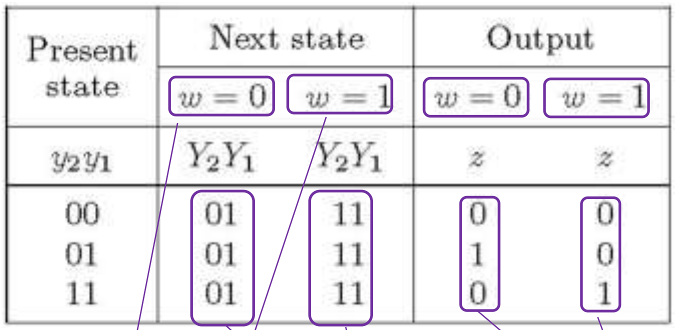
B1. Figure 2 shows a state-assigned table for a finite state machine.

(a) Is this a Moore's or Mealy's finite state machine? Support your answer with reason.
Answer: Mealy's since output depends on present states as well as input.

(b) Derive the excitation table using flips.



Present state	Next state		Output	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
$y_2 y_1$	$Y_2 Y_1$	$Y_2 Y_1$	z	z
00	01	11	0	0
01	01	11	1	0
11	01	11	0	1

Excitation table – truth table/K-map for generating the Next State and Output(s):

Input & Present State			**Next State**		**Output**
w	y_2 (Q_2)	y_1 (Q_1)	Y_2 (D_2)	Y_1 (D_1)	**z**
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	x	x	x
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	0
1	1	0	x	x	x
1	1	1	1	1	1

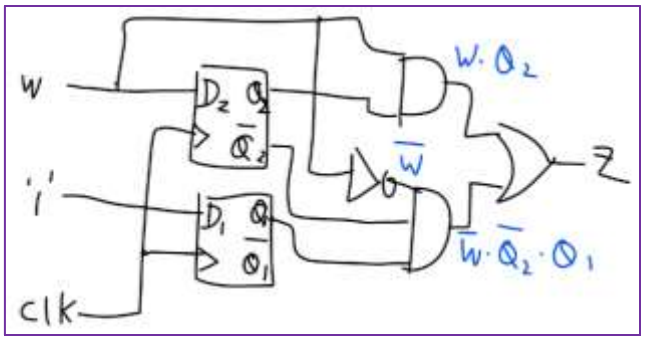
This state is not used.

(The following is not required in the question.)

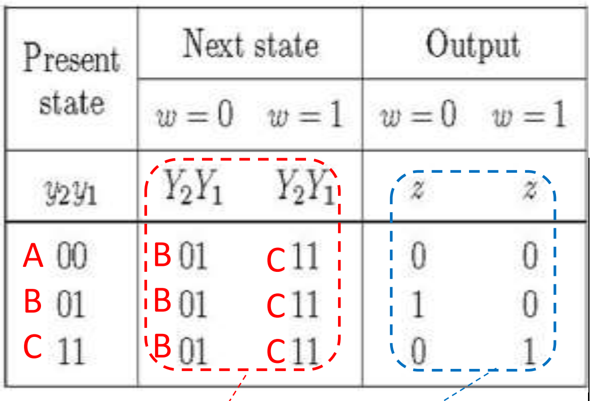
From K-map or inspection:

- $D_2 = w$
- $D_1 = 1$
- $Z = w'.Q_2'.Q_1 + w.Q_2$

Z		**Q2 Q1**			
		00	**01**	**11**	**10**
w	**0**	0	1	0	x
	1	0	0	1	x



B2. Write the Verilog code for the FSM in Figure 2. You may assume that the reset state is “00” and the Reset is asynchronous active low signal and Clock is an active low signal.



Present state	Next state		Output	
	$w=0$	$w=1$	$w=0$	$w=1$
y_2y_1	Y_2Y_1	Y_2Y_1	z	z
A 00	B 01	C 11	0	0
B 01	B 01	C 11	1	0
C 11	B 01	C 11	0	1

One possible solution:

```
module Tut5B2 (Clock, Reset, w, z);
        input wire Clock, Reset, w;
        output reg z;

        reg [2:1] y, Y; // Present & Next State
        parameter [2:1] A=0, B=1, C=3; // State codes

        // Define the next state and output combinational circuit:
        always @(w,y)
          case (y)
            A: if (w==1) begin  z=0; Y=C;  end
                     else begin z=0;  Y=B;  end

            B: if (w==1) begin  z=0;  Y=C; end
                      else begin  z=1; Y=B; end

            C: if (w==1)  begin  z=1; Y=C; end
                      else begin  z=0; Y=B;  end
```

The unused state(s)

```
             default:  begin  z=1'b0; Y=A; end
          endcase
```

Set next state to be the reset state instead of don’t care (as in the notes).

```
        // Define the sequential block:
        always @(negedge Resetn, negedge Clock)
           if (Resetn==0)  y<=A;
           else y <= Y;
              // The next state now (Y) will become the
              // present state (y) one clock cycle later.
endmodule
```