CS223 LAB5

Altay İlker Yiğitel

22203024

CS223 sec: 01

module traffic(

input SA,

input SB,

input reset,

output reg [5:0] current\_state,

input clk\_100MHz

);

reg[5:0] next\_state;

reg [25:0] counter\_reg = 0;

reg clk\_out\_reg = 0;

reg clk\_1Hz;

oneHz\_generator ohzz(.clk\_100MHz(clk\_100MHz),.clk\_1Hz(clk\_1Hz));

parameter BOTH\_RED1 = 6'b111111;

parameter BOTH\_RED2 = 6'b111111;

parameter LA\_GREEN = 6'b110111;

parameter LB\_GREEN = 6'b111110;

parameter LA\_YELLOW1 = 6'b100111;

parameter LB\_YELLOW1 = 6'b111100;

parameter LA\_YELLOW2 = 6'b111100;

parameter LB\_YELLOW2 = 6'b100111;

always@(posedge clk\_1Hz) begin

if(reset)begin

current\_state <= LA\_GREEN;

end else begin

current\_state <= next\_state;

end

end

always @(posedge clk\_1Hz)

case (current\_state)

LA\_GREEN: begin

if (SA)

next\_state <= current\_state;

else

next\_state <= LA\_YELLOW1;

end

LB\_GREEN: begin

if (SB)

next\_state <= current\_state;

else

next\_state <= LB\_YELLOW1;

end

BOTH\_RED1: begin

next\_state <= LA\_YELLOW2;

end

BOTH\_RED2: begin

next\_state <= LB\_YELLOW2;

end

LA\_YELLOW1: begin

next\_state <= BOTH\_RED2;

end

LB\_YELLOW1: begin

next\_state <= BOTH\_RED1;

end

LA\_YELLOW2: begin

next\_state <= LB\_GREEN;

end

LB\_YELLOW2: begin

next\_state <= LA\_GREEN;

end

endcase

endmodule

module BCD\_counter4(

input clk\_100MHz,

input res,

input lo,

input [9:0] in,

output [0:6] seg,

output [3:0] an

);

wire reset,load;

wire clk\_1Hz;

oneHz\_generator ohz(.clk\_100MHz(clk\_100MHz),.clk\_1Hz(clk\_1Hz));

btn\_debouncer bt(.clk\_100MHz(clk\_100MHz),.btn\_in(res),.btn\_out(reset));

btn\_debouncer btt(.clk\_100MHz(clk\_100MHz),.btn\_in(lo),.btn\_out(load));

reg [3:0] sec\_ones = 0;

reg [3:0] sec\_tens = 0;

reg [3:0] sec\_hundereds = 0;

reg [3:0] sec\_thousands = 0;

seg7\_control seg7(.clk\_100MHz(clk\_100MHz), .reset(res), .ones(sec\_ones), .tens(sec\_tens), .hundereds(sec\_hundereds),

.thousands(sec\_thousands), .seg(seg), .an(an));

always @(posedge clk\_1Hz or posedge reset or posedge load) begin

if(reset)begin

sec\_ones <= 0;

sec\_tens <= 0;

sec\_hundereds <= 0;

sec\_thousands <= 0;

end

else if(load)begin

sec\_ones <=in%10;

sec\_tens <=(in/10)%10;

sec\_hundereds <=(in/100)%10;

sec\_thousands <=(in/1000)%10;

end

else

if(sec\_ones == 9) begin

sec\_ones <= 0;

if(sec\_tens == 9)begin

sec\_tens <= 0;

if(sec\_hundereds ==9)begin

sec\_hundereds <= 0;

if(sec\_thousands == 9)

sec\_thousands <= 0;

else

sec\_thousands <= sec\_thousands+1;

end

else

sec\_hundereds <= sec\_hundereds+1;

end

else

sec\_tens <= sec\_tens+1;

end

else

sec\_ones <= sec\_ones + 1;

end

endmodule