LAB 5
SARTHAK BHAGAT
2016189

TOP

```
module top(
    input pushbutton1,
    input pushbutton2,
    input clock,
    input clear,
    output reg [7:0] cathode,
    output reg [3:0] anode
    );

    wire inp;
    wire [7:0] cathodea,cathodeb,cathodec;
    reg [14:0] i=0;
    wire [7:0] s0;
    wire [3:0]ones,tens,hundreds;
    debouncing d1 (clear,clear_out);
    fd #(19) f1 (clock,clock19);

    assign temp = pushbutton1 | pushbutton2;

    clean_pulse cp1 (clock19,temp,inp);
    shift_register sr1 (inp,clock19,clear_out,s0);
    b2b b1 (s0,ones,tens,hundreds);

    seven_segment s1 (ones,4'b1110,cathodea);
    seven_segment s2 (tens,4'b1101,cathodeb);
    seven_segment s3 (hundreds,4'b1011,cathodec);

    always @(posedge clock19)
    begin
        if (i < 3000)
        begin
            cathode <= cathodea;
            anode <= 4'b1110;
            i <= i + 1;
        end

        else if (i < 6000 & i > 3000)
        begin
            cathode <= cathodeb;
            anode <= 4'b1101;
            i <= i + 1;
        end

        else if (i < 9000 & i > 6000)
        begin
            cathode <= cathodec;
            anode <= 4'b1011;
            i <= i + 1;
        end

        else if (i == 9000)
        begin
            i <= 0;
        end
    end

endmodule
```

DEBOUNCING

```
module debouncing(
   input clock_in,
   input clr_in,
   output clr_out
   );

   reg D1,D2,D3;
       always@(posedge clock_in)
       begin
           D1 <= clr_in;
           D2 <= D1;
           D3 <= D2;
       end

       assign clr_out = D1 && D2 && D3 ;

endmodule
```

FREQ DIVIDER

```
module fd(
    input cin,
    output cout

    );

    parameter width = 26;

    reg [width-1:0] count;

    always @(posedge cin)
    count <= count + 1;

    assign cout = count[width-1];


endmodule
```

CLEAN PULSE

```
module clean_pulse(
   input clock_in,
   input inp,
   output clr_out
   );

   reg D1,D2,D3;
      always@(posedge clock_in)
      begin
         D1 <= inp;
         D2 <= D1;
         D3 <= ~D2;
      end

      assign clr_out = D1 && D2 && D3 ;

endmodule
```

SEVEN SEGMENT

```
module seven_segment (
            input [3:0] ones,
            output [3:0] anode,
            output [7:0] cathode
```

```
	);

	reg [6:0] sseg_temp;
	always@ (*)
	begin
		case(ones)
			4'd0 : sseg_temp = 7'b0000001;
			4'd1 : sseg_temp = 7'b1001111;
			4'd2 : sseg_temp = 7'b0010010;
			4'd3 : sseg_temp = 7'b0000110;
			4'd4 : sseg_temp = 7'b1001100;
			4'd5 : sseg_temp = 7'b0100100;
			4'd6 : sseg_temp = 7'b0100000;
			4'd7 : sseg_temp = 7'b0001111;
			4'd8 : sseg_temp = 7'b0000000;
			4'd9 : sseg_temp = 7'b0000100;
			default : sseg_temp = 7'b1111110;
		endcase;
	end;


	assign cathode = {sseg_temp, 1'b1};

endmodule
```

BINARY TO BCD

```
module b2b(
	input [7:0] number,
    output reg [3:0] ones,
    output reg [3:0] tens,
    output reg [3:0] hundreds
    );

    reg [19:0] shift;
    integer i;

    always @(number)
    begin
       shift[19:8] = 0;
       shift [7:0] = number;
```

```
        for(i=0; i<8; i=i+1)
        begin
            if (shift[11:8] >= 5)
                shift[11:8] = shift[11:8] + 3;
            if (shift[15:12] >= 5)
                shift[15:12] = shift[15:12] + 3;
            if (shift[19:16] >=5)
                shift[19:16] = shift[19:16] + 3;
            shift = shift << 1;
        end
     hundreds = shift[19:16];
     tens = shift[15:12];
     ones = shift[11:8];
     end

endmodule
```

SHIFT REGISTER

```
module shift_register(
    input numb,
    input clock,
    input clear,
    output [7:0] output_numb
    );

    reg [7:0] s;
    always @(posedge clock)
    begin
        if (clear == 1)
            s <= 8'b00000000;
        else
            begin
            s[7] <= s[6];
            s[6] <= s[5];
            s[5] <= s[4];
            s[4] <= s[3];
            s[3] <= s[2];
            s[2] <= s[1];
            s[1] <= s[0];
            s[0] <= numb;
            end
    end
    assign output_numb = s;

endmodule
```