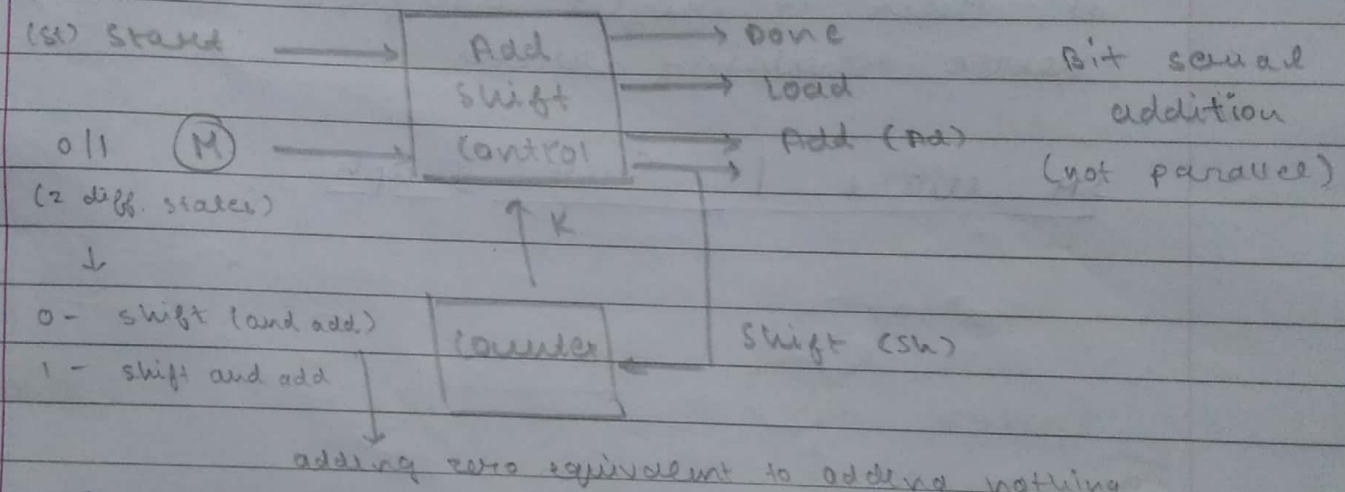
PHASE 3

→ To derive SM chart (or equivalent state graph), we need:

① Draw block diagram of system

② Define input and output.

→ 4 bit Multiplier — 8 bit output and a 9th cout bit (actually absent in final result)



Eg Accumulator — all bits are zero

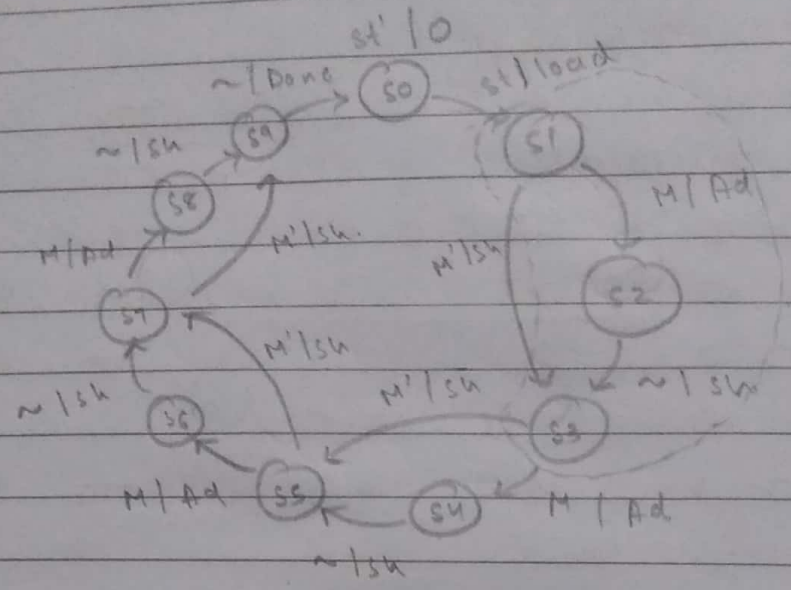
Consider
```
  1101      → Multiplicand
x 1011      → Multiplier
--------
10001111
```

```
  1101
x 1011
------
  1101
 1101
1101
------
1001111
0000
------
 100111
 1101
------
10001111
```

If multiplier bit is 1, multiplicand is added followed by a right shift of the sum (accumulator)

Scanned by CamScanner

→ A crude state diagram ....



→ This block repeats so use a counter and reduce the no. of states

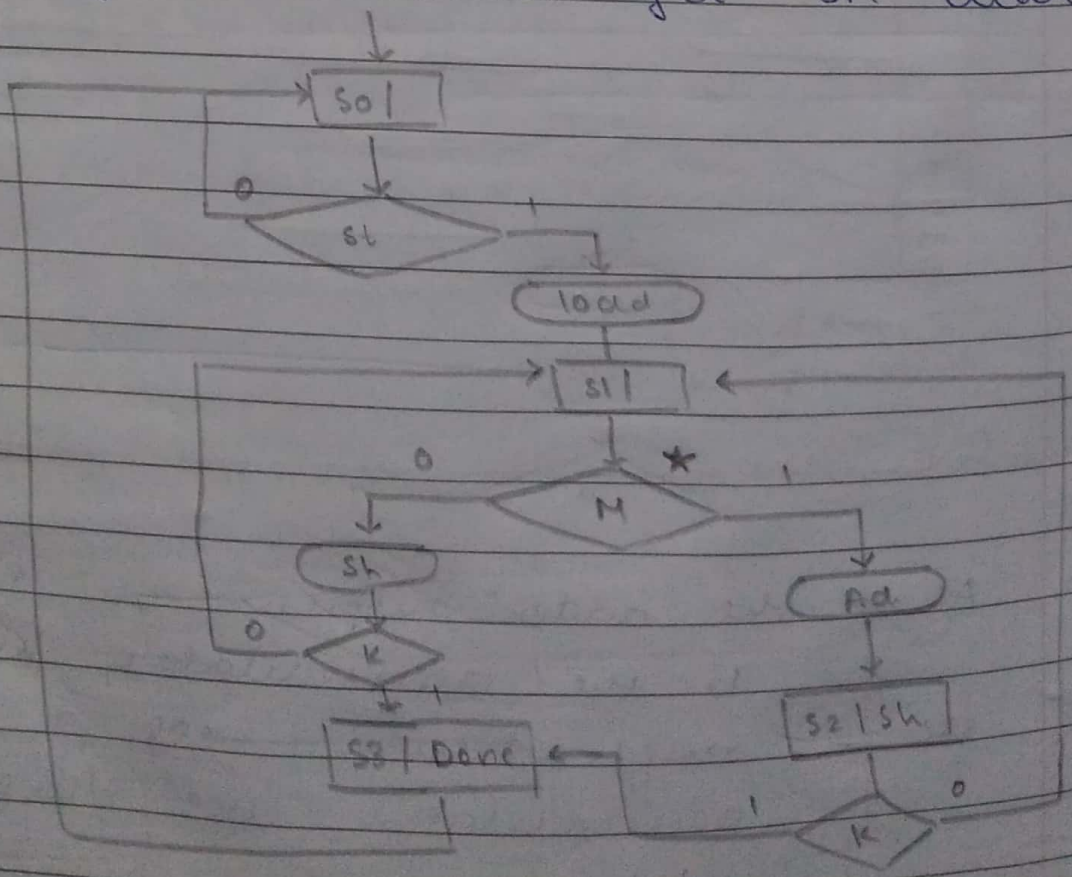
S1 to S3 count 1 → k=0

S3 to S5 count 2 → k=0

S5 to S7 count 3 → k=0

S7 to S4 count 4 → k=1

So now reduced to state graph [B] seen before. Now we get SM chart.



Scanned by CamScanner

→ Flow of making a SM chart ...

Ideation stage

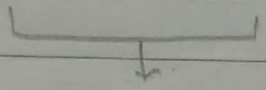
↓

Design specification

↓

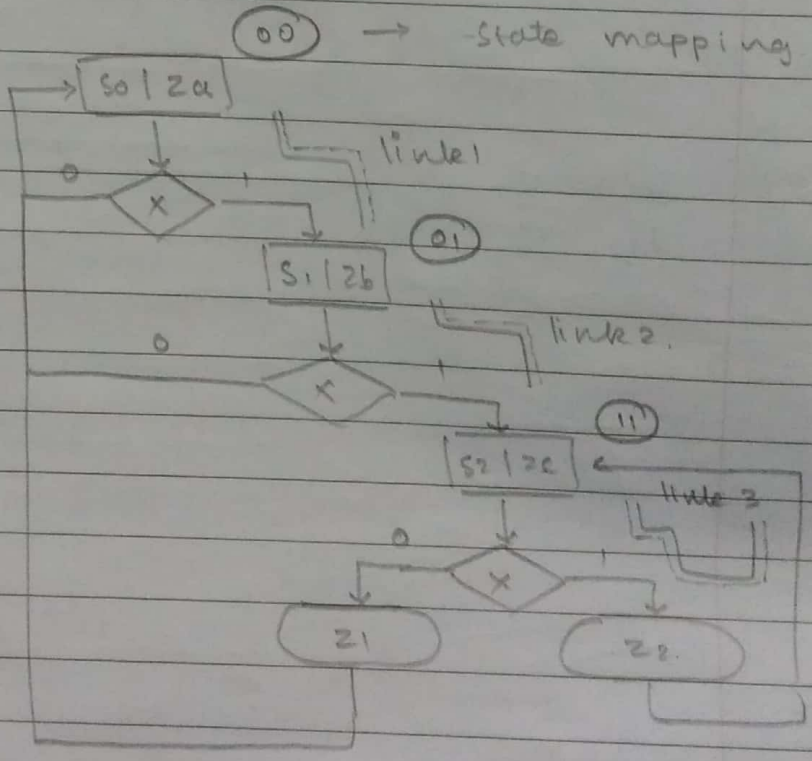
Decide I/O, control, timing strategy

State graph ⟷ SM chart



HDL program

→ Example :



All links are independent of each other

Za = 1 only in state 00 (A'B')

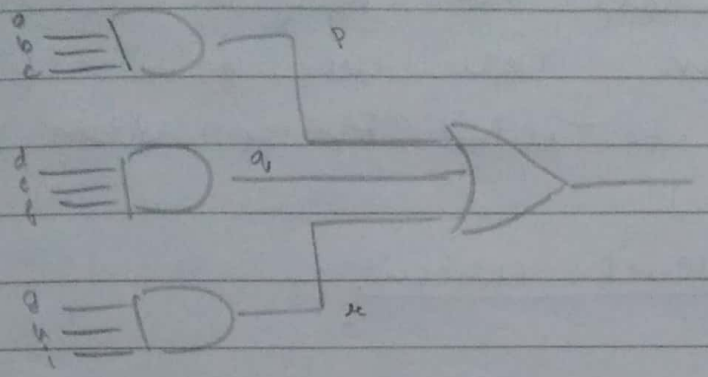
Zb = 1 only in state 01 (A'B)

Zc = 1 only in state 11 (AB)

Z1 — ABX'

Z2 — ABX } from link 3

Scanned by CamScanner



Make stuck at 0 and stuck at 1 test
(s-a-0 test
s-a-1 test)

Two level AND OR : Assume inputs are available but input to OR gate is not accessible.

$d=0$ → Path B gives zero
$g=0$ → Path C gives zero
$b=1$, $c=1$] output of OR must be 1, when we give $a=1$. But if we get output = 0, it means a is stuck at zero
$a=1$
$e,f,h,i = X$

$a=1$ $b=1$ $c=1$ $d=0$ $g=0$ other = X — this input vector helps identify a stuck at zero

Testing ....

a	b	c	d	e	f	g	h	i
1	1	1	0	x	x	0	x	x
0	x	x	1	1	1	0	x	x
0	x	x	0	x	x	1	1	1
0	1	1	0	1	1	0	1	1
1	0	1	1	0	1	1	0	1
1	1	0	1	1	0	1	1	0

Faults tested

stuck at zero tests

a0, b0, c0, p0
d0, e0, f0, g0
g0, h0, i0, r0

a1, d1, g1, p1, q1, r1
b1, e1, h1, p1, q1, r1
c1, f1, i1, p1, q1, r1

Stuck at one tests

Scanned by CamScanner

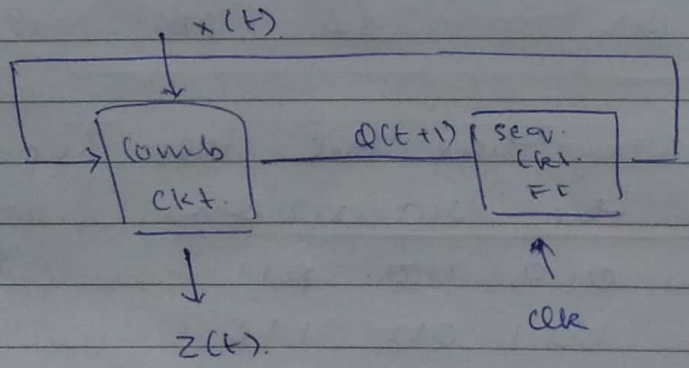
Testing sequential logic

→ If we can observe all the input and output sequences, and not the states, then a very large no. of flip flops are required.

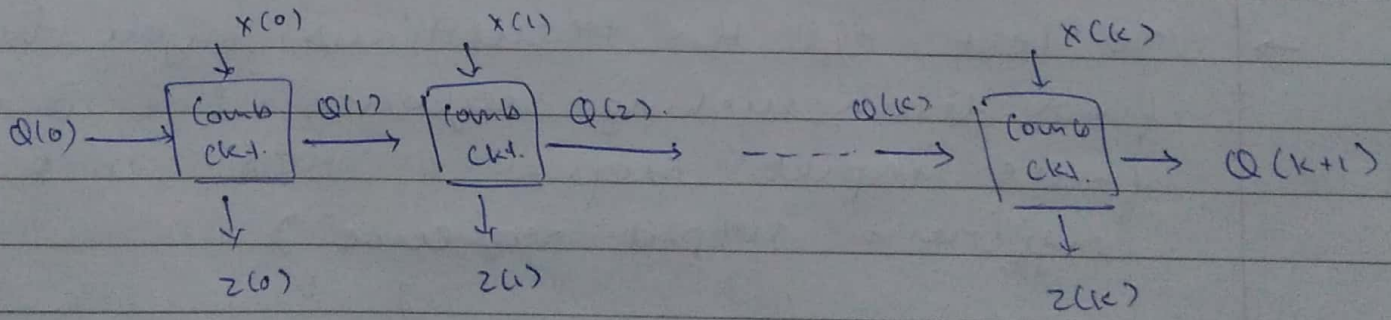
→ If a logic doesn't work, we resort to brute force method.

→ Reset → initial state → apply a test seq. → observe output sequence → if correct, repeat.

→ Convert to an iterative circuit :



Mealy machine

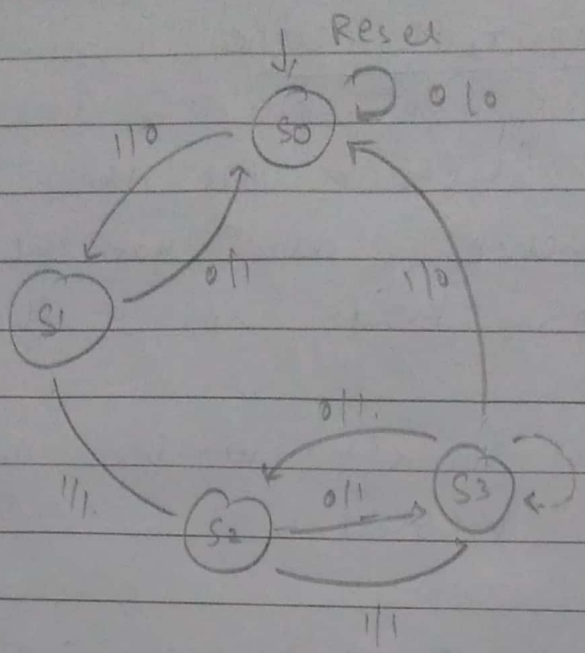


→ Here, z, x, Q can be single variables or vectors.

→ xs are inputs.

→ k+1 is the length of iterative circuits.

Scanned by CamScanner



			Next state		Output	
Q_1	Q_2	State	X=0	X=1	X=0	X=1
0	0	S0	S0	S1	0	0
1	0	S1	S0	S2	1	1
0	1	S2	S3	S3	1	1
1	1	S3.	S2	S0.	1	0

→ It is necessary that the test input goes through all the transitions (every line in the graph)

→ Eg. x = 010 110 011

z = 001 011 110

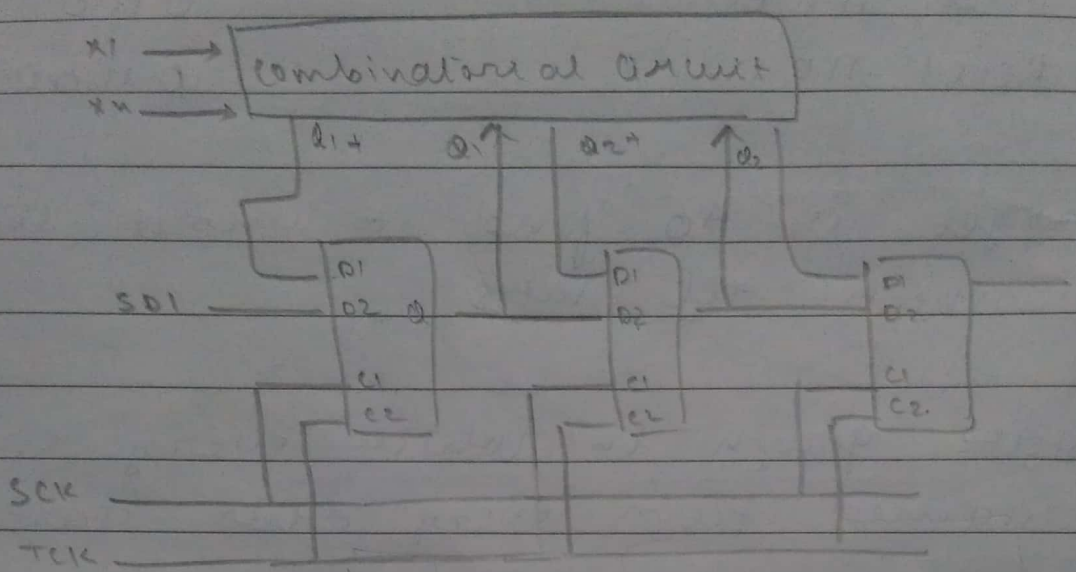
→ To test all the transitions, you have to find distinguishing sequence (ie one finite input sequence that cause different output sequence)

For above state machine, the distinguishing sequences:

11 (S0, S1, S2, S3) groups

S0, S3 S1, S2.

Scanned by CamScanner



C1 is pulsed - D1 is stored
C2 is pulsed - D2 is stored

NS: Q_1^+, Q_2^+ ---- generated by combinational logic.

when C1 is pulsed, Q_1, Q_2 --- feed back into comb. logic.

Circuit has 2 modes - operational (system) or testing.

circuit is not in test when { SCK = C1
$X_1 .. X_N$ applie
$Z_1 .. Z_N$ generate

Scanned by CamScanner