Lab 7

Top module

```
module top(
    input clk,
    input rst,
    input pb_read,
    output [1:0] led,
    output [7:0] cathode,
    output [3:0] anode
    );

    wire [3:0] thousands,hundreds,tens,ones;
    wire [3:0] dout1;
    reg [3:0] add=0;
    wire [4:0] dout2;
    reg [4:0] sum=0;
    reg i=0;
    reg [3:0] cntr=0;
    wire clk_190;
    wire deb_read,deb_rst;
    wire [4:0] fifo_out;

    frq_div #(19) fun1(clk,clk_190);

    pulse fun2(clk_190,deb_read,pb_read);
    pulse fun3(clk_190,deb_rst,rst);

    blk_mem_gen_1 fun5(.clka(clk),.addra(add),.douta(dout1));   // 0 1 2  3  4
5  6  7 8 9
```

```
    blk_mem_gen_2 fun6(.clka(clk),.addra(add),.douta(dout2));  //  7 0 17 18
7  8  19 0 0 0
                                                    //   7 1 19 21 11 13 25 7 8 9

    always @ (posedge clk)
    begin
        if(add<4'd12)
        begin
            sum<=dout1+dout2;
            i<=1;
            if(cntr<4'd3)
                add<=add;
            else
                add<=add+1;
        end
        else
        begin
            i<=0;
            cntr<=0;
        end

        cntr<=cntr+1;

    end

   fifo_generator fun7(.rst(deb_rst),
        .wr_clk(clk),.rd_clk(clk_190),

.din(sum),.wr_en(i),.rd_en(deb_read),.dout(fifo_out),.full(led[1]),.empty(led[
0]));

    bin2bcd fun8(fifo_out,thousands,hundreds,tens,ones);
```

```
    sevenseg
fun9(clk,deb_rst,ones,tens,hundreds,thousands,cathode,anode);

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
```

Pulse

```
module pulse(
    input clk,
    output deb_out,
    input deb_in
    );

    reg D1,D2,D3;
    always@(posedge clk)
    begin
        D1<=deb_in;
        D2<=D1;
        D3<=~D2; // inversion is done to generate a pulse
    end

    assign deb_out=D1&&D2&&D3;

endmodule
```

Bin to BCD

```
module bin2bcd(number, thousands, hundreds, tens, ones);
  // I/O Signal Definitions
  input [4:0] number;
  output reg [3:0] thousands;
  output reg [3:0] hundreds;
  output reg [3:0] tens;
  output reg [3:0] ones;

  // Internal variable for storing bits
```

```
  reg [19:0] shift;
  integer i;

  always @(number)
  begin
    // Clear previous number and store new number in shift register
    shift[19:8] = 0;
    shift[7:0] = number;

    // Loop eight times
    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 entire register left once
      shift = shift << 1;
    end

    // Push decimal numbers to output
    hundreds = shift[19:16];
    tens     = shift[15:12];
    ones     = shift[11:8];
    thousands= 4'b0000;

  end

Endmodule
```

Seven Segment

```
`timescale 1ns / 1ps
module sevenseg(
    input clk,
    input clr,              // clear pin used to reset the LED display
    input [3:0] ones,
    input [3:0] tens,
    input [3:0] hundreds,
    input [3:0] 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)
      4'd0 : sseg_temp = 7'b0000001; //to display 0
      4'd1 : sseg_temp = 7'b1001111; //to display 1
      4'd2 : sseg_temp = 7'b0010010; //to display 2
      4'd3 : sseg_temp = 7'b0000110; //to display 3
      4'd4 : sseg_temp = 7'b1001100; //to display 4
      4'd5 : sseg_temp = 7'b0100100; //to display 5
      4'd6 : sseg_temp = 7'b0100000; //to display 6
      4'd7 : sseg_temp = 7'b0001111; //to display 7
      4'd8 : sseg_temp = 7'b0000000; //to display 8
      4'd9 : sseg_temp = 7'b0000100; //to display 9
      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)
      4'd0 : sseg_temp = 7'b0000001; //to display 0
      4'd1 : sseg_temp = 7'b1001111; //to display 1
      4'd2 : sseg_temp = 7'b0010010; //to display 2
      4'd3 : sseg_temp = 7'b0000110; //to display 3
      4'd4 : sseg_temp = 7'b1001100; //to display 4
      4'd5 : sseg_temp = 7'b0100100; //to display 5
      4'd6 : sseg_temp = 7'b0100000; //to display 6
      4'd7 : sseg_temp = 7'b0001111; //to display 7
      4'd8 : sseg_temp = 7'b0000000; //to display 8
      4'd9 : sseg_temp = 7'b0000100; //to display 9
      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)
       4'd0 : sseg_temp = 7'b0000001; //to display 0
       4'd1 : sseg_temp = 7'b1001111; //to display 1
       4'd2 : sseg_temp = 7'b0010010; //to display 2
       4'd3 : sseg_temp = 7'b0000110; //to display 3
       4'd4 : sseg_temp = 7'b1001100; //to display 4
       4'd5 : sseg_temp = 7'b0100100; //to display 5
       4'd6 : sseg_temp = 7'b0100000; //to display 6
       4'd7 : sseg_temp = 7'b0001111; //to display 7
       4'd8 : sseg_temp = 7'b0000000; //to display 8
       4'd9 : sseg_temp = 7'b0000100; //to display 9
       default : sseg_temp = 7'b1111110; //dash
   endcase

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

2'b11:  //When the 2 MSB's are 11 enable the first display
 begin
  case(thousands)
       4'd0 : sseg_temp = 7'b0000001; //to display 0
       4'd1 : sseg_temp = 7'b1001111; //to display 1
       4'd2 : sseg_temp = 7'b0010010; //to display 2
       4'd3 : sseg_temp = 7'b0000110; //to display 3
       4'd4 : sseg_temp = 7'b1001100; //to display 4
```

```
            4'd5 : sseg_temp = 7'b0100100; //to display 5
            4'd6 : sseg_temp = 7'b0100000; //to display 6
            4'd7 : sseg_temp = 7'b0001111; //to display 7
            4'd8 : sseg_temp = 7'b0000000; //to display 8
            4'd9 : sseg_temp = 7'b0000100; //to display 9
            default : sseg_temp = 7'b1111110; //dash
        endcase

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

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

XDC File

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

        set_property IOSTANDARD LVCMOS33 [get_ports clk]

## LEDs
set_property PACKAGE_PIN U16 [get_ports {led[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
set_property PACKAGE_PIN E19 [get_ports {led[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
```

```
#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]
set_property PACKAGE_PIN T17 [get_ports pb_read]
       set_property IOSTANDARD LVCMOS33 [get_ports pb_read]
```

Output Waveform