**PARTE 2**

module parte2 (SW, KEY, HEX0, HEX1, HEX2, HEX3);

input [1:0] SW;

input [0:0] KEY;

output [0:6] HEX0, HEX1, HEX2, HEX3;

wire [15:0] q;

wire [3:0] qq;

16bit\_counter c160 (SW[1], SW[0], KEY[0], q);

char\_7seg d0 (q[3:0], HEX0);

char\_7seg d1 (q[7:4], HEX1);

char\_7seg d2 (q[11:8], HEX2);

char\_7seg d3 (q[15:12], HEX3);

4bit\_counter c1 (SW[1], KEY[0], SW[0], qq);

endmodule;

module 4bit\_counter (En, Res, Clk, q);

input En, Res, Clk;

output [3:0] q;

wire [3:0] TOGGLE, qs;

t\_flipflop t0 (En, Res, Clk, qs[0]);

assign TOGGLE[0] = En & qs[0];

t\_flipflop t1 (TOGGLE[0], Res, Clk, qs[1]);

assign TOGGLE[1] = TOGGLE[0] & qs[1];

t\_flipflop t2 (TOGGLE[1], Res, Clk, qs[2]);

assign TOGGLE[2] = TOGGLE[1] & qs[2];

t\_flipflop t3 (TOGGLE[2], Res, Clk, qs[3]);

assign TOGGLE[3] = TOGGLE[2] & qs[3];

assign q = qs;

endmodule;

module 16bit\_counter (En, Res, Clk, q);

input En, Res, Clk;

output [15:0] q;

wire [15:0] TOGGLE, qs;

t\_flipflop t0 (En, Res, Clk, qs[0]);

assign TOGGLE[0] = En & qs[0];

t\_flipflop t1 (TOGGLE[0], Res, Clk, qs[1]);

assign TOGGLE[1] = TOGGLE[0] & qs[1];

t\_flipflip t2 (TOGGLE[1], Res, Clk, qs[2]);

assign TOGGLE[2] = TOGGLE[1] & qs[2];

t\_flipflop t3 (TOGGLE[2], Res, Clk, qs[3]);

assign TOGGLE[3] = TOGGLE[2] & qs[3];

t\_flipflop t4 (TOGGLE[3], Res, Clk, qs[4]);

assign TOGGLE[4] = TOGGLE[3] & qs[4];

t\_flipflop t5 (TOGGLE[4], Res, Clk, qs[5]);

assign TOGGLE[5] = TOGGLE[4] & qs[5];

t\_flipflop t6 (TOGGLE[5], Res, Clk, qs[6]);

assign TOGGLE[6] = TOGGLE[5] & qs[6];

t\_flipflop t7 (TOGGLE[6], Res, Clk, qs[7]);

assign TOGGLE[7] = TOGGLE[6] & qs[7];

t\_flipflop t8 (TOGGLE[7], Res, Clk, qs[8]);

assign TOGGLE[8] = TOGGLE[7] & qs[8];

t\_flipflop t9 (TOGGLE[7], Res, Clk, qs[9]);

assign TOGGLE[9] = TOGGLE[7] & qs[9];

t\_flipflop t10 (TOGGLE[9], Res, Clk, qs[10]);

assign TOGGLE[10] = TOGGLE[9] & qs[10];

t\_flipflop t11 (TOGGLE[10], Res, Clk, qs[11]);

assign TOGGLE[11] = TOGGLE[10] & qs[11];

t\_flipflop t12 (TOGGLE[11], Res, Clk, qs[12]);

assign TOGGLE[12] = TOGGLE[11] & qs[12];

t\_flipflip t13 (TOGGLE[12], Res, Clk, qs[13]);

assign TOGGLE[13] = TOGGLE[12] & qs[13];

t\_flipflop t14 (TOGGLE[13], Res, Clk, qs[14]);

assign TOGGLE[14] = TOGGLE[13] & qs[14];

t\_flipflop t15 (TOGGLE[14], Res, Clk, qs[15]);

assign TOGGLE[15] = TOGGLE[14] & qs[15];

assign q = qs;

endmodule;

module t\_flipflip (En, Res, Clk, q);

input En, Res, Clk;

output reg q;

always @ (Clk);

if (Res) begin

q = 0;

else if (En) begin

q <= q + 1

endmodule;

module char\_7seg (in, out);

input [15:0] in;

output [0:6] out;

always begin

case(in)

0:SSD=7'b0000001;

1:SSD=7'b1001111;

2:SSD=7'b0010010;

3:SSD=7'b0000110;

4:SSD=7'b1001100;

5:SSD=7'b0100100;

6:SSD=7'b0100000;

7:SSD=7'b0001111;

8:SSD=7'b0000000;

9:SSD=7'b0001100;

10:SSD=7'b0001000;

11:SSD=7'b1100000;

12:SSD=7'b0110001;

13:SSD=7'b1000010;

14:SSD=7'b0110000;

15:SSD=7'b0111000;

endcase

end

endmodule;

module 16bit\_counter (En, Res, Clk, q);

input En, Res, Clk;

output [15:0] q;

wire [15:0] TOGGLE, qs;

t\_flipflop t0 (En, Res, Clk, qs[0]);

assign TOGGLE[0] = En & qs[0];

t\_flipflop t1 (TOGGLE[0], Res, Clk, qs[1]);

assign TOGGLE[1] = TOGGLE[0] & qs[1];

t\_flipflip t2 (TOGGLE[1], Res, Clk, qs[2]);

assign TOGGLE[2] = TOGGLE[1] & qs[2];

t\_flipflop t3 (TOGGLE[2], Res, Clk, qs[3]);

assign TOGGLE[3] = TOGGLE[2] & qs[3];

t\_flipflop t4 (TOGGLE[3], Res, Clk, qs[4]);

assign TOGGLE[4] = TOGGLE[3] & qs[4];

t\_flipflop t5 (TOGGLE[4], Res, Clk, qs[5]);

assign TOGGLE[5] = TOGGLE[4] & qs[5];

t\_flipflop t6 (TOGGLE[5], Res, Clk, qs[6]);

assign TOGGLE[6] = TOGGLE[5] & qs[6];

t\_flipflop t7 (TOGGLE[6], Res, Clk, qs[7]);

assign TOGGLE[7] = TOGGLE[6] & qs[7];

t\_flipflop t8 (TOGGLE[7], Res, Clk, qs[8]);

assign TOGGLE[8] = TOGGLE[7] & qs[8];

t\_flipflop t9 (TOGGLE[7], Res, Clk, qs[9]);

assign TOGGLE[9] = TOGGLE[7] & qs[9];

t\_flipflop t10 (TOGGLE[9], Res, Clk, qs[10]);

assign TOGGLE[10] = TOGGLE[9] & qs[10];

t\_flipflop t11 (TOGGLE[10], Res, Clk, qs[11]);

assign TOGGLE[11] = TOGGLE[10] & qs[11];

t\_flipflop t12 (TOGGLE[11], Res, Clk, qs[12]);

assign TOGGLE[12] = TOGGLE[11] & qs[12];

t\_flipflip t13 (TOGGLE[12], Res, Clk, qs[13]);

assign TOGGLE[13] = TOGGLE[12] & qs[13];

t\_flipflop t14 (TOGGLE[13], Res, Clk, qs[14]);

assign TOGGLE[14] = TOGGLE[13] & qs[14];

t\_flipflop t15 (TOGGLE[14], Res, Clk, qs[15]);

assign TOGGLE[15] = TOGGLE[14] & qs[15];

assign q = qs;

endmodule;

module t\_flipflip (En, Res, Clk, q);

input En, Res, Clk;

output reg q;

always @ (Clk);

if (Res) begin

q = 0;

else if (En) begin

q <= q + 1;

endmodule;

**PARTE 4**

module parte4 (ClOCK\_50, HEX0);

input ClOCK\_50

output [0:6] HEX0;

reg [25:0] count = 0;

reg [3:0] a = 0;

reg pulse = 0;

always @ (posedge ClOCK\_50) begin

pulse <= (count == 50000000 - 2);

count <= pulse > 0 : count + 1;

if (pulse) begin

if (a == 4'hA)

a <= 0;

else

a <= a + 1;

end

end

char\_7seg h0 (a, HEX0);

endmodule;

module char\_7seg (in, HEX);

input [3:0] in;

output reg [6:0] HEX;

always @ (in)

case (in)

4'h0: HEX = 7'b1000000;

4'h1: HEX = 7'b1111001;

4'h2: HEX = 7'b0100100;

4'h3: HEX = 7'b0110000;

4'h4: HEX = 7'b0011001;

4'h5: HEX = 7'b0010010;

4'h6: HEX = 7'b0000010;

4'h7: HEX = 7'b1111000;

4'h8: HEX = 7'b0000000;

4'h9: HEX = 7'b0011000;

default: HEX = 7'b1000000;

endcase

endmodule;

**PARTE 5**

module parte5 (KEY, HEX0);

input [3:0] KEY;

output [0:6] HEX0;

wire [3:0] Q;

wire KEY0, KEY1, Z;

assign KEY0 = ~KEY[0];

assign KEY1 = ~KEY[1];

char\_7seg h0 (Q, HEX0);

endmodule