⊛ Testing Of Digital ~~Circuits~~ systems

- In any digital system, many faults may occur in both the chip level and the board level, so it is very important ~~for the man~~ to test digital systems before sending them to the market.

- The later a fault is noticed, the more expensive to repair. So it is important to spend a lot of effort on designing digital systems which are easy to be tested (Design For Testability DFT).

⊛ There are - in general - two approaches of testing

1- Functional Testing (Exhaustive Testing):
In this approach we apply all possible combinations to the inputs of the system and test the output for each combination.

2- Structural Testing (Non-Exhaustive Testing):
In this approach we apply some combinations to check the existence of faults or not.

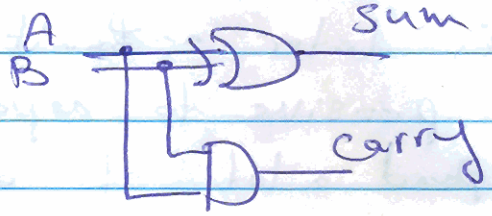
⊛ Basic Testing Procedure :-

The basic procedure is very simple

1- Initialise the circuit into a known state
2- Apply the test inputs
3- observe the outputs and compare with expected

Ex: Half Adder

A	B	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



- Now let's imagine that a fault has occured such that the node sum has been short circuited to the Vcc. ⇒ the new truth table is

A	B	sum	carry
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	1

1- Exhaustive Testing : apply all possible combination ($2^2 = 4$ combinations) and observe the output.

In general if we have a combinational circuit with n input ⇒ we need 2^n combinations of test inputs.

For example, if we have 32 inputs ⇒ we need $2^{32} \approx 4 \times 10^9$ test inputs. If we have 1 GHz clock for test generation ⇒ this will take 4 seconds,

but if we have a circuit with 64 inputs ⇒ we need $2^{64} \approx 2 \times 10^{19}$ test inputs. If we have 1 GHz clock for test generation ⇒ this will take 585 years.

2- Non-Exhaustive Testing :-

if we can apply only two input tests to test a fault in the sum ⇒ we can take

1) A = B = 0
2) A = 0 , B = 1

A	B	Sum	Carry
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	1

These two tests are enough to test whether the sum node is short circuited or not.

But if we choose 1) A = 0, B = 1
2) A = 1, B = 0
⇒ we will not detect the fault.

⇒ if we cannot use all possible test vectors in a non-exhaustive test, and have to choose a subset, then, in general, some set of test vectors will detect more faults than this is called the

Fault modelling :-

There are many types of faults may occur in digital systems, But there are two types are very common

1- node is stuck-at-1 (s-a-1), in which the node is short circuited with Vcc

2- node is stuck-at-0 (s-a-0) in which the node is short circuited with the ground

In our test procedures we will assume that these are the only types of fault that can be present at any node.

⇒ slow increase between the number of nodes & the number of test vectors.

Path Sensitisation Method (2 value logic)

Procedure :-

For each node in the circuit

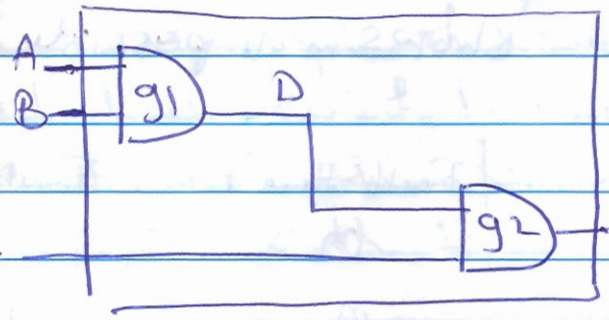
Assume the fault type

1- do Backtrace phase : Attempt to drive the node to the non-fault condition

2- do propagation phase : steer the content of the node to an output where it can be observed.

(control of inputs & observe the outputs)

Ex:



(we can control A and B, we can observe E);

To test node D:

1- Assume D is stuck-at-0 (D/0)

2- Backtrace phase: Find the inputs which will attempt to drive node D to 1

$\Rightarrow (A=1, B=1)$

3- Find a set of inputs which will steer the contents of node D to node E

$\Rightarrow C=1$

Now we have the following test vector (A,B,C) = (1,1,1) that tests whether D is s-a-0 or not.

Now if E=1 $\Rightarrow$ circuit is OK.

if E=0 $\Rightarrow$ if D is s-a-0. (bad).

4. assume D is s-a-1

5- Backtrace phase $\Longrightarrow$ (A,B) = (0,0) or (0,1) or (1,0)

6- propagation phase $\Longrightarrow C = 1$

$\Rightarrow$ test vector to test if D is s-a-1 are (0,0,1) or (0,1,1) or (1,0,1).

$\Rightarrow$ E=0 if no-fault

E=1 if D is s-a-1 (bad).

* The following table shows all possible fault

Fault	Inputs: ABC	Fault Free E	Faulty E
A s-a-1	011	0	1
B s-a-1	101	0	1
C s-a-1	110	0	1
D s-a-1	(001) or (101) or (011)	0	1
E s-a-1	(000) or (001) or (010) or (101) or (110) or (001) or (100)	0	1
A s-a-0	111	1	0
B s-a-0	111	1	0
C s-a-0	111	1	0
D s-a-0	111	1	0
E s-a-0	111	1	0

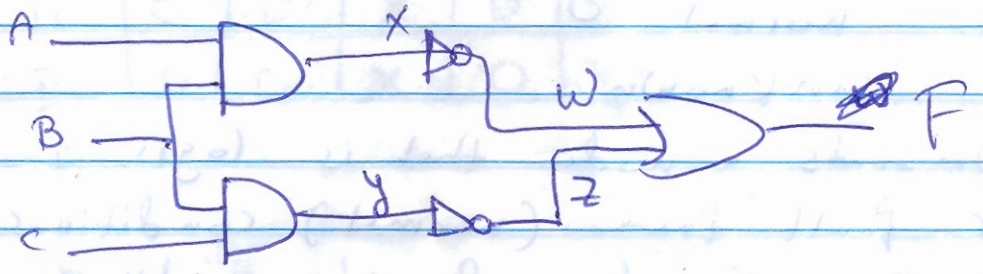
notes

1- inputs 111 can be used to test many faults at the same time. This is good if we want to distinguish between good and bad circuit and through out the bad circuit.
This bad if we want to repair the fault (diagnosis)

2- 101 can test many faults (efficient but ambig
but 000 to test if E is s-a-1
(inefficient but unambiguous).

* The disadvantage of the simple path sensitisation method is its failure in some cases to propagate the value of the required node in the proper path

Ex:



To test if B is sa1 ⇒

1) in order to test if B is s-a-1 ⇒ B = 0
2) in order to propagate the value of B to point x ⇒ A = 1.
3) in order to propagate the value in node w to node F ⇒ z = 0
4) if z = 0 ⇒ y = 1
5) if y = 1 ⇒ c = 1 and B = 1

↳ CONTRADICTION?!

⊛ 5-Value Logic (D-Algorithm).

- The 5 value Logic is specially designed to provide a convenient treatment of s-a-1 and s-a-0 faults.
- The five value logic are

1) 1 : ~~no fault~~ normal 1
2) 0 : normal 0
3) X : unknown
4) D : represents a node that is logic 1 under fault free (normal) conditions and logic 0 under faulty conditions.
5) $\bar{D}$: represents a node that is logic 0 under fault free (normal) conditions and logic 1 under faulty conditions

⊛ The following are the main logical operation using 5 value logic

1) not operation : Z = not A

A	Z
0	1
1	0
X	X
D	$\bar{D}$
$\bar{D}$	D

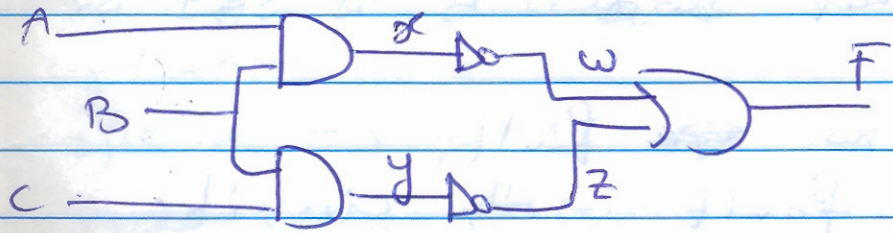
2) AND operation : $Z = A \cdot B$

A \ B	0	1	X	D	$\bar{D}$
0	0	0	0	0	0
1	0	1	X	D	$\bar{D}$
X	0	X	X	X	X
D	0	D	X	D	0
$\bar{D}$	0	$\bar{D}$	X	0	$\bar{D}$

3) OR operation : $Z = A + B$

A \ B	0	1	X	D	$\bar{D}$
0	0	1	X	D	$\bar{D}$
1	1	1	1	1	1
X	X	1	X	X	X
D	D	1	X	D	1
$\bar{D}$	$\bar{D}$	1	X	1	$\bar{D}$

Exc.



To test if B is sa1 ⇒ try the ~~Prop~~ propagation path BxwF only

1) B is sa1 ⇒ we must put 0 ⇒ $\bar{D}$

2) to propagate $\bar{D}$ to point x ⇒ A = 1

⇒ w = D

3) to propagate D from w to F we need 0 at z (or $\bar{D}$)

4) if $z=0 \Rightarrow y=1 \Rightarrow B=C=1$

↳ contradiction !!

- so path BxwF alone is failed to propagate the value of B to F output.

- also path ByzF alone will fail (the same w

- if we try to propagate B using both paths (BxwF & ByzF) ⇒

1) A = 1 to propagate node B to node x

C = 1 " " " B to node y

now $x = \bar{D}$, $y = \bar{D}$ ⇒ $w = D$, $z = D$

⇒ $F = D$

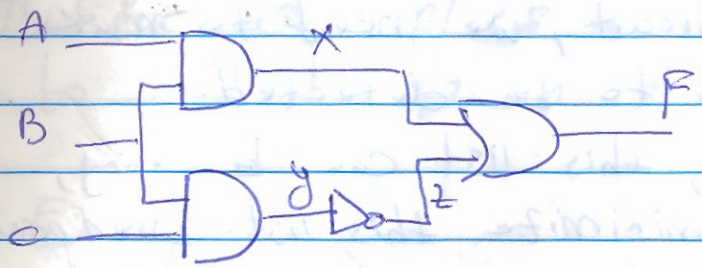
⇒ input test (A, B, C) = (1, 0, 1) (101)

Can test whether node B is sa1 or not.

if F = 1 ⇒ no fault

if F = 0 ⇒ fault in the circuit.

Ex.



test if B is sa0

① Path B → x → F

1) B = D

2) A = 1 ⇒ x = D

3) z = 0 ⇒ y = 1 ⇒ B = 1 , C = 1

↓ not sure ??

⇒ Path fail

② using Path B - x - F & B - y - z - F

1) B = D

2) A = 1 C = 1

⇒ x = D

y = D ⇒ z = $\bar{D}$

D or $\bar{D}$ = 1 (Fail)

③ using path B → y → z → F

1) B = D

2) C = 1 ⇒ y = D ⇒ z = $\bar{D}$

3) x = 0 ⇒ A = 0

⇒ F = $\bar{D}$

if F = 0 (no fault)

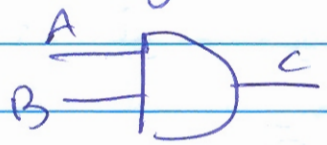
if F = 1 (fault) ~~in the system~~

⊛ Fault Collapsing:

- to test a digital circuit, we need to make a list of all faults to be detected.
- for large circuits, this list can be long, so it is good to minimize this list when possible
- Fault collapsing can reduce the size of fault list with two concepts: fault equivalence and fault dominance.

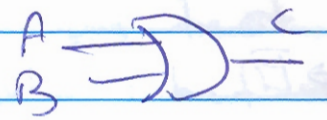
- Fault equivalence: Two faults are said to be equivalent if every test pattern that detects one of the faults also detects the other fault. (i.e their test sets are identical).

Ex: AND gate



A- S-a-0
B-S-a 0
c S-a-0
} equivalent
⇒ (nand A/0, B/0, c

OR gat



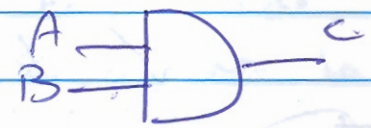
A S-a-1
B S-a-1
c S-a-1
} equivalent faults
⇒ (nor A/1, B/1, c/0

(not A/0, 2/1)
A/1, 2/0}

- Fault Dominance: ~~A fault, f_1, is said~~
  - a fault (f_1) dominates another fault f_2 if the test set of f_2 is a subset of the test set of f_1. Any test pattern that detects f_2 will also detect f_1.

Therefor, f_2 implies f_1 and it is sufficient to include f_2 in the fault list.

Ex:



A s-a-1 ⇒ 01

B s-a-1 ⇒ 10

C s-a-1 ⇒ ~~00, 01, 10~~ 00 or 01 or 10

⇒ C s-a-1 dominates A s-a-1 & B s-a-1

So it is possible to drop this fault since it will be discovered by either A s-a-1 or B s-a-1.

OR gate

~~A s-a-0~~

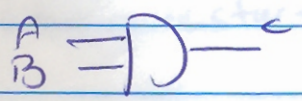
A s-a-0, B s-a-0 → C s-a-0

nand

A s-a-1, B s-a-1 → C s-a-0

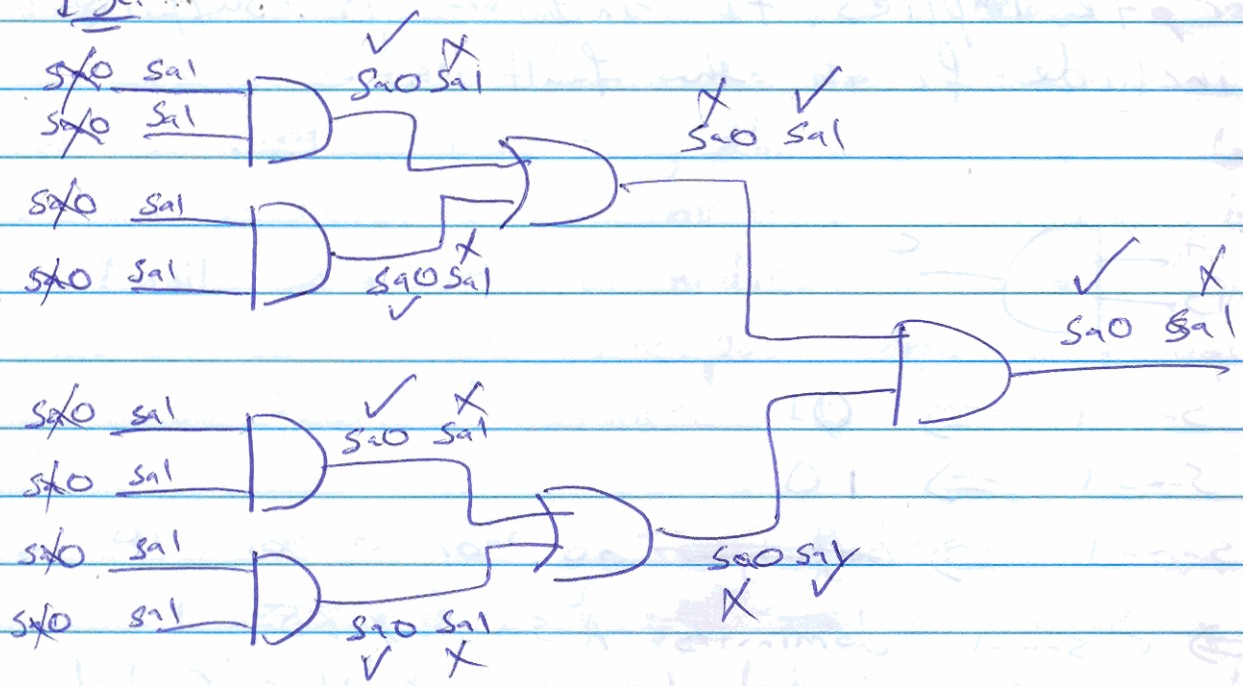
nor A s-a-0, B s-a-0 → C s-a-1

Ex. AND gate



3 test vectors are enough 1) A s-a-0
2) A s-a-1
3) B s-a-1

Ex:



1) ~~no~~ exhaustive testing $\Rightarrow 2^8 = 256$ test vector

2) non-exhaustive without collapsing = 30 ~~test vector~~ type of faults

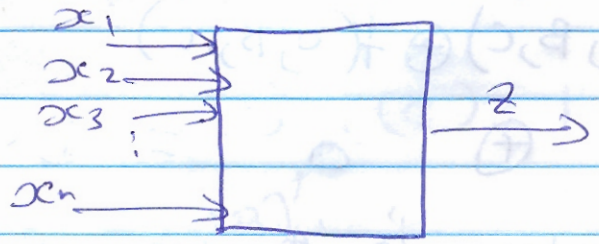
3) with collapsing $\Rightarrow$ 15 types of faults.

⊛ Boolean Difference Method :-

- in complicated circuits, it is not easy to find a test pattern that tests a specific node. For such circuits, it is useful to have a more formal way to decide when the output of a circuit is sensitive to the value at one of its nodes.

- Boolean difference is the digital equivalent of the derivative in math. It tells us whether the output of a function is sensitive to one of its inputs.

- In general, if we have a Boolean function Z of n inputs x



$$Z(x) = f(x_1, x_2, x_3, \cdots, x_n)$$

then the Boolean difference of Z with respect to x_i is

$$\frac{dz}{dx_i} = f(x_i = 1) \oplus f(x_i = 0)$$

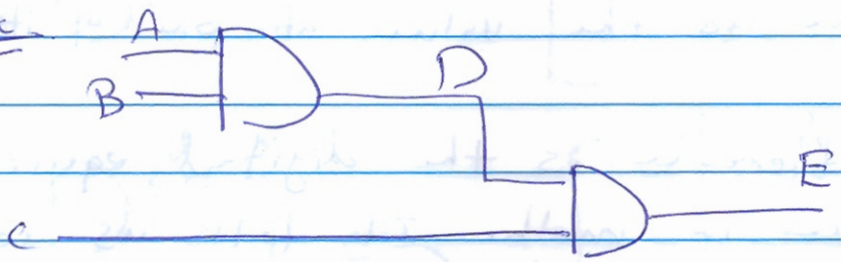
and the test vectors that test if x_i is s-a-0 are vectors which satisfy $(x_i) \cdot [f(x_i = 1) \oplus f(x_i = 0)] = 1$

and the test vectors that test if x_i is s-a-1

are vectors which satisfy that

$$(x_i)' \cdot [f(x_i=1) \oplus f(x_i=0)] = 1$$

Exc



to test stucks at node A

1) find when E is sensitive to A

$$\Rightarrow \frac{dE}{dA} \quad E = f(A,B,C) = A \cdot B \cdot C$$

$$\Rightarrow \frac{dE}{dA} = f(1,B,C) \oplus f(0,B,C)$$

$$= BC \oplus 0 = BC$$

$\Rightarrow$ E is sensitive to A when $BC=1$

$\Rightarrow$ B = 1 C = 1

to test A sa0 $\Rightarrow$

$ABC = 1$ $\Rightarrow$ ABC = 111

to test A sa1 $\Rightarrow$

$\bar{A}BC = 1$ $\Rightarrow$ $\bar{A}BC$ = 011

To test if D is stuck

$\Rightarrow E = f(D, C) = DC$

$\Rightarrow \frac{dE}{dD} = f(1, c) \oplus f(0, c)$

$= c \oplus 0 = c$

$\Rightarrow$ E is sensitive to D when $\underline{c = 1}$

$D = AB$

to find vectors ~~token~~ to test D s-a-0

$\Rightarrow (D) . (C) = 1$

$\Rightarrow (AB) . (C) = 1 \quad \Rightarrow$ **A = B = C = 1**

to find vectors to test D s-a-1

$\Rightarrow (D)' . (C) = 1$

$(AB)' . c = 1$

$(A' + B') c = A'c + B'c = 1$

A	B	C	result
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

$\Rightarrow$ vector

ABC = $\underline{001}$ or $\underline{011}$ or $\underline{101}$