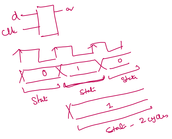
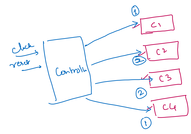
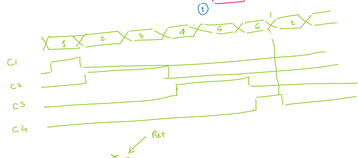
1. FINITE STATE MACHINES

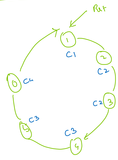
Condition → binary value → FFs (output)
Persist for a certain amount of time



A valid State should persist for atleast 1 clock cycle







⇒ State diagram
Captures the functionality of the controller

At a given point of time only one state can be there

Coding the State machine:

→ come up with a state diagram

→ Identify the number of states

→ Find the minimum number of FFs needed to generate the 'n' states

N=6

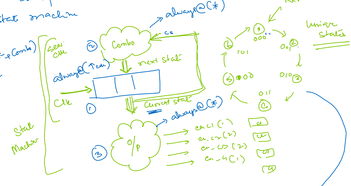
$2^n \geq N$

$n<3$ ✗
$n \geq 3$ ✓

$2^n \geq 6$
$n = 3$ optimal

→ Now come up with the standard template for a state machine

State machine is nothing but a Sequential circuit (FF+Comb)



→ FSM is generally coded with 3 always blocks

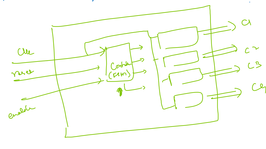
reg [2:0] cs, ns;

① always@(↑ clk)
if (reset)
cs ⇐ initial_state;
else cs ⇐ next_state;

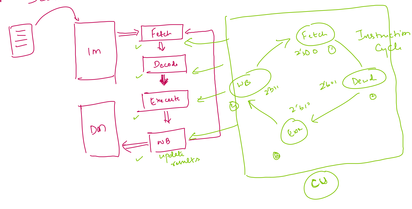
② always@(*)
case (cs)
3'b000 : Next state = 3'b001; next state = 3'b010;
3'b001 : ...
endcase

③ always@(*)
case (cs)
3'b000 : en_c1 = 1
3'b001 : en_c2 = 1
3'b010 : en_c3 = 1
endcase

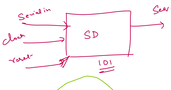
→ How does the top level module look like?

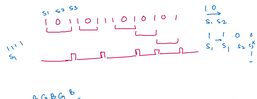


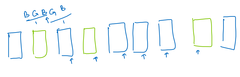
2. Another Scenario



3. Sequence detector







No. of states = 4
No. of FFs = 2

2 part state machine