LAB4
SARTHAK BHAGAT
2016189

TOP

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 09/23/2017 03:45:35 AM
// Design Name:
// Module Name: top
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module top(
        input cin,
        output reg[3:0] anode,
        output reg[7:0] cathode
    );
    wire [7:0] counter_output;
    wire [3:0] ones,tens,hundreds;
    wire [7:0] cathode1,cathode2,cathode3;

    initial anode = 4'b1110;

    one_hertz f1(.clk(cin),.outclk(cout));

    counter c1(.clk_in(cout),.count(counter_output));

    b2b b1(.number(counter_output),.ones(ones),.tens(tens),.hundreds(hundreds));
```

```
    fd #(18) f2(.cin(cin),.cout(cout2));

    seven_segment ss1(.ones(ones),.cathode(cathode1));
          seven_segment ss2(.ones(tens),.cathode(cathode2));
          seven_segment ss3(.ones(hundreds),.cathode(cathode3));

    always @(posedge cout2)
    begin
          case (anode)
            4'b1110 :
            begin
               anode = 4'b1101;
            cathode = cathode2;
            end
            4'b1101 :
            begin
               anode = 4'b1011;
            cathode = cathode3;
            end
            default :
            begin
               anode = 4'b1110;
            cathode = cathode1;
            end
          endcase
    end
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
```

COUNTER

```
module counter(
    input clk_in,
    output count
    );

    parameter width = 8;

    reg [width-1:0] count;

    always @(posedge clk_in)
    begin
        if (count<256)
        begin
        count <= count + 1;
        end
        else
        begin
        count <= 0;
        end
    end

endmodule
```

ONE HERTZ FREQUENCY

```
module one_hertz(
    input clk,
    output reg outclk
    );
    reg [26:0] count;
```

```
    always @(posedge clk)
        begin
            count <= count + 1;
            if (count == 50000000)
                begin
                    count <= 0;
                    outclk <= ~outclk;
                end
        end
endmodule
```