CSCE 342

Assignment #3

ThunderBird Taillight on the BASYS3

Mark Quimpo

(31113576)

tBirdTaillightTop.sv

```
module tBirdTaillightTop(   input logic clk,
                            input logic [15:0] sw,
                            input logic btnC,
                            input logic btnU,
                            output logic [15:0] led
                        );

    logic clk_1Hz;

    logic [2:0] leftLEDs;
    logic [2:0] rightLEDs;

    slowClock slowClockInstance(.clk(clk), .reset(btnU), .clk_1Hz(clk_1Hz));

    taillight leftTaillight(.clk(clk_1Hz), .reset(btnU), .brake(btnC), .turnSignal(sw[15]), .taillights(leftLEDs));
    taillight rightTaillight(.clk(clk_1Hz), .reset(btnU), .brake(btnC), .turnSignal(sw[0]), .taillights(rightLEDs));



    assign led[15:13] = {leftLEDs[0], leftLEDs[1], leftLEDs[2]};
    assign led[12:3] = 10'b00000_00000;
    assign led[2:0] = {rightLEDs[2], rightLEDs[1], rightLEDs[0]};

endmodule
```

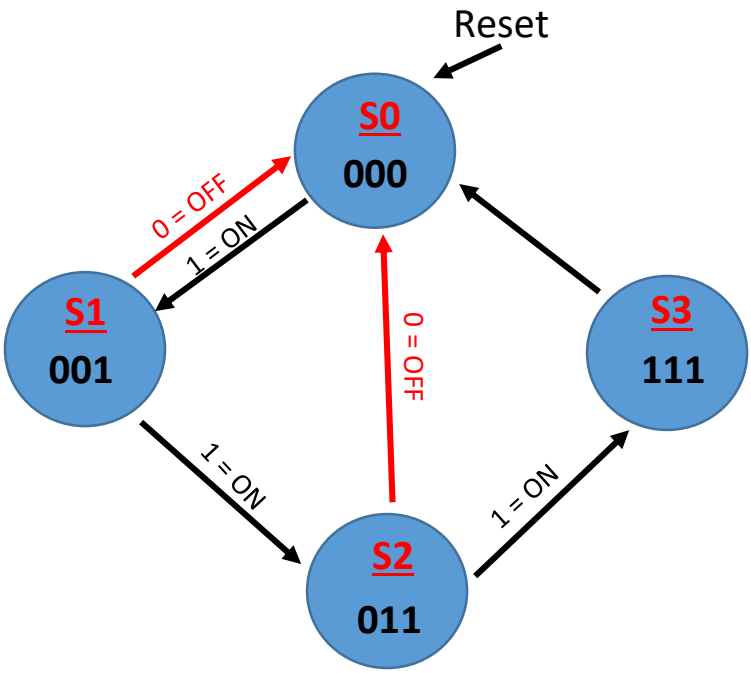
taillight.sv

```
module taillight(   input logic clk,
                    input logic reset,
                    input logic brake,
                    input logic turnSignal,
                    output logic [2:0] taillights
                );

    logic [2:0] blinkingLights;
    blinkers blinkersInstance( .clk(clk), .reset(reset), .turnSignal(turnSignal), .blinkingLights(blinkingLights));

    always_comb
    begin
            if (turnSignal == 1)
                begin
                    taillights = blinkingLights;
                end
            else if (brake == 1)
                begin
                    taillights = 3'b111;
                end
            else
                taillights = 3'b000;
    end
endmodule
```

STATE DIAGRAM



blinkers.sv

```
module blinkers( input  logic       clk,
                 input  logic       reset,
                 input  logic       turnSignal,
                 output logic [2:0] blinkingLights );

    typedef enum logic [1:0] {S0, S1, S2, S3} statetype;
    statetype state, nextstate;

    always_ff @( posedge clk, posedge reset)
    begin
        if( reset )
            state <= S0;
        else
            state <= nextstate;
    end


    always_comb
    begin
        case(state)
        S0:         if(turnSignal == 0) nextstate = S0;
                    else                nextstate = S1;
        S1:         if(turnSignal == 1) nextstate = S2;
                    else                nextstate = S0;
        S2:         if(turnSignal == 1) nextstate = S3;
                    else                nextstate = S0;
        S3:                             nextstate = S0;
        default:                        nextstate = S0;

        endcase;
    end

    assign blinkingLights[2] = (state == S1) || (state == S2) || (state == S3);
    assign blinkingLights[1] = (state == S2) || (state == S3);
    assign blinkingLights[0] = (state == S3);

endmodule
```

blinkers.tv

```
10_000  // reset blinkers FSM, all bits of blinkingLights turn off
01_100  // turn signal on, so most significant bit of blinkingLights turns on
00_000
01_100
01_110  // turn signal remains on, two most significant bits of blinkingLights are on
00_000
01_100
01_110
01_111  // turn signal remains on, all bits of blinkingLights are on
00_000
01_100
01_110
01_111
01_000  // turn signal remains on, all bits of blinkingLights turn off
00_000
//continue checking the FSM so that *all* the transitions have been enumerated, the above is just to help you get a start
```

Simulation