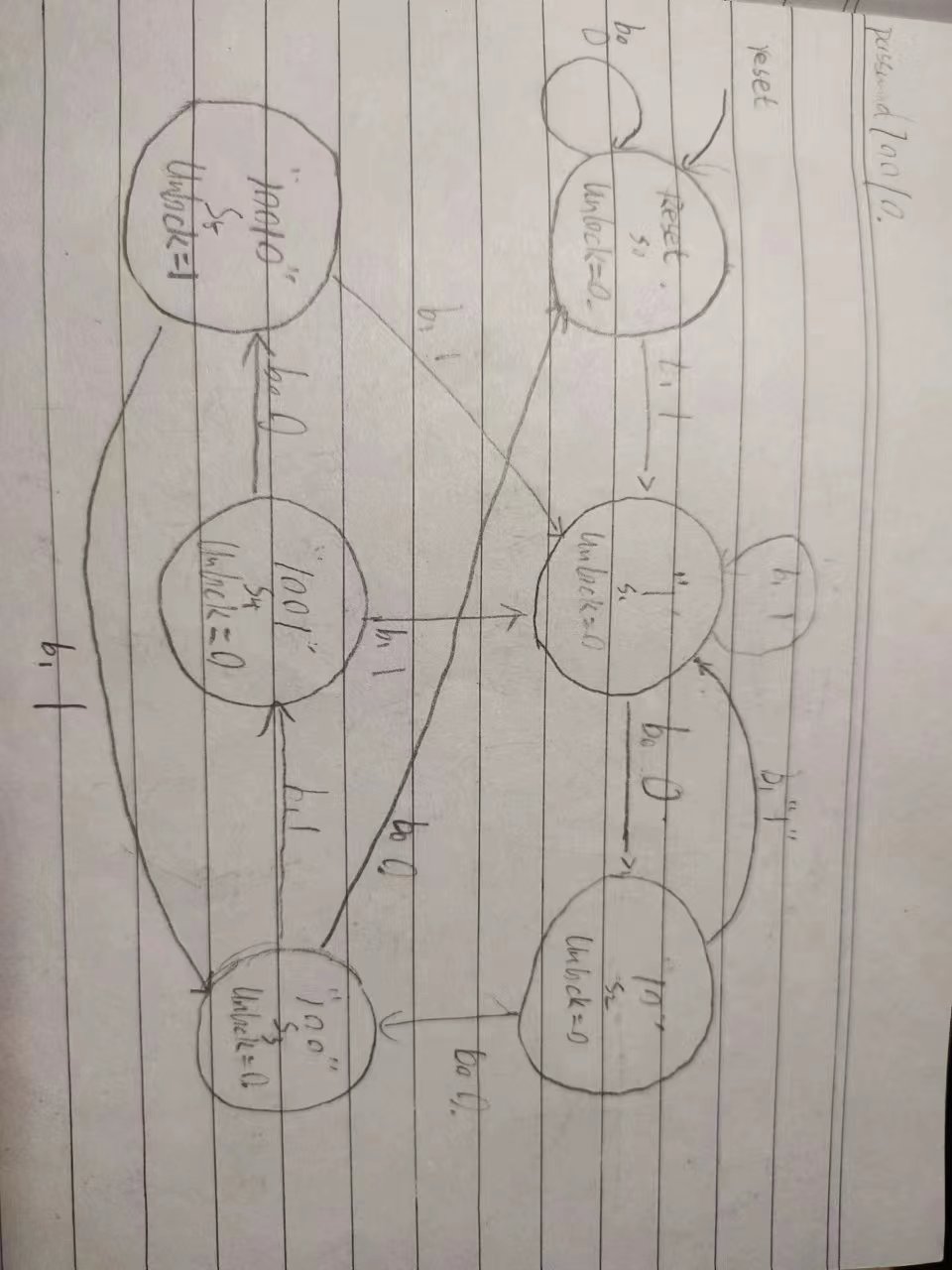


1. Draw the state diagram of the FSM in the above figure.

Set the password:10010



2)Write the behavior level Verilog module of the FSM.

module FSM (

    input clk,

    input rst,

    input b0,b1,

    output  reg  unlock,

    output  [2:0] debugstate

    );

    parameter s\_reset = 0;

    parameter s\_1 = 1;

    parameter s\_10 = 2;

    parameter s\_100 = 3;

    parameter s\_1001 = 4;

    parameter s\_10010 = 5;

    reg[2:0] state,next\_state;

    assign debugstate=state;

    always@(posedge clk or negedge rst) begin

        if(!rst) begin

            state <= s\_reset;

        end

        else begin

            state <= next\_state;

        end

    end

    always @(posedge clk or negedge rst) begin

        if (!rst) unlock=0;

        else begin case (next\_state)

            s\_reset: unlock<=0;

            s\_1: unlock<=0;

            s\_10: unlock<=0;

            s\_100: unlock<=0;

            s\_1001: unlock<=0;

            s\_10010: unlock<=1;

            default: unlock<=0;

        endcase

        end

    end

    always @(state or b0 or b1)

    begin

        case (state)

        s\_reset: begin

            if ({b0,b1}==2'b01) next\_state=s\_1;

            else next\_state=s\_reset;

        end

        s\_1: begin

            if ({b0,b1}==2'b10) next\_state=s\_10;

            else next\_state=s\_1;

        end

        s\_10: begin

            if ({b0,b1}==2'b10) next\_state=s\_100;

            else if ({b0,b1}==2'b01) next\_state=s\_1;

            else next\_state=s\_10;

        end

        s\_100: begin

            if ({b0,b1}==2'b01) next\_state=s\_1001;

            else if ({b0,b1}==2'b10) next\_state=s\_reset;

            else next\_state=s\_100;

        end

        s\_1001: begin

            if ({b0,b1}==2'b10) next\_state=s\_10010;

            else if ({b0,b1}==2'b01) next\_state=s\_1;

            else next\_state=s\_1001;

        end

        s\_10010: begin

            if ({b0,b1}==2'b10) next\_state=s\_reset;

            else if ({b0,b1}==2'b01) next\_state=s\_100;

            else next\_state=s\_10010;

        end

        endcase

    end

endmodule

3)

Write a testbench to simulate the FSM by using the method in Lab 3.

//~ `New testbench

`include "FSM.v"

`timescale  1ns / 1ps

module tb\_FSM;

// FSM Parameters

parameter PERIOD   = 10;

parameter s\_reset  = 0;

parameter s\_1      = 1;

parameter s\_10     = 2;

parameter s\_100    = 3;

parameter s\_1001   = 4;

parameter s\_10010  = 5;

// FSM Inputs

reg   clk                                  = 0 ;

reg   rst                                  = 0 ;

reg   b0                                   = 0 ;

reg   b1                                   = 0 ;

// FSM Outputs

wire  unlock                               ;

wire  [2:0]  debugstate                    ;

initial

begin

    forever #(PERIOD/2)  clk=~clk;

end

FSM #(

    .s\_reset ( s\_reset ),

    .s\_1     ( s\_1     ),

    .s\_10    ( s\_10    ),

    .s\_100   ( s\_100   ),

    .s\_1001  ( s\_1001  ),

    .s\_10010 ( s\_10010 ))

 u\_FSM (

    .clk                     ( clk               ),

    .rst                     ( rst               ),

    .b0                      ( b0                ),

    .b1                      ( b1                ),

    .unlock                  ( unlock            ),

    .debugstate              ( debugstate  [2:0] )

);

initial

begin

    $dumpfile("tb\_FSM.vcd");

    $dumpvars(0,tb\_FSM);

    b0 = 0;

    b1 = 0;

    rst = 0;

    #5

    rst = 1;

    #10

    b1=1;

    #10

    b1=0;

    b0=1;

    #10

    b0=1;

    b1=0;

    #10

    b0=0;

    b1=0;

    #30

    b0=0;

    b1=1;

    #10

    b0=1;

    b1=0;

    #10

    b0=0;

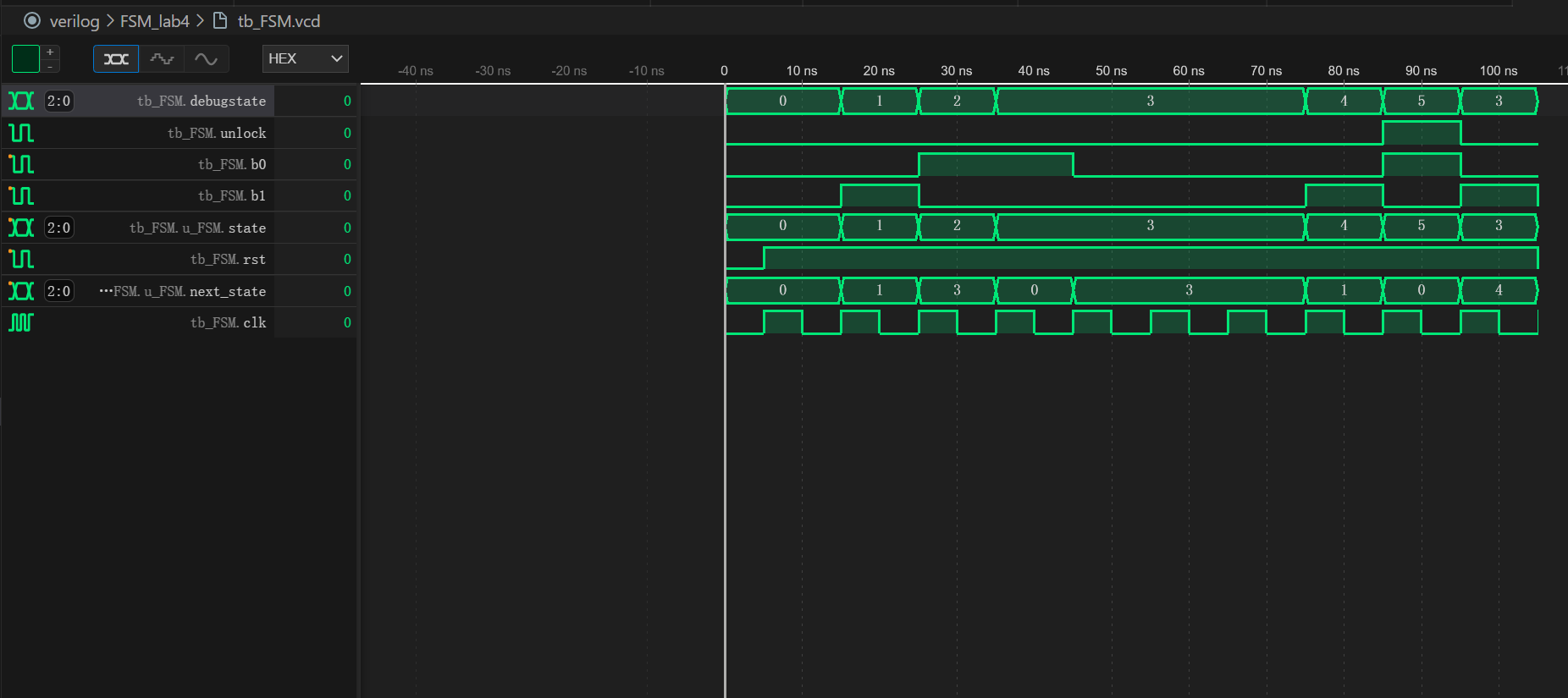
    b1=1;

    #10

    $finish;

end

endmodule



4)

Write the Verilog description of the whole design in the above figure and test it in your Basys3 board.

`include "botton.v"

`include "FSM.v"

`include "Segment.v"

module top (

    input clk,

    input rst,

    input key0,key1,

    output  [0:0]unlock,

    output  [7:0]Sel,

    output  [7:0]DisPlay,

    output  [7:0]DisPlay1

);

   wire b0,b1;

   wire [2:0] current\_state;

   reg  [31:0] data;

   botton u\_botton(

       .clk(clk),

       .rst\_n(rst),

       .key(key0),

       .press(b0)

   );

    botton u\_botton1(

         .clk(clk),

         .rst\_n(rst),

         .key(key1),

         .press(b1)

    );

   FSM u\_FSM(

       .clk(clk),

       .rst(rst),

       .b0(b0),

       .b1(b1),

       .unlock(unlock),

       .debugstate(current\_state)

   );

    always @(posedge clk ) begin

        case (current\_state)

   3'b000: data<=32'h00000000;

   3'b001: data<=32'h00000001;

   3'b010: data<=32'h00000010;

   3'b011: data<=32'h00000100;

   3'b100: data<=32'h00001001;

   3'b101: data<=32'h00010010;

        endcase

    end

    Segment u\_Segment(

         .Clk(clk),

         .Reset\_N(rst),

         .Disp\_Data(data),

         .state(current\_state),

         .Sel(Sel),

         .DisPlay(DisPlay),

         .DisPlay1(DisPlay1)

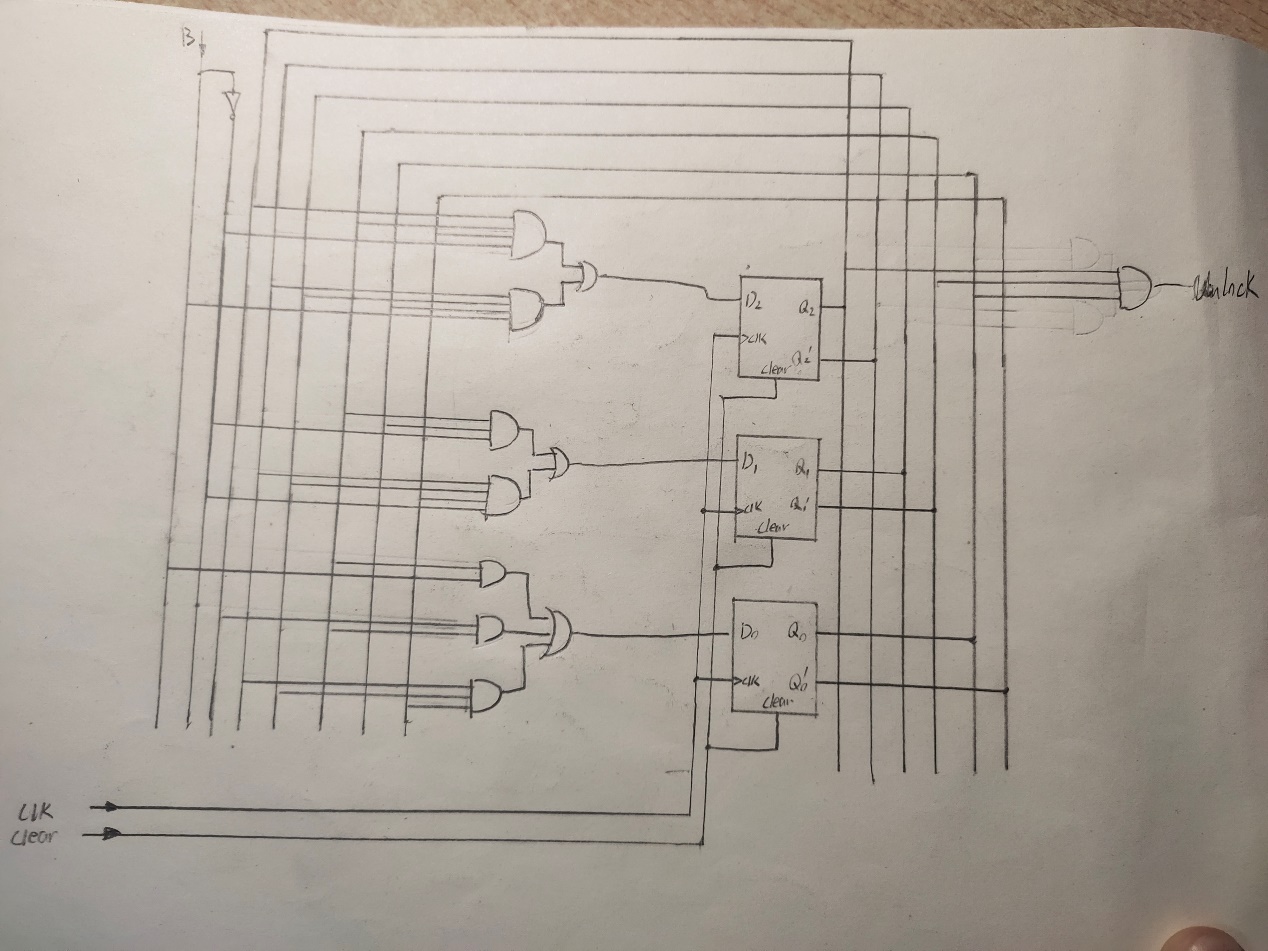
    );

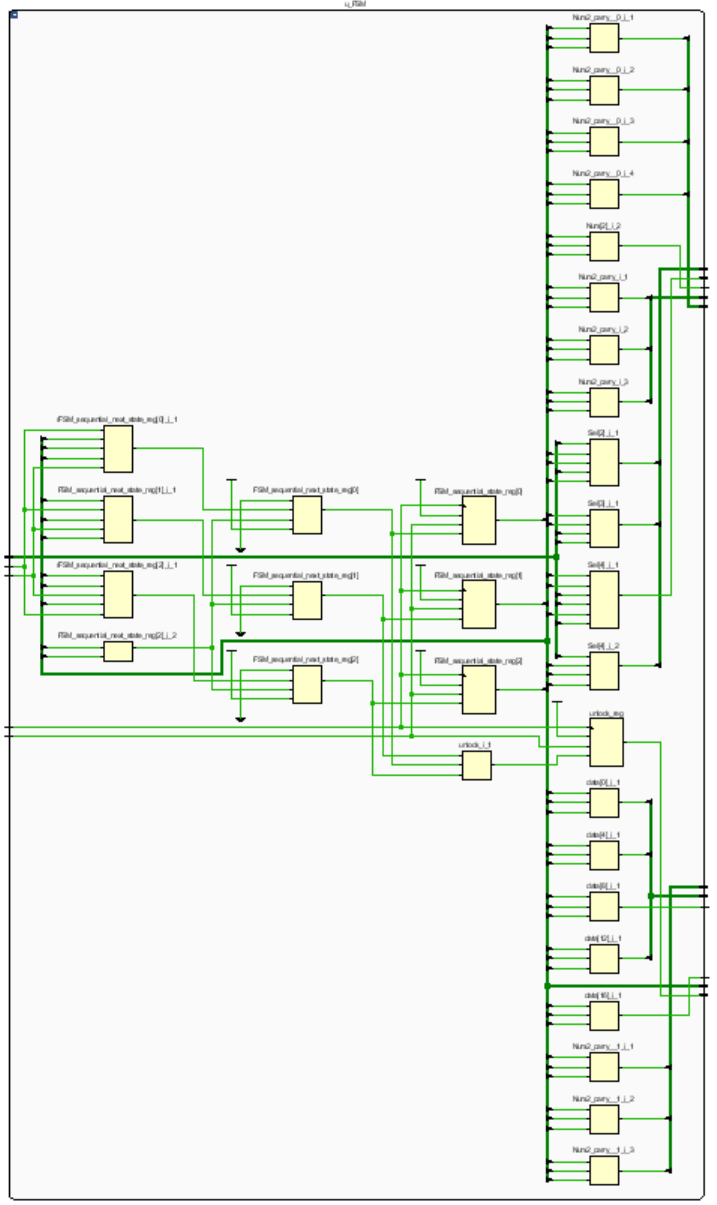
endmodule

按钮模块使用pre lab4中代码，FSM模块使用（2）中代码，Segment模块为数码管显示模块。

5)Synthesis the FSM by hand to draw the circuit of the FSM with only the D flipflops and AND/

OR/NOT gate. Verify the circuit you synthesized and the circuit synthesized by Vivado.





6)

From the synthesized circuit by your own, write the Verilog module of the FSM: For the next-state logic and the output logic, use “assign” statement to describe the combinational logic,and for the D flipflop, use “always” statement to describe it.

7)For the whole design, replace the FSM module designed in step 2) with that designed in step6), and implement the whole design in your board.

经测试，两种状态机效果实现电路效果相同