

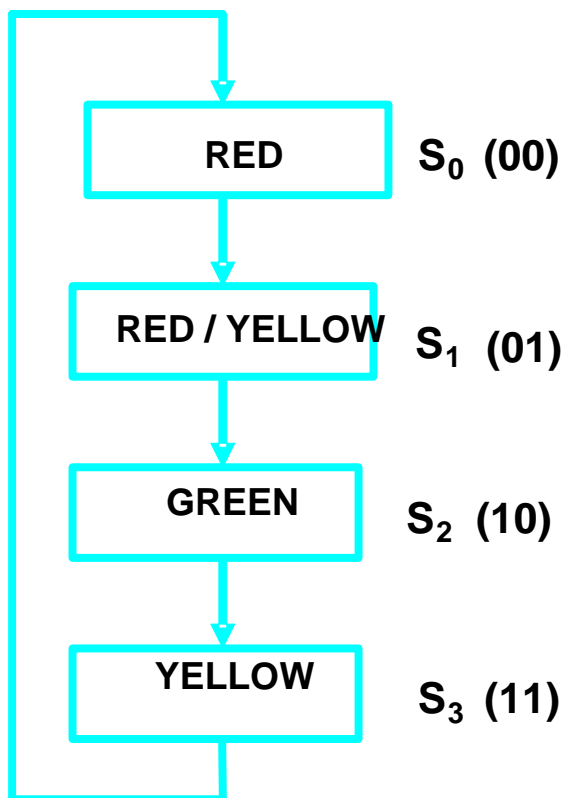
CS226 Lab 14

Design using HDL(Verilog)

The goal of this lab12 is to familiarize the students with describing FSMs in verilog Hardware Description Language (HDL). In this lab presented in the form of problems and answers.

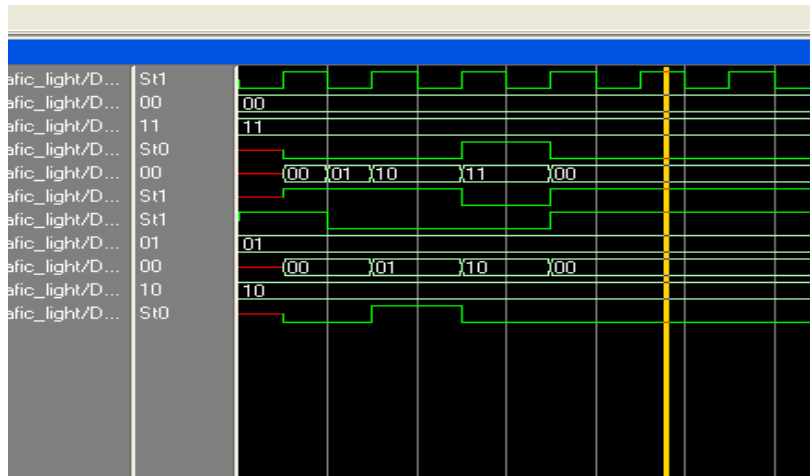
Task1: Simulate traffic light controller.

Finite state machine consists of combinational, sequential and output logic. Combinational logic is used to decide the next state of the FSM, sequential logic is used to store the current state of the FSM. The output depends on the current state of the machine in this example.

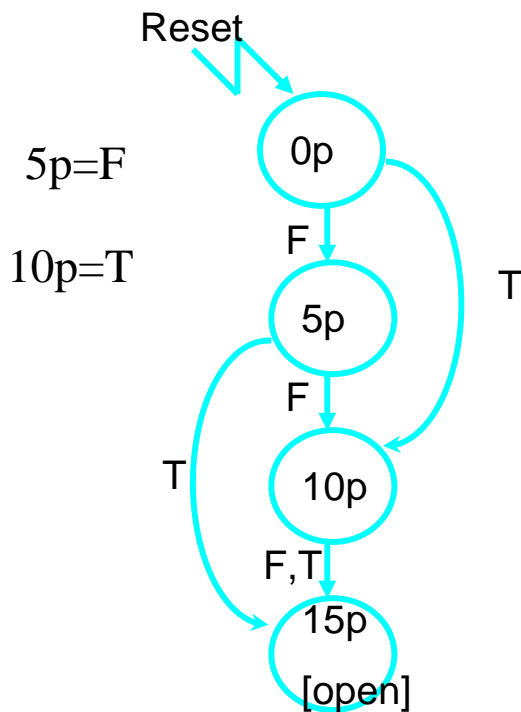


```
module fsm_traffic_light(clk,reset, red,yellow, green);
input clk, reset;
//output [1:0] state;
output red,yellow,green;
reg [1:0] next_state;
reg [1:0] state;
parameter [1:0] FIRST= 2'b00;
parameter [1:0] SECOND= 2'b01;
parameter [1:0] THIRD = 2'b10;
parameter [1:0] FOURTH=2'b11;
always @(posedge clk) // sequential
begin
if (reset) state <= FIRST;
else state <= next_state;
end
always @(state) // combinational
begin
case(state)
FIRST: if (reset)
next_state = FIRST;
else next_state = SECOND;
SECOND: if (reset)
next_state = FIRST;
else next_state = THIRD;
THIRD: if (reset)
next_state = FIRST;
else next_state = FOURTH;
FOURTH: if (reset)
next_state = FIRST;
else next_state = FIRST;
endcase end
// output logic described using continuous assignment

assign red = (state == FIRST) | (state == SECOND);
assign yellow = (state == SECOND | (state ==
FOURTH));
assign green = (state == THIRD);
endmodule
```



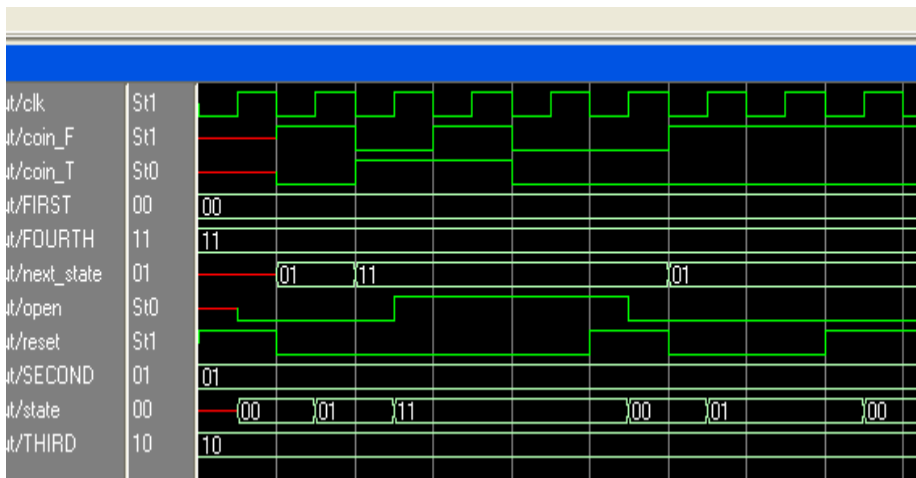
Task2: Simulate vending machine.



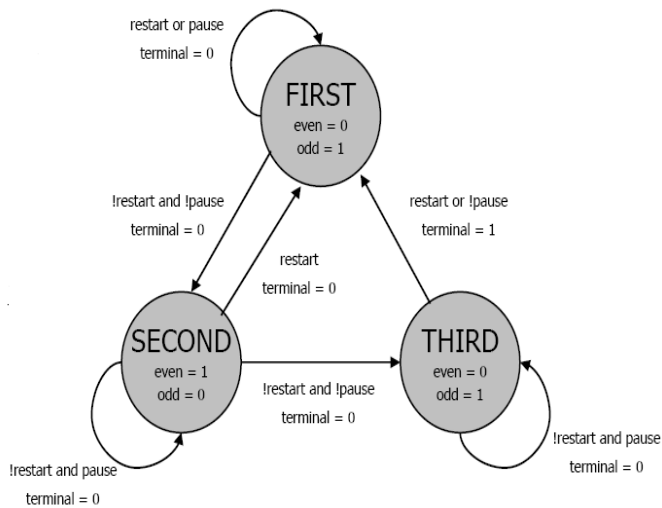
```

module fsm_vendingmachine(open,clk,reset,
    coin_T,coin_F);
    input clk, reset;
    input coin_T,coin_F;
    output open;
    reg [1:0] next_state;
    reg [1:0] state;
    parameter [1:0] FIRST= 2'b00;
    parameter [1:0] SECOND= 2'b01;
    parameter [1:0] THIRD = 2'b10;
    parameter [1:0] FOURTH=2'b11;
    always @(posedge clk) // sequential
    begin
        if (reset) state <= FIRST;
        else state <= next_state;
    end
    always @(state,coin_T,coin_F) // combinational
    begin
        case(state)
            FIRST: if (coin_F)
                    next_state = SECOND;
                    else if (coin_T) next_state = THIRD;
            SECOND: if (coin_F)
                    next_state = SECOND;
                    else next_state = FOURTH;
            THIRD: if (coin_F)
                    next_state = FOURTH;
                    else next_state = FOURTH;
            FOURTH: if (coin_F)
                    next_state = FOURTH;
                    else next_state = FOURTH;
        endcase end
    // output logic described using continuous assignment
    assign open = (state == FOURTH);
endmodule

```



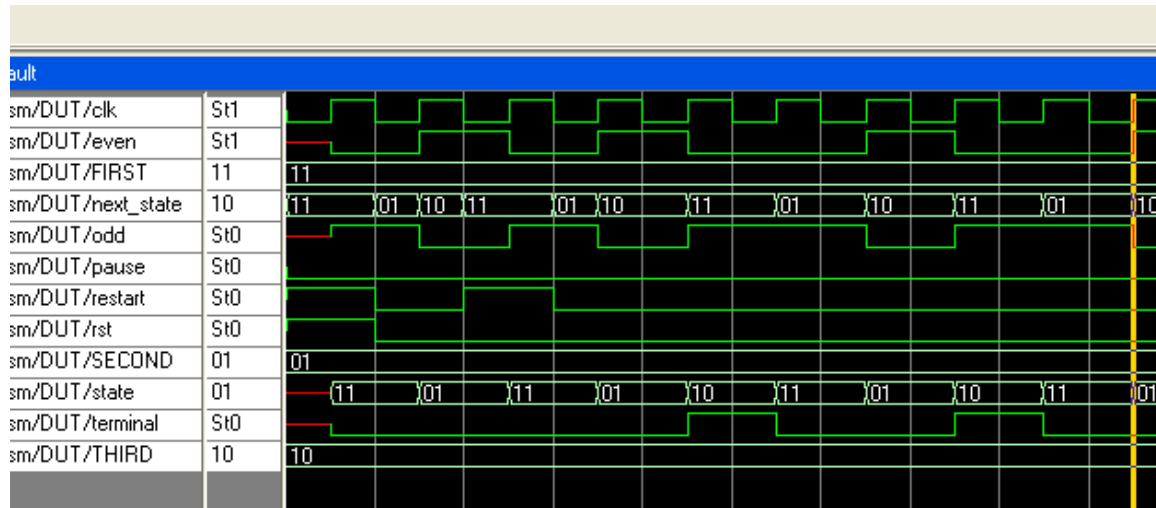
Task3: Simulate the state diagram description in verilog.



```

module fsm(pause,restart,clk, rst, state,odd,even,
terminal);
input pause,restart,clk, rst, odd,even;
output [1:0] state;
output terminal;
reg[1:0] next_state,state;
parameter [1:0] FIRST= 2'b11;
parameter [1:0] SECOND= 2'b01;
parameter [1:0] THIRD = 2'b10;
always @(posedge clk) // sequential
begin
if (rst) state <= FIRST;
else state <= next_state;
end

always @* // combinational
begin
case(state)
FIRST: if (restart | pause)
next_state = FIRST;
else next_state = SECOND;
SECOND: if (restart)
next_state = FIRST;
else if (pause) next_state =
SECOND;
else next_state = THIRD;
THIRD: if (!restart & pause) next_state =
THIRD;
else next_state = FIRST;
default: next_state = FIRST;
endcase
end
  
```



Your Lab Assignment 14 (100 points)

Submission Your submission must contain- Verilog description and test in a single file:

Submission Your submission must contain:

- The source code of your design/testbench/logic sim file if any (for each of the problem) - reasonably well documented.
- A word document showing Verilog code/test bench and simulation waveform(screen shots)

Course work submission through

<https://my.pcloud.com/#page=puplink&code=tns7ZSgK1dEn3j9yYfBscCi7YTkH6Pzry>

File Name: YourrollNo_Lab14.

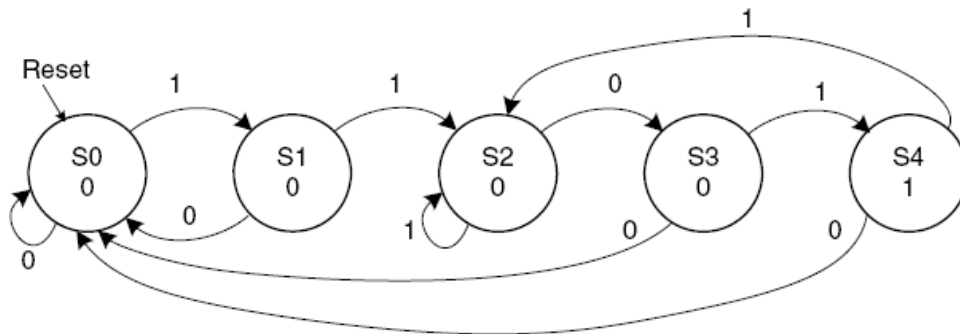
Due on: 24th May 2020

Question 1

Bob owns a pet snail with an FSM brain. The snail crawls from left to right along a paper tape containing a sequence of 1's and 0's. On each clock cycle, the snail crawls to the next bit. The snail smiles when the last four bits that it has crawled over are, from left to right, 1101. Design the FSM to compute when the snail should smile. The input A is the bit underneath the snail's antennae. The output Y is '1' when the snail smiles. Sketch a timing diagram (modelsim simulation output) showing the input, states, and output as your snail crawls along the sequence 11101101001001101101. (20 points)



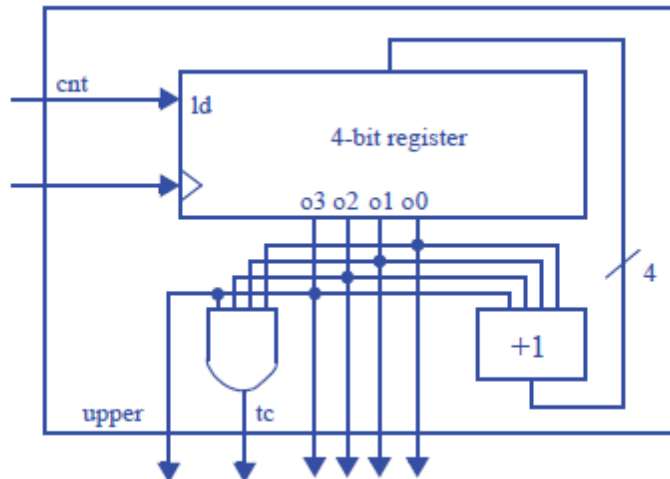
Hint : The state diagram has been solved for you (see below)



Q2.

Design a 4-bit up-counter with an additional output *upper*. *upper* outputs a 1 whenever the counter is within the upper half of the counter's range, 8 to 15. Use a basic 4-bit up-counter.

(hint: basic architecture is given below) . Create a Verilog model.



(20 points)

Q3: (a) Simulate the block given blow in logic-sim

(b) Create the Verilog model and a test bench to simulate the behavior block behavior.

(c) Design a simple state machine to add two numbers that is stored in the registers(R1-R7) and store the result in another Register(R1-R7). Hint: generate control signals (SELA, SELB, OPR etc) . Integrate this FSM to Part (a) and Part (b)

(points 10+20+40)

