


Computation II: embedded system design (5EIB0)



[Dashboard](#) / [My courses](#) / [5EIB0](#) / [Final Exam 2022-04-21](#)
/ [Part 2c - 2022-04-21 Finite State Machine \(Debug FSM Moore Sequence Detector - 4pt\)](#)


 Description

 [Submission view](#)

Part 2c - 2022-04-21 Finite State Machine (Debug FSM Moore Sequence Detector - 4pt)

 **Due date:** Thursday, 21 April 2022, 5:45 PM

 **Requested files:** seq_top.v ( [Download](#))

Type of work:  Individual work

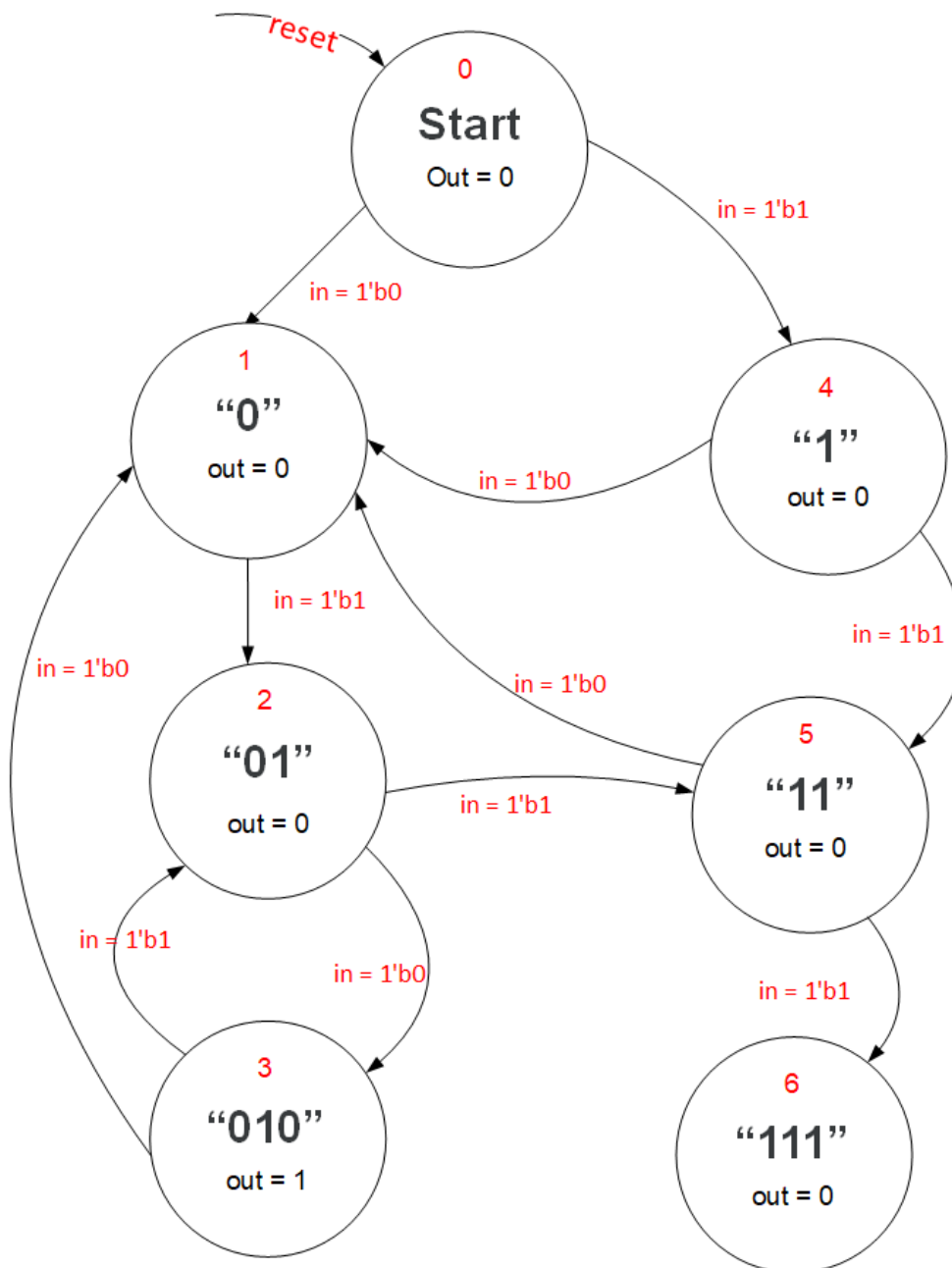
Introduction

In this exercise, you will debug the Verilog description of a module for a Finite State Machine (FSM) which detects a finite string pattern. The FSM has the following properties:

- one input (**in**) and one output (**out**)
- output is asserted whenever the input sequence ... **010** ... is observed, as long as the sequence ...**111**... is not seen.

Example input/output behavior:

in:	0	0	1	0	1	0	1	1	1	0	1	0	...
out:	0	0	0	1	0	1	0	0	0	0	0	0	...



The diagram above illustrates the transitions of the FSM. It shows a Moore Machine where the output is defined by the state. The machine has seven states. The state numbers are marked in red.

Note: All state transitions should happen when next rises from 0 to 1. DO NOT USE NEXT'S POSITIVE EDGE IN YOUR SENSITIVITY LIST (i.e no always @ (posedge next)).

Assignment

You will be provided with a buggy implementation of the above fsm. You need to find the bug and change it to make it working as per the state transition diagram given above. The FSM module you will modify is named **seq_top**. It takes four input signals **clk**, **reset**, **in** and **next** and produces two output signals **out** and **state_display**. The **reset** input always makes a transition to the "Start" state. Assume all input and output signals to be 1 bit wide except state_display which is 3 bits wide and little endian. The **state_display** must output the number of the current state using the same numbers as the states in the FSM diagram above (shown in red).



Requested files

seq_top.v

```

1  timescale 1ns / 100ps
2
3  module seq_top (
4      input wire [0:0] clk,
5      input wire [0:0] reset,
6      input wire [0:0] next,
7      input wire [0:0] in,
8      output wire [2:0] state_display,
9      output wire [0:0] out
10 );
11
12 /* Define state parameters */
13 localparam
14     START = 0,
15     STATE_1 = 1,
16     STATE_2 = 2,
17     STATE_3 = 3,
18     STATE_4 = 4,
19     STATE_5 = 5,
20     STATE_6 = 6;
21
22 /* Internal state registers */
23 reg [2:0] state;
24 reg [2:0] next_state;
25
26 /* Internal next register */
27 reg [0:0] next_last;
28
29 /* Clock Logic */
30 always @(posedge clk) begin
31     if(reset == 1'b1) begin
32         state <= START;
33         next_last <= 1'b0;
34     end else begin
35         state <= next_state;
36         next_last <= next;
37     end
38 end
39
40 always @(*) begin
41     next_state = state;
42
43     if(next == 1'b1 && next_last == 1'b0) begin
44         case (state)
45             START: begin
46                 if ((in == 1'b0)) begin
47                     next_state = STATE_2;
48                 end else if ((in == 1'b1)) begin
49                     next_state = STATE_2;
50                 end
51             end
52             STATE_1: begin
53                 if ((in == 1'b0)) begin
54                     next_state = STATE_4;
55                 end else if (in == 1'b1) begin
56                     next_state = STATE_2;
57                 end
58             end
59             STATE_2: begin
60                 if ((in == 1'b0)) begin
61                     next_state = STATE_3;
62                 end else if ((in == 1'b1)) begin
63                     next_state = STATE_4;
64                 end
65             end
66             STATE_3: begin
67                 if ((in == 1'b1)) begin
68                     next_state = STATE_2;
69                 end else if ((in == 1'b0)) begin
70                     next_state = STATE_1;
71                 end
72             end
73             STATE_4: begin
74                 if ((in == 1'b0)) begin
75                     next_state = STATE_1;
76                 end else if ((in == 1'b1)) begin
77                     next_state = STATE_5;
78                 end
79             end
80             STATE_5: begin
81                 if ((in == 1'b1)) begin
82                     next_state = STATE_6;
83                 end else if ((in == 1'b0)) begin
84                     next_state = STATE_1;
85                 end
86             end
87             STATE_6: begin
88                 if ((in == 1'b0)) begin
89                     next_state = STATE_6;
90                 end else if ((in == 1'b1)) begin
91                     next_state = STATE_6;
92                 end
93             end
94         endcase
95     end
96
97 end
98 assign out = (state == STATE_3);
99 assign state_display = state;
100
101 endmodule

```

[VPL](#)

◀ Part 2b - 2022-04-21 Finite State Machine (Write testbench for FSM Moore Sequence Detector - 3pt)

Jump to...

[Part 2d: 2022-04-21 Finite State Machine \(Design Moore AC controller - 6pt\) ►](#)

You are logged in as Alexandru Tănasă (Log out)
5EIB0

[Data retention summary](#)
[Get the mobile app](#)

