Mini Project 1:

Real time clock

The main aim of the project is to develop a real time clock generator using seven segment decoder and counter. In this project we need to develop the 6 decoders and counters so we are using the concept of instantiation.

Code:

//top module

module top;

reg clk,rst,en;

wire [6:0]sec\_l,sec\_m;

wire [6:0]min\_l,min\_m;

wire [6:0]hr\_l,hr\_m;

wire [3:0] countsec\_L,countsec\_M;

wire [3:0] countmin\_L,countmin\_M;

wire [3:0] counthr\_L,counthr\_M;

wire secm\_en, minl\_en , minm\_en , hrl\_en , hrm\_en;

wire clr;

counter #(9)

lsb\_sec(.clk(clk),.rst(rst),.en(en),.clr(clr),.count(countsec\_L));

sevenseg lsb\_out(.sel(countsec\_L),.out(sec\_l));

assign secm\_en = (countsec\_L == 4'b1001);

counter #(5)

msb\_sec(.clk(clk),.rst(rst),.en(secm\_en),.clr(clr),.count(countsec\_M));

sevenseg msb\_out(.sel(countsec\_M),.out(sec\_m));

assign minl\_en = (countsec\_M == 4'b0101 && countsec\_L == 4'b1001);

counter #(9)

lsb\_min(.clk(clk),.rst(rst),.en(minl\_en),.clr(clr),.count(countmin\_L));

sevenseg lsb\_outmin(.sel(countmin\_L),.out(min\_l));

assign minm\_en = (countmin\_L == 4'b1001 && countsec\_M == 4'b0101 && countsec\_L == 4'b1001);

counter #(5)

msb\_min(.clk(clk),.rst(rst),.en(minm\_en),.clr(clr),.count(countmin\_M));

sevenseg msb\_outmin(.sel(countmin\_M),.out(min\_m));

assign hrl\_en = (countmin\_M == 4'b0101 && countmin\_L == 4'b1001 && countsec\_M == 4'b0101 && countsec\_L == 4'b1001);

counter #(9)

lsb\_hr(.clk(clk),.rst(rst),.en(hrl\_en),.clr(clr),.count(counthr\_L));

sevenseg lsb\_outhr(.sel(counthr\_L),.out(hr\_l));

assign hrm\_en = (counthr\_L == 4'b1001 && countmin\_M == 4'b0101 && countmin\_L == 4'b1001 && countsec\_M == 4'b0101 && countsec\_L == 4'b1001);

counter #(2)

msb\_hr(.clk(clk),.rst(rst),.en(hrm\_en),.clr(clr),.count(counthr\_M));

sevenseg msb\_outhr(.sel(counthr\_M),.out(hr\_m));

assign clr = (counthr\_M == 4'b0010 && counthr\_L == 4'b0011 && countmin\_M == 4'b0101 && countmin\_L == 4'b1001 && countsec\_M == 4'b0101 && countsec\_L == 4'b1001);

initial

begin

clk = 0;

forever #5 clk = ~clk;

end

initial begin

#10 rst = 1;

#10 rst = 0;

#10 en=1;

#86500 $finish;

end

endmodule

//seven segment

module sevenseg(input [3:0] sel,

output reg [6:0] out);

always @(sel)

begin

case(sel)

4'b0000: out=7'b1111110;

4'b0001: out=7'b0110000;

4'b0010: out=7'b1101101;

4'b0011: out=7'b1111001;

4'b0100: out=7'b0110011;

4'b0101: out=7'b1011011;

4'b0110: out=7'b1011111;

4'b0111: out=7'b1110000;

4'b1000: out=7'b1111111;

4'b1001: out=7'b1111011;

default: out=7'bzzzzzzz;

endcase

end

endmodule

//counter

module counter #(parameter iterations=9)(input clk,rst,en,clr, output reg [3:0] count);

always @(posedge clk or posedge rst)

begin

if (rst) begin

count <= 4'b0000;

end else if (clr) begin

count <= 4'b0000;

end

else if (en) begin

if (count == iterations ) begin

count <= 4'b0000;

end else begin

count <= count + 1; end

end

/\*else

count <= count +1;\*/

end

endmodule