

3. The following Verilog code defines a Finite State Machine (FSM).

```

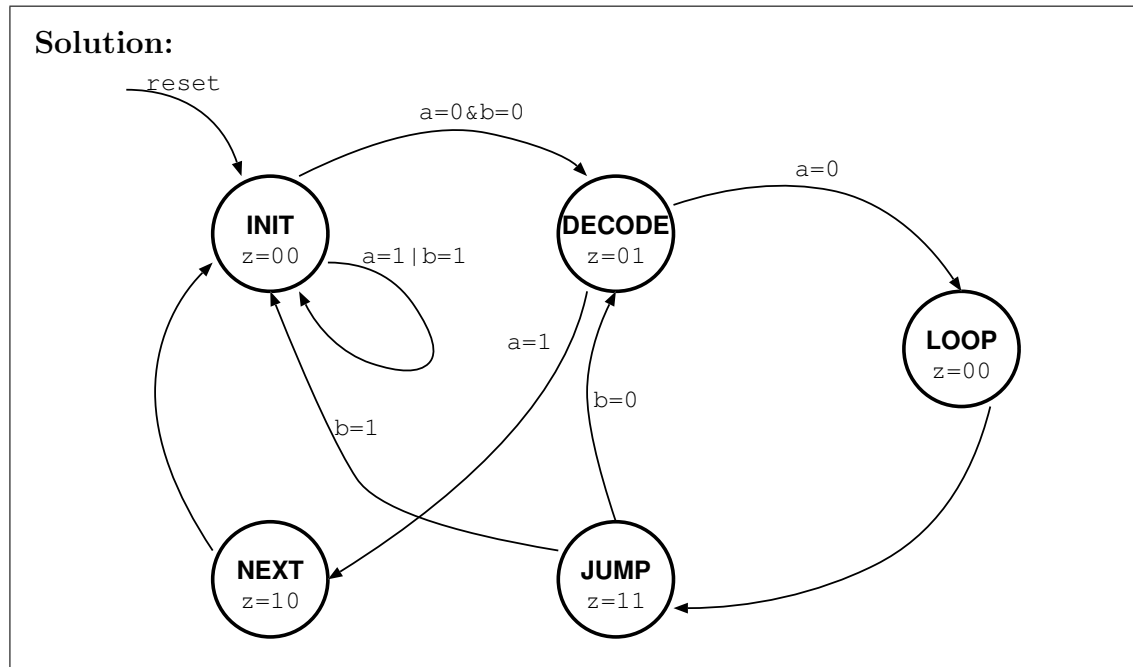
1  module fsm ( input a , input b , output [1:0] z,
2              input clk, input reset);
3
4  reg [2:0] state, nextstate;
5
6  parameter INIT    = 3'b000;
7  parameter DECODE  = 3'b001;
8  parameter LOOP    = 3'b100;
9  parameter JUMP    = 3'b111;
10 parameter NEXT    = 3'b010;
11 //      DEF1      = 3'b011;
12 //      DEF2      = 3'b101;
13 //      DEF3      = 3'b110;
14
15 // next state calculation
16 always @( * )
17     case (state)
18         INIT:    if ((a==1'b0) & (b==1'b0) ) nextstate = DECODE;
19                 else                                nextstate = INIT;
20         DECODE:  if (a) nextstate = NEXT;
21                 else  nextstate = LOOP;
22         LOOP:   nextstate = JUMP;
23         JUMP:   if (b) nextstate = INIT;
24                 else  nextstate = DECODE;
25         NEXT:   nextstate = INIT;
26         default: nextstate = INIT;
27     endcase
28
29 // state register
30 always @ (posedge clk, negedge reset)
31     if (reset == 1'b0) state <= INIT;
32     else                state <= nextstate;
33
34 // output logic
35 assign z = state[1:0];
36
37 endmodule

```

(a) (1 point) Is this a Moore or a Mealy FSM? Briefly explain.

Solution: Moore, outputs depend only on the present state and nothing else.

- (b) (4 points) Draw the State Transition Diagram corresponding to the Verilog code given above.



- (c) (5 points) Complete the following state transition table for the FSM described by the Verilog code. To make writing easier, denote the state bits by S_2, S_1, S_0 and the nextstate bits by N_2, N_1, N_0 . Note that the default behavior for the nextstate is to move to the INIT state. Since only five states have been defined, there are three additional states which we named DEF1, DEF2, DEF3. As an example, entries for these three default states and the NEXT state have been entered.

State				Inputs		Next State			
name	S_2	S_1	S_0	A	B	name	N_2	N_1	N_0
INIT	0	0	0	0	0	DECODE	0	0	1
INIT	0	0	0	0	1	INIT	0	0	0
INIT	0	0	0	1	0	INIT	0	0	0
INIT	0	0	0	1	1	INIT	0	0	0
DECODE	0	0	1	0	X	LOOP	1	0	0
DECODE	0	0	1	1	X	NEXT	0	1	0
LOOP	1	0	0	X	X	JUMP	1	1	1
JUMP	1	1	1	X	0	DECODE	0	0	1
JUMP	1	1	1	X	1	INIT	0	0	0
NEXT	0	1	0	X	X	INIT	0	0	0
DEF1	0	1	1	X	X	INIT	0	0	0
DEF2	1	0	1	X	X	INIT	0	0	0
DEF3	1	1	0	X	X	INIT	0	0	0

Note that there are different ways of writing this table to represent the same result.

- (d) (1 point) For describing the `nextstate` is it better to use Products-of-Sums (POS) or Sums-of-Products (SOP)? Briefly explain.

Solution: Sums-of-Products, there are fewer entries with a 1 in them, and SOP requires one entry for each

- (e) (3 points) Write down the Boolean Equations for the `nextstate` bits N_2, N_1, N_0 , in either POS or SOP.
You don't need to minimize the equations.

Solution:

$$N_0 = \overline{S_2} \overline{S_1} \overline{S_0} \overline{A} \overline{B} + S_2 \overline{S_1} \overline{S_0} + S_2 S_1 S_0 \overline{B}$$

$$N_1 = \overline{S_2} \overline{S_1} S_0 A + S_2 \overline{S_1} \overline{S_0}$$

$$N_2 = \overline{S_2} \overline{S_1} S_0 \overline{A} + S_2 \overline{S_1} \overline{S_0}$$

- (f) (1 point) Briefly explain how the output `z` could be obtained in an actual circuit implementation, what kind of logic circuit would be needed?

Solution: The output is directly obtained from the 2 least significant bits of the state, or $S_1 S_0$.