Lab 3 Finite State Machine

FSM

Before talking about the details of this assignment, here's an example to help you get familiar with FSM description in Verilog.

Suppose we want to design a model "lab3", and it has three inputs and one output.

- Input

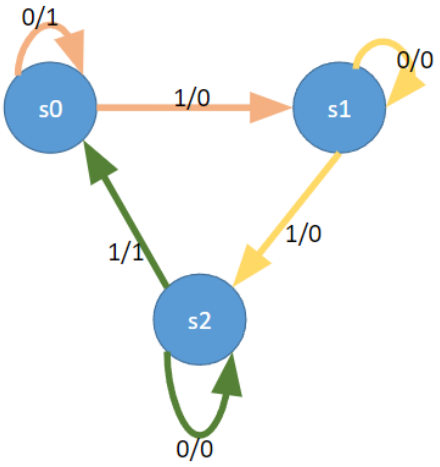
clk	1-bit integer
reset	1-bit integer
X	1-bit integer

- Output

Y	1-bit integer

When the total number of 1's received is a multiple of 3, the output Y will be 1; otherwise, it will be 0.

The following diagram is the state diagram of the module. "s0" represents the total number of 1's received as a multiple of 3, "s1" represents the total number of 1's received as a multiple of 3 plus 1, and "s2" represents the total number of 1's received as a multiple of 3 plus 2.



```
module lab3(clk, reset, X, Y)
    input clk, reset, X;
    output reg Y;
    reg [1: 0] state;
    reg [1: 0] next_state;

    always @(*) begin
        if(reset)
            next_state = 3'b000;
        else begin
            if(state = 2'd0)begin
                if(X = 1'd0)
                    next_state = state;
                else
                    next_state = state + 2'd1;
            end
            else if(state = 2'd1)begin
                if(X = 1'd0)
                    next_state = state;
                else
                    next_state = state + 2'd1;
            end
            else begin
                if(X = 1'd0)
                    next_state = state;
                else
                    next_state = 2'd0;
            end
        end
    end
```

```
    always @(*) begin
        if(reset)
            Y = 1'b0;
        else begin
            if(state = 2'd0)begin
                if(X = 1'd0)
                    Y = 1'b1;
                else
                    Y = 1'b0;
            end
            else if(state = 2'd1)begin
                if(X = 1'd0)
                    Y = 1'b01;
                else
                    Y = 1'b0;
            end
            else begin
                if(X = 1'd0)
                    Y = 1'b0;
                else
                    Y = 1'b1;
            end
        end
    end

```

```
    always@(posedge clk)begin
        if(reset)
            state <= 2'b00;
        else
            state <= next_state;
    end

endmodule
```

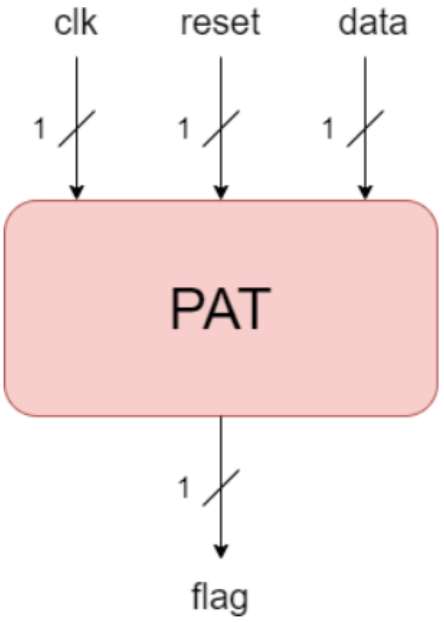
- 1st "always" block is used to define the next state.
- 2nd "always" block defines the output in each condition.
- 3rd "always" block will be triggered at the posedge of clock, and update the state with next_state.

Question 1: Patterns Matching

The tokens of the given sequence are binary bits, either “1” or “0”. In this lab, we aim to find patterns “1011” or “1101”.

Module

There is one module, “PAT”, we need to design for this question.



Input and Output

- Input

Input		Description
clk	1-bit	clock signal
reset	1-bit	Synchronous reset. When reset == 1’b1 at positive clock edge, reset the module to initial state and stay at initial stage in that cycle.
data	1-bit	Details given below.

- Output

Output		Description
flag	1-bit	Details given below.

Details of Input and Output

- Trigger the flip-flops in your module with posedge “clk”.
- “data” is a token of the sequence, and the module will receive it one by one in every clock cycle.
- If either of the patterns are detected, the output variable “flag” should be “1”, otherwise it’s “0”.

Example

cycle	reset	data	flag	cycle	reset	data	flag
0	1	1	0	6	0	1	1
1	0	0	0	7	0	0	0
2	0	1	0	8	0	1	0
3	0	1	0	9	0	1	1
4	0	1	0	10	1	0	0
5	0	0	0	11	0	1	0

- In cycle “0”, since “reset” is “1”, the input data is ignored.
- We start detecting the desired patterns at cycle 1.
- In cycle 6, the pattern “1101” we want is detected, so the “flag” is “1”.
- In cycle 9, the pattern “1011” we want is detected, so the “flag” is “1”.
- In cycle 10, since “reset” is “1”, it resets the module to the initial state.

Testbench

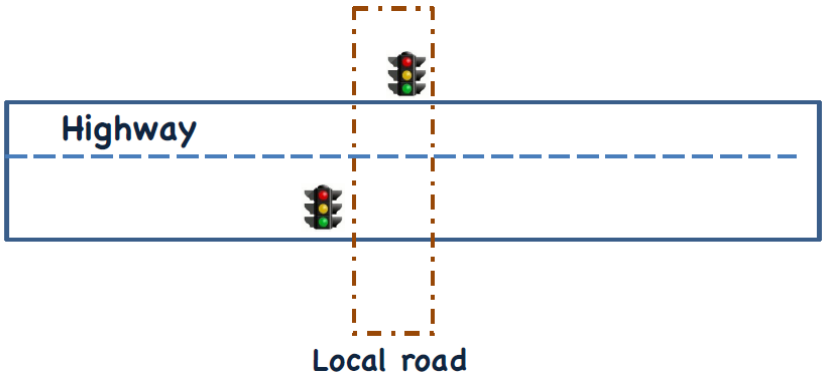
Follow the rules when creating the testbench.

- We assume the flip-flops are triggered by posedge “clk”, so in the testbench we only change the values of input “data” and “reset” at negedge “clk” to make sure that the setup time and hold time constraints of the flip-flops are satisfied.
- If the value of “flag” is 1’bx or 1’bz, or it is different from the right value, the answer is wrong.

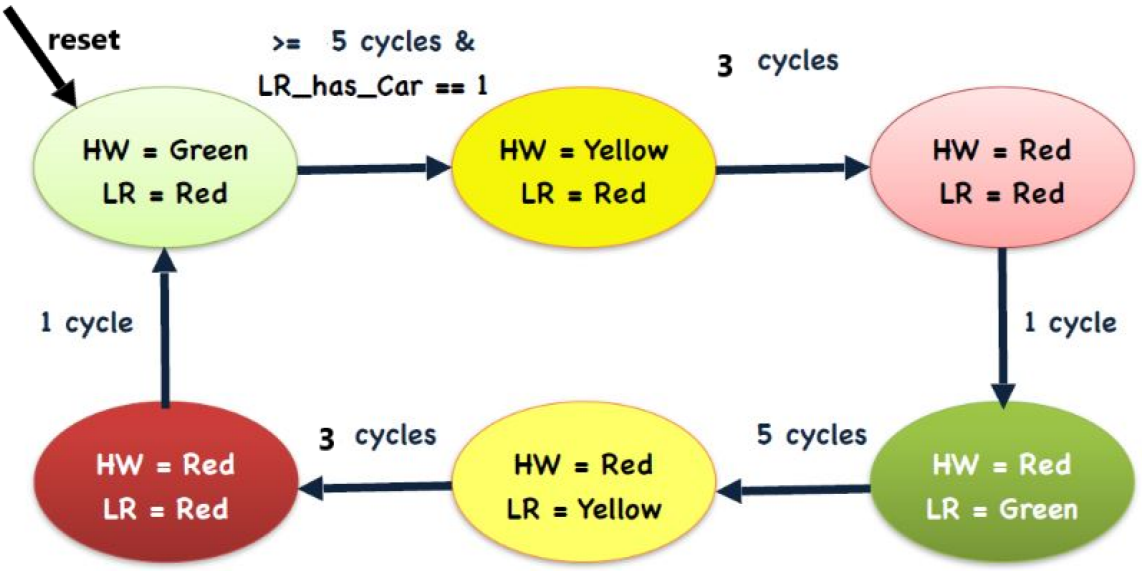
Question 2: Traffic Light Controller

Traffic light controller for highway (**HW**) and local road (**LR**) intersection

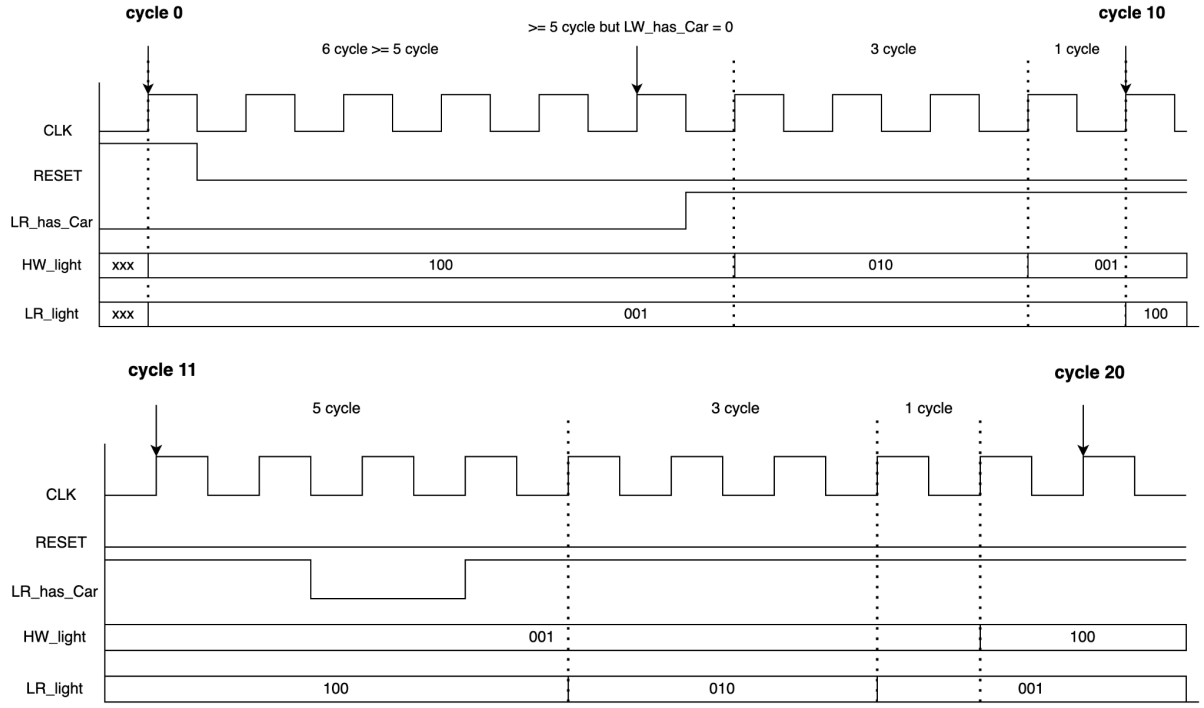
- **HW** has higher priority and should be green as long as possible
- **LR** has a sensor to detect cars on it. When a car is sensed, LR turns green shortly
- Green light is **at least 5 clock cycles** and yellow light is **3 clock cycles**
- **Input**: **CLK**, **RESET**, LR_has_Car; **Output**: HW_light[2:0], LR_light[2:0]
- HW_light[2:0] & LR_light[2:0]: the **bits [2:0]** in HW_light[2:0] and LR_light[2:0] represent **Green**, **Yellow**, and **Red**, respectively.
- In your testbench, you must test that your program functions properly in all states of the state diagram.



Traffic light controller Finite State Machine:



Example



Testbench

Follow the rule when creating the testbench.

- We assume the flip-flops are triggered by posedge "CLK", so in the testbench we only change the values of input "LR_has_car" and "RESET" at negedge "CLK" to make sure that the setup time and hold time constraints of the flip-flops are satisfied.
- Print the clock cycle and the values of "HW_light" and "LR_light" when state changes."HW_light" and "LR_light" should be in binary form.

Requirements

1. Two modules to implement
   - Patterns Matching
     - module name: PAT
     - input name: clk, reset, data
     - output name: flag
   - Traffic Light Controller
     - module name: Traffic_Light_Controller
     - input name: CLK, RESET, LR_has_Car
     - output name: HW_light, LR_light
2. Put the two modules in a single file "lab3.v"
   - Do not put your testbenches in the file.

Files to upload

1. lab3.v (80%)
2. {studentID}_report.pdf (For example, 111062000_report.pdf) (20%)
   - Patterns Matching
     - State diagram (5%)
     - Execution result (screen shot, use the example as input data) (5%)
   - Traffic Light Controller
     - Execution result (screen shot, use the example as input data) (5%)
   - The problem you faced and how you deal with it. (5%)

Please upload these two files on the eeclass system.
Be careful of the file name, module name, and whatever listed above.
Wrong format will result in the failure (0 points) of your assignment, please double check before uploading it.

Deadline

2023/06/05 23:58

Upload the files before the deadline!
Late assignment will get a 10% penalty per day, and no assignment will be accepted after 3 days from the given due date.
Do not plagiarize!