12/7

- Levels of Abstraction

Requirement
↓
Specification
↓
Behavioural Description
↓
RTL (register transfer level)
↓
Gate level
↓
Transistor level
↓
Physical level design
↓
Fabrication

- Rule of 10

- Verification → checks the design
  Validation → rigorous testing.
  functionality testing.
  time consuming

Diagonosis → what is the fault? is found.
Device characterization → How can be corrected.
Failure mode analysis.
Detection
Burn-in / Stress test
Acceptance / Incoming

14/7

Verification — done on behavioural description.
Synthesis — • conversion from behavioural description to RTL
• conversion from one level to other.

Gate level Netlist — Gate delays are verified
Transistor level — delays in interconnects, i.e., in propagation of data is verified.

Fabrication / Manufacturability — • fabrication of small paths is possible or not
• Design Rule is checked whether manufacturing is possible with present machines. → done after Physical layout is made.

- Verification of millions of transistors — possible only by software
but if few transistors are present then verification can be done manually.

- How error can be removed
(i) Automation
(ii) Poka-Yoke
(iii) Redundancy
→ used only when repetative designs are used. ⇒ ∴ not used.

- Verification Methods.
Functional
Formal
~~Informal~~ Semiformal

- Functional Verification Approach
  Black box → mostly used now-a-days.
  White box
  Grey box

17/7 Test Bench code components → I/P generator
O/P " comparator.

19/7 → Limitation of Functional Verification
- Synchronous design — clock present bcoz of which clock reaches each module with some delay ⇒ causes misinterpretations.
- Clock Tree Synthesis (CTS) — used to remove — causes changes in RTL design.
- Formal Verification. → Mathematical Checking of design.

Static power — due to leakage current
Dynamic power — causes during transition from 1 to 0.

- Clock gating circuitry — used to reduce power
  — done by making a path in sleep mode by adding AND gates with some signal to turn a path ON.
  functionality of circuit doesnot change.

- Critical path → present only in combinational ckt
  — path having maximum delay.
  → In sequential path — present only if "delay > clock period"

- Adding test design in ckt (Design for test) — used for verification — causes less time to verify a design by verfication engineers
  — a extra circuit is added for testing

- Area ↑ ⇒ Speed ↑ Power ↓ ← flattening (area ↑)

VERIFICATION TOOLS.

(1) Linting. — Syntax and logical (till some extent) is checked.

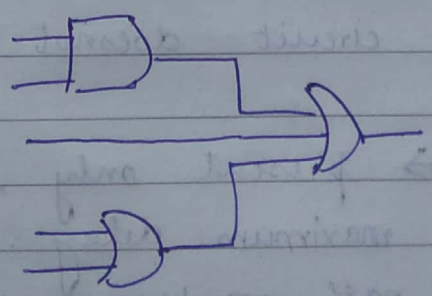
(2) Simulator — • virtual environmental for system
• functional, timing simulation
• will just provide o/ps — won't check them
• o/ps are calculated by creating mathematical model of design and stored in a log file

Can't be exhaustive to non-trivial designs — means all the input combination for n i/ps cannot be done if n is very large.

RTL SIMULATORS

— Event-based Regression — no. of calculations for a certain o/p is high.

non-trivial design



- require many iterations
- time consuming

~~faster~~

catches small glitches in seq., combination, Sync etc.

- cycle-driven — based on clock's edge — as o/p changes only when clock edge occurs and at that time i/ps change.
  - faster
  - less memory usage.
  - setup & hold time violations are checked
- co-simulator — work on all platforms
- waveform viewer → part of simulator.

21/7.

→ Coverage.

- 100 lines of code & 70 of them are executed then coverage = 70%. this is 'Code Coverge'.
- 'Functional coverage' — provides the coverage of how many functions are covered or how many i/p combinations have been covered.
- Functional coverage is more important.
- code coverage — completeness. — sometimes all lines are not needed. That would cause generation of unnecessary h/w and simulation time & memory.
- functional " — correctness.

So 1st code coverage is checked. ⇐

- code coverage provides just the path
- both coverage go hand-in-hand
- code coverage provides —
  (1) which i/p should be given next
  (2) which lines to be removed
  (3) which lines are to be optimized.

Profiling — perfect code to ↓ simulation time.

J. Bhaskar → for test bench

Types of code coverage
(1) Statement / Line coverage
(2) Block coverage
(3) Branch / decision
(4) Condition / expression
(5) Toggle
(6) State / FSM

Bug Tracking System (BTS)
Regression & Revision Tool.
Hardware Modeller

24/7

① Design for Verification (DFV)
- verification methodology.
- Addition of extra code for verification like addition of display, counter, adder, etc.
- Code for verification — should be mentioned to not be present in hardware synthesis

synthesis off
code for verify. } → added in code so that h/w not developed.
synthesis on.

② Verification of a Resuable design.
- IP cores.
↳ intellectual property.

③ Verification Reuse
RVM (Reusable Verification Methodology)

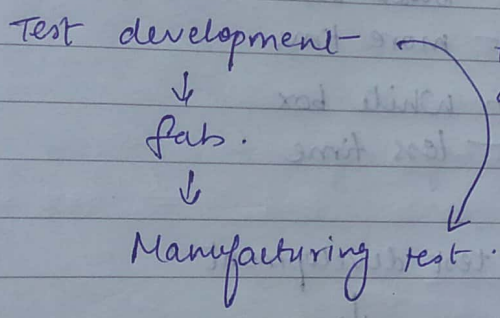
Generic code → Not synthesised

26/7

- off-line testing → testing of design done when it is removed from actual usage place
  on-line " → normal function & testing done simultaneously

26/7

- Verification → checks the functionality before fab.
- Test plan development → designs ckt for testing chip after fabrication. — reduces actual test time.
  - doesnot comes in time of time-to-market.
  - as done before fab. (during specification simultaneously Test is developed).
- Test application → done after fab. using CRO, probes, etc.
- Netlist, schematic, layout → test developed depending upon design in these.



- DFT (Design For Test) →
  - physical H/W adding in IC
  - not req. for functionality
  - Reduces test efforts.
  - Extra area, power consumption, man-power, delay

Built in Self Test (BIST) → almost same as DFT.

- Effective Testing

- Component Testing is done in 2 ways.
  - Functional testing
  - Structural testing
    (schematic, Layout size — transistor size, and all).

- Functional Verification
  - RTL, Layout → Formal Verification takes place after every stage.
  - After fab → If any problem occurs then structural testing takes place 1st, as functionality was fine

ATE → automatic test Equipment.

- Functional test — Black box, more time
  Structural test — White box, less time

- Verification. → test development.
  - 1st slot fabricated is validated rigorously
  - Then it is tested on mass fabrication

- Defect — not done properly at physical layers.
  eg:- doping n^+ for certain time, temp.

Fault — defect in form of logical manner.
eg:- functionality of gate will be wrong.
o/p comes to be wrong. ~~due~~ due to extra connection

Error — o/p is wrong.

- Testing → purpose is not to repair the defect just says whether we can use or not.

Diagonosis → tries to repair the defect

31/7

- Error — can be detected by some i/p combinations Not all i/p comb" will give us wrong o/p.
- Faults. in ⌊ gates / interconnects

- K = 2 → 2 faults ⌊ stuck at 1 / stuck at 0.

n → no. of wires/ports. = 3.

no. of multiple faults = $(k+1)^n - 1$

$= (2+1)^3 - 1$

$= 3^3 - 1 = 26.$

SSa single stuck at faults ⌊
- a s-a-0. — z s@0
- a s-a-1 — z s@1.
- b s-b-0
- b s-b-1

→ 1 fault at a time.

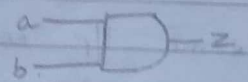
MSa multiple stuck at faults ⌊
- {a sa0, b sa0} (can also be written at b0. some goes with others)
- {a sa0, b sa1}.
- :
- {a sa0, b sa0, z sa0} → 3 fault at a time
- :

→ 2 fault at a time.

Assume — 1 fault at a time
No bridge faults.

no. of faults checked = $2 \times n$ $(k * n)$

2/6



$z_g \rightarrow$ golden o/p (ideal o/p)
$z_f \rightarrow$ faulty o/p.

$z_g = ab$.

If $z_g = 0$ then for given test vector $z_f = 1$
$z_g = 1$ " $z_f = 0$.

$z_g \oplus z_f = 1$ (Taking a sa1)

$ab \oplus 1 \cdot b = 1$

$(ab)\,b' + (ab)'\,b = 1$

$0 + (a' + b')\,b = 1$

$a'b = 1$

$\Rightarrow a = 0 \quad b = 1. \Rightarrow$ test vector = 01.

	test vectors.
a1	01
a0	11
b1	10
b0	11
z1	00, 01, 10
z0	11

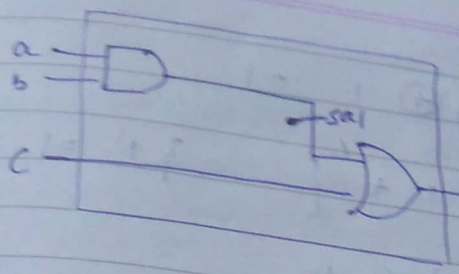
$\{a_0, b_0\} \equiv z_0$
$\{a_0, b_1\} \equiv z_0$
$\{a_0, z_0\} \equiv z_0$
$\{a_0, z_1\} \equiv z_1$

⇓

all these msa can be removed as they get ~~cover~~ covered in ssa test vectors.

$\Rightarrow$ 95 - 98 % of msa is covered in ssa

g.



$z_g = ab + c$

$z_f = 1$

000 , 010 , 100 ← Test vectors.

$z_g \oplus z_f = 1$

$(ab+c) \oplus 1 = 1$

$(ab+c)' \cdot 1 + (ab+c) \cdot 0 = 1$

$(ab+c)' = 1$

$(ab)' c' = 1$

$(a'+b')c' = 1$

$a'c' + b'c' = 1$

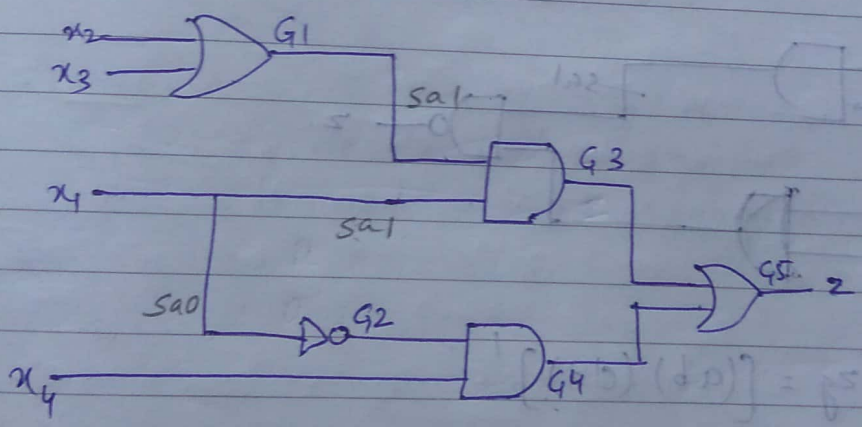
$a'c'(b+b') + b'c'(a+a') = 1$

$a'bc' + a'b'c' + ab'c' + a'b'c' = 1$

$a'bc' + a'b'c' + ab'c' = 1$

010 000 100

eg.



$z_g = (x_2 + x_3)x_1 + x_1' x_4$

$z_f = 1 \cdot 1 + 1 \cdot x_4 = 1 + x_4 = 1$

$z_g \oplus z_f = 1$

$[(x_2 + x_3)x_1 + x_1' x_4) \oplus 1 = 1$

$\{(x_2 + x_3)x_1 + x_1' x_4\}' \cdot 1 + \{\quad\}\, 1' = 1$

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

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

$(x_2' x_3' + x_1')(x_1 + x_4') = 1$

$x_1 x_2' x_3' + x_2' x_3' x_4' + 0 + x_1' x_4' = 1$

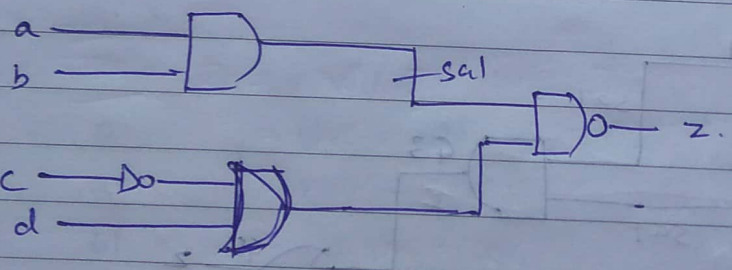
$x_1\ x_2\ x_3\ x_4$

$1\,0\,0\,x$ $\quad x\,0\,0\,0$ $\quad 0\,x\,x\,0$

1000	1000	0000
1001	0000	0010
		0100
		0110

Test vector → 0000, 1000, 1001, 0010, 0100, 0110.

eg.



$z_g = [(ab)(c'd)]'$

$z_f = [1 \cdot (c'd)]' = (c'd)'$

$z_g \oplus z_f = 1$

$[(ab)(c'd)]' \oplus (c'd)' = 1$

$[(ab)' + (c'd)'] \oplus (c'd)' = 1$

$[(ab)' + (c'd)']\, c'd + [(ab') + (c'd)']'\, (c'd)' = 1$

$(ab)'\, c'd + (ab')'(c'd) \cdot (c'd)' = 1$

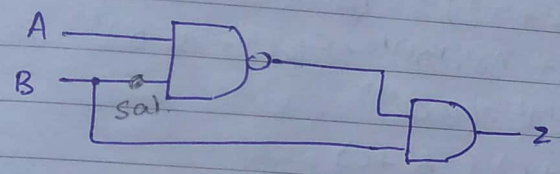
$(ab)' c'd = 1$

$(a' + b')\, c'd = 1$

$a'c'd + b'c'd = 1$

$a'c'd = 1$	$b'c'd = 1$	abcd.
0 x 0 1	x 0 0 1	{ 0 0 0 1
0 0 0 1	0 0 0 1	0 1 0 1
0 1 0 1	1 0 0 1	1 0 0 1 }

eg.



$Z_g = \bar{A}\bar{B}\, B = \bar{A}B.$

$Z_f = \overline{\bar{A} \cdot 1}\, B = \bar{A}B$

$Z_g \oplus Z_f = 1$ ⟶ $\bar{A}B \oplus \bar{A}B = 1$

$0 = 1$

$\bar{A}\bar{B}\, B \oplus \bar{A}B = 1$

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

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

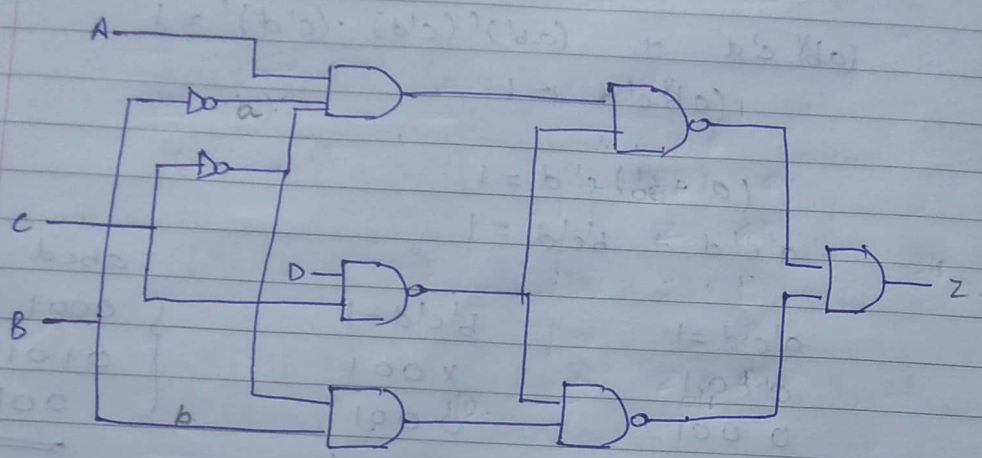
$(\bar{A}B)'\, \bar{A}B + (\quad) = 1$

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

$0 = 1$

⇒ The defect can be detected through any test vector.

eg. Find out test vector set for:
ssa → a1 and b0
msa → {a1, b0}



$z_g = (AB'C' + \overline{BC})' (BC' \overline{CD})'$

$= (AB'C'(C'+D'))' (BC'(C'+D'))'$

$= (AB'C' + AB'C'D')' (BC' + BC'D')'$

$= (AB'C')' (BC')'$

$= (A'+B+C)(B'+C)$

$= A'B' + A'C + BC + B'C + C = A'B' + A'C + C$

a1 $z_f = (AC' \overline{CD})' (BC' \cdot \overline{CD})'$

$= (AC'(C'+D'))' (BC'(C'+D'))'$

$= (AC' + AC'D')' (BC' + BC'D')'$

$= (AC')' (BC')'$

$= (A'+C)(B'+C)$

$= A'B' + A'C + B'C + C = A'B' + A'C + C$

b0 $z_f = 0$.

a1, b0 $z_f = 0$.

9/8

00
01
10
11

a1. $z_g \oplus z_f = 1$

$(A'B' + A'C + C) \oplus (A'B' + A'C + C) = 1$

$0 = 0$ → no test vector.

b0. $z_g \oplus z_f = 1$

$(A'B' + A'C + C) \oplus 0 = 1$

$A'B' + A'C + C = 1$.

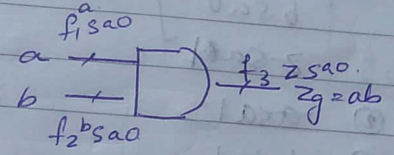
00xx 0x1x xx1x

{0000, 0010, 0001, 0011
0100, 0111
1010, 1011, 1110, 1111}

9/8

FAULT EQUIVALENCE

→



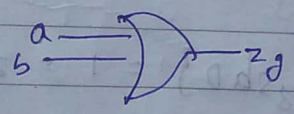
① a sa0, b sa0, z sa0 ② a sa1 ③ b sa1 ④ z sa1.

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

$z_{f1} = 0$
$z_{f2} = 0$
$z_{f3} = 0$.
→ all these 3 faults are similiar. So any one of them can be used

Earlier we were supposed to consider 6 faults but here we require to consider only 4 faults.

→



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

① a sa1, b sa1, z sa1 → same.
② a sa0
③ b sa0
④ z sa0.

→

0 1 ; } sa1

0 1 0 } sa0

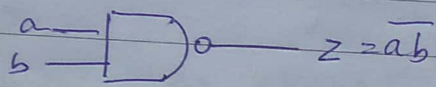
① a sa0
② a sa1 } ③ z sa0 }
④ z sa1

→

① a sa1
b sa1
z sa0 }

② a sa0
③ b sa0
④ z sa1.

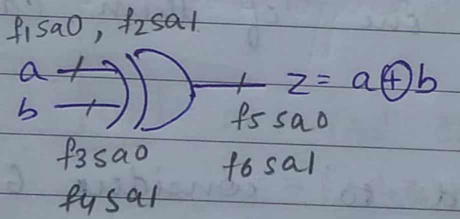
→



① a sa0
b sa0
z sa1 }

② a sa1
③ b sa1
④ z sa0

→



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

$z_{f_1} = b$

$z_{f_2} = \bar{b}$

$z_{f_3} = a$

$z_{f_4} = \bar{a}$

$z_{f_5} = 0$

$z_{f_6} = 1$

$\{f_1 sa0,\ f_4 sa1\} = 1 = z_{f_6}$.

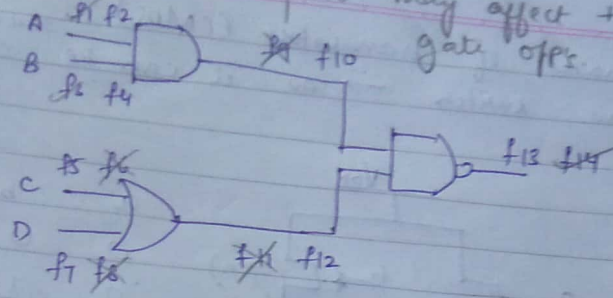
$\{f_2 sa1,\ f_3 sa0\} = 1 = z_{f_6}$.

Multiple 'stuck at' in inputs are not to be checked as they are covered in ssa

Scanned by CamScanner

as f_9 is removed so at i/p side both f_1 & f_3 are kept as it may affect the previous gate o/p's.

eg.

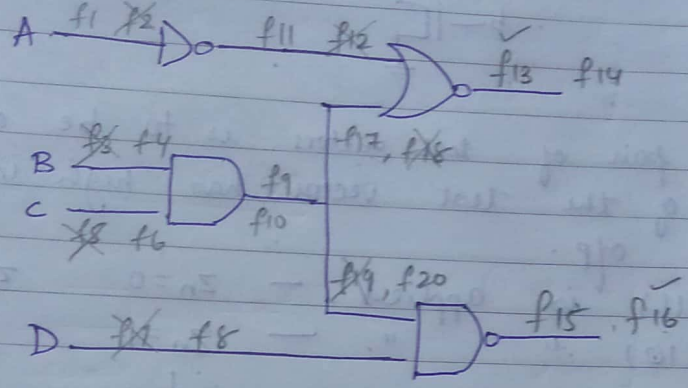


odd no – sa0
even no – sa1

collapse ratio
$= \frac{10}{14}$.

10 are considered instead of 14.

eg.

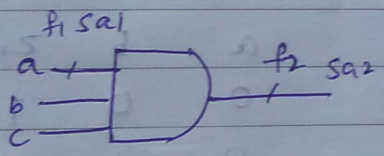


collapse ratio
$= \frac{13}{20}$.

All the faults at stems kept as it is. They are not removed. ⇒ f_9 & f_{10} are not cancelled.

11/8

FAULT DOMINANCE



Test vectors for

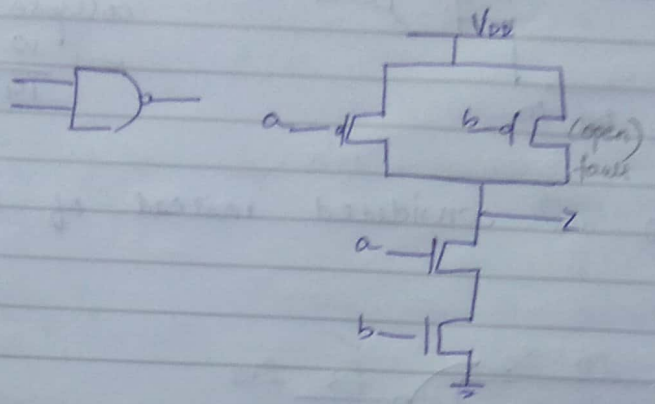
f_1 (abc)	f_2 (abc)
011	000
	001
	010
	011
	100
	101
	110.

t_{f_2} contains t_{f_1} Test vector so as f_1 test vector is small

⇒ f_1 dominant on f_2 as combination of f_1 will also cover error of f_2.

- Gates are made of transistors (CMOS)
- CMOS acts as switch → so it can be stuck at close (1) or open (0)

→



a	b	z_n	z_f
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Here a pair of test vectors is to be applied as one of the test vector has high impedance state as o/p.

⇒ 1st (11) is applied — $z_n = 0$ $z_f = 0$

2nd (10) " " — $z_n = 1$ $z_f = 0$

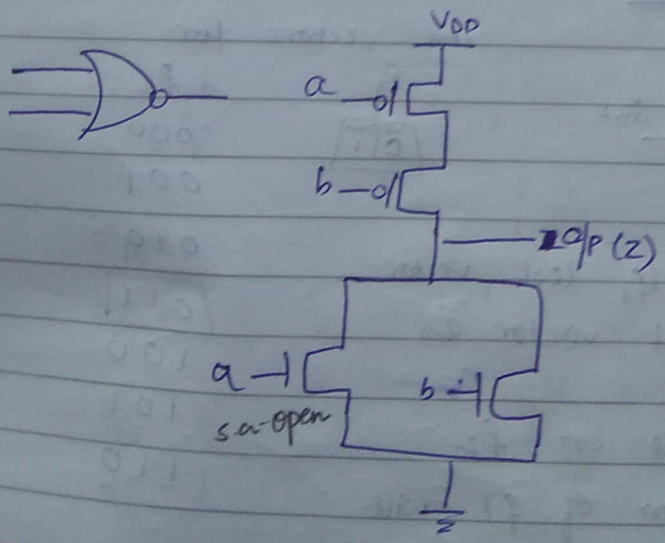
↓ transition takes place

↓ no transition taken place at o/p

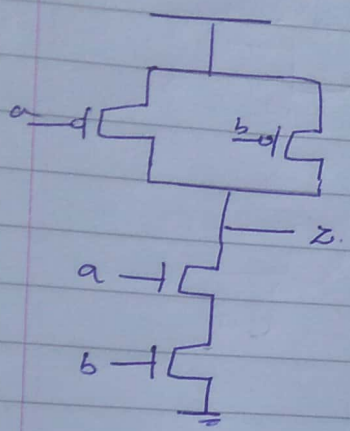
⇒ fault detected

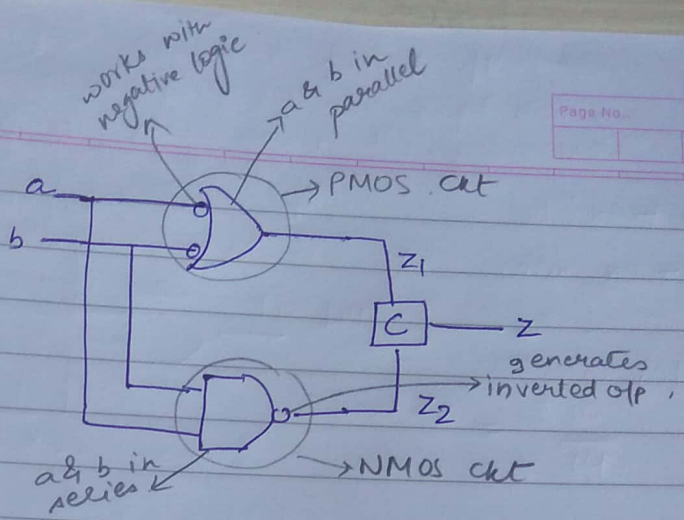
11, 10 → test vector to be applied

→



a	b	z_n	z_f
0	0	1	1
0	1	0	0
1	0	0	z
1	1	0	0





z_1	z_2	C (combine)
1	0	1
0	1	0
1	1	short
0	0	open