**PARTE 1**

module parte1 (SW, KEY, LEDR, LEDG);

input [17:0] SW;

input [3:0] KEY;

output [17:0] LEDR;

output [8:0] LEDG;

wire [8:0] D, Y;

wire w;

assign w = SW[1];

my\_lpm\_ff FF (~KEY[1], KEY[0], D[8:0], Y[8:0]);

assign D[0] = ~(~(~Y[0] | Y[1] | Y[2] | Y[3] | Y[4] | Y[5] | Y[6] | Y[7] | Y[8]) | ~(D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] | D[8]));

assign D[1] = (~Y[0] | Y[5] | Y[6] | Y[7] | Y[8]) & ~w; // B

assign D[2] = Y[1] & ~w; // C

assign D[3] = Y[2] & ~w; // D

assign D[4] = (Y[3] | Y[4]) & ~w; // E

assign D[5] = (~Y[0] | Y[1] | Y[2] | Y[3] | Y[4]) & w; // F

assign D[6] = Y[5] & w; // G

assign D[7] = Y[6] & w; // H

assign D[8] = (Y[7] | Y[8]) & w; // I

assign LEDG = Y;

assign LEDR[8:0] = D;

endmodule

**PARTE 2**

module parte2 (SW, KEY, LEDR, LEDG);

input [17:0] SW;

input [3:0] KEY;

output [17:0] LEDR;

output [8:0] LEDG;

wire w;

assign w = SW[1];

wire Clock;

assign Clock = KEY[0];

reg z;

reg [3:0] y\_Q, Y\_D;

parameter A = 4'b0000, B = 4'b0001, C = 4'b0010, D = 4'b0011, E = 4'b0100, F = 4'b0101,

G = 4'b0110, H = 4'b0111, I = 4'b1000;

always @(w, y\_Q)

begin: state\_table

case (y\_Q)

A: if (!w) Y\_D = B;

else Y\_D = F;

B: if (!w) Y\_D = C;

else Y\_D = F;

C: if (!w) Y\_D = D;

else Y\_D = F;

D: if (!w) Y\_D = E;

else Y\_D = F;

E: if (!w) Y\_D = E;

else Y\_D = F;

F: if (w) Y\_D = G;

else Y\_D = B;

G: if (w) Y\_D = H;

else Y\_D = B;

H: if (w) Y\_D = I;

else Y\_D = B;

I: if (w) Y\_D = I;

else Y\_D = B;

default: Y\_D = 4'bxxxx;

endcase

end // state\_table

always @(posedge Clock)

begin: state\_FFs

y\_Q = Y\_D;

end // state\_FFS

always

begin: zset

case (y\_Q)

E: z = 1;

I: z = 1;

default: z = 0;

endcase

end

assign LEDG[3:0] = y\_Q;

assign LEDR[3:0] = Y\_D;

assign LEDR[17] = z;

test t1 (SW[17], SW[16], LEDR[17]);

endmodule

module test (A, B, Out);

input A, B;

output reg Out;

always

case (A)

0: Out = 0;

1: Out = B;

endcase

endmodule

**PARTE 2.2**

module parte22 (SW, KEY, LEDR, LEDG);

input [17:0] SW;

input [3:0] KEY;

output [17:0] LEDR;

output [8:0] LEDG;

wire w;

assign w = KEY[1];

wire Clock;

assign Clock = KEY[0];

reg z;

reg [3:0] y\_Q, Y\_D; parameter A = 4'b0000, B = 4'b0001, C = 4'b0010, D = 4'b0011,

E = 4'b0100, F = 4'b0101, G = 4'b0110, H = 4'b0111, I = 4'b1000;

always @(w, y\_Q)

begin: state\_table

case (y\_Q)

A: if (!w) Y\_D = B;

else Y\_D = F;

B: if (!w) Y\_D = C;

else Y\_D = F;

C: if (!w) Y\_D = D;

else Y\_D = F;

D: if (!w) Y\_D = E;

else Y\_D = F;

E: if (!w) Y\_D = E;

else Y\_D = F;

F: if (w) Y\_D = G;

else Y\_D = B;

G: if (w) Y\_D = H;

else Y\_D = B;

H: if (w) Y\_D = I;

else Y\_D = B;

I: if (w) Y\_D = I;

else Y\_D = B;

default: Y\_D = 4'bxxxx;

endcase

end // state\_table

always @(posedge (Clock && KEY[1]))

begin: state\_FFs

y\_Q = Y\_D;

end // state\_FFS

always

begin: zset

case (y\_Q)

E: z = 1;

I: z = 1;

default: z = 0;

endcase

end

assign LEDG[3:0] = y\_Q;

assign LEDR[3:0] = Y\_D;

assign LEDR[17] = z;

endmodule

module test (A, B, Out);

input A, B;

output reg Out;

always

case (A)

0: Out = 0;

1: Out = B;

endcase

endmodule

**PARTE 3**

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

input [17:0] SW;

input [3:0] KEY;

output [17:0] LEDR;

output [8:0] LEDG;

wire w;

assign w = SW[1];

wire Clock;

assign Clock = KEY[0];

reg z;

wire s0Clock, s1Clock, s0In, s1In, s0Clr, s1Clr;

wire [3:0] s0Q, s1Q;

assign s0Clock = (/\*w &\*/ ~s0Q[3] | ~w) & ~Clock;

assign s1Clock = (/\*~w &\*/ ~s1Q[3] | w) & ~Clock;

assign s0Clr = ~w;

assign s1Clr = w;

assign s0In = ~(s0Q[3] | s0Q[2] | s0Q[1] | s0Q[0]) & w;

assign s1In = ~(s1Q[3] | s1Q[2] | s1Q[1] | s1Q[0]) & ~w;

lpm\_shiftreg\_4bit S0 (s0Clock, s0Clr, s0In, s0Q);

lpm\_shiftreg\_4bit S1 (s1Clock, s1Clr, s1In, s1Q);

always

begin: zset

if (s0Q[3] == 1 | s1Q[3] == 1)

z = 1;

else

z = 0;

end

assign LEDG[3:0] = s0Q;

assign LEDR[3:0] = s1Q;

assign LEDR[17] = z;

Endmodule

**PARTE 4**

module parte4 (SW, KEY, LEDR, LEDG, HEX0);

input [17:0] SW;

input [3:0] KEY;

output [17:0] LEDR;

output [8:0] LEDG;

output [0:6] HEX0;

reg [3:0] state;

wire [1:0] w;

assign w[0] = SW[1];

assign w[1] = SW[2];

always @ (negedge KEY[0])

begin

if (SW[0])

state = 0;

else begin

case (w)

1: state = state + 1;

2: state = state + 2;

3: if(state == 0)

state = 9;

else

state = state - 1;

endcase

if (state == 10)

state = 0;

else if (state == 11)

state = 1;

end

end

b2d\_ssd H0 (state, HEX0);

endmodule

**PARTE 5**

module parte5 (SW, KEY, LEDR, LEDG, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7);

input [17:0] SW;

input [3:0] KEY;

output [17:0] LEDR;

output [8:0] LEDG;

output [0:6] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7;

reg [3:0] state;

reg [0:6] H0, H1, H2, H3, H4, H5, H6, H7, H8;

always @ (negedge KEY[0])

begin

if (~SW[0]) begin

H8 = H7;

H7 = H6;

H6 = H5;

H5 = H4;

H4 = H3;

H3 = H2;

H2 = H1;

H1 = H0;

case (state)

0: H0 = 7'b1001000;

1: H0 = 7'b0110000;

2: H0 = 7'b1110001;

3: H0 = 7'b1110001;

4: H0 = 7'b0000001;

5: H0 = 7'b1111111;

6: H0 = 7'b1111111;

7: H0 = 7'b1111111;

8: H0 = H8;

endcase

if (state < 8) begin

state = state + 1;

end

end else begin

state = 0;

H0 = 7'b1111111;

H8 = 7'b1111111;

H7 = 7'b1111111;

H6 = 7'b1111111;

H5 = 7'b1111111;

H4 = 7'b1111111;

H3 = 7'b1111111;

H2 = 7'b1111111;

H1 = 7'b1111111;

end

end

assign HEX0 = H0;

assign HEX1 = H1;

assign HEX2 = H2;

assign HEX3 = H3;

assign HEX4 = H4;

assign HEX5 = H5;

assign HEX6 = H6;

assign HEX7 = H7;

assign LEDG[3:0] = state;

endmodule

**PARTE 6**

module parte6 (CLOCK\_50, SW, KEY, LEDR, LEDG, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7);

input CLOCK\_50;

input [17:0] SW;

input [3:0] KEY;

output [17:0] LEDR;

output [8:0] LEDG;

output [0:6] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7;

reg [3:0] state;

reg [0:6] H0, H1, H2, H3, H4, H5, H6, H7, H8;

wire [25:0] counter;

counter\_modk C0 (CLOCK\_50, 1, counter);

defparam C0.n = 26;

defparam C0.k = 50000000;

always @ (posedge counter[25])

begin

if (~SW[0]) begin

H8 = H7;

H7 = H6;

H6 = H5;

H5 = H4;

H4 = H3;

H3 = H2;

H2 = H1;

H1 = H0;

case (state)

0: H0 = 7'b1001000;

1: H0 = 7'b0110000;

2: H0 = 7'b1110001;

3: H0 = 7'b1110001;

4: H0 = 7'b0000001;

5: H0 = 7'b1111111;

6: H0 = 7'b1111111;

7: H0 = 7'b1111111;

8: H0 = H8;

endcase

if (state < 8) begin

state = state + 1;

end

end else begin

state = 0;

H0 = 7'b1111111;

H8 = 7'b1111111;

H7 = 7'b1111111;

H6 = 7'b1111111;

H5 = 7'b1111111;

H4 = 7'b1111111;

H3 = 7'b1111111;

H2 = 7'b1111111;

H1 = 7'b1111111;

end

end

assign HEX0 = H0;

assign HEX1 = H1;

assign HEX2 = H2;

assign HEX3 = H3;

assign HEX4 = H4;

assign HEX5 = H5;

assign HEX6 = H6;

assign HEX7 = H7;

assign LEDG[3:0] = state;

endmodule

**PARTE 7**

module parte7 (CLOCK\_50, SW, KEY, LEDR, LEDG, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7);

input CLOCK\_50;

input [17:0] SW;

input [3:0] KEY;

output [17:0] LEDR;

output [8:0] LEDG;

output [0:6] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7;

reg [3:0] state;

reg [0:6] H0, H1, H2, H3, H4, H5, H6, H7, H8;

wire [22:0] counter;

wire [4:0] counter2;

reg trigger;

counter\_modk C0 (CLOCK\_50, 1, counter);

defparam C0.n = 23;

defparam C0.k = 6250000;

counter\_modk C1 (counter[22], 1, counter2);

defparam C1.n = 5;

defparam C1.k = 32;

always @ (posedge trigger)begin

if (~SW[0]) begin

H8 = H7;

H7 = H6;

H6 = H5;

H5 = H4;

H4 = H3;

H3 = H2;

H2 = H1;

H1 = H0;

case (state)

0: H0 = 7'b1001000;

1: H0 = 7'b0110000;

2: H0 = 7'b1110001;

3: H0 = 7'b1110001;

4: H0 = 7'b0000001;

5: H0 = 7'b1111111;

6: H0 = 7'b1111111;

7: H0 = 7'b1111111;

8: H0 = H8;

endcase

if (state < 8) begin

state = state + 1;

end

end else begin

state = 0;

H0 = 7'b1111111;

H8 = 7'b1111111;

H7 = 7'b1111111;

H6 = 7'b1111111;

H5 = 7'b1111111;

H4 = 7'b1111111;

H3 = 7'b1111111;

H2 = 7'b1111111;

H1 = 7'b1111111;

end

end

assign HEX0 = H0;

assign HEX1 = H1;

assign HEX2 = H2;

assign HEX3 = H3;

assign HEX4 = H4;

assign HEX5 = H5;

assign HEX6 = H6;

assign HEX7 = H7;

assign LEDG[3:0] = state;

reg [3:0] y\_Q, Y\_D;

parameter A = 4'b0000,

B = 4'b0001,

C = 4'b0010,

D = 4'b0011,

E = 4'b0100,

F = 4'b0101,

G = 4'b0110,

H = 4'b0111,

I = 4'b1000;

always @(KEY[0], SW[0], KEY[1], y\_Q) begin:

state\_table

case (y\_Q)

A: if (SW[0]) Y\_D = C;

else if (~KEY[0]) Y\_D = B;

else if (~KEY[1]) Y\_D = A;

B: if (SW[0]) Y\_D = C;

else if (~KEY[0]) Y\_D = C;

else if (~KEY[1]) Y\_D = A;

C: if (SW[0]) Y\_D = C;

else if (~KEY[0]) Y\_D = D;

else if (~KEY[1]) Y\_D = B;

D: if (SW[0]) Y\_D = C;

else if (~KEY[0]) Y\_D = E;

else if (~KEY[1]) Y\_D = C;

E: if (SW[0]) Y\_D = C;

else if (~KEY[0]) Y\_D = E;

else if (~KEY[1]) Y\_D = D;

default: Y\_D = 4'bxxxx;

endcase

end

always @(posedge (KEY[0] && KEY[1]))begin:

state\_FFs y\_Q = Y\_D;

end

assign LEDR[3:0] = y\_Q;

reg [27:0] z;

parameter z1 = 28'b0000101111101011110000100000,

z2 = 28'b0001011111010111100001000000,

z3 = 28'b0010111110101111000010000000,

z4 = 28'b0101111101011110000100000000,

z5 = 28'b1011111010111100001000000000;

always @(counter2, y\_Q)begin

case (y\_Q)

A: trigger = counter2[0];

B: trigger = counter2[1];

C: trigger = counter2[2];

D: trigger = counter2[3];

E: trigger = counter2[4];

default: trigger = 0;

endcase

end

assign LEDR[17:13] = counter2;

endmodule