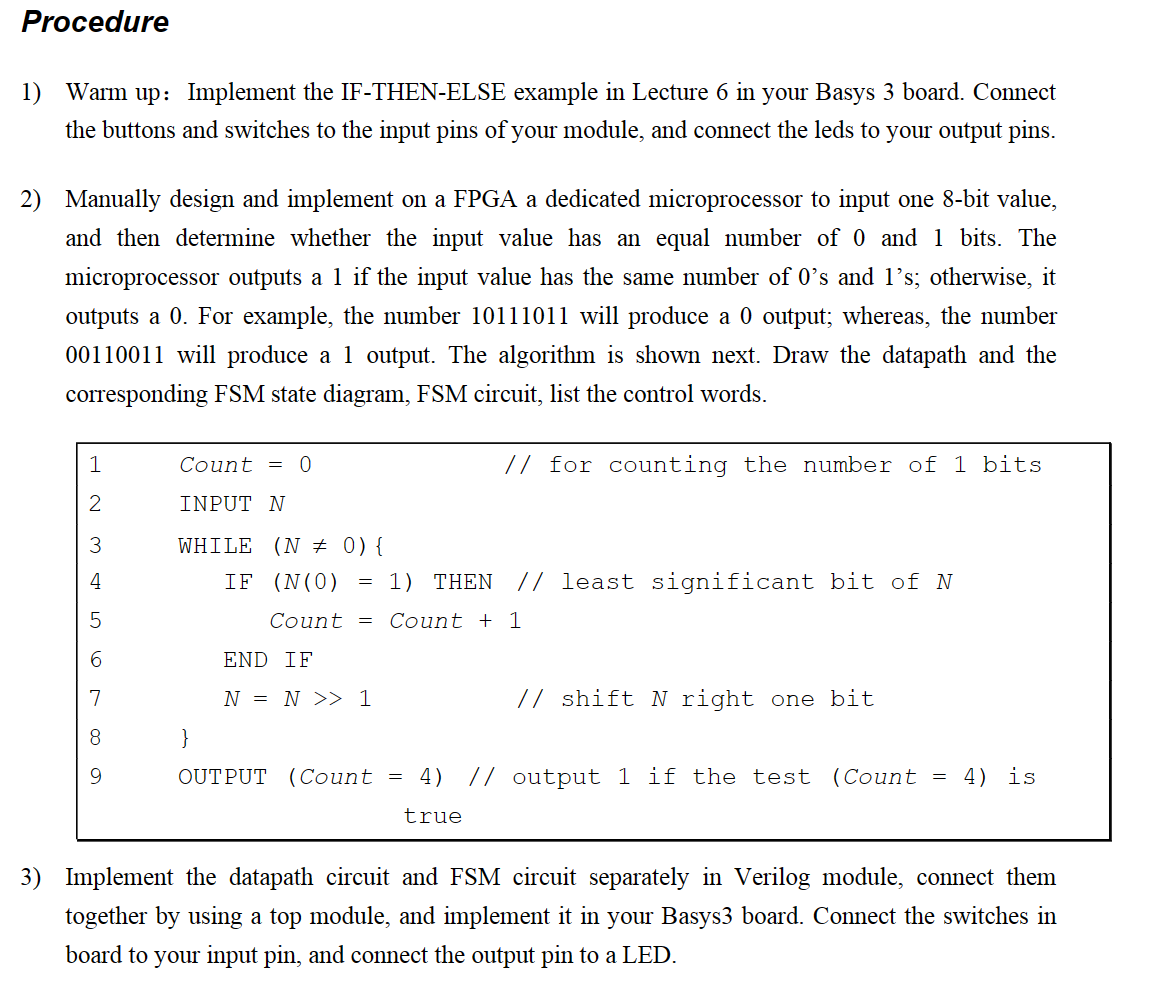
**1 实验目的和要求**

Goal To learn how to implement a dedicated microprocessor.



**2 实验内容**

（分点阐述实验步骤）

1. 编写Verilog，实现要求一。
2. 上手动设计数据通路，控制字和状态机。
3. 根据数据通路，控制字和状态机，编写Verilog代码实现要求三。

**3 源代码与分析**

**Warmup：代码如下，运行结构正常。**

module if\_then\_else(

    input clk,

    input [3:0]A,

    output reg [3:0]B

    );

    always @ (posedge clk) begin

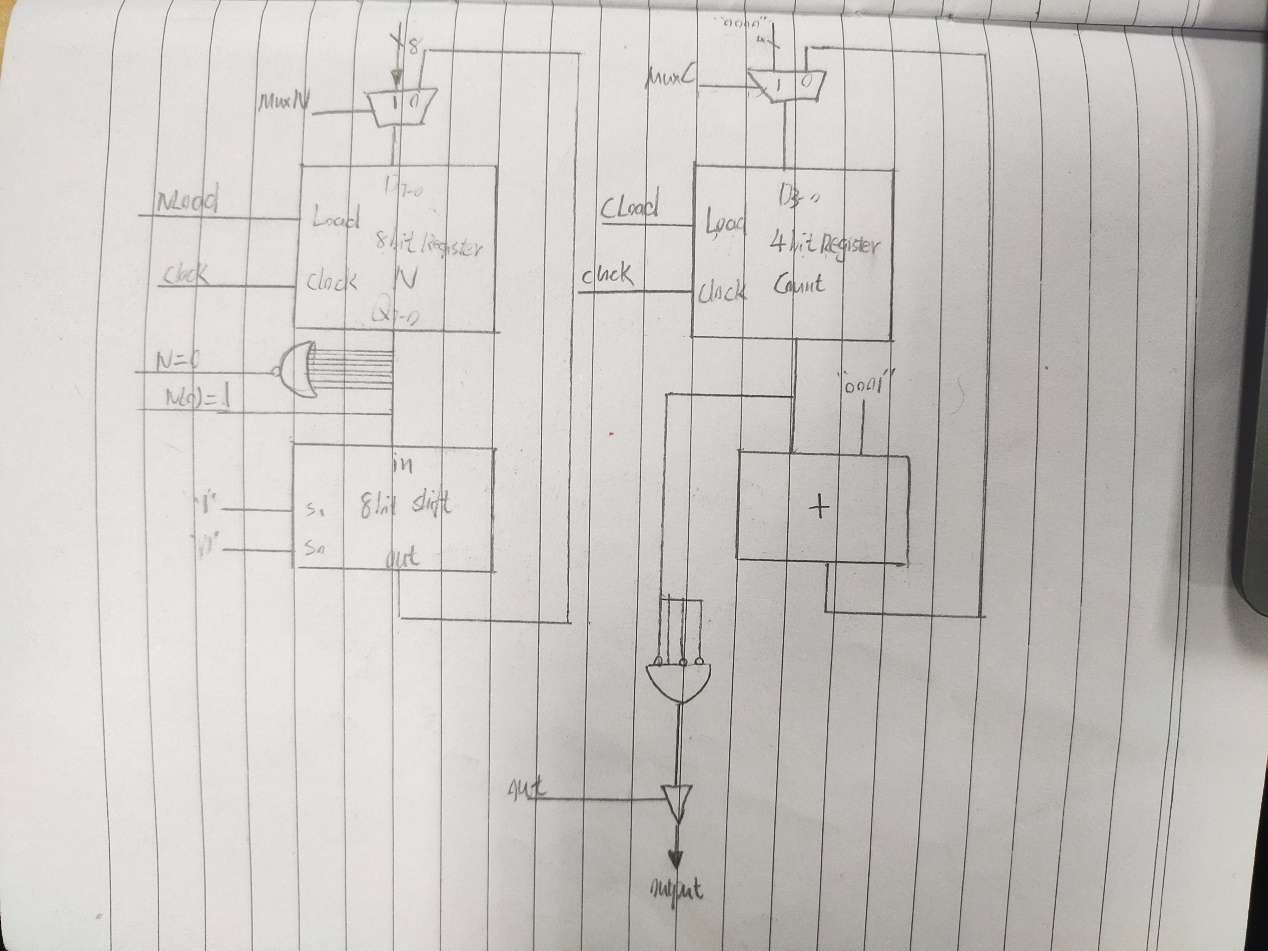
        if(A==4'b0101) b<=4'B1000;

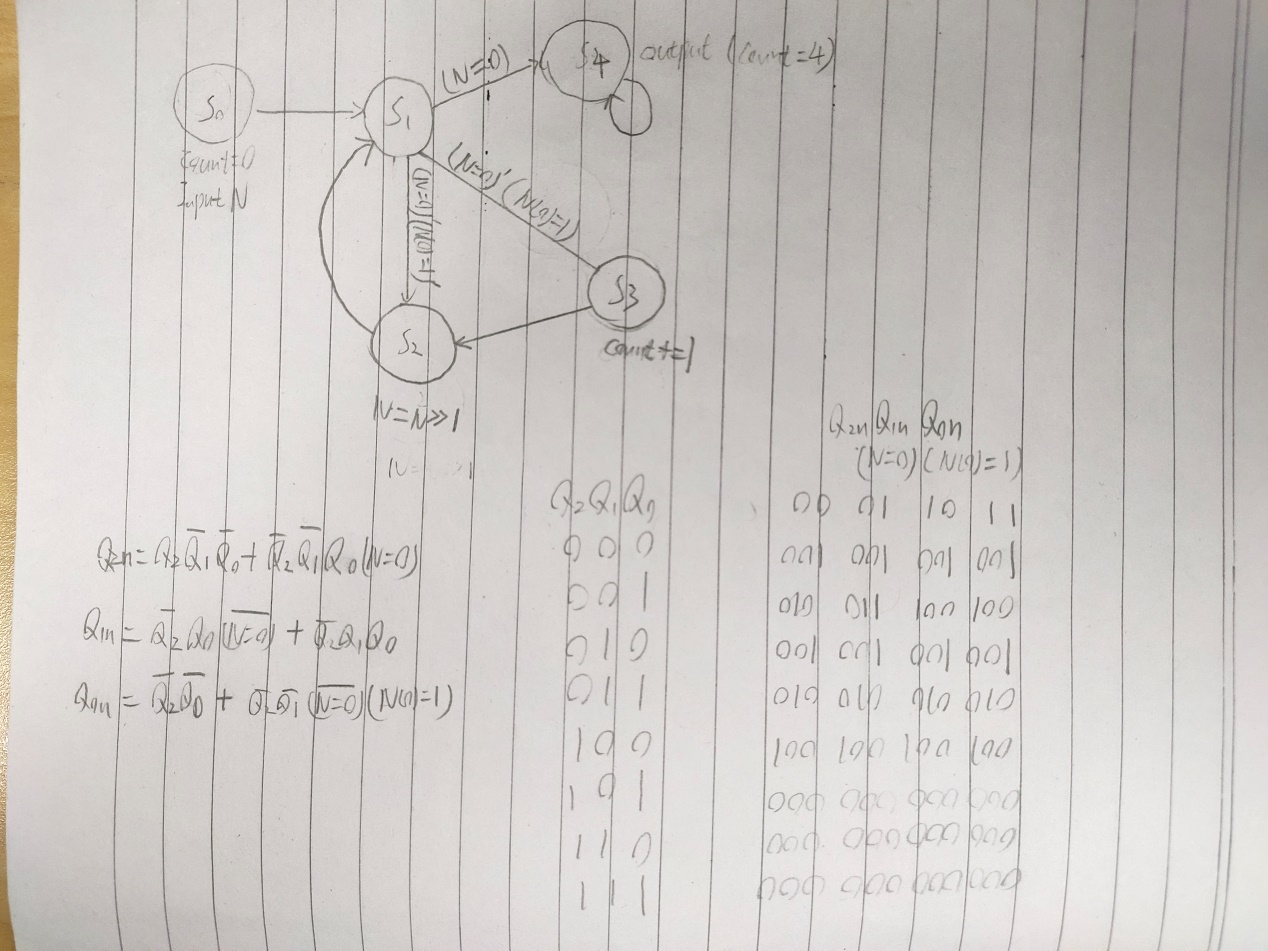
        else b<=4'b1101;

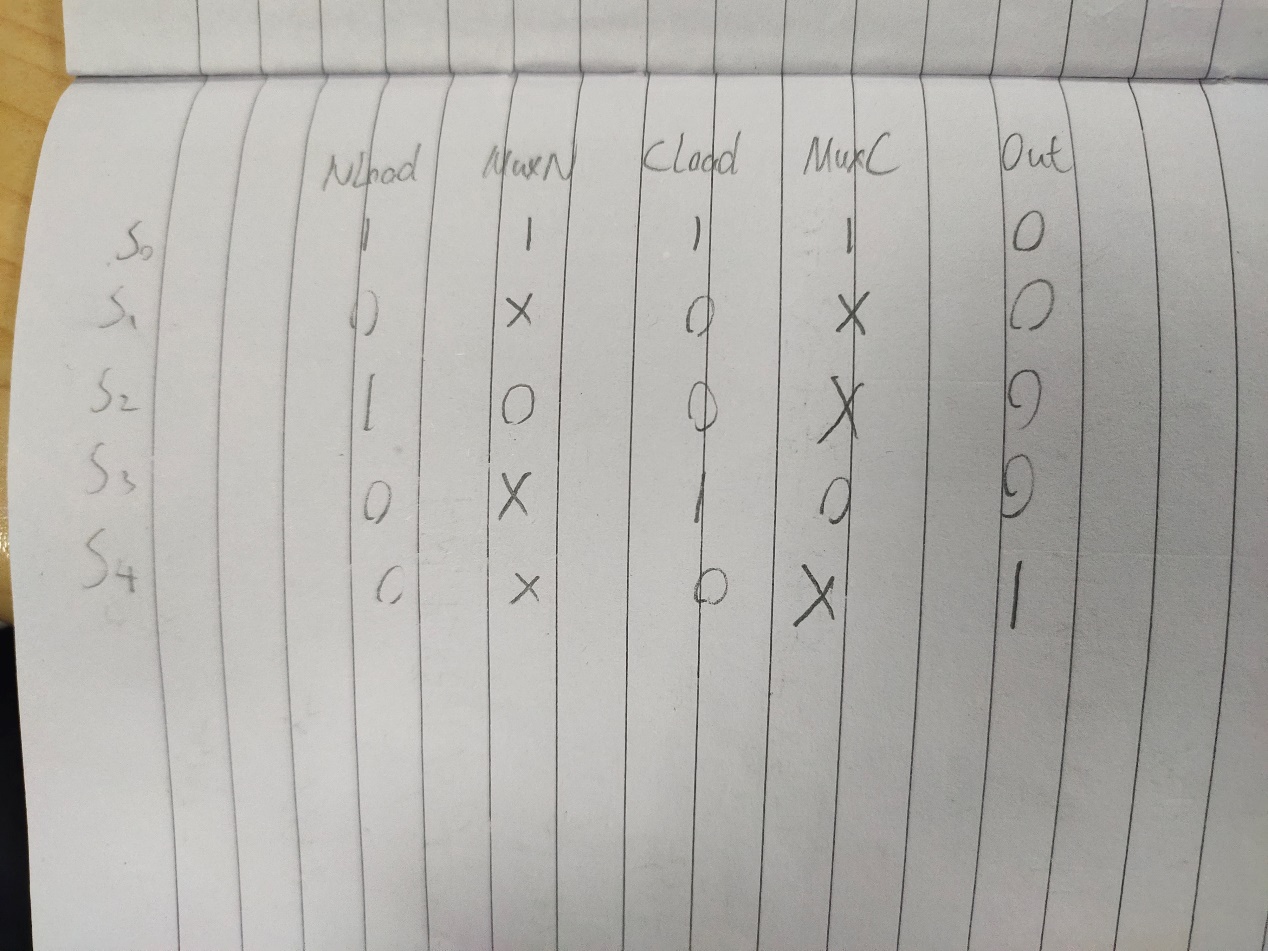
    end

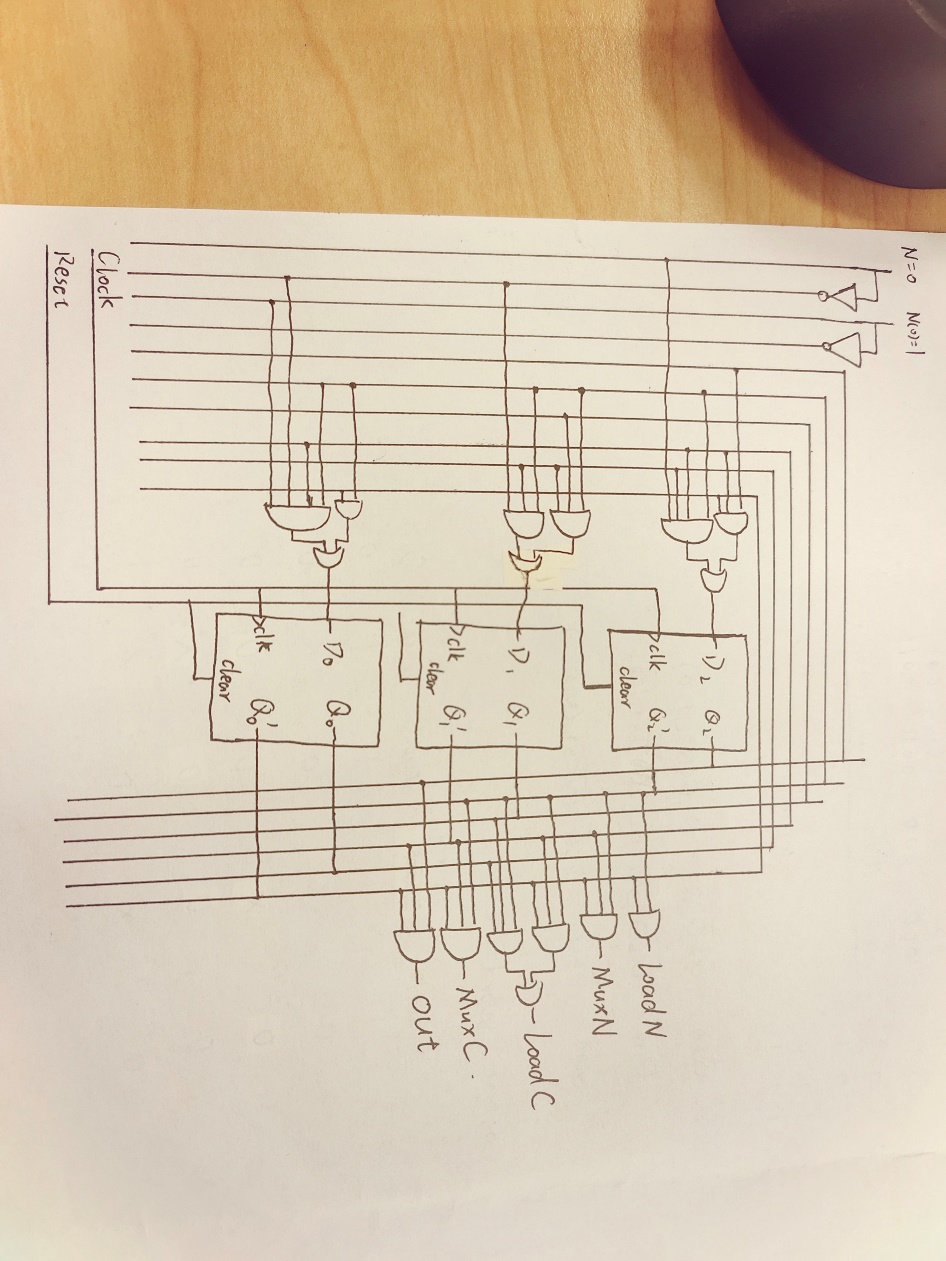
endmodule

**数据通路、控制字与状态机**

****

****

****

****

**各模块代码如下：**

**加法器：**

module adder

(

      input [3:0]a,b,

      output [3:0]sum

);

 assign sum=a+b;

endmodule

**选择器：**

module mux21

#(parameter n=4)

(

    input S,

    input [n-1:0]D1,

    input [n-1:0]D0,

    output [n-1:0]Y

);

    assign Y = S ? D1 : D0;

endmodule

**寄存器：**

module register

#(parameter n=8)

(

 input clock,clear,load,

 input [n-1:0] D,

    output reg [n-1:0] Q

);

    always @(posedge clock or negedge clear ) begin

        if (clear==0) begin

            Q<=0;

        end

        else if (load==1) begin

            Q<=D;

        end

    end

endmodule

**移位器：**

module shifter

 #(parameter n = 8)

(

    input [1:0]s,

    input [n-1:0]date,

    output reg [n-1:0]outdate

);

always @(s or date) begin

    case(s)

        2'b00: outdate = date;

        2'b01: outdate = date<<1;

        2'b10: outdate = date>>1;

        2'b11: outdate = {date[0],date[n-1:1]};

    endcase

end

endmodule

**三态门：**

module tri\_buffer

    #(parameter n = 4)

    (input E,

    input [n-1:0] D,

    output[n-1:0] Y

    );

    assign Y = E ? D : {n{1'bz}} ;

endmodule

**数据通路：**

`include "adder.v"

`include "mux21.v"

`include "register.v"

`include "shifter.v"

`include "tri\_buffer.v"

module dp(

    input Clock,Reset,

    input MuxN,LoadN,MuxC,LoadC,Out,

    input [7:0]N,

    output Data,

    output Nd0,N0d0

);

    wire[7:0] dp\_MuxN,dp\_shift,dp\_N;

    wire[3:0] dp\_MuxC,dp\_sum,dp\_C;

    wire C4;

    mux21#(8) muxN(

        .S(MuxN),

        .D1(N),

        .D0(dp\_shift),

        .Y(dp\_MuxN)

    );

    mux21#(4) muxC(

        .S(MuxC),

        .D1(4'b0000),

        .D0(dp\_sum),

        .Y(dp\_MuxC)

    );

    register#(8) regN(

        .clock(Clock),

        .clear(Reset),

        .load(LoadN),

        .D(dp\_MuxN),

        .Q(dp\_N)

    );

    register#(4) regC(

        .clock(Clock),

        .clear(Reset),

        .load(LoadC),

        .D(dp\_MuxC),

        .Q(dp\_C)

    );

    shifter#(8) shift(

        .s(2'b10),

        .date(dp\_N),

        .outdate(dp\_shift)

    );

    adder sum(

        .a(dp\_C),

        .b(4'b0001),

        .sum(dp\_sum)

    );

    assign C4=(dp\_C==4'b0100)?1'b1:1'b0;

    assign Nd0=(dp\_N==0)?1'b1:1'b0;

    assign N0d0=dp\_N[0];

    tri\_buffer#(1) buffer(

        .E(Out),

        .D(C4),

        .Y(Data)

    );

endmodule

**状态机：**

module fsm (

     input Clock,Reset,

     input N0d0,Nd0,

     output reg Nload,Cload,Out,MuxC,MuxN

);

    parameter s0 = 0,s1 = 1,s2 = 2,s3 = 3,s4 = 4;

    reg [2:0] state;

    always @(posedge Clock or negedge Reset) begin

        if(~Reset)

            state <= s0;

        else

            case(state)

                s0: state <= s1;

                s1:begin

                  if (Nd0) begin

                    state <= s4;

                  end

                    else if (N0d0) begin

                        state <= s3;

                    end

                    else begin

                        state <= s2;

                    end

                end

                s2: state <= s1;

                s3: state <= s2;

                s4: state <= s4;

                default : state <= s0;

            endcase

    end

    always @(state) begin

        case (state)

            s0: begin

                Nload <= 1'b1;

                Cload <= 1'b1;

                Out <= 1'b0;

                MuxC <= 1'b1;

                MuxN <= 1'b1;

            end

            s1: begin

                Nload <= 1'b0;

                Cload <= 1'b0;

                Out <= 1'b0;

                MuxC <= 1'b0;

                MuxN <= 1'b0;

            end

            s2: begin

                Nload <= 1'b1;

                Cload <= 1'b0;

                Out <= 1'b0;

                MuxC <= 1'b0;

                MuxN <= 1'b0;

            end

            s3: begin

                Nload <= 1'b0;

                Cload <= 1'b1;

                Out <= 1'b0;

                MuxC <= 1'b0;

                MuxN <= 1'b0;

            end

            s4: begin

                Nload <= 1'b0;

                Cload <= 1'b0;

                Out <= 1'b1;

                MuxC <= 1'b0;

                MuxN <= 1'b0;

            end

            default: begin

                Nload <= 1'b0;

                Cload <= 1'b0;

                Out <= 1'b0;

                MuxC <= 1'b0;

                MuxN <= 1'b0;

            end

        endcase

    end

endmodule

**顶层模块：**

`include "dp.v"

`include "fsm.v"

module top (

    input Clock,Reset,

    output Data,

    input [7:0]N

);

wire MuxN,LoadN,MuxC,LoadC,Out,Nd0,N0d0;

  dp dp(

      .Clock(Clock),

      .Reset(Reset),

      .MuxN(MuxN),

      .LoadN(LoadN),

      .MuxC(MuxC),

      .LoadC(LoadC),

      .Out(Out),

      .N(N),

      .Data(Data),

      .Nd0(Nd0),

      .N0d0(N0d0)

  );

  fsm fsm(

      .Clock(Clock),

      .Reset(Reset),

      .N0d0(N0d0),

      .Nd0(Nd0),

      .Nload(LoadN),

      .Cload(LoadC),

      .Out(Out),

      .MuxC(MuxC),

      .MuxN(MuxN)

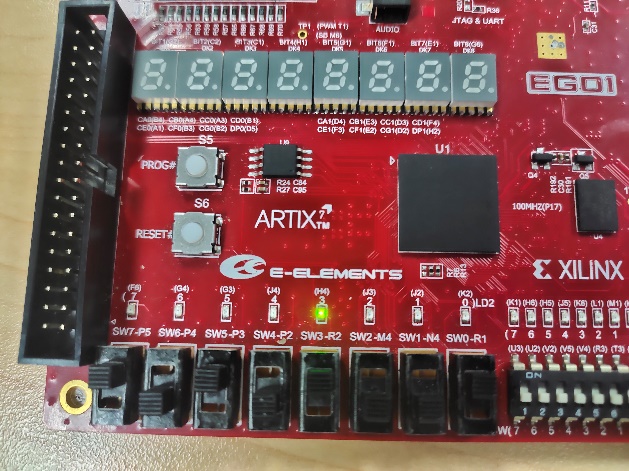
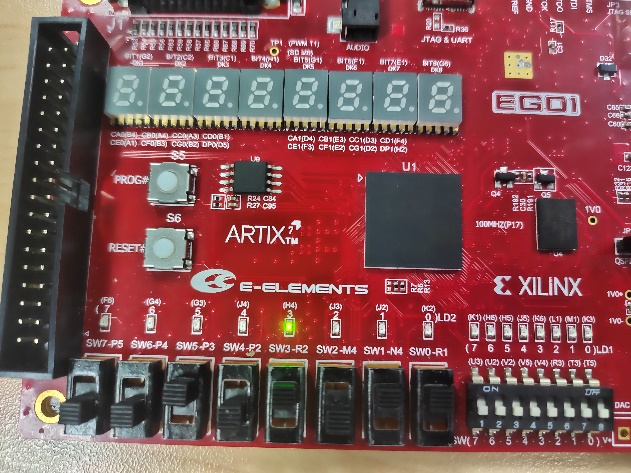
  );

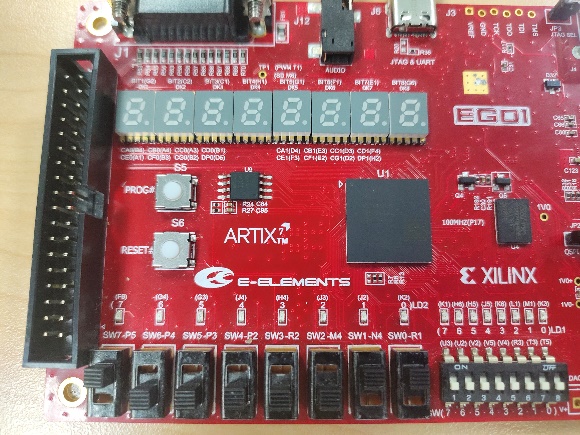
endmodule

**4 实验结果和分析**=

测试效果如下功能正常

将八个输入约束到八个拨码开关上，输出约束到一个led灯上，上板测试。



如图所示功能正常。

**5 实验结论**

（基于实验结果和分析，得出结论）

成功使用Verilog代码，实现了lab所要求的功能。