Lab 6

Top module

```
module top(
   input clk,
   input rst,
   output [7:0] cathode,
   output [3:0] anode,
   output [6:0] ssd, //JB[3:0] JC[2:0]
   output ssdcat   //JC[3]
   );

   wire deb_out,clk_20,clk_1;
   wire [5:0] num;

   frq_div #(20) fun1(clk,clk_20);
   frq_one fun2(clk,clk_1);

   debounce fun3(clk_20,deb_out,rst);
   up_counter fun4(clk_1,num,deb_out);
   sevenseg fun5(clk,num[2],num[3],num[4],num[5],cathode,anode);
   ssd fun6(clk,num[1],num[0],ssd,ssdcat);

endmodule
```

External ssd

```
module ssd(
input clk,
input ones,
input tens,
output[6:0] ssd,
output reg ssdcat
 );

reg[17:0] count;
   always @(posedge clk)
   begin
```

```
        count<=count+1;
    end

    reg[6:0] sseg_temp;
    //reg[3:0] antemp=4'b1110;
    always @(*)
    begin
    case(count[17])
    1'b0:
    begin
    case(ones)
        1'b0 : sseg_temp = 7'b0111111; //to display 0
        1'b1 : sseg_temp = 7'b0000110; //to display 1 //1111001
        default : sseg_temp = 7'b1000000; //dash

        endcase
        ssdcat=1'b1;
        end
       1'b1:
       begin
       case(tens)
            1'b0 : sseg_temp = 7'b0111111; //to display 0
            1'b1 : sseg_temp = 7'b0000110; //to display 1 //1111001
            default : sseg_temp = 7'b1000000; //dash
               endcase
               ssdcat=1'b0;
       end
       endcase
    end
    assign ssd=sseg_temp;
endmodule
```

Internal ssd

```
module sevenseg(
    input clk,
    input ones,
    input tens,
    input hundreds,
    input thousands,
    output [7:0] cathode,
    output [3:0] anode
    );
     reg [6:0]sseg_temp;
     reg [3:0]an_temp = 4'b1110 ;
     reg [17:0] count; //the 18 bit counter which allows us to multiplex at 1000Hz

     always @ (posedge clk)
      begin
        count <= count + 1;
      end

            //code for display multiple digits (do not initialize an_temp in line 41 and comment out
lines 44 to 60)
     always @ (*)
      begin
       case(count[17:16]) //using only the 2 MSB's of the counter

        2'b00 :  //When the 2 MSB's are 00 enable the fourth display
         begin
          case(ones)
               1'b0 : sseg_temp = 7'b0000001; //to display 0
               1'b1 : sseg_temp = 7'b1001111; //to display 1
               default : sseg_temp = 7'b1111110; //dash
           endcase

          //sseg = ones;
          an_temp = 4'b1110;
         end

        2'b01:  //When the 2 MSB's are 01 enable the third display
         begin
            case(tens)
               1'b0 : sseg_temp = 7'b0000001; //to display 0
               1'b1 : sseg_temp = 7'b1001111; //to display 1
               default : sseg_temp = 7'b1111110; //dash
            endcase

          //sseg = tens;
          an_temp = 4'b1101;
         end

        2'b10:  //When the 2 MSB's are 10 enable the second display
```

```
         begin
          case(hundreds)
               1'b0 : sseg_temp = 7'b0000001; //to display 0
               1'b1 : sseg_temp = 7'b1001111; //to display 1
           endcase

          //sseg = hundreds;
          an_temp = 4'b1011;
         end

        2'b11:  //When the 2 MSB's are 11 enable the first display
         begin
          case(thousands)
                1'b0 : sseg_temp = 7'b0000001; //to display 0
                1'b1 : sseg_temp = 7'b1001111; //to display 1
          endcase

          //sseg = thousands;
          an_temp = 4'b0111;
         end
       endcase
      end

     assign anode = an_temp;
     assign cathode = {sseg_temp, 1'b1};
 Endmodule
```

1 Hz frequency

```
module frq_one(
input clk,
output reg out_clk=0
);

reg [26:0] count=0;

always @ (posedge clk)
begin
    count<=count+1;
    if(count==50000000)
    begin
```

```
        count<=0;
        out_clk<=~out_clk;
    end
end

endmodule
```

Counter

```
module up_counter(
    input clk,
    output reg [5:0] q=0,
    input rst
    );

// always @ (posedge clk | rst) // asynchronous clk
always @ (posedge clk)  // synchronous clk
begin
    if(rst | q==63)
        q<=6'b0;
    else
        q<=q+1;
end

endmodule
```

Debounce

```
module debounce(
    input clk_deb,
    output deb_out,
    input pb_press
    );

    reg [2:0] ff;

    always @ (posedge clk_deb)
    begin
        ff[2]<=pb_press;
        ff[1]<=ff[2];
        ff[0]<=ff[1];
    end

    assign deb_out=ff[0]&ff[1]&ff[2];

endmodule
```

Frequency divider

```
module frq_div(
    input clk,
    output div_clk
    );

parameter n=25;

reg [n-1:0] q=0;

always @ (posedge clk)
begin

q<=q+1;

end

assign div_clk=q[n-1];

endmodule
```

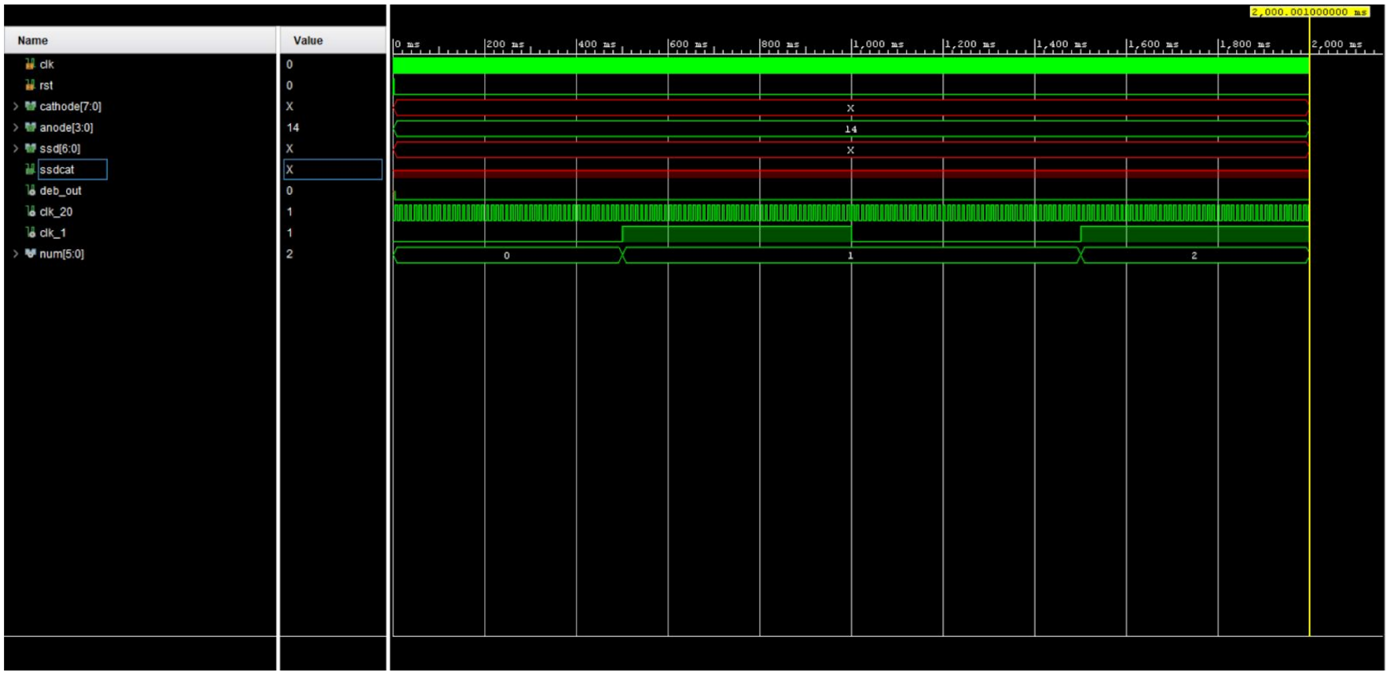
XDC file

```
# Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
            set_property IOSTANDARD LVCMOS33 [get_ports clk]

#7 segment display
    set_property PACKAGE_PIN W7 [get_ports {cathode[7]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {cathode[7]}]
```

```
   set_property PACKAGE_PIN W6 [get_ports {cathode[6]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {cathode[6]}]
   set_property PACKAGE_PIN U8 [get_ports {cathode[5]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {cathode[5]}]
   set_property PACKAGE_PIN V8 [get_ports {cathode[4]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {cathode[4]}]
   set_property PACKAGE_PIN U5 [get_ports {cathode[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {cathode[3]}]
   set_property PACKAGE_PIN V5 [get_ports {cathode[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {cathode[2]}]
   set_property PACKAGE_PIN U7 [get_ports {cathode[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {cathode[1]}]
   set_property PACKAGE_PIN V7 [get_ports cathode[0]]
      set_property IOSTANDARD LVCMOS33 [get_ports cathode[0]]

   set_property PACKAGE_PIN U2 [get_ports {anode[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {anode[0]}]
   set_property PACKAGE_PIN U4 [get_ports {anode[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {anode[1]}]
   set_property PACKAGE_PIN V4 [get_ports {anode[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {anode[2]}]
   set_property PACKAGE_PIN W4 [get_ports {anode[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {anode[3]}]

##Buttons
set_property PACKAGE_PIN U18 [get_ports rst]
        set_property IOSTANDARD LVCMOS33 [get_ports rst]

##Pmod Header JB
##Sch name = JB1
set_property PACKAGE_PIN A14 [get_ports {ssd[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {ssd[0]}]
##Sch name = JB2
set_property PACKAGE_PIN A16 [get_ports {ssd[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {ssd[1]}]
##Sch name = JB3
set_property PACKAGE_PIN B15 [get_ports {ssd[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {ssd[2]}]
##Sch name = JB4
set_property PACKAGE_PIN B16 [get_ports {ssd[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {ssd[3]}]

##Pmod Header JC
```

```
##Sch name = JC1
set_property PACKAGE_PIN K17 [get_ports {ssd[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {ssd[4]}]
##Sch name = JC2
set_property PACKAGE_PIN M18 [get_ports {ssd[5]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {ssd[5]}]
##Sch name = JC3
set_property PACKAGE_PIN N17 [get_ports {ssd[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {ssd[6]}]
##Sch name = JC4
set_property PACKAGE_PIN P18 [get_ports {ssdcat}]
        set_property IOSTANDARD LVCMOS33 [get_ports {ssdcat}]
```

Output



Name	Value
clk	0
rst	0
cathode[7:0]	X
anode[3:0]	14
ssd[6:0]	X
ssdcat	X
deb_out	0
clk_20	1
clk_1	1
num[5:0]	2