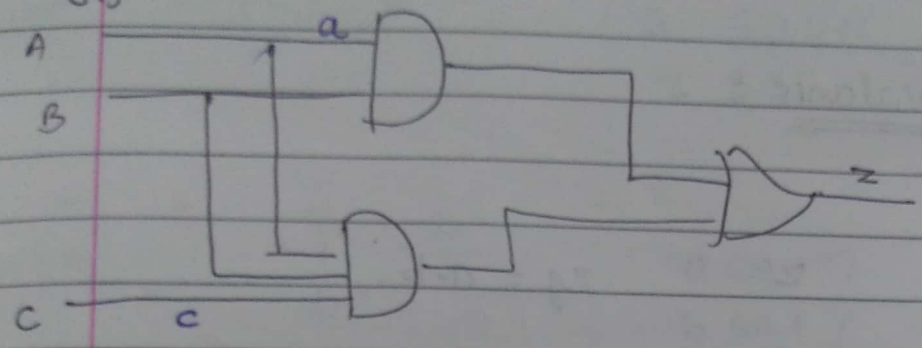
$z_f = \bar{A} \bar{B} + C = (\bar{A} + C)(\bar{B} + C)$

$z_g = \bar{A} \bar{B} + C = (\bar{A} + B + C)(\bar{B} + C)$

not detectable

FLAMINGO Date 9-Aug Page

Ex:



i) c sa1

ii) a sa0

(iii) { c sa1, a sa0 }

(i) $Z_g = A \cdot B + ABC = AB(1+C) = AB$

c, sa1

$Z_f = AB + AB = AB$

not detectable

(ii) a sa0

$Z_g = AB$

$Z_f = 0 + ABC = ABC$

$Z_g \oplus Z_f = 1 \Rightarrow AB \cdot \bar{0} = 1 \Rightarrow$ ~~11X~~

$AB \oplus ABC = 1 \Rightarrow AB\bar{C} = 1 \Rightarrow 110 \rightarrow$

(iii) c, sa1 and a sa0

$Z_f = 0 + AB = AB$

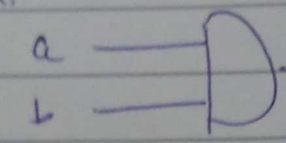
is undetectable

f_2 is called second genr redundant fault.

Fault Equivalance :

for ex:

AND



$z_g = a \cdot b$

f_1 a sa0 $\quad z_{f_1} = 0$

f_2 b sa0 $\quad z_{f_2} = 0$

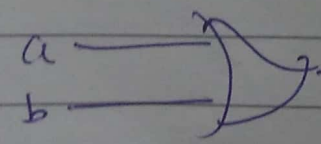
f_3 z sa0 $\quad z_{f_3} = 0$

these three faults not need to XOR, to find out the test vector, separatly.

$$
\begin{aligned}
& z_g \oplus z_{f_1} = 1 \\
& (a \cdot b) \oplus (\bar{a} b) = 1 \\
& \bar{a}\bar{b} \cdot \bar{a} b + a b \, \overline{\bar{a} b} = 1 \\
& (\bar{a} + \bar{b}) \bar{a} b + a b (a + \bar{b}) = 1 \\
& \bar{a} b + a b = 1
\end{aligned}
$$

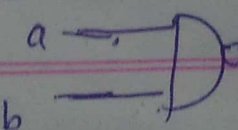
so sa0 is fault equivalent for AND.

OR



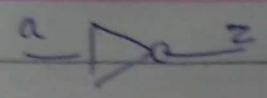
a sa1
b sa1 } — fault equivalent
z sa1

NAND



a sa0 } a sa1
b sa0 } b sa1
z sa1 } z sa0

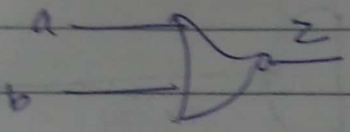
NOT



a sa0 ↔ z sa1

z sa0 ↔ a sa1

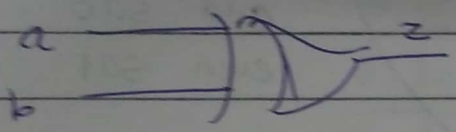
NOR



a sa1 }
b sa1 } z sa0

a sa0
b sa0
z sa1

XOR



0 0	0
0 1	1
1 0	1
1 1	0

f_1 sa0 }
f_2 sa1 } a

f_3 sa0 }
f_4 sa1 } b

f_5 sa0 }
f_6 sa1 } z

$Zf_1 = b$, $Zf_2 = \bar{b}$, $Zf_3 = a$, $Zf_4 = \bar{a}$

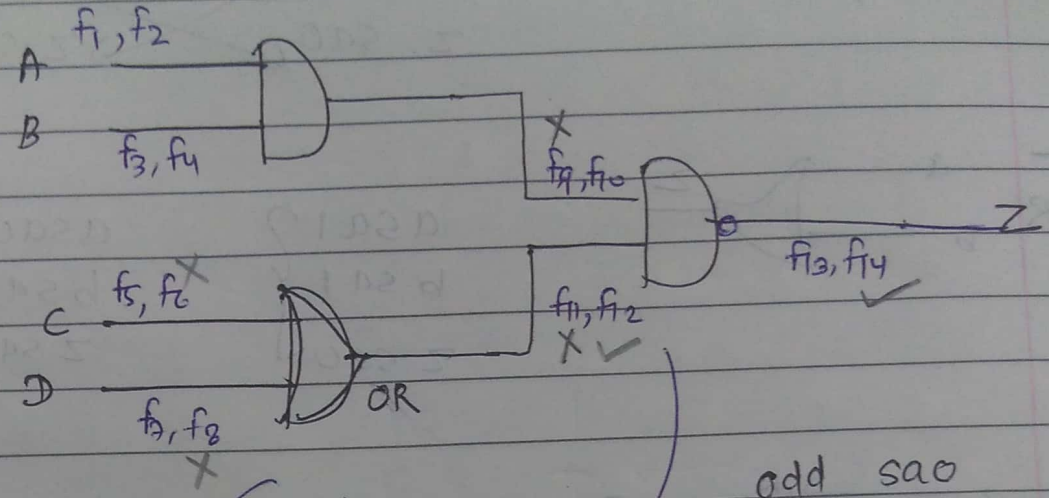
$Zf_5 = 0$, $Zf_6 = 1$

here individuals faults are not equivale

- $\{ f_1$ sa0 , f_4 sa1 $\} = 1 = Zf_6$

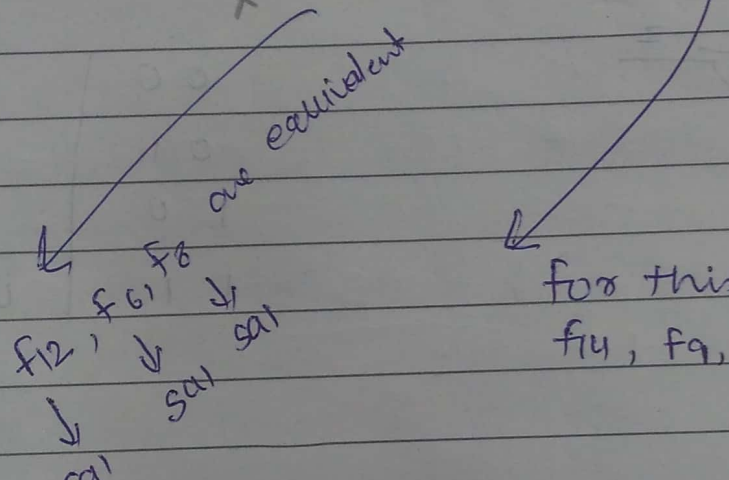
- $\{ f_2$ sa1 , f_3 sa0 $\} = 1 = Zf_6$

Ex:-



Sol:

odd sao
even sa1

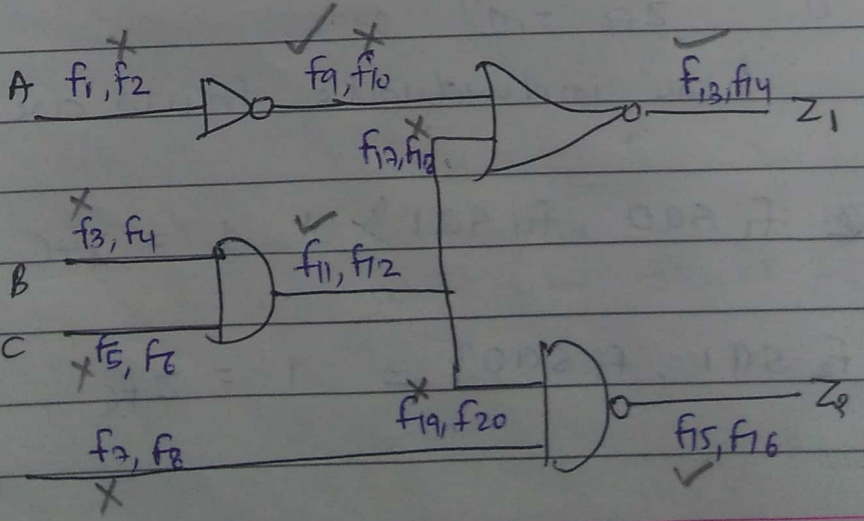


for this
f_{14}, f_9, f_{11} → sa0 & sa1 equivalent

10/14

$f_1, f_2, f_3, f_4, f_5, f_6, f_{10}, f_{12}, f_{13}, f_{14}$ → faults

Ex:

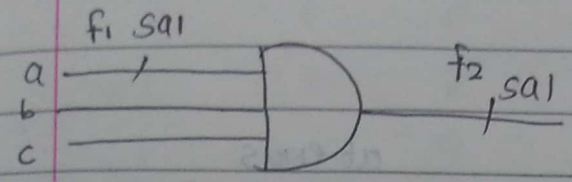


13/20

Collapse ratio = 13/20

FLAMINGO
Date 11-Aug-20
Page

Fault Dominance :

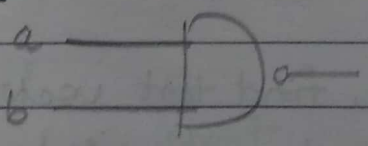


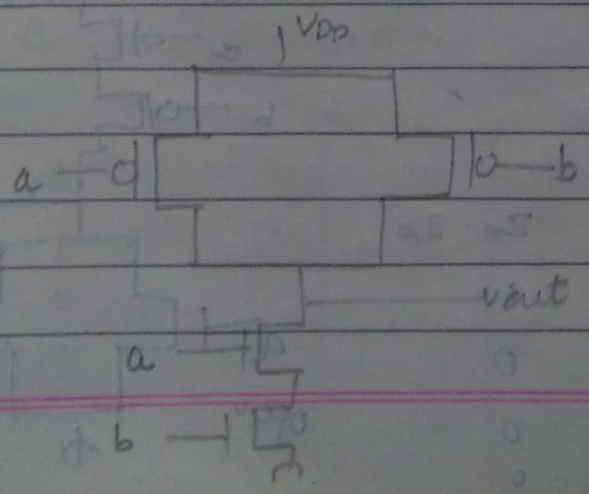
f_1	f_2
0 1 1	0 0 0
	0 0 1
	0 1 0
	0 1 1
	1 0 0
	1 0 1
	1 1 0

f_2 are removed from fault list

f_2 dominant on f_1.

Gate fault :





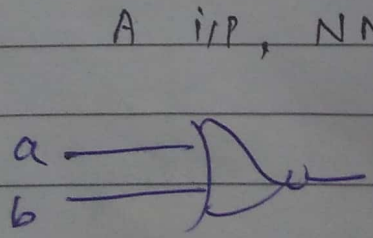
- In digital domain, i/p's are either ON or OFF.
- switch [open / short]

a	b	Z_n	Z_f (when b open) at PMOS
0	0	1	1
0	1	1	1
1	0	1	Z ← (bcz b is not connected)
1	1	0	0

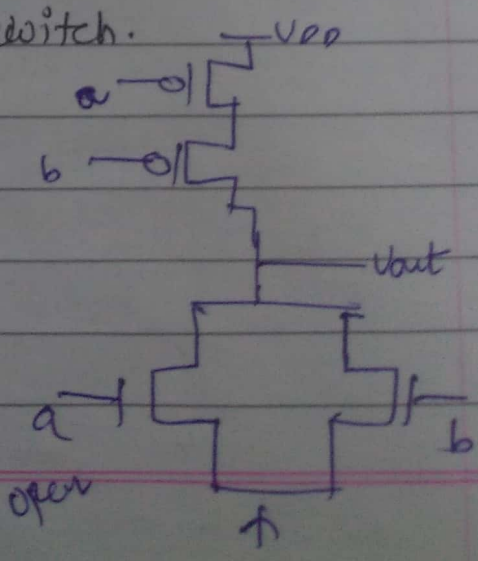
pair of test vector →

- When wire is not connected to VDD or gnd then considered as High impedence (Z).
- when we applied a=b=1 then O/P = 0 and after that if we applied a=1, b=0, then O/P = 1 But in fault condn O/P changes from 0 to Z, then we detect the fault.
- for NOR gate, find test vector.

A i/p, NMOS at open switch.

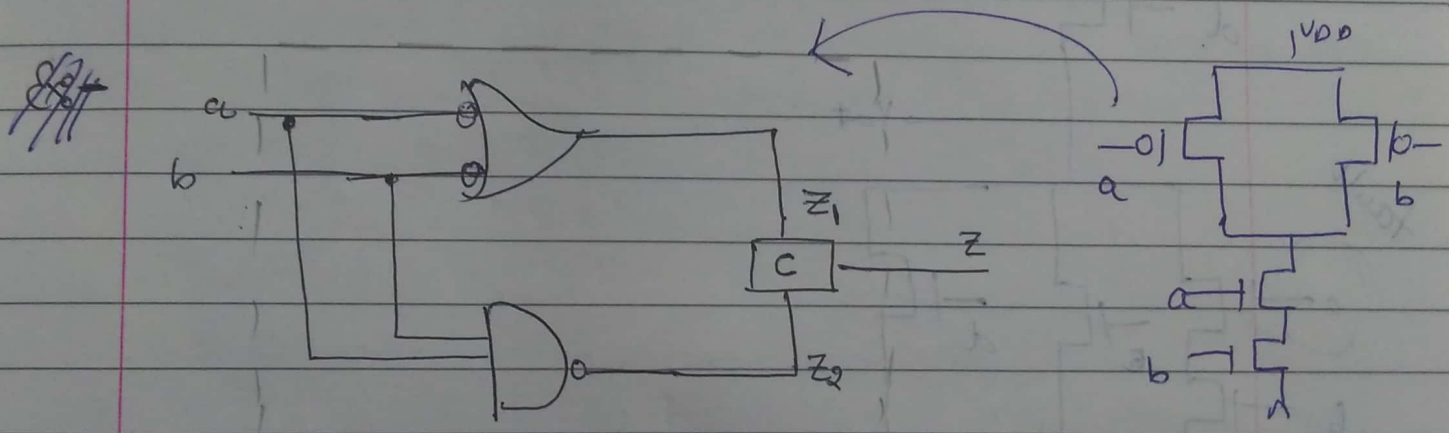


a	b	Z_n	Z_f
0	0	1	
0	1	0	
1	0	0	
1	1	0	



a	b	z_f	z_n
0	0	1	1
0	1	0	0
1	0	Z	0 ←
1	1	0	0

- High impedence Z means that it is stored the previous value of O/P.



b open, b sa1
b short, b sa0

z_1	z_2	C
1	0	1
0	1	0
1	1	short
0	0	open

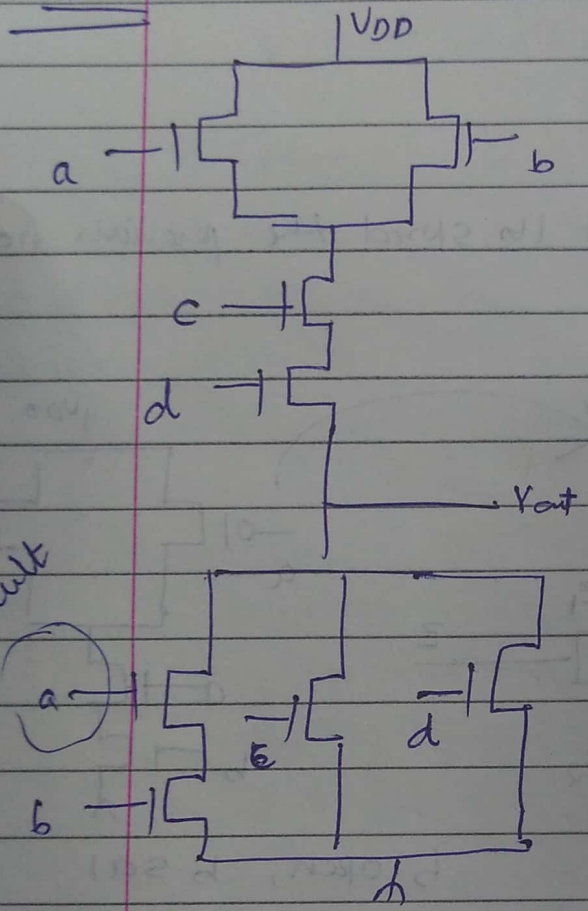
Solo

a	b	y	z_1	z_2
0	0		0	ϕ
0	1		1	1
1	0		1	1
1	1		1	0

Ex: $Z = ab + c + d$

fault at a → in n-mos open

Sol:



a	b	c	d	Z_n	Z_p
1	1	0	0	0	Z
0	0	0	0	1	1

- in P-MOS, open ckt. sa1 } for open ckt. testing
  in n-MOS, open ckt sa0 } we require two test vectors
- in P-MOS (a.b is in ~~parallel~~ series
  a+b is series)
  in N-MOS (a.b is → series
  a+b → Parallel)

Verification	Testing
• On Design	• On Device
• Pre-silicon	• Post silicon
• One time	• One all IC's
• By simulation, emulation formal methods	• By test gen^r (develop and test applicati
• A Design BUG (makes all IC's fabricated useless)	• A fabrication defect (may cause all ICs or some of IC's useless)

Defects, Errors and faults:

(I) Defects:
Process, age, material, Package defects

(II) Error: A wrong o/p signal produced by a defective sis

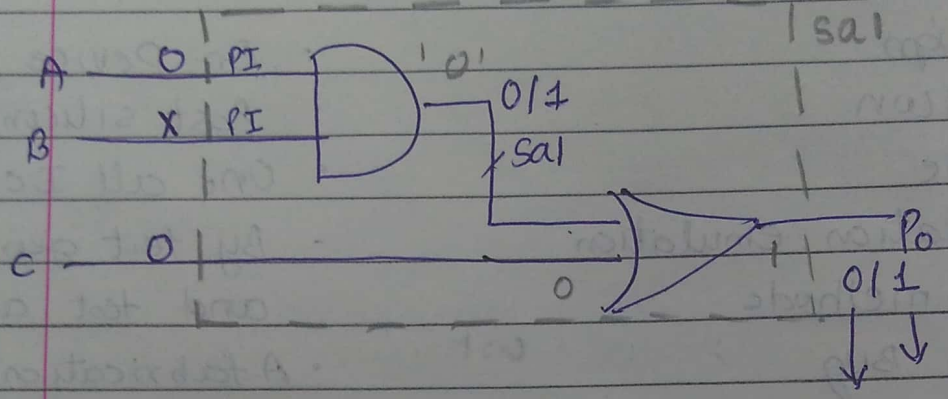
(III) fault: fault is imperfection in function
Defect " " " Hardware

• A representation of defect at the abstracted level → fault

stuck at faults : "classical faults" (?)
- simple to use
- they are found capable to detect other type of faults also
- Relates to yield modeling

FLAMINGO Date 20-Augo Page

Test Pattern Generation (Automated) (ATPG)



(1) fault activation

net 'l' sa v, $v \in \{0,1\}$

'l' $\bar{v}$

(2) line justification

Po → sa1 0/1

without fault | with fault

(3) fault propagation

C = 0

0 X 0

subset of XORing/Exhaustive { 0 0 1, 0 1 0 }

to get the algorithm

While XORing,

$z_f = 1 + c = 1$

$z_g = a \cdot b + c$

$z_f \oplus z_g = 1$

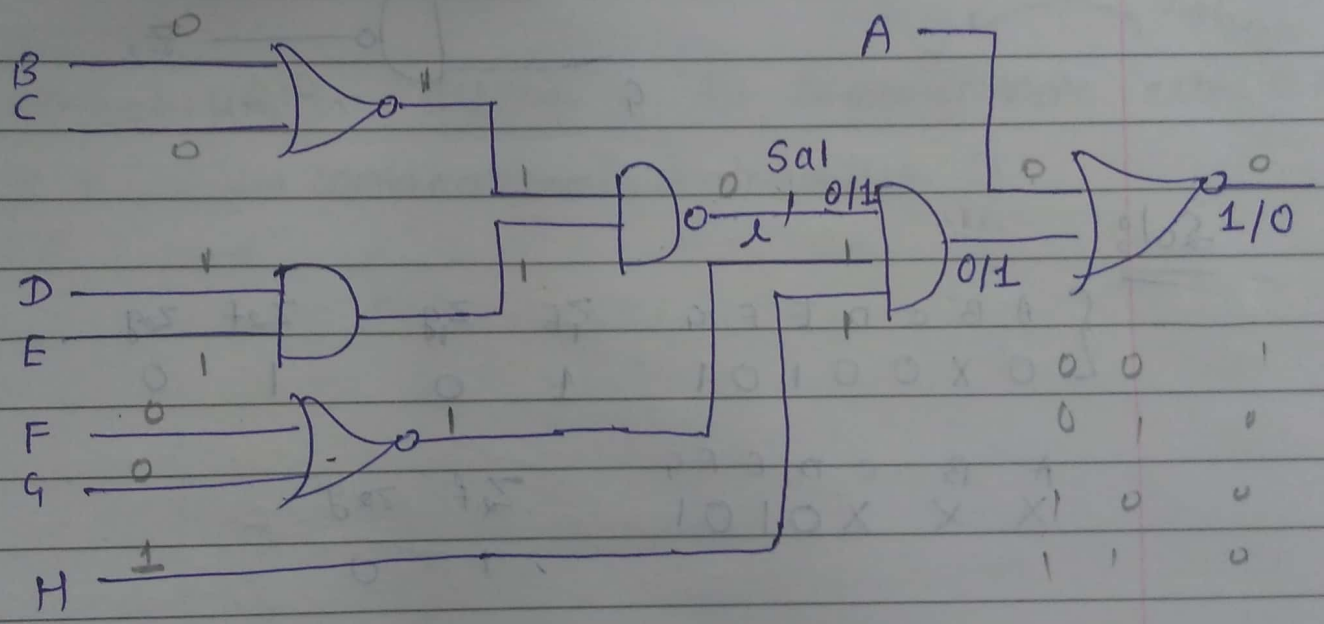
$\overline{(ab+c)} \cdot 1 = 1$

$(\bar{a}+\bar{b}) \cdot \bar{c} = 1$

$\bar{a}\bar{c} + \bar{b}\bar{c} = 1$

000
010
100
} Exhausted list of faults

Ex:



Sol:

"0 0 0 1 1 0 0 1" → Ans

$z_f = 0$, $z_g = 1$

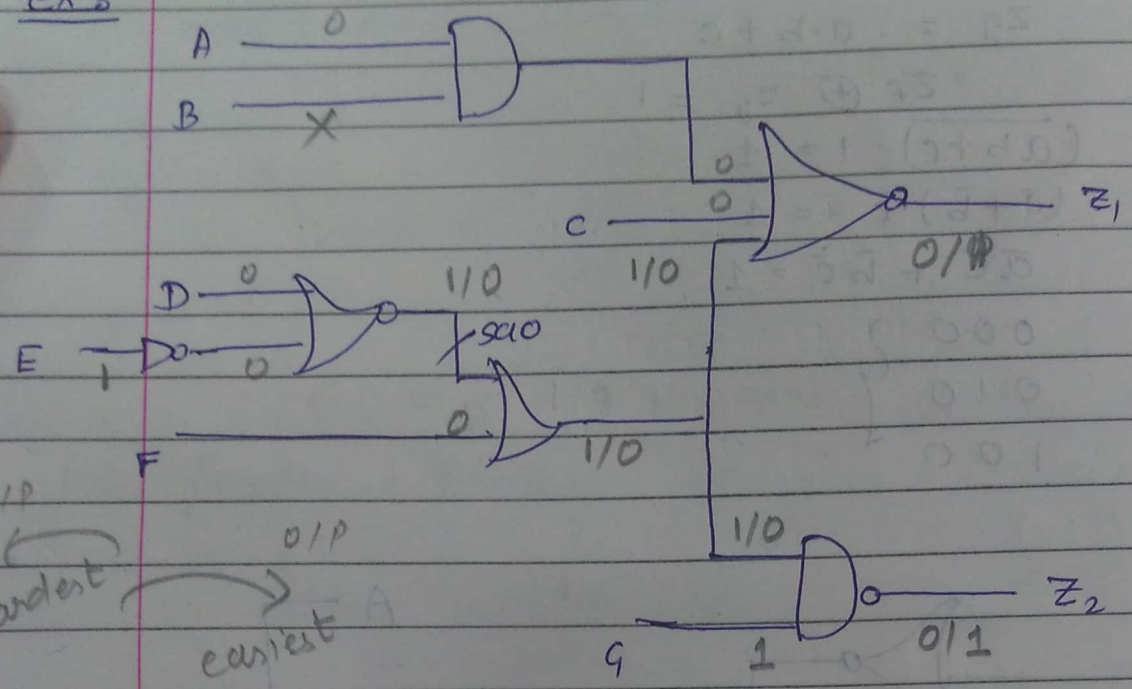
fault activation → sal

line justification → B = C = 0, D = E = 1

fault propagation (func Propagation
func justify)

⇒ 'l' sal 0 0 0 1 1 0 0 1 f_G 1 f_F 0

Ex :-

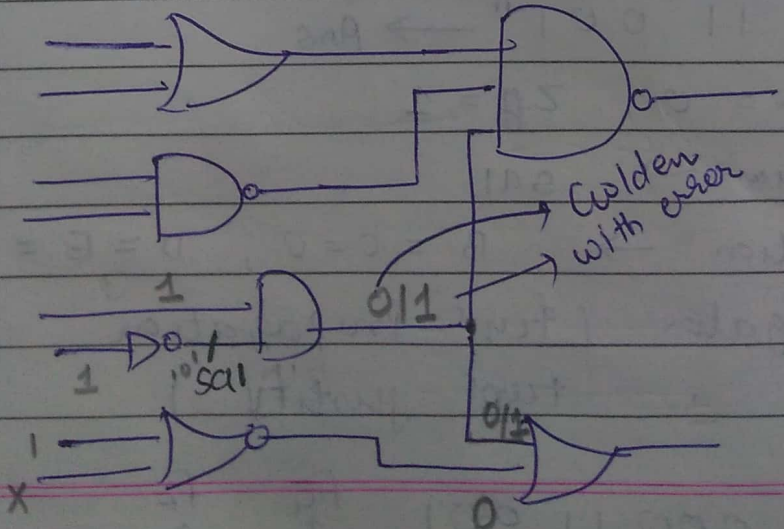


Sol :

A	B	C	D	E	F	G	Z_1f	Z_1g	Z_2f	Z_2g
0	X	0	0	1	0	1	1	0	1	0

A	B	C	D	E	F	G	Z_1f	Z_1g
X	X	X	0	1	0	1	1	0

Ex :-



Vector :

X X X X 11 1X

or

X X X X 11 X 1

FLAMINGO
Date 28-Aug
Page

* controllability } testability
observability } measures

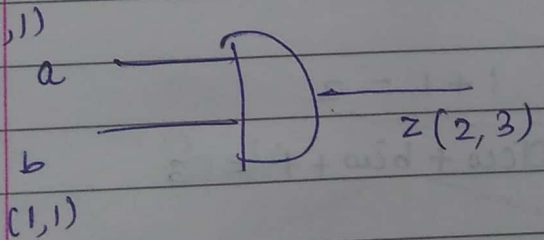
- Primary I/P controls easily and hardest to observed.
- Primary o/p easiest to observed and hardest to control.

	Control.	Observ.
PI	'1'	'1'
PO	'1'	'1'

- controllability is always 1 for primary I/Ps either 0 or 1.
- CC1 → combinational controllability '1'
- CC0 → " " '0'

(CC0, CC1)

Ex:



$Z_{CC0} = \min(a_{CC0}, b_{CC0}) + 1$

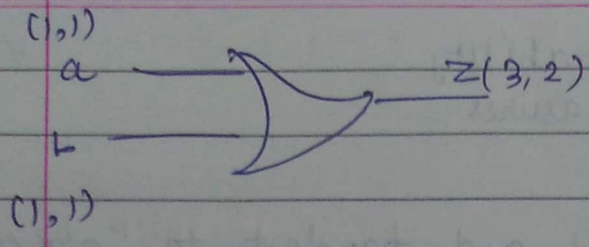
$= 1 + 1 = 2$

$Z_{CC1} = a_{CC1} + b_{CC1} + 1$

$= 3$

FLAMINGO
Date
Page

* OR



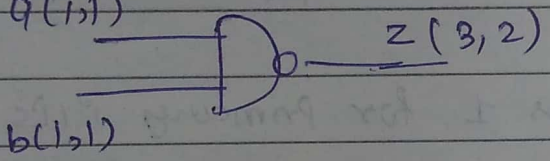
$Z_{CC0} = a_{CC0} + b_{CC0} + 1$

$= 3$

$Z_{CC1} = \min(a_{CC1}, b_{CC1}) + 1$

$= 1 + 1 = 2$

* NAND

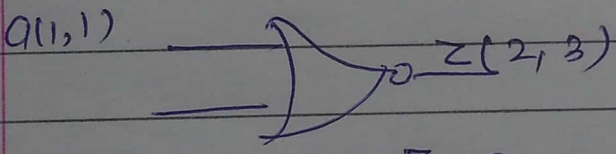


0	0	1
0	1	1
1	0	1
1	1	0

$Z_{CC0} = a_{CC1} + b_{CC1} + 1 = 3$

$Z_{CC1} = 1 + 1 = 2$

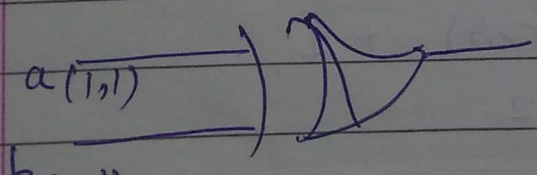
* NOR



$Z_{CC0} = 1 + 1 = 2$

$Z_{CC1} = a_{CC0} + b_{CC0} + 1 = 3$

* XOR



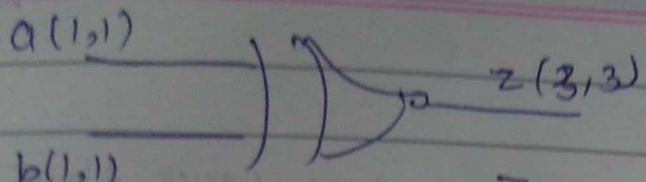
$Z_{CC0} = \cancel{1+1=2}$ $a_{CC0} + b_{CC0} + 1 = 3$

$Z_{CC1} = \cancel{1+1=2}$ $\cancel{1+1=2}$

$\min\{(a_{CC1} + b_{CC0}), (a_{CC0} + b_{CC1})\} + 1$

$= 3$

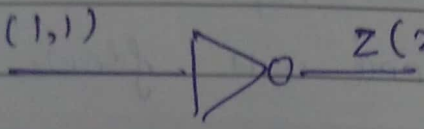
* XNOR



Z(c0 = 1+1 = 3

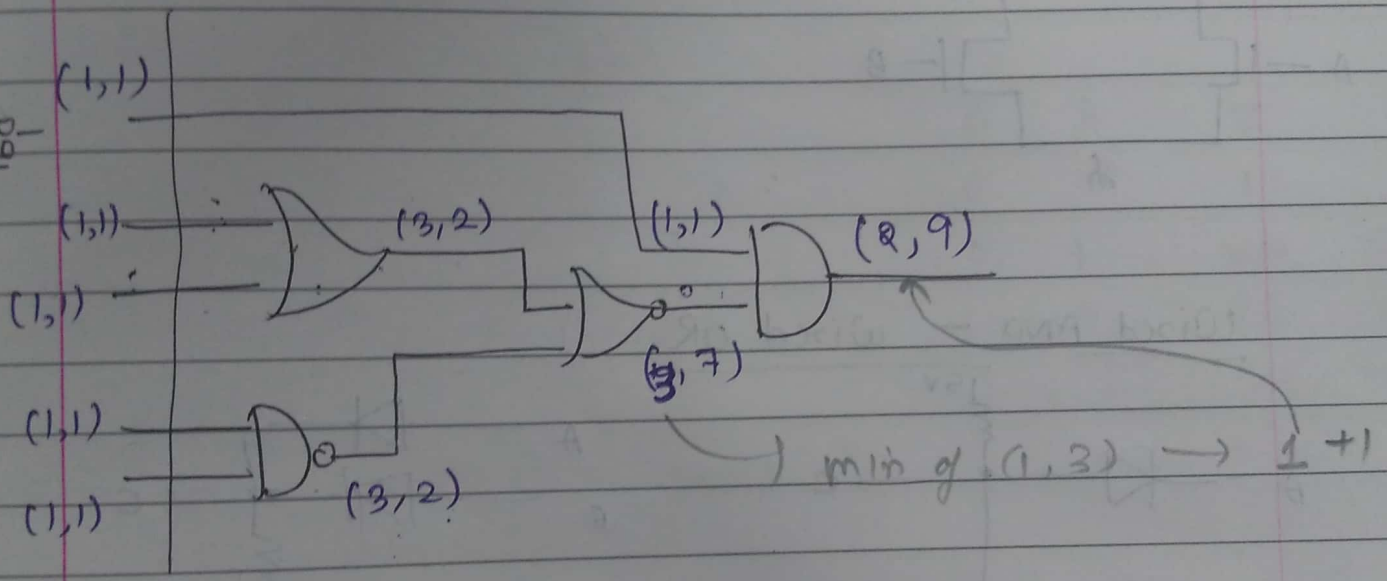
Z(c1 = 1+1+1 = 3

* NOT



Z(c0 = 1+1 = 2

Ex:-



30-Aug

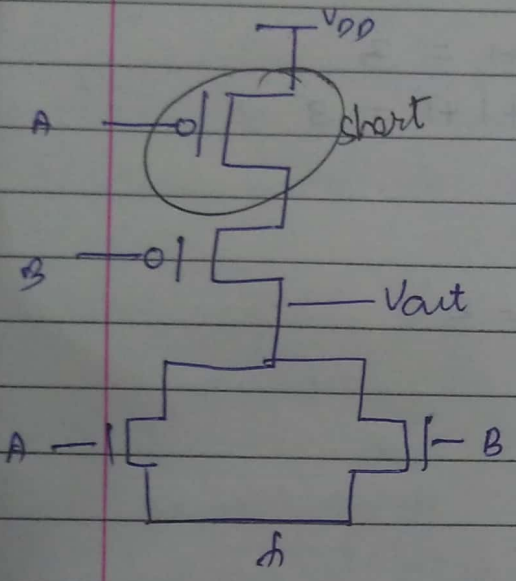
⇒ Faults

Nets
- Stuck @ 0
- " @ 1
- bridge

Gate
- transistor open (Pair of test vectors)
- " short

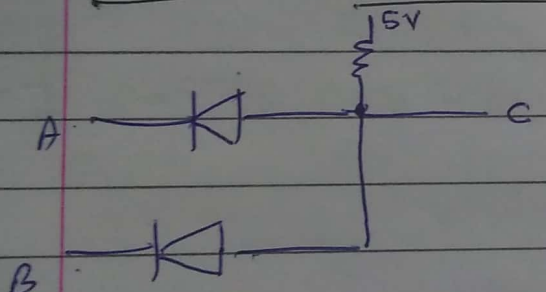
FLAMINGO
Date 30-Aug-0
Page

Stuck - short fault :

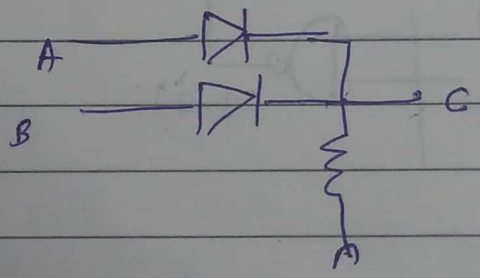


When A PMOS stuck short then A = 1, B = 0 ;
then high amount of I_{DDQ} will flow.

Wired AND - Wired OR

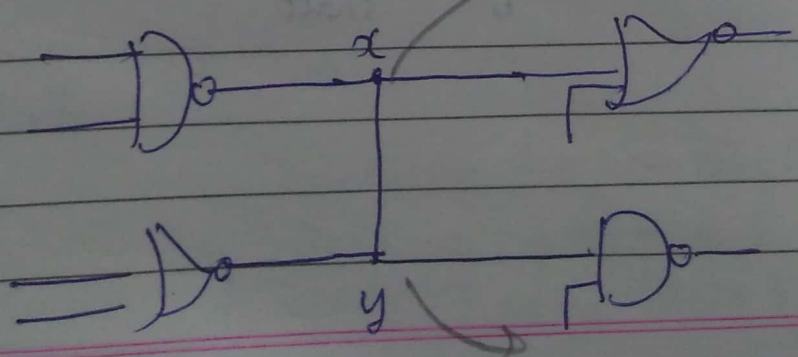


'AND'



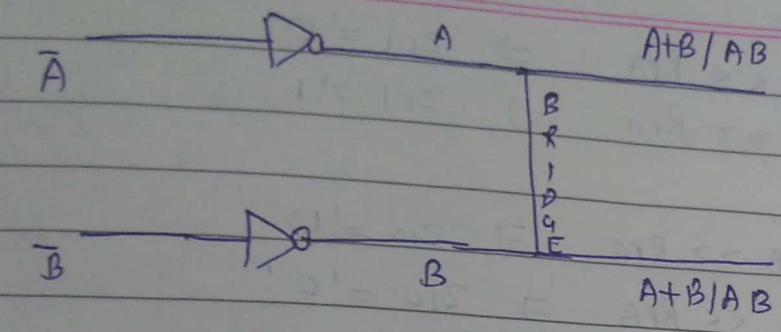
'OR'

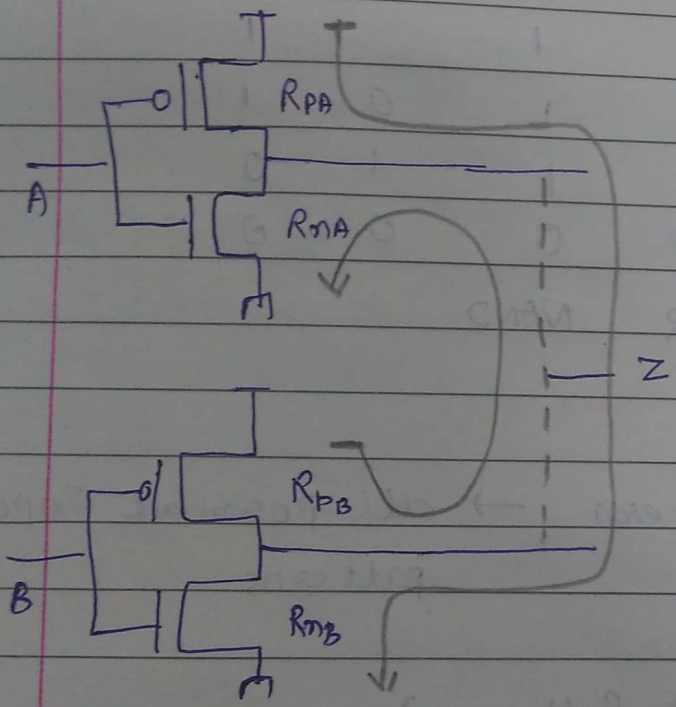
Bridge fault:



Here either x·y or x+y
Wired OR / AND
Here either x·y or x+y

Ex:





A	B	Z	
0	0	1	
0	1	Z_{01}	0/1
1	0	Z_{10}	1/0
1	1	0	

(ii) – When $A = 0$, $B = 1$

R_{PA} and R_{nB} conducting so

$$Z_{01} = \frac{R_{nB}}{R_{PA} + R_{nB}} \times V_{DD}$$

(iii) – When $A = 1$, $B = 0$

R_{nA} and R_{PB} conducting so

$$Z_{10} = \frac{R_{nA}}{R_{nA} + R_{PB}} \times V_{DD}$$

for

(ii) When $R_{nB} << R_{PA} \Rightarrow Z_{01} = '0'$

$R_{nB} >> R_{PA} \Rightarrow Z_{01} = '1'$

for (iii) when $R_{nA} >> R_{PB} \Rightarrow Z_{10} = '1'$

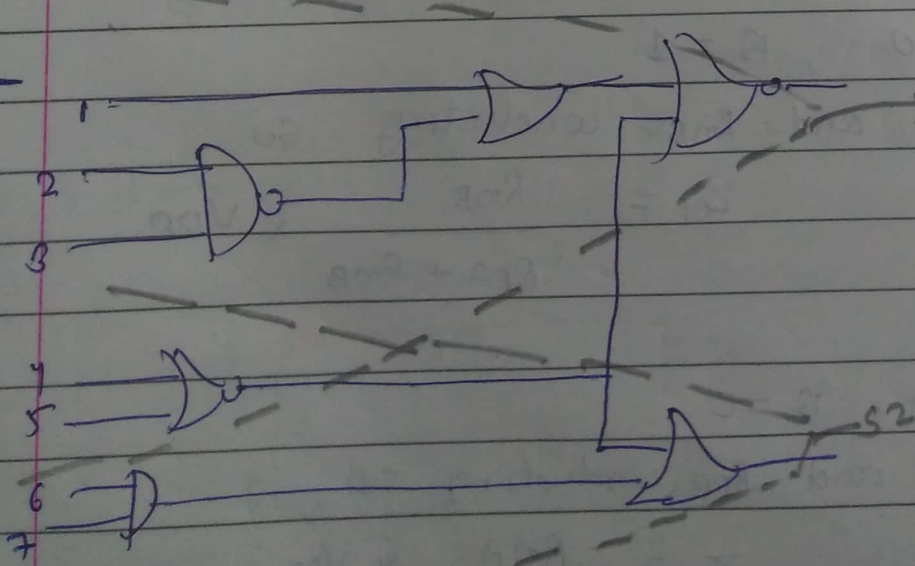
$R_{nA} << R_{PB} \Rightarrow Z_{10} = '0'$

A	B	Z			
0	0	1	1	1	1
0	1	0	1	0	1
1	0	0	1	1	0
1	1	0	0	0	0
		NOR	NAND		

Test Pattern :

(A)- Exhaustive test pattern → all possible input test patterns

(B) → Pseudo exhaustive test Pattern →



divide into sub ckts

$2^5 + 2^4$

$S_1 \quad S_2$

Exhaustive 2^7

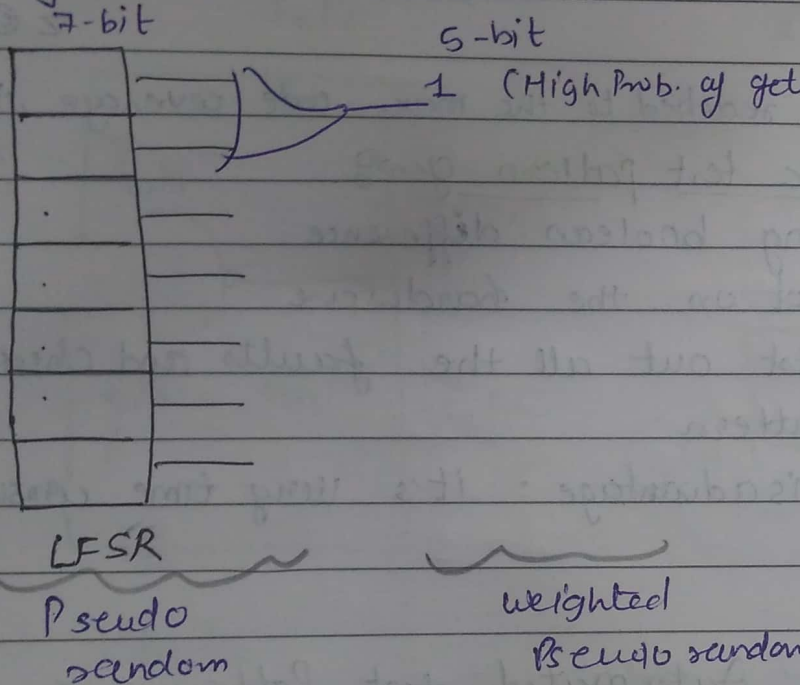
LFSR

Linear feedback shift register

- partition the ckt into subckts.
- each subckt are tested exhaustively.

(c) Random test Pattern

Weighted Pseudo random



1-Sept

Pseudo random pattern resistor fault → PRPR faults

for ex: if 8 i/p AND gate is there, if it is stuck at 0, then getting the o/p '1' all inputs are set to 1.

and for stuck at 1, to get the o/p '0'

$2^8 - 1 = 255$ → possible test vectors for getting o/p 0

FLAMINGO
Date 1-Sep
Page

We are planning for the test

(I) Fault simulation: We gives the random pattern, and how many fault can detect by this pattern → fault simulation

for ex: AND gate A=1, B=1, O/P = 1

it can detect A@0
B@0
Z@0

- if we reached to the max. code coverage, the stopped the process

(II) Deterministic test pattern gen:
- Using boolean difference
- not on the hardware
- list out all the faults and checks the test pattern
- Disadvantage: it is very time consuming

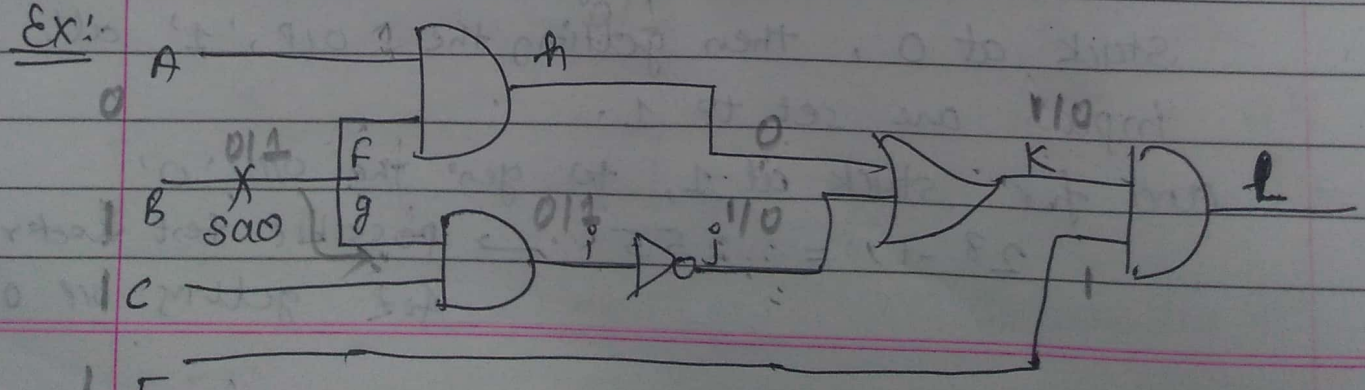
ATPG: Automated test Pattern genr

5 value Algebra: 0, 1, x, D, D bar

D → 1/0

$\bar{D}$ → 0/1

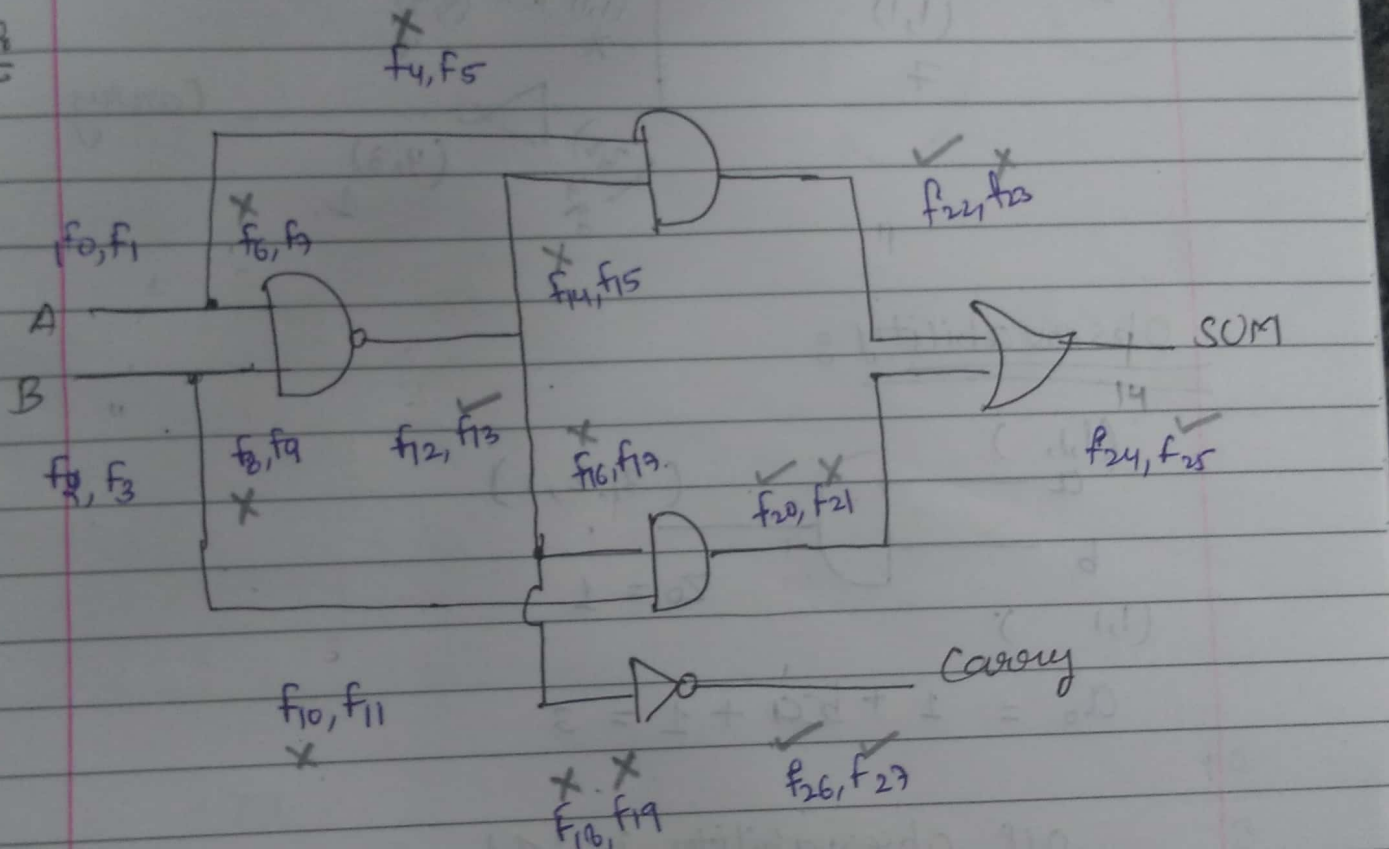
Ex:



Sol:

A	B	C	E	
0	1	1	1	→ test pattern

Ex9

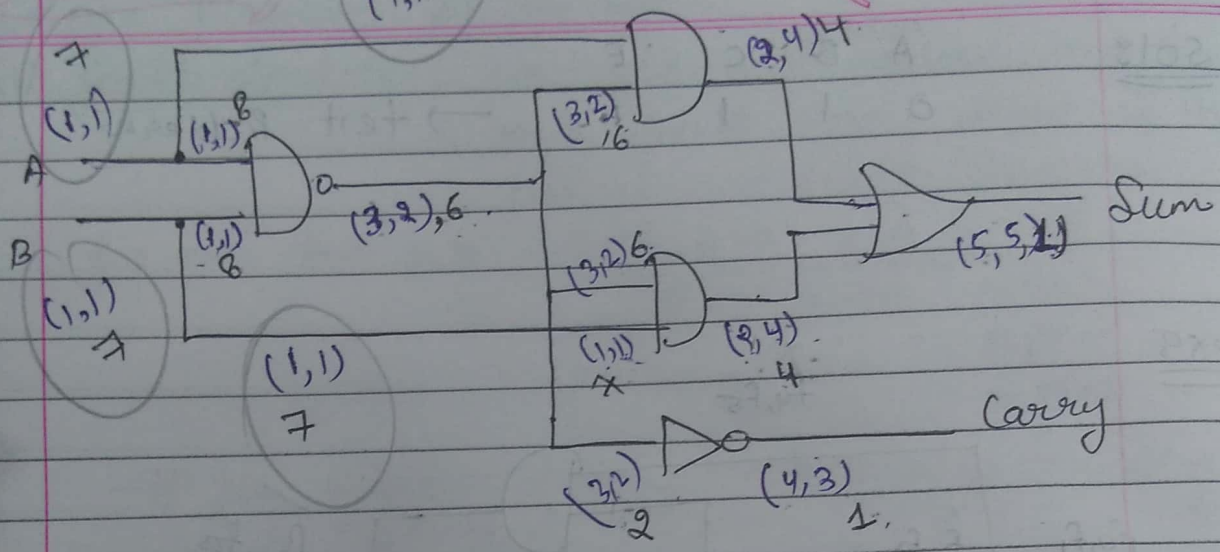


(i) fault equivalent ⇒ collapse ratio = $\frac{18}{28}$

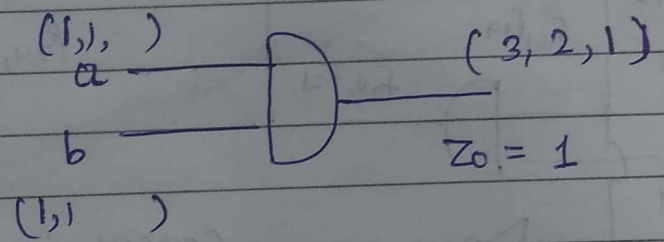
- 10 faults are removed by equivalency.

(ii) Controlability

(CC0; CC1)



Observability :



$a_0 = 1 + bca + 1 = 3$

O/P observability is '1'