Computation II: embedded system design (5EIB0)

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

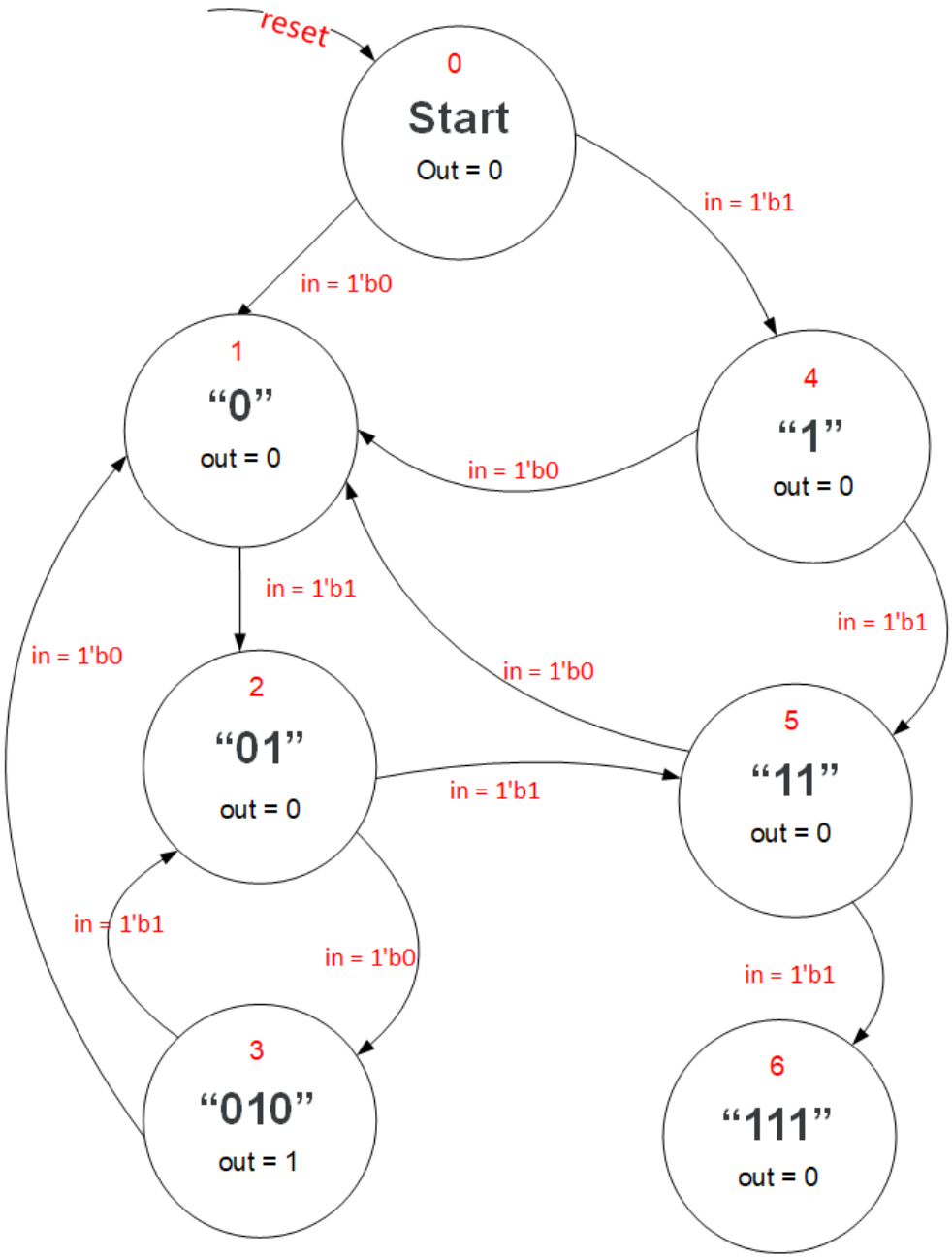
Introduction

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

```
`timescale 1ns / 100ps

module seq_top (
    input  wire [0:0] clk,
    input  wire [0:0] reset,
    input  wire [0:0] next,
    input  wire [0:0] in,
    output wire [2:0] state_display,
    output wire  [0:0] out
    );

    /* Define state paramaters */
    localparam
    START   = 0,
    STATE_1 = 1,
    STATE_2 = 2,
    STATE_3 = 3,
    STATE_4 = 4,
    STATE_5 = 5,
    STATE_6 = 6;

    /* Internal state registers */
    reg [2:0] state;
    reg [2:0] next_state;

    /* Internal next register */
    reg [0:0] next_last;

    /* Clock Logic */
    always @(posedge clk) begin
        if(reset == 1'b1) begin
            state <= START;
            next_last <= 1'b0;
        end else begin
            state <= next_state;
            next_last <= next;
        end
    end

    always @(*) begin
        next_state = state;

        if(next == 1'b1 && next_last == 1'b0) begin
           case (state)
                START: begin
                    if ((in == 1'b0)) begin
                        next_state = STATE_2;
                    end else if ((in == 1'b1)) begin
                        next_state = STATE_2;
                    end
                end
                STATE_1: begin
                    if ((in == 1'b0)) begin
                        next_state = STATE_4;
                    end else if (in == 1'b1) begin
                        next_state = STATE_2;
                    end
                end
                STATE_2: begin
                    if ((in == 1'b0)) begin
                        next_state = STATE_3;
                    end else if ((in == 1'b1)) begin
                        next_state = STATE_4;
                    end
                end
                STATE_3: begin
                    if ((in == 1'b1)) begin
                        next_state = STATE_2;
                    end else if ((in == 1'b0)) begin
                        next_state = STATE_1;
                    end
                end
                STATE_4: begin
                    if ((in == 1'b0)) begin
                        next_state = STATE_1;
                    end else if ((in == 1'b1)) begin
                        next_state = STATE_5;
                    end
                end
                STATE_5: begin
                    if ((in == 1'b1)) begin
                        next_state = STATE_6;
                    end else if ((in == 1'b0)) begin
                        next_state = STATE_1;
                    end
                end
                STATE_6: begin
                    if ((in == 1'b0)) begin
                        next_state = STATE_6;
                    end else if ((in == 1'b1)) begin
                        next_state = STATE_6;
                    end
                end
             endcase
        end

    end
assign out = (state == STATE_3);
assign state_display = state;

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) ►