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

**Bài tập tuần 10**

**Bài 1:**

*module* cau1(SW, LEDR);

    input [2:0]SW;

    output [0:0]LEDR;

    wire f0;

    //Cau a

    //mux2\_1(SW[2], ~SW[2], SW[1], f0);

    //mux2\_1(f0, ~f0, SW[0], LEDR[0]);

    //Cau b

    mux2\_1(SW[1]&SW[2], SW[1]|SW[2], SW[0], LEDR[0]);

*endmodule*

*module* mux2\_1(x, y, s, m\_out);

    input x, y, s;

    output m\_out;

    assign m\_out = s?y:x;

*endmodule*

**Bài 2:**

*module* cau2(SW, LEDR);

    input [3:0]SW;

    output [3:0]LEDR;

    demux1\_4(SW[0], SW[2:1], SW[3], LEDR);

*endmodule*

*module* demux1\_4(d, s, en, f);

    input d, en;

    input [1:0]s;

    output reg [3:0]f;

    always@(en, s)

    if(~en)

        begin

            f[0] = 0;

            f[1] = 0;

            f[2] = 0;

            f[3] = 0;

        end

    else

        begin

            //always@(s)

            case(s)

                2'b00: f[0] = d;

                2'b01: f[1] = d;

                2'b10: f[2] = d;

                2'b11: f[3] = d;

            endcase

        end

*endmodule*

**Bài 3:**

*module* cau3(SW, LEDR, HEX0);

    input [0:9]SW;

    output [3:0]LEDR;

    output [6:0]HEX0;

    decode\_10\_4(SW, LEDR);

    display\_0\_9(LEDR, HEX0);

*endmodule*

*module* decode\_10\_4(i, f);

    input [9:0]i;

    output reg [3:0]f;

    always@(i)

    case(i)

        10'b1000000000: f = 4'b0000;

        10'b0100000000: f = 4'b0001;

        10'b0010000000: f = 4'b0010;

        10'b0001000000: f = 4'b0011;

        10'b0000100000: f = 4'b0100;

        10'b0000010000: f = 4'b0101;

        10'b0000001000: f = 4'b0110;

        10'b0000000100: f = 4'b0111;

        10'b0000000010: f = 4'b1000;

        10'b0000000001: f = 4'b1001;

    endcase

*endmodule*

*module* display\_0\_9(c,hex);

    input [3:0]c;

    output 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;

            default :hex =7'b1111111;

        endcase

*endmodule*

**Bài 4:**

*module* cau4(SW, HEX0, HEX1);

    input [3:0]SW;

    output [6:0] HEX0, HEX1;

    wire [3:0]d1, d0;

    bin\_to\_dec(SW, d1, d0);

    display\_0\_9(d1, HEX1);

    display\_0\_9(d0, HEX0);

*endmodule*

*module* bin\_to\_dec(b, d1, d0);

    input [3:0]b;

    output reg [3:0]d1, d0;

    always@(b)

    case(b)

        4'b0000:

            begin

                d0 = 0;

                d1 = 0;

            end

        4'b0001:

            begin

                d0 = 1;

                d1 = 0;

            end

        4'b0010:

            begin

                d0 = 2;

                d1 = 0;

            end

        4'b0011:

            begin

                d0 = 3;

                d1 = 0;

            end

        4'b0100:

            begin

                d0 = 4;

                d1 = 0;

            end

        4'b0101:

            begin

                d0 = 5;

                d1 = 0;

            end

        4'b0110:

            begin

                d0 = 6;

                d1 = 0;

            end

        4'b0111:

            begin

                d0 = 7;

                d1 = 0;

            end

        4'b1000:

            begin

                d0 = 8;

                d1 = 0;

            end

        4'b1001:

            begin

                d0 = 0;

                d1 = 0;

            end

        4'b1010:

            begin

                d0 = 0;

                d1 = 1;

            end

        4'b1011:

            begin

                d0 = 1;

                d1 = 1;

            end

        4'b1100:

            begin

                d0 = 2;

                d1 = 1;

            end

        4'b1101:

            begin

                d0 = 3;

                d1 = 1;

            end

        4'b1110:

            begin

                d0 = 4;

                d1 = 1;

            end

        4'b1111:

            begin

                d0 = 5;

                d1 = 1;

            end

    endcase

*endmodule*

*module* display\_0\_9(c,hex);

    input [3:0]c;

    output reg [6:0]hex;

    always@(c)

        case(c)

            0 :hex =7'b1000000;

            1 :hex =7'b1111001;

            2 :hex =7'b0100100;

            3 :hex =7'b0110000;

            4 :hex =7'b0011001;

            5 :hex =7'b0010010;

            6 :hex =7'b0000010;

            7 :hex =7'b1111000;

            8 :hex =7'b0000000;

            9 :hex =7'b0010000;

            default :hex =7'b1111111;

        endcase

*endmodule*

**Bài 5:**

*module* cau5(SW,LEDR);

    input [8:0] SW;

    output [4:0] LEDR;

    add4bit(SW[4], SW[3:0], SW[8:5], LEDR[3:0], LEDR[4]);

*endmodule*

*module* add4bit(cin,a,b,s,c);

    input cin;

    input [3:0]a,b;

    output [3:0]s;

    output [4:1] c;

    fulladder(cin, a[0], b[0], s[0], c[1]);

    fulladder(c[1:1], a[1:1], b[1:1], s[1:1], c[2:2]);

    fulladder(c[2:2], a[2:2], b[2:2], s[2:2], c[3:3]);

    fulladder(c[3:3], a[3:3], b[3:3], s[3:3], c[4:4]);

*endmodule*

*module* fulladder(cin,a,b,s,cout);

    input cin,a,b;

    output s,cout;

    assign s=a^b^cin;

    assign cout=(a&b)|(cin&a)|(cin&b);

*endmodule*

**Bài 6:**

*module* cau6(SW,LEDR);

    input [8:0]SW;

    output [4:0]LEDR;

    fulladder4bit (SW[8], SW[3:0], SW[7:4], LEDR[3:0], LEDR[4]);

*endmodule*

*module* fulladder4bit(cin, a[3:0], b[3:0], s[3:0], cout);

    parameter n=4;

    input [n-1:0]a,b;

    input cin;

    output reg [n-1:0]s;

    output reg cout;

    reg [n:0]c;

    integer k;

    always @(a,b,cin)

    begin

        c[0]=cin;

        for (k=0;k<n;k=k+1)

            begin

                s[k]=a[k]^b[k]^c[k];

                c[k+1]=(a[k]&b[k])|(a[k]&c[k])|(c[k]&b[k]);

            end

        cout = c[n];

    end

*endmodule*

**Bài 7:**

*module* cau2(SW, KEY, LEDR);

    input [9:0]SW; // SW[3:0] = D0, SW[7:4] = D1, SW[8] = s

    input [0:0]KEY;

    output wire [3:0]LEDR; // LEDR[3:0] = q

    wire [3:0]d;

    mux\_21\_4b(SW[3:0], SW[7:4], SW[9], d);

    FF\_D\_4b(d, KEY[0], LEDR[3:0]);

*endmodule*

*module* mux\_21\_4b(x, y, s, m);

    input [3:0]x, y;

    input s;

    output [3:0]m;

    assign m = s?y:x;

*endmodule*

*module* FF\_D\_4b(d, clk, q);

    input [3:0]d;

    input clk;

    output reg [3:0]q;

    always @(posedge clk)

        q <= d;

*endmodule*

**Bài 8:**

*module* cau8(SW, KEY, LEDR);

    input [9:9]SW;

    input [0:0]KEY; // t = SW9, clk = KEY0

    output [0:0]LEDR; // q = LEDR

    FF\_T(SW[9], KEY[0], LEDR[0]);

*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*

**Bài 9:**

*module* cau9(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*

**Bài 10:**

*module* cau10(CLOCK\_50,KEY,SW,HEX0, HEX1);

    input CLOCK\_50;

    input [0:0]KEY;

    input [1:0]SW;

    output [6:0]HEX0, HEX1;

    wire [3:0]a,b;

        counter (CLOCK\_50, KEY[0],SW[0],SW[1], a,b);

        display\_0\_9 (a, HEX0);

        display\_0\_9 (b, HEX1);

*endmodule*

*module* counter(CLOCK\_50,reset,enable,select,BCD0, BCD1);

    input reset,CLOCK\_50,enable,select;

    output reg[3:0]BCD0, BCD1;

    reg [25:0]delay;

    //tao delay 1s

    always @(posedge CLOCK\_50)

        begin

            if(delay==49999999)

                delay<=0;

            else

            delay=delay+1;

        end

    always@(posedge CLOCK\_50 or negedge reset)

        if(!reset)

            begin

                BCD0<=9;

                BCD1<=9;

            end

        else

        if (enable)

        begin

            if(select==0)

            begin

            if(delay==0)

                    begin

                        if(BCD0==0)

                            begin

                                BCD0<=9;

                                    if(BCD1==0)

                                        BCD1<=9;

                                    else

                                        BCD1<=BCD1-1;

                            end

                        else

                            BCD0<=BCD0-1;

                    end

                    end

                    else

                        begin

            if(delay==0)

                    begin

                        if(BCD0==9)

                            begin

                                BCD0<=0;

                                    if(BCD1==9)

                                        BCD1<=0;

                                    else

                                        BCD1<=BCD1+1;

                                end

                        else

                            BCD0<=BCD0+1;

                    end

                    end

                    end

*endmodule*

*module* display\_0\_9(c,hex);

    input [3:0]c;

    output 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;

            default :hex =7'b1111111;

        endcase

*endmodule*