**PARTE 3**

module parte3(SW,KEY,LEDG,LEDR);

input [1:0]SW;

input [0:0]KEY;

output [8:0]LEDR;

output [0:0]LEDG;

parameter[8:0] A = 9'b000000001,

parameter[8:0] B = 9'b000000010,

parameter[8:0] C = 9'b000000100,

parameter[8:0] D = 9'b000001000,

parameter[8:0] E = 9'b000010000,

parameter[8:0] F = 9'b000100000,

parameter[8:0] G = 9'b001000000,

parameter[8:0] H = 9'b010000000,

parameter[8:0] I = 9'b100000000;

reg [8:0] estado\_atual, prox\_estado;

always @(posedge KEY[0])

if (~SW[0])

estado\_atual <= A;

else

estado\_atual <= prox\_estado;

always @(estado\_atual,SW[1])

case(estado\_atual)

A: if(SW[1])

prox\_estado = F;

else

prox\_estado = B;

B: if(SW[1])

prox\_estado = F;

else

prox\_estado = C;

C: if(SW[1])

prox\_estado = F;

else

prox\_estado = D;

D: if(SW[1])

prox\_estado = F;

else

prox\_estado = E;

E: if(SW[1])

prox\_estado = F;

else

prox\_estado = E;

F: if(SW[1])

prox\_estado = G;

else

prox\_estado = B;

G: if(SW[1])

prox\_estado = H;

else

prox\_estado = B;

H: if(SW[1])

prox\_estado = I;

else

prox\_estado = B;

I: if(SW[1])

prox\_estado = I;

else

prox\_estado = B;

endcase

assign LEDG[0] = (estado\_atual == (E|I));

endmodule;

**PARTE 4**

module ps2lab1(

input CLOCK\_50,

input [3:0] KEY;

input [17:0] SW;

output [6:0] LCD;

output [8:0] LEDG;

output [17:0] LEDR;

input PS2\_DAT,

input PS2\_CLK,

inout [35:0] GPIO\_0, GPIO\_1

assign GPIO\_0 = 36'hzzzzzzzzz;

assign GPIO\_1 = 36'hzzzzzzzzz;

wire RST;

assign RST = KEY[0];

assign LEDR[17:0] = SW[17:0];

assign LEDG = 0;

wire reset = 1'b0;

wire [7:0] scan\_code;

reg [7:0] history[1:4];

wire read, scan\_ready;

oneshot pulser(

.pulse\_out(read),

.trigger\_in(scan\_ready),

.clk(CLOCK\_50)

);

keyboard kbd(

.keyboard\_clk(PS2\_CLK),

.keyboard\_data(PS2\_DAT),

.clock50(CLOCK\_50),

.reset(reset),

.read(read),

.scan\_ready(scan\_ready),

.scan\_code(scan\_code)

);

lcd dsp0(history[1][3:0],HEX0);

always @(posedge scan\_ready)

begin

history[4] <= history[3];

history[3] <= history[2];

history[2] <= history[1];

history[1] <= scan\_code;

end

endmodule;

module LCDmodule(clk, RxD, LCD\_RS, LCD\_RW, LCD\_E, LCD\_DataBus);

input clk, RxD;

output LCD\_RS, LCD\_RW, LCD\_E;

output [7:0] LCD\_DataBus;

wire RxD\_data\_ready;

wire [7:0] RxD\_data;

async\_receiver deserialer(.clk(clk), .RxD(RxD), .RxD\_data\_ready(RxD\_data\_ready), .RxD\_data(RxD\_data));

assign LCD\_RS = RxD\_data[7];

assign LCD\_DataBus = {1'b0, RxD\_data[6:0]}; // sends only 7 bits to the module, padded with a '0' in front to make 8 bits

assign LCD\_RW = 0;

reg [2:0] count;

always @(posedge clk) if(RxD\_data\_ready | (count!=0)) count <= count + 1;

reg LCD\_E;

always @(posedge clk) LCD\_E <= (count!=0);

endmodule