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

**Bài tập tuần 7**

**Bài 1:**

*module* cau1(SW,LEDR);

    input [2:0]SW;

    output [3:0]LEDR;

    wire [3:0]A = 4'b0110;

    wire [3:0]B = 4'b0101;

    alu(A, B, SW[2:0], LEDR);

*endmodule*

*module* alu(A, B, opcode, ALU\_OUT);

    input [2:0]opcode;

    input wire [3:0]A,B;

    output reg [3:0]ALU\_OUT;

    always@(\*)

    case (opcode)

        3'b000: ALU\_OUT = ~A;

        3'b001: ALU\_OUT = A & B;

        3'b010: ALU\_OUT = A | B;

        3'b100: ALU\_OUT = A ^ B;

        default: ALU\_OUT = 4'b0000;

    endcase

*endmodule*

**Bài 2:**

*module* cau2(SW,LEDR,HEX0);

    input [2:0]SW;

    output [3:0]LEDR;

    output [6:0]HEX0;

    wire [3:0]A = 4'b0110;

    wire [3:0]B = 4'b0101;

    alu(A, B, SW[2:0], LEDR);

    decoder(LEDR, HEX0);

*endmodule*

*module* alu(A, B, opcode, ALU\_OUT);

    input [2:0]opcode;

    input wire [3:0]A,B;

    output reg [3:0]ALU\_OUT;

    always@(\*)

    case (opcode)

        3'b000: ALU\_OUT = ~A;

        3'b001: ALU\_OUT = A & B;

        3'b010: ALU\_OUT = A | B;

        3'b100: ALU\_OUT = A ^ B;

        default: ALU\_OUT = 4'b0000;

    endcase

*endmodule*

*module* decoder(c,hex);

    input [3:0]c;

    output reg [6:0]hex;

    always@(c)

    case (c)

        4'b 0000: hex = 7'b 1000000;

        4'b 0001: hex = 7'b 1111001;

        4'b 0010: hex = 7'b 0100100;

        4'b 0011: hex = 7'b 0110000;

        4'b 0100: hex = 7'b 0011001;

        4'b 0101: hex = 7'b 0010010;

        4'b 0110: hex = 7'b 0000010;

        4'b 0111: hex = 7'b 1111000;

        4'b 1000: hex = 7'b 0000000;

        4'b 1001: hex = 7'b 0010000;

    endcase

*endmodule*

**Bài 3:**

*module* cau3(SW,LEDR,HEX0,HEX1);

    input [2:0]SW;

    output [7:0]LEDR;

    output [6:0]HEX0,HEX1;

    wire [7:0]A = 8'b01010110;

    wire [7:0]B = 8'b10110101;

    alu(A, B, SW[2:0], LEDR[7:0]);

    decoder(LEDR[3:0], HEX0);

    decoder(LEDR[7:4], HEX1);

*endmodule*

*module* alu(A, B, opcode, ALU\_OUT);

    input [2:0]opcode;

    input wire [7:0]A,B;

    output reg [7:0]ALU\_OUT;

    always@(\*)

    case (opcode)

        3'b000: ALU\_OUT = ~A;

        3'b001: ALU\_OUT = A & B;

        3'b010: ALU\_OUT = A | B;

        3'b100: ALU\_OUT = A ^ B;

        default: ALU\_OUT = 4'b00000000;

    endcase

*endmodule*

*module* decoder(c,hex);

    input [3:0]c;

    output reg [6:0]hex;

    always@(c)

    case (c)

        4'b 0000: hex = 7'b 1000000;

        4'b 0001: hex = 7'b 1111001;

        4'b 0010: hex = 7'b 0100100;

        4'b 0011: hex = 7'b 0110000;

        4'b 0100: hex = 7'b 0011001;

        4'b 0101: hex = 7'b 0100100;

        4'b 0110: hex = 7'b 0100000;

        4'b 0111: hex = 7'b 1111000;

        4'b 1000: hex = 7'b 0000000;

        4'b 1001: hex = 7'b 0011000;

        4'b 1010: hex = 7'b 0001000;

        4'b 1011: hex = 7'b 0000011;

        4'b 1100: hex = 7'b 1000110;

        4'b 1101: hex = 7'b 0100001;

        4'b 1110: hex = 7'b 0000110;

        4'b 1111: hex = 7'b 0001110;

    endcase

*endmodule*

**Bài 4:**

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

    input [5:0]SW;

    output [7:0]LEDR;

    output [6:0]HEX0,HEX1;

    wire [7:0]x=8'b01010110;

    wire [7:0]y=8'b10110101;

    alu(SW[4:0],SW[5],x,y,LEDR);

    decoder(LEDR[3:0],HEX0);

    decoder(LEDR[7:4],HEX1);

*endmodule*

*module* alu(sel,car\_in,A,B,Y);

    input [4:0]sel;

    input car\_in;

    input wire [7:0]A,B;

    output reg [7:0]Y;

    reg [7:0]alu\_logic, alu\_arith;

    always @ (sel or A or B or car\_in)

    begin

    case (sel[1:0])

        2'b00:alu\_logic=A&B;

        2'b01:alu\_logic=A|B;

        2'b10:alu\_logic=A^B;

        2'b11:alu\_logic=!A;

    endcase

    case ({sel[1:0],car\_in})

        3'b000:alu\_arith=A;

        3'b001:alu\_arith=A+1;

        3'b010:alu\_arith=A+B;

        3'b011:alu\_arith=A+B+1;

        3'b100:alu\_arith= A+!B;

        3'b101:alu\_arith=A-!B+1;

        3'b110:alu\_arith=A-1;

        3'b111:alu\_arith=A;

    endcase

    if(sel[2])

        Y = alu\_logic;

    case(sel[4:3])

        2'b00: Y=alu\_arith;

        2'b01: Y<= (A<<1);

        2'b10: Y<= (A>>1);

        2'b11: Y<=0;

    endcase

    end

*endmodule*

*module* decoder (bcd, HEX);

input [3:0] bcd;

output reg [6:0]HEX;

always @(bcd)

case (bcd)

        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;

        4'b1011: HEX = 7'b0000011;

        4'b1100: HEX = 7'b1000110;

        4'b1101: HEX = 7'b0100001;

        4'b1110: HEX = 7'b0000110;

        4'b1111: HEX = 7'b0001110;

        default: HEX = 7'b1111111;

    endcase

*endmodule*