CS223 LAB5

Altay İlker Yiğitel
22203024
CS223 sec: 01

```
module traffic(
    input SA,
    input SB,
    input reset,
    output reg [5:0] current_state,
    input clk_100MHz
);
    reg[5:0] next_state;
    reg [25:0] counter_reg = 0;
    reg clk_out_reg = 0;
    reg clk_1Hz;
oneHz_generator ohzz(.clk_100MHz(clk_100MHz),.clk_1Hz(clk_1Hz));


    parameter BOTH_RED1 = 6'b111111;
    parameter BOTH_RED2 = 6'b111111;
    parameter LA_GREEN = 6'b110111;
    parameter LB_GREEN = 6'b111110;
    parameter LA_YELLOW1 = 6'b100111;
    parameter LB_YELLOW1 = 6'b111100;
    parameter LA_YELLOW2 = 6'b111100;
    parameter LB_YELLOW2 = 6'b100111;
```

```
always@(posedge clk_1Hz) begin
    if(reset)begin
        current_state <= LA_GREEN;
    end else begin
        current_state <= next_state;
        end
end
always @(posedge clk_1Hz)
    case (current_state)
        LA_GREEN: begin
            if (SA)
                next_state <= current_state;
            else
                next_state <= LA_YELLOW1;
            end
            LB_GREEN: begin
                if (SB)
                    next_state <= current_state;
                else
                    next_state <= LB_YELLOW1;
                end
            BOTH_RED1: begin
                    next_state <= LA_YELLOW2;
```

```
            end
            BOTH_RED2: begin
                next_state <= LB_YELLOW2;
            end
            LA_YELLOW1: begin
                next_state <= BOTH_RED2;
            end
            LB_YELLOW1: begin
                next_state <= BOTH_RED1;
            end
            LA_YELLOW2: begin
                next_state <= LB_GREEN;
            end
            LB_YELLOW2: begin
                next_state <= LA_GREEN;
            end
        endcase

endmodule

module BCD_counter4(
```

```
    input clk_100MHz,
    input res,
    input lo,
    input [9:0] in,
    output [0:6] seg,
    output [3:0] an

    );
    wire reset,load;
    wire clk_1Hz;
    oneHz_generator ohz(.clk_100MHz(clk_100MHz),.clk_1Hz(clk_1Hz));
    btn_debouncer bt(.clk_100MHz(clk_100MHz),.btn_in(res),.btn_out(reset));
    btn_debouncer btt(.clk_100MHz(clk_100MHz),.btn_in(lo),.btn_out(load));

    reg [3:0] sec_ones = 0;
    reg [3:0] sec_tens = 0;
    reg [3:0] sec_hundereds = 0;
    reg [3:0] sec_thousands = 0;

    seg7_control seg7(.clk_100MHz(clk_100MHz), .reset(res), .ones(sec_ones), .tens(sec_tens),
.hundereds(sec_hundereds),
                .thousands(sec_thousands), .seg(seg), .an(an));

    always @(posedge clk_1Hz or posedge reset or posedge load) begin
       if(reset)begin
          sec_ones <= 0;
          sec_tens <= 0;
```

```
    sec_hundereds <= 0;
    sec_thousands <= 0;
end
else if(load)begin
    sec_ones <=in%10;
    sec_tens <=(in/10)%10;
    sec_hundereds <=(in/100)%10;
    sec_thousands <=(in/1000)%10;
end
else
    if(sec_ones == 9) begin
        sec_ones <= 0;
        if(sec_tens == 9)begin
            sec_tens <= 0;
            if(sec_hundereds ==9)begin
                sec_hundereds <= 0;
                if(sec_thousands == 9)
                    sec_thousands <= 0;
                else
                    sec_thousands <= sec_thousands+1;
            end
            else
                sec_hundereds <= sec_hundereds+1;
        end
        else
            sec_tens <= sec_tens+1;
    end
```

```
        else
            sec_ones <= sec_ones + 1;
    end


endmodule
```