

# IRIS LABS ASSIGNMENT ANSWERS

-HARSHAK SACHDEVA

21133230

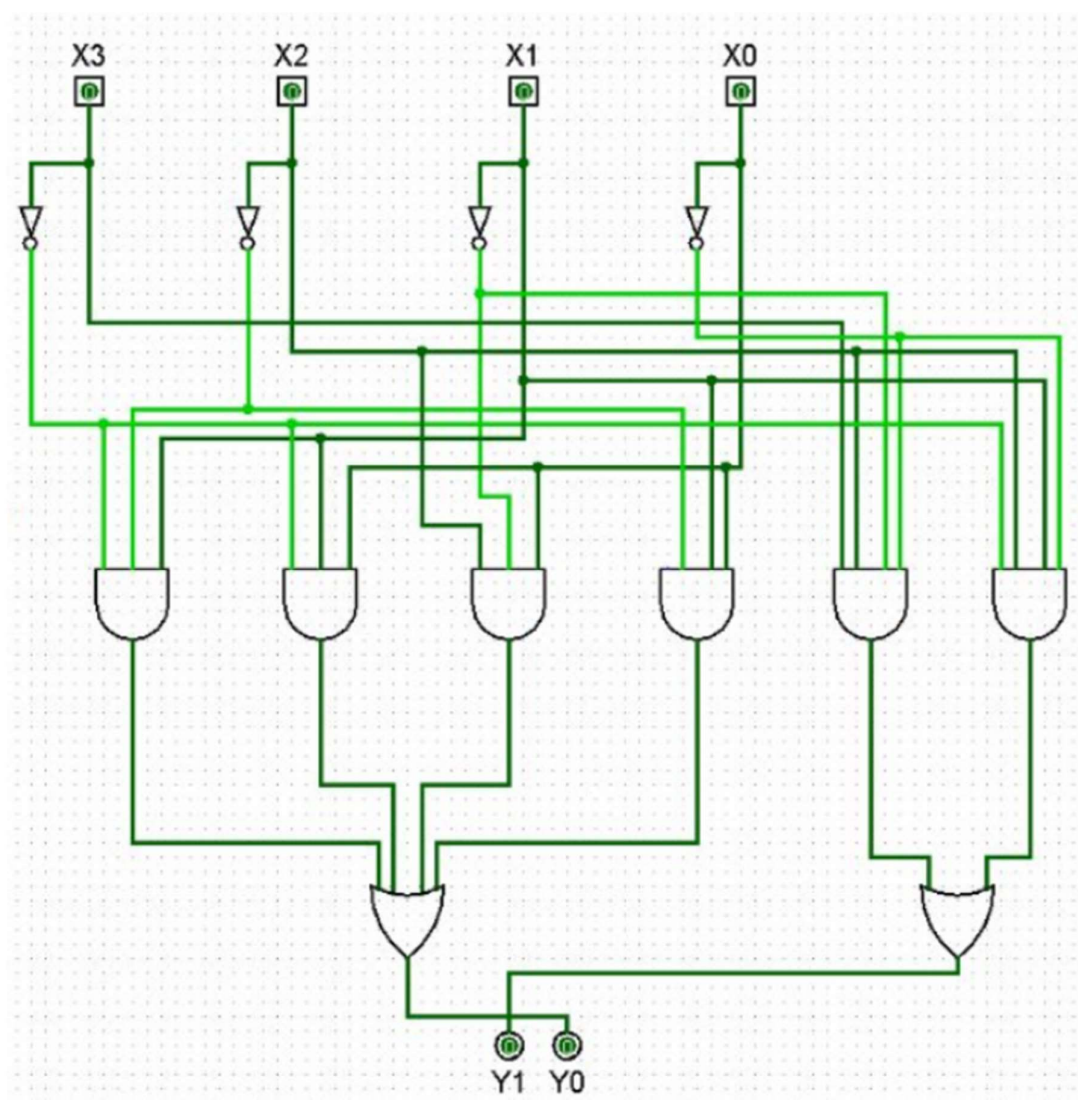
Ans 1: Truth Table :

A	B	C	D	X	Y
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	0	1
1	1	1	0	0	0
1	1	1	1	0	0

Solving using K Map method yields the following expressions for X and Y :

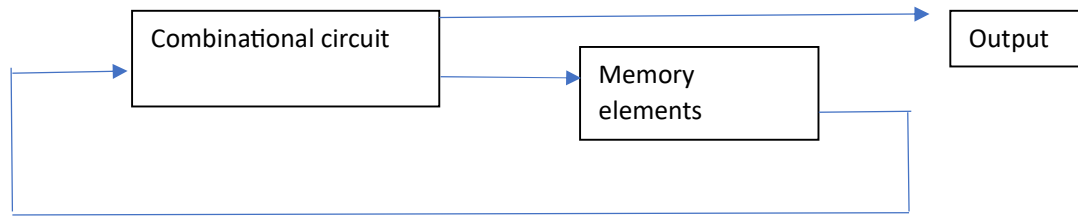
$$X = A'BCD' + ABC'D'$$

$$Y = A'B'C + A'CD + BC'D + B'CD$$



Ans2:

- Sequential circuits consists of a combinational circuit to which memory elements are connected to form a feedback path.
- The memory elements are devices capable of storing binary information within them.
- The binary information stored in the memory elements at any given time defines the “state of the sequential circuit”.
- The sequential circuit receives binary information from external inputs , these inputs together with present state of memory elements, determine the binary value at output terminals.
- There are mainly 2 types of sequential circuits.
- Synchronous sequential circuit: It is a system whose behaviour can be defined from knowledge of its signals at discrete instants of time.
- Behaviour of asynchronous sequential circuits depends upon the order in which its input signals change & can be affected at any instant of time.
- The memory elements commonly used in asynchronous sequential circuits are time delay devices. The time delay is due to finite time taken for the signal to propagate through a device.



The memory elements used in clocked sequential circuits are called “flip-flops”. These circuits are binary cells capable of storing 1 bit of information.

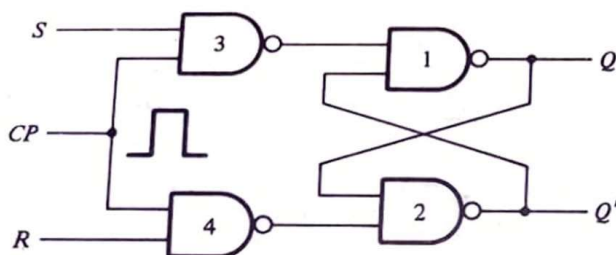
- In combination circuits, output is only dependent on present input, whereas, in sequential circuit, output depends on present input as well as previous output.
- Eg: Adder is a combination circuit, whereas counter is a sequential circuit as counter adds 1 to previous output.

Ans 3: In digital electronics, flip flop is a circuit that can be used to store binary data. The stored data can be changed by applying varying input. Flip flops and latches are fundamental building blocks of digital electronics systems used in computers, communications etc.

Flip flops are basically storage element in sequential logic. They differ from latches by being edge triggered in contrast to latches which are level-triggered.

Flip flops are categorised into: SRFF, JKFF, T flip flop, D flip flop.

SRFF:



(a) Logic diagram

$Q$	$S$	$R$	$Q(t + 1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Indeterminate

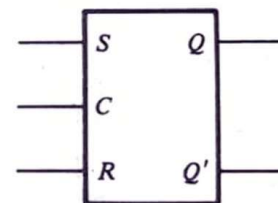
(b) Characteristic table

		$S$			
		$SR$			
		00	01	11	10
$Q$	0			X	1
$Q$	1	1		X	1
		$R$			

$$Q(t + 1) = S + R'Q$$

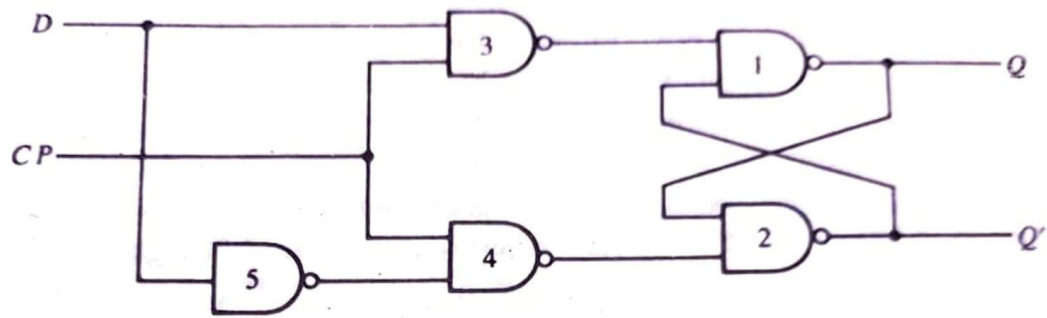
$$SR = 0$$

(c) Characteristic equation



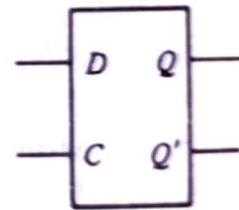
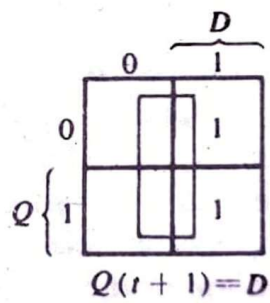
(d) Graphic symbol

D FF:



(a) Logic diagram

$Q$	$D$	$Q(t+1)$
0	0	0
0	1	1
1	0	0
1	1	1

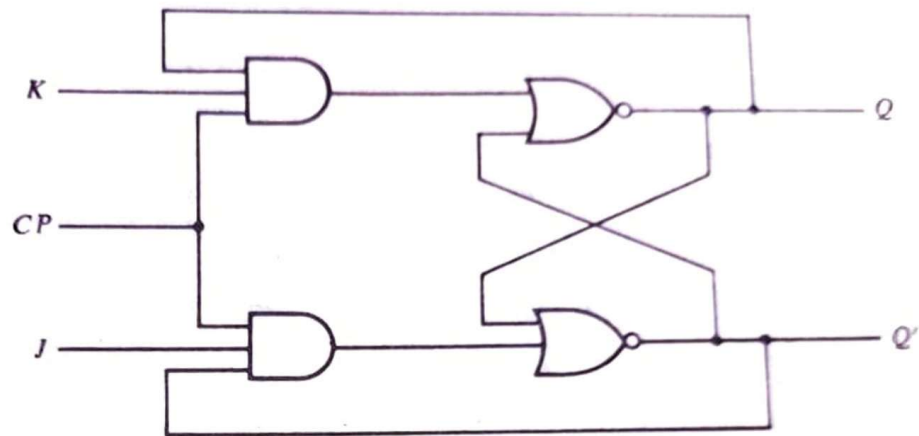


(b) Characteristic table

(c) Characteristic equation

(d) Graphic symbol

JKFF:



(a) Logic diagram

$Q$	$J$	$K$	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

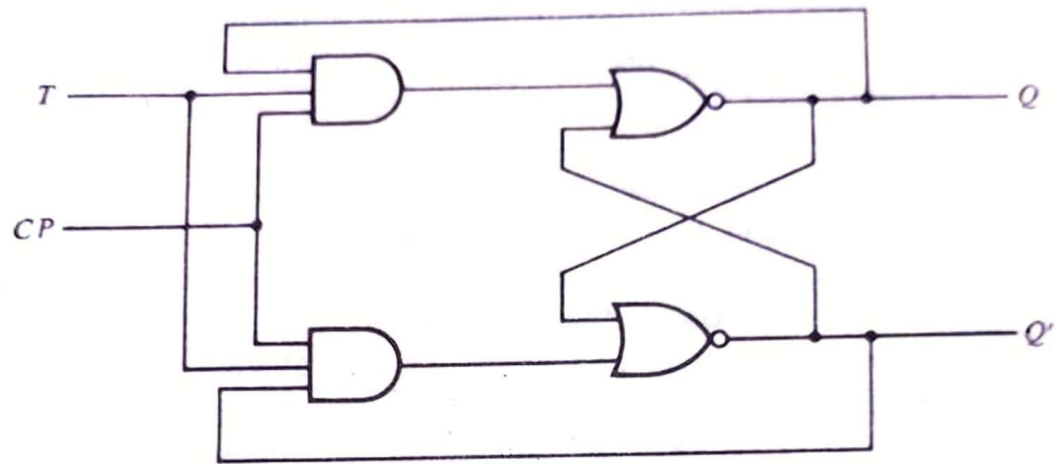
(b) Characteristic table

		$J$			
		$JK$		11	10
$Q$	0			1	1
	1	1			1
		$K$			

$$Q(t+1) = JQ' + K'Q$$

(c) Characteristic equation

T FF(toggle FF):



(a) Logic diagram

$Q$	$T$	$Q(t+1)$
0	0	0
0	1	1
1	0	1
1	1	0

(b) Characteristic table

			$T$
		0	1
0			1
1	1		

$$Q(t+1) = TQ' + T'Q$$

(c) Characteristic equation



## OPTIONAL QUESTION 3 Ans:

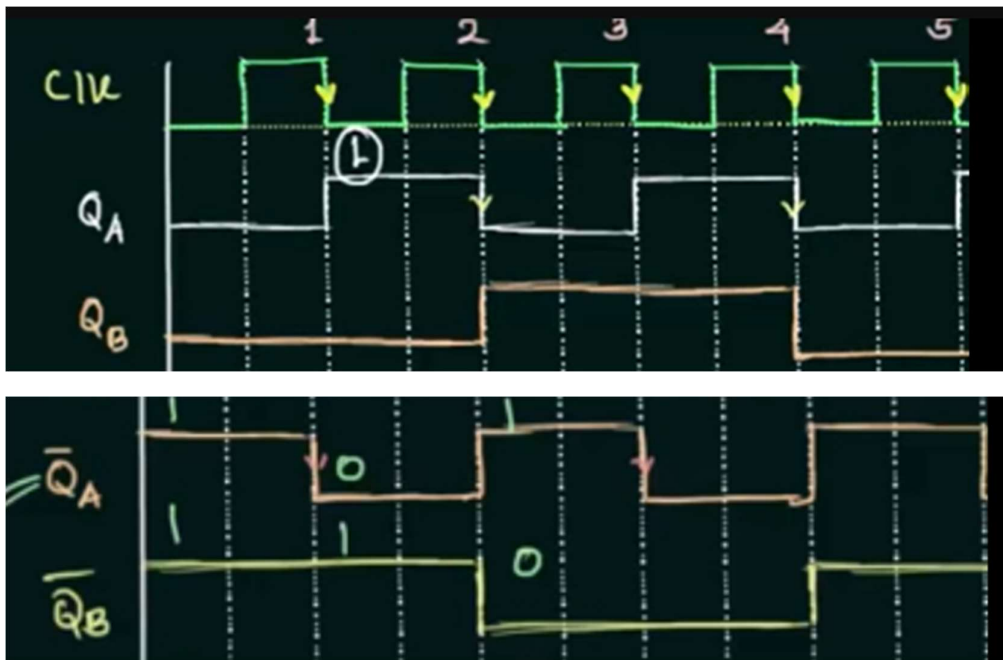
The top screenshot shows the Vivado 2022.2 IDE with the Verilog code for the DFF circuit and its testbench. The code is as follows:

```
22 module dff_final(clk,d,rst,q);
23   input d,clk,rst;
24   output reg q;
25   always @(negedge clk or negedge rst)
26   begin
27     if (rst == 1'b0)
28       q <= 1'b0;
29     else
30       q <= d;
31     end
32   endmodule
33 module testbench();
34   reg d,clk,rst;
35   // ... (testbench code) ...
36 endmodule
```

The bottom screenshot shows the simulation results in a waveform viewer. The waveform displays the signals d, clk, rst, and q over time. The signals d, clk, and rst are shown as blue lines, and q is shown as a red line. The time scale is 100 ns. The simulation time is 1 us.

Name	Value
d	Z
clk	Z
rst	Z
q	X

Ans 4:



A 2 bit up-down counter usually has 2 main inputs, the clock & the counter mode. Depending on the customization of the counter, the mode bit can use different bit guides. In this case, if the mode bit is high, then the counter counts up from zero(or last value in memory)to 3 & starts again from 0. If the mode bit is zero, then the counter counts down from 3 to 0 & starts again from 1. The values are 0 & 3 because the counter is for 2 bits & the value is the output.

Ans. 5: Procedural assignments in verilog are broadly classified into 2 types:

- (a) Blocking
- (b) Non-blocking

The left hand side of a procedural assignment can be either:

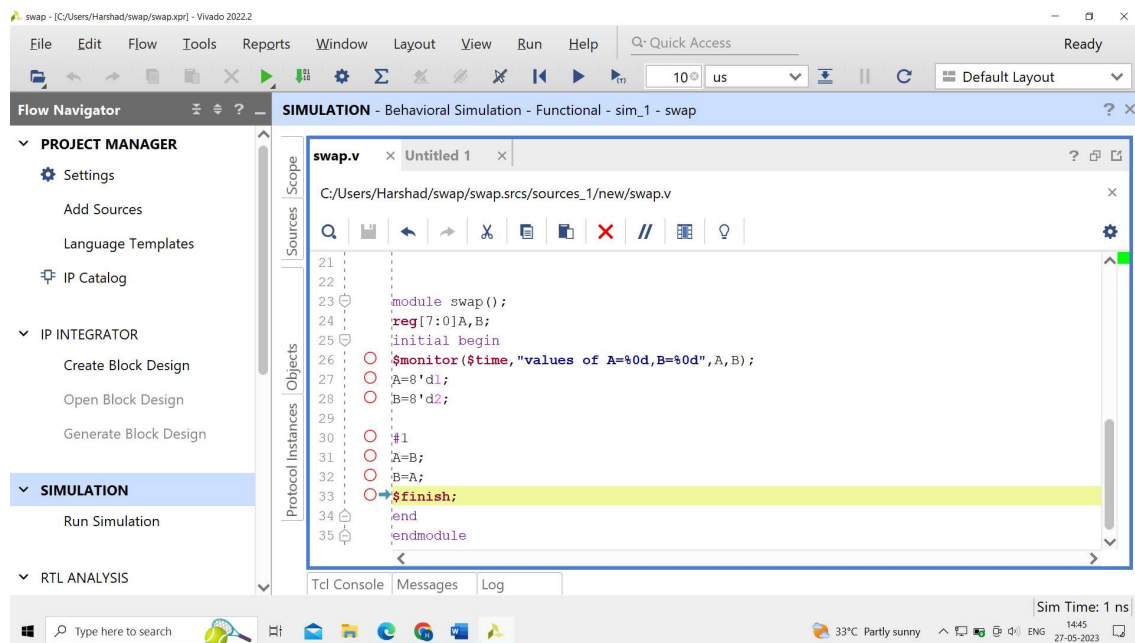
- (a) A register type variable(reg, integer, real or time)
- (b) A bit select of these variables(e.g.: sum[15])
- (c) A part select of these variables(e.g.: IR[31:26])

Procedural assignment statements can only appear within procedural blocks(“initial” or “always”) and these statements can be used to change the value of variable. The value that we assign to a variable remains unchanged until you assign some other value to that variable again.

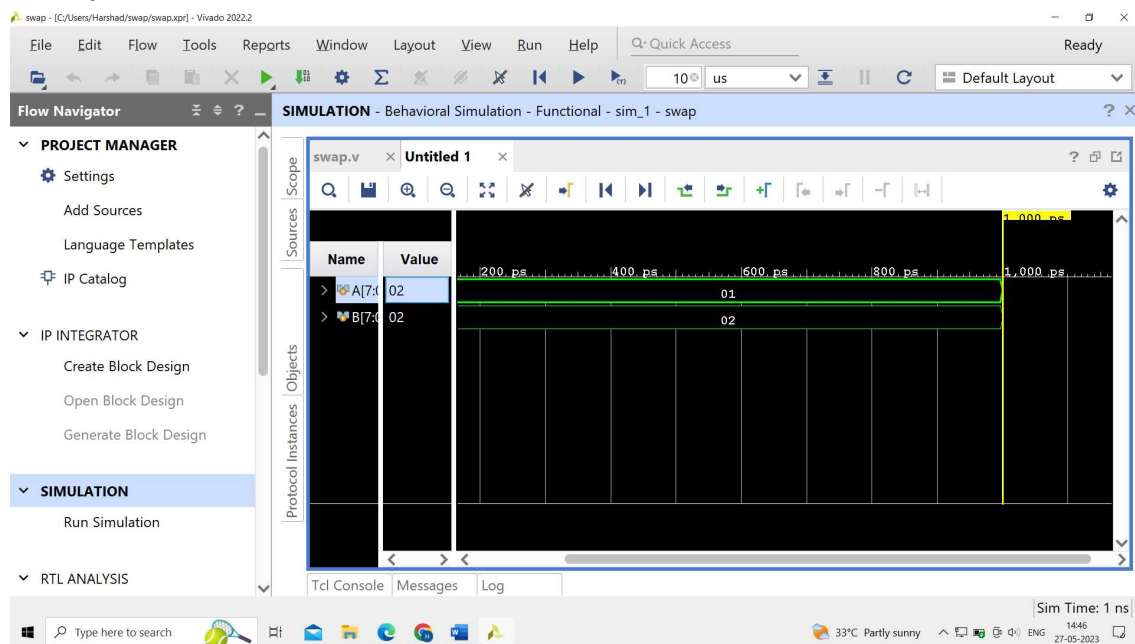
Blocking assignments are recommended style for modelling combinational logics whereas, non-blocking assignments are recommended for modelling sequential logic.

A good example can be of swapping values fo two variables a & b.

## Code for swapping 2 numbers:



## Output:



As we can see, after #1(1 nanosecond), both values A and B are assigned value of 2.(swapping not done).

Using non blocking assignment statements:

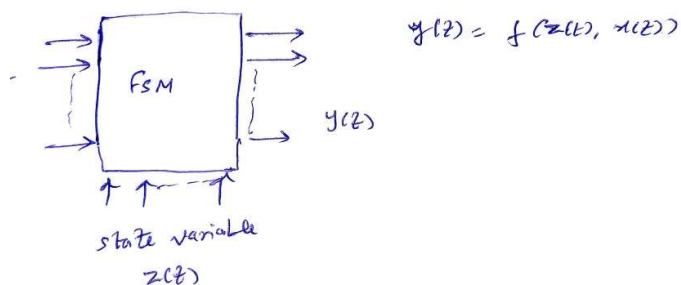
```

always @(posedge clk)
    a <= b;
always @(posedge clk)
    b <= a;

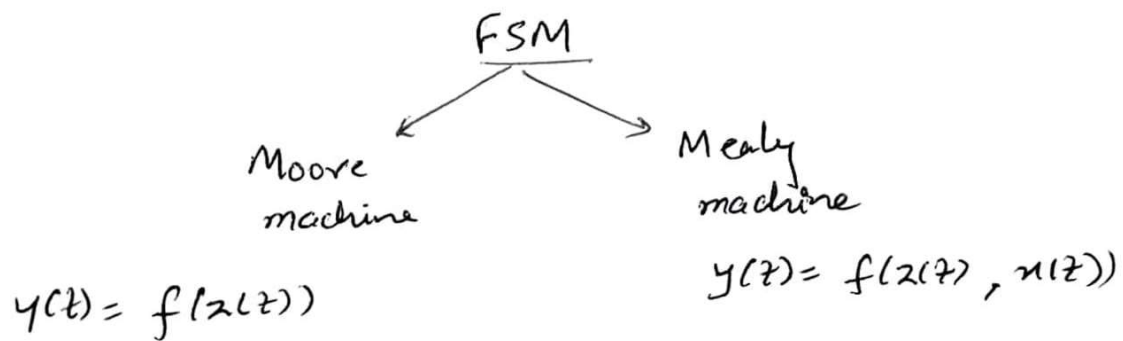
```

- Here, the values are correctly swapped.
- All RHS variables are read first, & assigned to LHS variables at the positive clock edge.

Ans. 6: The output of a FSM depends on state variable and the input. FSM is a digital system in which there is only finite number of states.



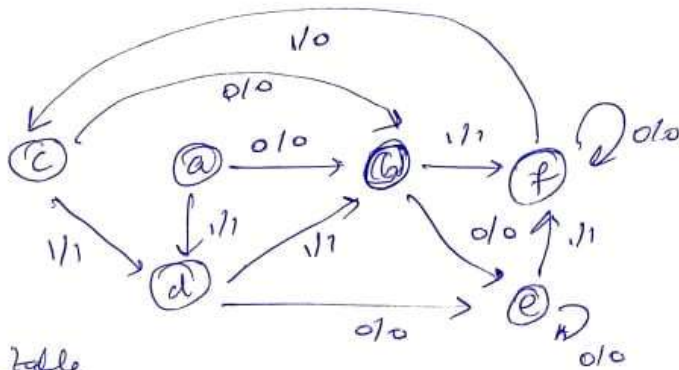
FSM is categorised into 2 types: Moore machine, Mealy machine.



State diagram is a pictorial representation of present state, input, next state & output of a sequential circuit.

As design process must consider the problem of minimizing the cost of final circuit. The 2 most obvious cost reductions are reductions in number of FF and the number of gates.

Since  $m$  FFs produce  $2^m$  states, a reduction in the number of states may or may not result in reduction in number of FFs. Therefore, we must proceed to reduce the redundant states(if any). So for that, we need state tables.



State table

Present state	Next state, o/p	
	x=0	x=1
a	b, 0	d, 1
b	e, 0	f, 1
c	b, 0	d, 1
d	e, 0	b, 1
e	e, 0	f, 1
f	f, 0	e, 0

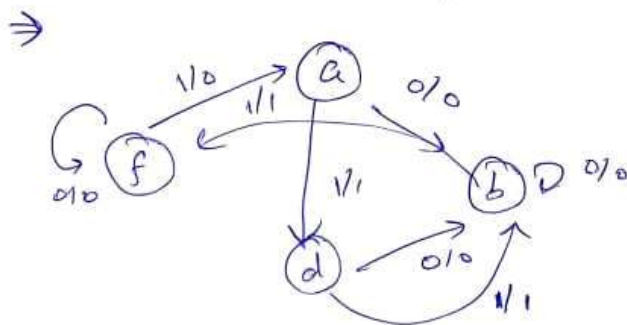
no. of FF =  $2^3 = 8$   
(apparent)

a & c are redundant  
 $\therefore$  c can be replaced by a.  
 similarly, b & e are redundant.

$\Rightarrow$

PS	NS, o/p	
	x=0	x=1
a	b, 0	d, 1
b	b, 0	f, 1
d	b, 0	b, 1
f	f, 0	a, 0

$\Rightarrow$  no. of FF required =  $2^2 = 4$



Mealy machines have fewer states & are faster because the state is dependent on the input. Thus, the state can change asynchronously & for hardware implementation too, mealy machine requires less hardware in their circuits, thus in my opinion, mealy machines are generally preferred over moore machine.

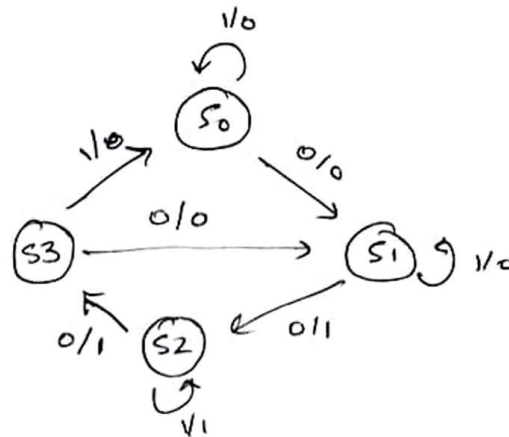


Q.6 (Optional) we need to detect more than 1 0's in any sequence of 3 samples

⇒

	$p_2$	$p_1$	$p_0$	y (output)
required states ⇒	0	0	0	1
	0	0	1	1
	0	1	0	1
	0	1	1	
	1	0	0	1
	1	0	1	
	1	1	0	
	1	1	1	

State Diagram:-



Let  $S_0$  be the state with no zeros,  $S_1$  → state with one 0.

$S_2$  → state with two 0,  $S_3$  → state with three zeros.

∴ If input of  $S_0$  is 1,  $op = 0$  & same for  $S_1$

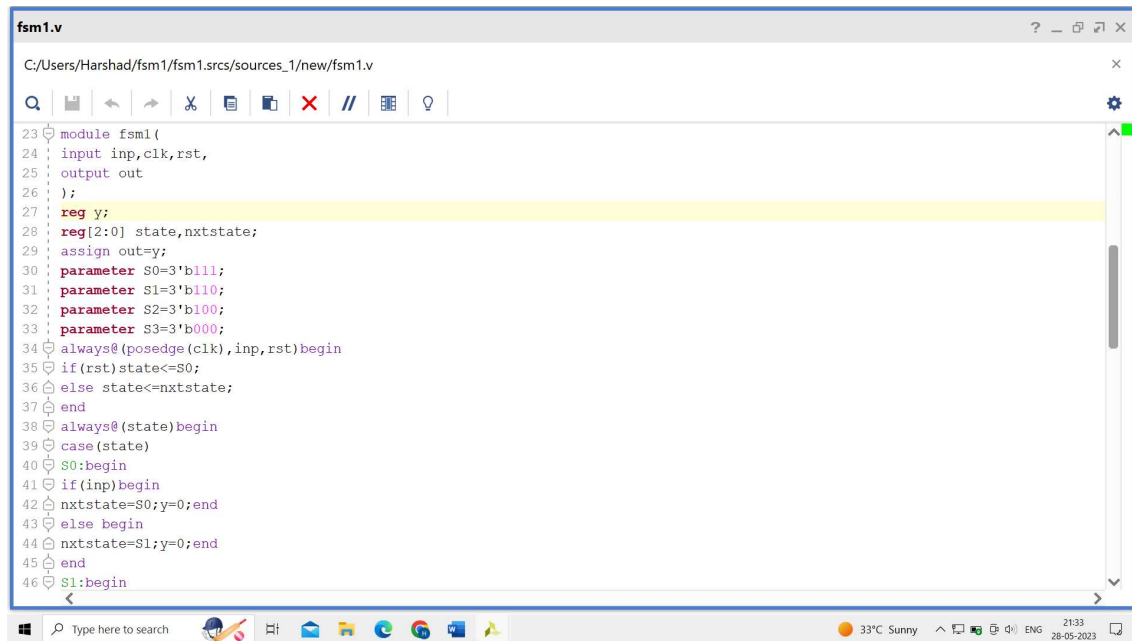
But if input to  $S_2$  is 1, then  $S_2$  shall have 3 zeros

(required state), similarly, if  $S_1$  has input = 0,

output shall have two zeros. (required state), ∴  $op = 1$ .

(Verilog code is attached in the github repo)

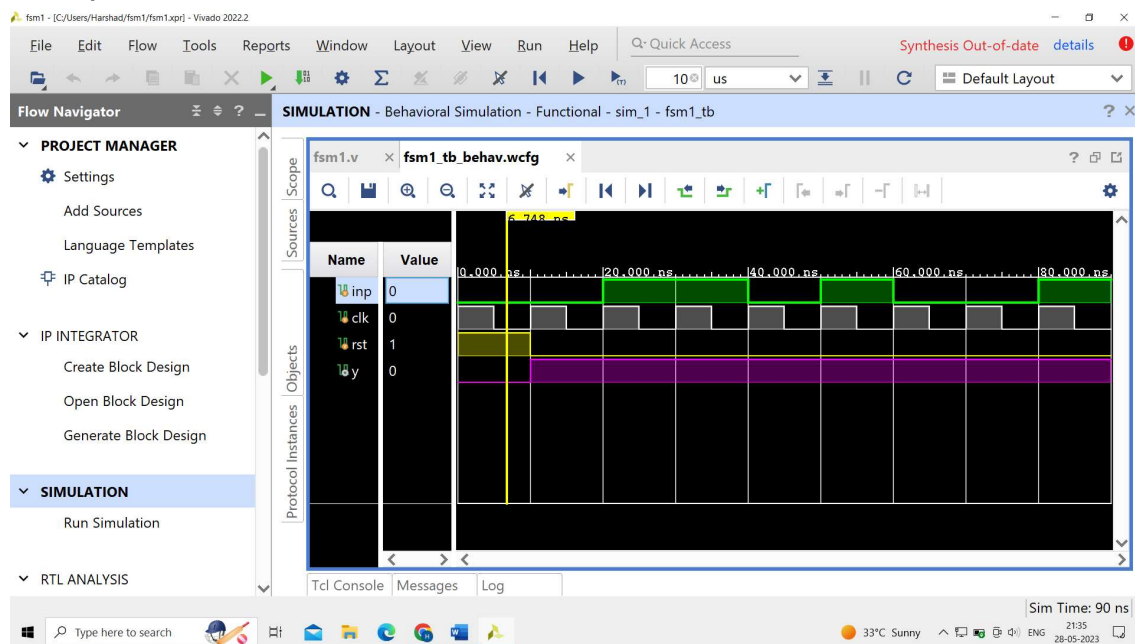
## CODE SCREENSHOT:



```
fsm1.v
C:/Users/Harshad/fsm1/fsm1.srcs/sources_1/new/fsm1.v

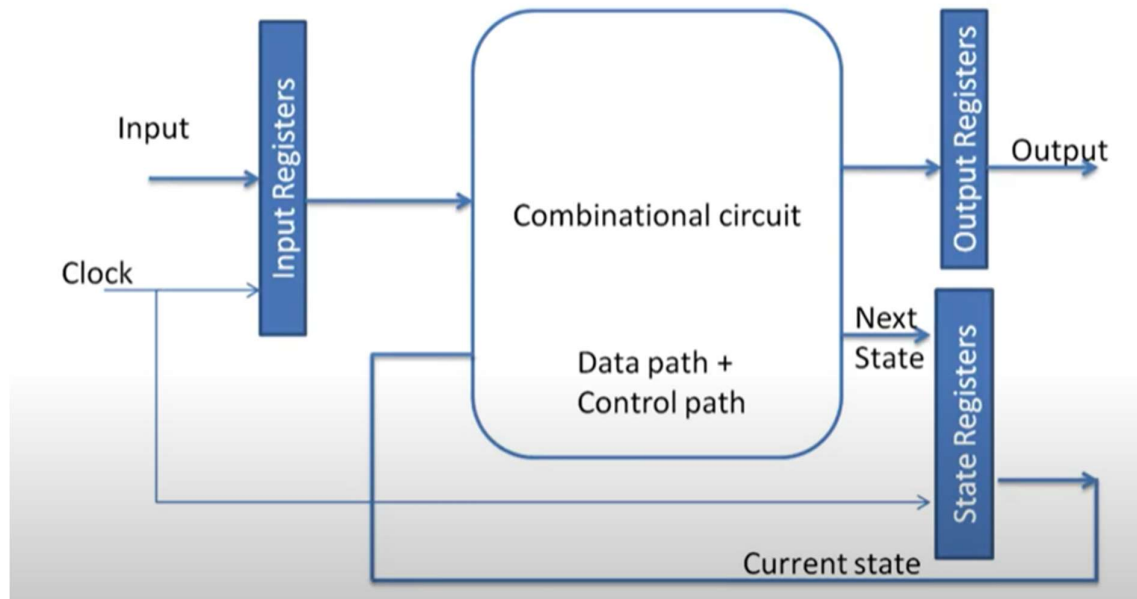
23 module fsm1(
24     input inp,clk,rst,
25     output out
26 );
27 reg y;
28 reg[2:0] state,nxtstate;
29 assign out=y;
30 parameter S0=3'b111;
31 parameter S1=3'b110;
32 parameter S2=3'b100;
33 parameter S3=3'b000;
34 always@(posedge(clk),inp,rst)begin
35     if(rst)state<=S0;
36     else state<=nxtstate;
37 end
38 always@(state)begin
39     case(state)
40     S0:begin
41         if(inp)begin
42             nxtstate=S0;y=0;end
43         else begin
44             nxtstate=S1;y=0;end
45         end
46     S1:begin
```

## Output:



Ans 7: RTL(Register-Transfer-Level) Design method:

- Input is read from register & output gets written to registers & registers are global clock driven.



• Implication of RTL specifications: Operations performed in each clock cycle is clear.

- How many clock cycles required?
- How many hardware resources required?
- What is worst case latency?
- Separation of data path & control path

Clock period discussion:

1. Find the worst case critical path & decide the clock period.
  - (a) One operation may slow entire circuit
  - (b) Overall execution time will be large
2. Decide the clock period & design the data path accordingly.

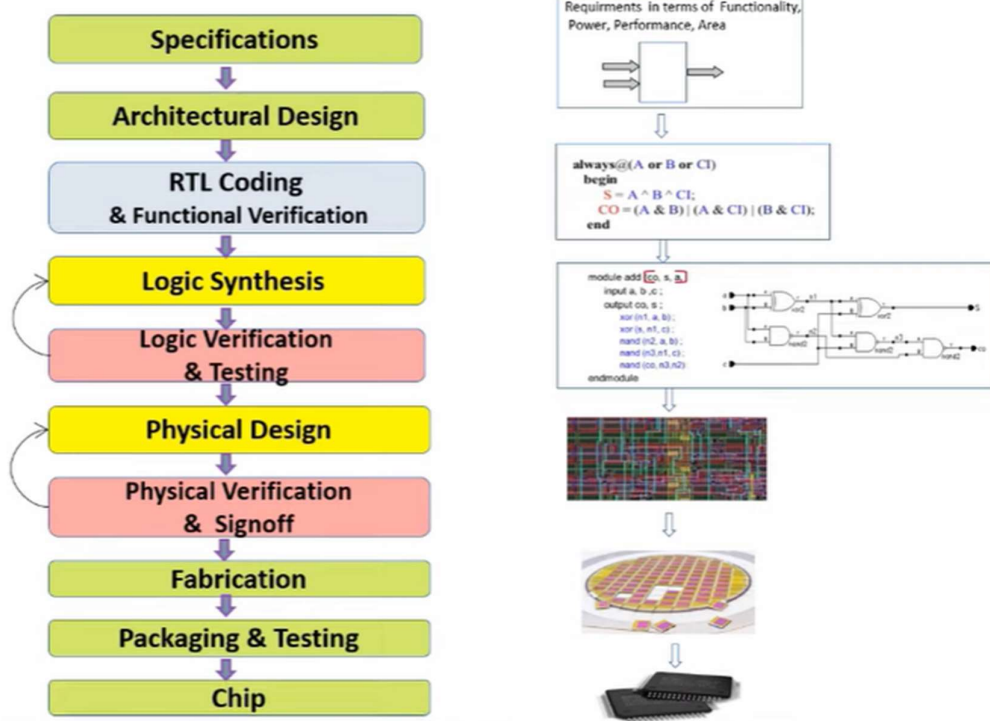
- (a) Put a constraint on design.
- (b) Slow operations are converted into multi-cycle operations
- (c) Clock period is decided by fastest operation.

### RTL Modelling in Verilog

- Combinational logic
  - Use data flow statements
  - Eg: assign s = a+b;
- Use of always block for sequential statement
  - always@(posedge clock)begin
  - RA <= RB + RC
  - End
  - Blocking as well as non blocking statements are permissible.

Ans 8: Stages involved in VLSI design flow are:

## VLSI-IC Design Flow / ASIC Flow / RTL to GDS Flow



The design flow starts with a given set of specifications or requirements that the chip must possess that is in terms of functionality, means what logic/function it must perform, how much area it may consume, with what clock frequency it must operate etc.

Now as per the given specifications, architecture of the chip is designed in the form of block diagram that is, what different blocks it must contain like, ALU, memory unit etc. & its interconnections. The cost of the chip is also estimated at this stage & if everything is fine, it is proceeded to the RTL design.

Now RTL code is written using HDL(Hardware Description Language) eg. VHDL(Verilog). RTL is the

Register-transfer-level which includes the interconnections of the combinational elements like logic gates & the sequential elements like registers as per the functionality.

Now, next step is the functional verification i.e. to verify & check the RTL code whether it satisfies the functionality & the given specifications. If there are any errors, the code is modified & iterations are made until the specifications are satisfied.

The next step is logic synthesis. In this stage, the tool converts the RTL behavioural model code into the structural verilog code which is a gate level netlist. Inputs required for the logic synthesis are RTL code, library files & SDC constraints file.

The logic synthesis happens in 3 stages: translate, mapping & optimize. Translate means the tool converts the high level RTL ckt. Into its corresponding logic gates.

The tool does optimization in terms of power, area, timing. The Design for Test(DFT insertion) is also done by the DFT team.

Now, the gate level simulation i.e. logic verification & testing is done to check whether it performs the required logic & iterations are made until there are no errors.

In physical design, the gate level netlist is converted to the GDS format. GDS is Graphical Data Stream, it is nothing but a transistor level layout format i.e. the physical representation of a design which can be manufactured.

Now, the obtained GDS layout file is sent for verification & signoff where different checks are performed like design rule check, electrical rule check, timing analysis, power analysis etc.

The GDS file after verification is sent to the foundry. In foundry, the mask is generated for the design layout & chip is fabricated and packaged to protect from external damages.

Thank You

---