**Họ và tên: Hà Minh Khuê MSSV: 1513079**

**Bài tập tuần 5**

**Bài 1:**

module cau1(SW, KEY, HEX0);

    input [0:0]SW;

    input [0:0]KEY;

    output [6:0]HEX0;

    wire t2, t3;

    wire [3:0]q;

    FF\_T(SW[0], KEY[0], q[0]);

    FF\_T(q[0], KEY[0], q[1]);

    and(t2, q[1], q[0]);

    FF\_T(t2, KEY[0], q[2]);

    and(t3, q[2], t2);

    FF\_T(t3, KEY[0], q[3]);

    Decoder\_HEX(q, HEX0);

endmodule

module FF\_T(t, clk, q);

    input t, clk;

    output q;

    wire m1, m2, d;

    and(m1, ~t, q);

    and(m2, t, ~q);

    or(d, m1, m2);

    FF\_D(d, clk, q);

endmodule

module FF\_D(d, clk, q);

    input d, clk;

    output reg q;

    always @(posedge clk)

        q <= d;

endmodule

module Decoder\_HEX(c,HEX);

    input[3:0]c;

    output [6:0]HEX;

    reg[6:0]HEX;

    always@(c)

        case(c)

        4'b0000: HEX = 7'b1000000;

        4'b0001: HEX = 7'b1111001;

        4'b0010: HEX = 7'b0100100;

        4'b0011: HEX = 7'b0110000;

        4'b0100: HEX = 7'b0011001;

        4'b0101: HEX = 7'b0010010;

        4'b0110: HEX = 7'b0000010;

        4'b0111: HEX = 7'b1111000;

        4'b1000: HEX = 7'b0000000;

        4'b1001: HEX = 7'b0010000;

        4'b1010: HEX = 7'b0001000; //A

        4'b1011: HEX = 7'b0000011; //B

        4'b1100: HEX = 7'b1000110; //C

        4'b1101: HEX = 7'b0100001; //D

        4'b1110: HEX = 7'b0000110; //E

        4'b1111: HEX = 7'b0001110; //F

        default: HEX = 7'b1111111;

    endcase

endmodule

**Bài 2:**

module cau2(SW, KEY, HEX0);

    input [0:0]SW; // T

    input [1:0]KEY; // clk = key0, rst = key1

    output [6:0]HEX0;

    wire [3:0]q, t;

    FF\_T(SW[0], KEY[1], KEY[0], q[0]);

    and(t[1], SW[0], q[0]);

    FF\_T(t[1], KEY[1], KEY[0], q[1]);

    and(t[2], t[1], q[1]);

    FF\_T(t[2], KEY[1], KEY[0], q[2]);

    and(t[3], t[2], q[2]);

    FF\_T(t[3], KEY[1], KEY[0], q[3]);

    Decoder\_HEX(q, HEX0);

endmodule

module FF\_T(t, rst, clk, q);

    input t, rst, clk;

    output reg q;

    always@(posedge clk or negedge rst)

        if(!rst)

            q <= 1'b0;

        else

            if(t)

                q <= ~q;

endmodule

module Decoder\_HEX(c,HEX);

    input[3:0]c;

    output [6:0]HEX;

    reg[6:0]HEX;

    always@(c)

        case(c)

        4'b0000: HEX = 7'b1000000;

        4'b0001: HEX = 7'b1111001;

        4'b0010: HEX = 7'b0100100;

        4'b0011: HEX = 7'b0110000;

        4'b0100: HEX = 7'b0011001;

        4'b0101: HEX = 7'b0010010;

        4'b0110: HEX = 7'b0000010;

        4'b0111: HEX = 7'b1111000;

        4'b1000: HEX = 7'b0000000;

        4'b1001: HEX = 7'b0010000;

        4'b1010: HEX = 7'b0001000; //A

        4'b1011: HEX = 7'b0000011; //B

        4'b1100: HEX = 7'b1000110; //C

        4'b1101: HEX = 7'b0100001; //D

        4'b1110: HEX = 7'b0000110; //E

        4'b1111: HEX = 7'b0001110; //F

        default: HEX = 7'b1111111;

    endcase

endmodule

**Bài 3:**

module cau3(SW, KEY, HEX0, HEX1);

    input [0:0]SW; // T

    input [1:0]KEY; // clk = key0, rst = key1

    output [6:0]HEX0, HEX1;

    wire [7:0]q, t;

    // 4 High bits

    FF\_T(SW[0], KEY[1], KEY[0], q[0]);

    and(t[1], SW[0], q[0]);

    FF\_T(t[1], KEY[1], KEY[0], q[1]);

    and(t[2], t[1], q[1]);

    FF\_T(t[2], KEY[1], KEY[0], q[2]);

    and(t[3], t[2], q[2]);

    FF\_T(t[3], KEY[1], KEY[0], q[3]);

    // 4 Low bits

    and(t[4], t[3], q[3]);

    FF\_T(t[4], KEY[1], KEY[0], q[4]);

    and(t[5], t[4], q[4]);

    FF\_T(t[5], KEY[1], KEY[0], q[5]);

    and(t[6], t[5], q[5]);

    FF\_T(t[6], KEY[1], KEY[0], q[6]);

    and(t[7], t[6], q[6]);

    FF\_T(t[7], KEY[1], KEY[0], q[7]);

    Decoder\_HEX(q[3:0], HEX0);

    Decoder\_HEX(q[7:4], HEX1);

endmodule

module FF\_T(t, rst, clk, q);

    input t, rst, clk;

    output reg q;

    always@(posedge clk or negedge rst)

        if(!rst)

            q <= 1'b0;

        else

            if(t)

                q <= ~q;

endmodule

module Decoder\_HEX(c,HEX);

    input[3:0]c;

    output [6:0]HEX;

    reg[6:0]HEX;

    always@(c)

        case(c)

        4'b0000: HEX = 7'b1000000;

        4'b0001: HEX = 7'b1111001;

        4'b0010: HEX = 7'b0100100;

        4'b0011: HEX = 7'b0110000;

        4'b0100: HEX = 7'b0011001;

        4'b0101: HEX = 7'b0010010;

        4'b0110: HEX = 7'b0000010;

        4'b0111: HEX = 7'b1111000;

        4'b1000: HEX = 7'b0000000;

        4'b1001: HEX = 7'b0010000;

        4'b1010: HEX = 7'b0001000; //A

        4'b1011: HEX = 7'b0000011; //B

        4'b1100: HEX = 7'b1000110; //C

        4'b1101: HEX = 7'b0100001; //D

        4'b1110: HEX = 7'b0000110; //E

        4'b1111: HEX = 7'b0001110; //F

        default: HEX = 7'b1111111;

    endcase

endmodule

**Bài 4:**

module cau4(SW, KEY, HEX0);

    input [0:0]SW; // T

    input [1:0]KEY; // clk = key0, rst = key1

    output [6:0]HEX0;

    wire [3:0]q, t;

    FF\_T(SW[0], KEY[1], KEY[0], q[0]);

    and(t[1], SW[0], ~q[0]);

    FF\_T(t[1], KEY[1], KEY[0], q[1]);

    and(t[2], t[1], ~q[1]);

    FF\_T(t[2], KEY[1], KEY[0], q[2]);

    and(t[3], t[2], ~q[2]);

    FF\_T(t[3], KEY[1], KEY[0], q[3]);

    Decoder\_HEX(q, HEX0);

endmodule

module FF\_T(t, rst, clk, q);

    input t, rst, clk;

    output reg q;

    always@(posedge clk or negedge rst)

        if(!rst)

            q <= 1'b0;

        else

            if(t)

                q <= ~q;

endmodule

module Decoder\_HEX(c,HEX);

    input[3:0]c;

    output [6:0]HEX;

    reg[6:0]HEX;

    always@(c)

        case(c)

        4'b0000: HEX = 7'b1000000;

        4'b0001: HEX = 7'b1111001;

        4'b0010: HEX = 7'b0100100;

        4'b0011: HEX = 7'b0110000;

        4'b0100: HEX = 7'b0011001;

        4'b0101: HEX = 7'b0010010;

        4'b0110: HEX = 7'b0000010;

        4'b0111: HEX = 7'b1111000;

        4'b1000: HEX = 7'b0000000;

        4'b1001: HEX = 7'b0010000;

        4'b1010: HEX = 7'b0001000; //A

        4'b1011: HEX = 7'b0000011; //B

        4'b1100: HEX = 7'b1000110; //C

        4'b1101: HEX = 7'b0100001; //D

        4'b1110: HEX = 7'b0000110; //E

        4'b1111: HEX = 7'b0001110; //F

        default: HEX = 7'b1111111;

    endcase

endmodule

**Bài 5:**

module cau5(SW, KEY, HEX0);

    input [0:0]SW;

    input [1:0]KEY;

    output [6:0]HEX0;

    wire [3:0]bcd;

    counter(SW[0], KEY[1], KEY[0], bcd);

    Decoder\_HEX(bcd, HEX0);

endmodule

module counter(en, rst, clk, bcd);

    parameter n = 4;

    input en, rst, clk;

    output reg [n-1:0]bcd;

    always@(posedge clk or negedge rst)

        if(!rst)

            bcd <= 0;

        else if(en)

            bcd <= bcd + 1;

endmodule

module Decoder\_HEX(c,HEX);

    input[3:0]c;

    output [6:0]HEX;

    reg[6:0]HEX;

    always@(c)

        case(c)

        4'b0000: HEX = 7'b1000000;

        4'b0001: HEX = 7'b1111001;

        4'b0010: HEX = 7'b0100100;

        4'b0011: HEX = 7'b0110000;

        4'b0100: HEX = 7'b0011001;

        4'b0101: HEX = 7'b0010010;

        4'b0110: HEX = 7'b0000010;

        4'b0111: HEX = 7'b1111000;

        4'b1000: HEX = 7'b0000000;

        4'b1001: HEX = 7'b0010000;

        4'b1010: HEX = 7'b0001000; //A

        4'b1011: HEX = 7'b0000011; //B

        4'b1100: HEX = 7'b1000110; //C

        4'b1101: HEX = 7'b0100001; //D

        4'b1110: HEX = 7'b0000110; //E

        4'b1111: HEX = 7'b0001110; //F

        default: HEX = 7'b1111111;

    endcase

endmodule

**Bài 6:**

module cau6(SW, KEY, HEX0);

    input [0:0]SW;

    input [1:0]KEY;

    output [6:0]HEX0;

    wire [3:0]bcd;

    counter(SW[0], KEY[1], KEY[0], bcd);

    Decoder\_HEX(bcd, HEX0);

endmodule

module counter(en, rst, clk, bcd);

    parameter n = 4;

    input en, rst, clk;

    output reg [n-1:0]bcd;

    always@(posedge clk or negedge rst)

        if(!rst)

            bcd <= 0;

        else if(en)

            bcd <= bcd - 1;

endmodule

module Decoder\_HEX(c,HEX);

    input[3:0]c;

    output [6:0]HEX;

    reg[6:0]HEX;

    always@(c)

        case(c)

        4'b0000: HEX = 7'b1000000;

        4'b0001: HEX = 7'b1111001;

        4'b0010: HEX = 7'b0100100;

        4'b0011: HEX = 7'b0110000;

        4'b0100: HEX = 7'b0011001;

        4'b0101: HEX = 7'b0010010;

        4'b0110: HEX = 7'b0000010;

        4'b0111: HEX = 7'b1111000;

        4'b1000: HEX = 7'b0000000;

        4'b1001: HEX = 7'b0010000;

        4'b1010: HEX = 7'b0001000; //A

        4'b1011: HEX = 7'b0000011; //B

        4'b1100: HEX = 7'b1000110; //C

        4'b1101: HEX = 7'b0100001; //D

        4'b1110: HEX = 7'b0000110; //E

        4'b1111: HEX = 7'b0001110; //F

        default: HEX = 7'b1111111;

    endcase

endmodule

**Bài 7:**

module cau7(SW, KEY, HEX0);

    input [1:0]SW;

    input [1:0]KEY;

    output [6:0]HEX0;

    wire [3:0]bcd;

    counter(SW[0], SW[1], KEY[1], KEY[0], bcd);

    Decoder\_HEX(bcd, HEX0);

endmodule

module counter(en, s, rst, clk, bcd);

    parameter n = 4;

    input s, en, rst, clk;

    output reg [n-1:0]bcd;

    always@(posedge clk or negedge rst)

        if(!rst)

            bcd <= 0;

        else if(en)

            bcd <= bcd + (s?1:-1);

endmodule

module Decoder\_HEX(c,HEX);

    input[3:0]c;

    output [6:0]HEX;

    reg[6:0]HEX;

    always@(c)

        case(c)

        4'b0000: HEX = 7'b1000000;

        4'b0001: HEX = 7'b1111001;

        4'b0010: HEX = 7'b0100100;

        4'b0011: HEX = 7'b0110000;

        4'b0100: HEX = 7'b0011001;

        4'b0101: HEX = 7'b0010010;

        4'b0110: HEX = 7'b0000010;

        4'b0111: HEX = 7'b1111000;

        4'b1000: HEX = 7'b0000000;

        4'b1001: HEX = 7'b0010000;

        4'b1010: HEX = 7'b0001000; //A

        4'b1011: HEX = 7'b0000011; //B

        4'b1100: HEX = 7'b1000110; //C

        4'b1101: HEX = 7'b0100001; //D

        4'b1110: HEX = 7'b0000110; //E

        4'b1111: HEX = 7'b0001110; //F

        default: HEX = 7'b1111111;

    endcase

endmodule

**Bài 8:**

module cau8(SW, KEY, HEX0, HEX1);

    input [1:0]SW;

    input [1:0]KEY;

    output [6:0]HEX0;

    output [6:0]HEX1;

    wire [7:0]bcd;

    counter(SW[0], SW[1], KEY[1], KEY[0], bcd);

    Decoder\_HEX(bcd[3:0], HEX0);

    Decoder\_HEX(bcd[7:4], HEX1);

endmodule

module counter(en, s, rst, clk, bcd);

    parameter n = 8;

    input s, en, rst, clk;

    output reg [n-1:0]bcd;

    always@(posedge clk or negedge rst)

        if(!rst)

            bcd <= 0;

        else if(en)

            bcd <= bcd + (s?1:-1);

endmodule

module Decoder\_HEX(c,HEX);

    input[3:0]c;

    output [6:0]HEX;

    reg[6:0]HEX;

    always@(c)

        case(c)

        4'b0000: HEX = 7'b1000000;

        4'b0001: HEX = 7'b1111001;

        4'b0010: HEX = 7'b0100100;

        4'b0011: HEX = 7'b0110000;

        4'b0100: HEX = 7'b0011001;

        4'b0101: HEX = 7'b0010010;

        4'b0110: HEX = 7'b0000010;

        4'b0111: HEX = 7'b1111000;

        4'b1000: HEX = 7'b0000000;

        4'b1001: HEX = 7'b0010000;

        4'b1010: HEX = 7'b0001000; //A

        4'b1011: HEX = 7'b0000011; //B

        4'b1100: HEX = 7'b1000110; //C

        4'b1101: HEX = 7'b0100001; //D

        4'b1110: HEX = 7'b0000110; //E

        4'b1111: HEX = 7'b0001110; //F

        default: HEX = 7'b1111111;

    endcase

endmodule