

**1 实验目的和要求**

To learn how to implement displaying a character string circularly by using FSM.

**2 实验内容**

1. Synthesize a FSM circuit (i.e., draw the complete FSM circuit) to display the character string

2017 04 27 ddl

in one 7-segment LED circularly at a rate of 0.5 Hz for each digital. Blank the display (i.e., turn off all the segments of the LED) for the spaces in the string. Assume that your main clock frequency is 100MHz. Write the Verilog code for your design.

1. Implement the above circuit in your EGO1/Basys3 board.
2. Display the above character string in four 7-segment LED circularly at a rate of 0.5 Hz in your EGO1/Basys3 board

**3 实验内容**

（分点阐述实验步骤）

1. 编写代码实现要求一。
2. 上板验证。
3. 修改代码实现要求三。

**4 源代码与分析**

本实验使用两种方式实现要求功能：

**状态机方式代码：**

功能一：

module ledplus(

    input reset,

    input clk,

    output enable,

    output reg [6:0]led

    );

    parameter CLKDIV=9999\_9999;

    reg [28:0] count=0;

    reg [3:0] state,next\_state;

//状态机状态参数

    parameter s0=0,s1=1,s2=2,s3=3,s4=4,s5=5,s6=6,s7=7,s8=8,s9=9,s10=10,s11=11,s12=12,s13=13,s14=14;

//时钟分频

    always @(posedge clk or negedge reset)begin

        if(!reset) count<=0;

        else begin

            if(count==CLKDIV-1) count<=0;

            else count<=count+1'b1;

        end

    end

//状态机

    always @(posedge clk or negedge reset)begin

        if(!reset) next\_state<=s0;

       else case (state)

            s0:next\_state<=s1;

            s1:next\_state<=s2;

            s2:next\_state<=s3;

            s3:next\_state<=s4;

            s4:next\_state<=s5;

            s5:next\_state<=s6;

            s6:next\_state<=s7;

            s7:next\_state<=s8;

            s8:next\_state<=s9;

            s9:next\_state<=s10;

            s10:next\_state<=s11;

            s11:next\_state<=s12;

            s12:next\_state<=s13;

            s13:next\_state<=s14;

            s14:next\_state<=s0;

        endcase

    end

//状态机更新

    always @(posedge clk)begin

        if(count==CLKDIV-1) state<=next\_state;

    end

//状态机输出

    always @(posedge clk)begin

        case(state)

                    s0:led<=7'b110\_1101;

                    s1:led<=7'b111\_1110;

                    s2:led<=7'b011\_0000;

                    s3:led<=7'b111\_0000;

                    s4:led<=7'b000\_0000;

                    s5:led<=7'b111\_1110;

                    s6:led<=7'b011\_0011;

                    s7:led<=7'b000\_0000;

                    s8:led<=7'b110\_1101;

                    s9:led<=7'b111\_0000;

                    s10:led<=7'b000\_0000;

                    s11:led<=7'b011\_1101;

                    s12:led<=7'b011\_1101;

                    s13:led<=7'b000\_1110;

                    s14:led<=7'b000\_0000;

        endcase

    end

    assign enable=1;

endmodule

功能二：

module ledplus(

    input reset,

    input clk,

    output reg [3:0]enable,

    output reg [6:0]led

    );

    reg [27:0] ledchoose;

    reg [20:0] countdiv=0;

    reg [1:0] countled=0;

    parameter CLKDIV=9999\_9999;

    parameter COUNT\_DIV=45\_0000;

    reg [28:0] count=0;

    reg [3:0] state,next\_state;

//状态机状态参数

        parameter s0=0,s1=1,s2=2,s3=3, s4=4,s5=5,s6=6,s7=7, s8=8,s9=9,s10=10,s11=11,s12=12,s13=13,s14=14;

//时钟分频

    always @(posedge clk or negedge reset)begin

          if(!reset) count<=0;

          else begin

              if(count==CLKDIV-1) count<=0;

              else count<=count+1'b1;

          end

    end

//状态机

   always @(posedge clk or negedge reset)begin

       if(!reset) next\_state<=s0;

       else case (state)

            s0:next\_state<=s1;

            s1:next\_state<=s2;

            s2:next\_state<=s3;

            s3:next\_state<=s4;

            s4:next\_state<=s5;

            s5:next\_state<=s6;

            s6:next\_state<=s7;

            s7:next\_state<=s8;

            s8:next\_state<=s9;

            s9:next\_state<=s10;

            s10:next\_state<=s11;

            s11:next\_state<=s12;

            s12:next\_state<=s13;

            s13:next\_state<=s14;

            s14:next\_state<=s0;

        endcase

    end

//状态机更新

    always @(posedge clk)begin

        if(count==CLKDIV-1) state<=next\_state;

    end

//状态机输出

always @(posedge clk)begin

       case(state)

           s0:ledchoose<=28'b0000000\_0000000\_0000000\_0000000;

           s1:ledchoose<=28'b0000000\_0000000\_0000000\_1101101;

           s2:ledchoose<=28'b0000000\_0000000\_1101101\_1111110;

           s3:ledchoose<=28'b0000000\_1101101\_1111110\_0110000;

           s4:ledchoose<=28'b1101101\_1111110\_0110000\_1110000;

           s5:ledchoose<=28'b1111110\_0110000\_1110000\_1111110;

           s6:ledchoose<=28'b0110000\_1110000\_1111110\_0110011;

           s7:ledchoose<=28'b1110000\_1111110\_0110011\_1101101;

           s8:ledchoose<=28'b1111110\_0110011\_1101101\_1110000;

           s9:ledchoose<=28'b0110011\_1101101\_1110000\_0111101;

           s10:ledchoose<=28'b1101101\_1110000\_0111101\_0111101;

           s11:ledchoose<=28'b1110000\_0111101\_0111101\_0001110;

           s12:ledchoose<=28'b0111101\_0111101\_0001110\_0000000;

           s13:ledchoose<=28'b0111101\_0001110\_0000000\_0000000;

           s14:ledchoose<=28'b0001110\_0000000\_0000000\_0000000;

       endcase

   end

//时钟分频

   always@(posedge clk or negedge reset)begin

        if(!reset)countdiv<=0;

        else begin

            if(countdiv==COUNT\_DIV-1 ) countdiv<=0;

            else countdiv = countdiv+1'b1;

        end

   end

//led选择

   always@(posedge clk or negedge reset)begin

        if(!reset)countled<=0;

        else begin

            if(countdiv==COUNT\_DIV-1)begin

                if(countled==3) countled<=0;

                else countled =countled+1'b1;

            end

        end

   end

//led输出

   always @(posedge clk)begin

        case(countled)

            2'b00:led<=ledchoose[27:21];

            2'b01:led<=ledchoose[20:14];

            2'b10:led<=ledchoose[13:7];

            2'b11:led<=ledchoose[6:0];

        endcase

   end

//使能输出

   always @(posedge clk)begin

        case(countled)

           2'b00:enable=4'b1000;

           2'b01:enable=4'b0100;

           2'b10:enable=4'b0010;

           2'b11:enable=4'b0001;

       endcase

   end

endmodule

**不使用状态机 通过移位方式实现功能：**

`include "clk\_division.v"

`include "segment.v"

module top

(

    input clk\_sys,

    input mode,

    input reset,

    output [7:0] led,

    output [3:0] segena

);

    reg [63:0]bcd=64'hBBB2\_017B\_04B2\_7DDE;//字符编码

    reg [15:0] seg\_selection;//段选

    reg mode1;//模式选择

    wire ena = 1'b1;

    wire clk;   //0.5Hz时钟

    wire clk1; // 20ms时钟

// 时钟分频

    clk\_division #(.F\_CLK\_SYS( 200\_000\_000 ) , .F\_CLK\_DIVISION( 1 ) ) clk\_division\_inst (

        .ena( 1'b1 ),

        .clk\_in( clk\_sys ),

        .clk\_out( clk )

    );

// 时钟分频

        clk\_division #(.F\_CLK\_SYS( 100\_000\_000 ) , .F\_CLK\_DIVISION( 500 ) ) clk\_division\_inst1 (

        .ena( 1'b1 ),

        .clk\_in( clk\_sys ),

        .clk\_out( clk1 )

    );

    always @(posedge clk or negedge reset) begin

        if (~reset) begin //复位

            bcd<=64'hBBB2\_017B\_04B2\_7DDE;

            seg\_selection <= 16'hBBBB;

        end

        else begin//单位显示

            if (~mode) begin

                if(mode1) begin

                    bcd <= 64'hBBB2\_017B\_04B2\_7DDE;

                    mode1<=mode;

                    end

                else begin

             seg\_selection <= bcd[63:48];

            bcd <= {bcd[63:52],bcd[47:0],bcd[51:48]};

            mode1<=mode;

                end

        end

        else begin//四位显示

            if(~mode1) begin

                    bcd <= 64'hBBB2\_017B\_04B2\_7DDE;

                    mode1<=mode;

                    end

                else begin

            seg\_selection <= bcd[63:48];

            bcd <= {bcd[59:0],bcd[63:60]};

            mode1<=mode;

                end

        end

        end

           end

// 数码管模块，输入时钟和seg\_selection，输出数码管段选和位选信号

           segment segment\_inst (

        .clk\_sys(clk\_sys),

        .clk(clk1),

        .reset(reset),

        .Disp\_Data(seg\_selection),

        .Sel(segena),

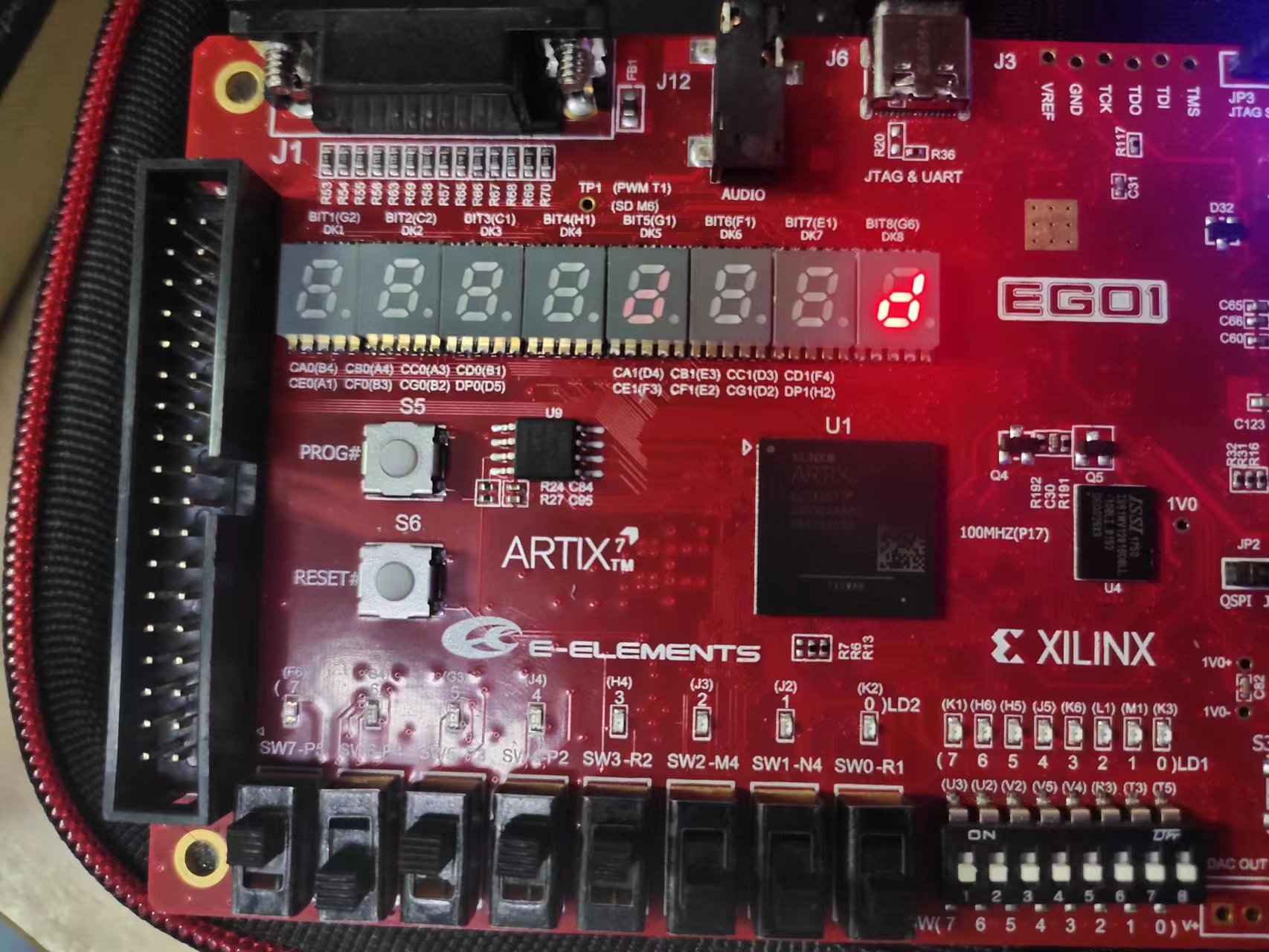
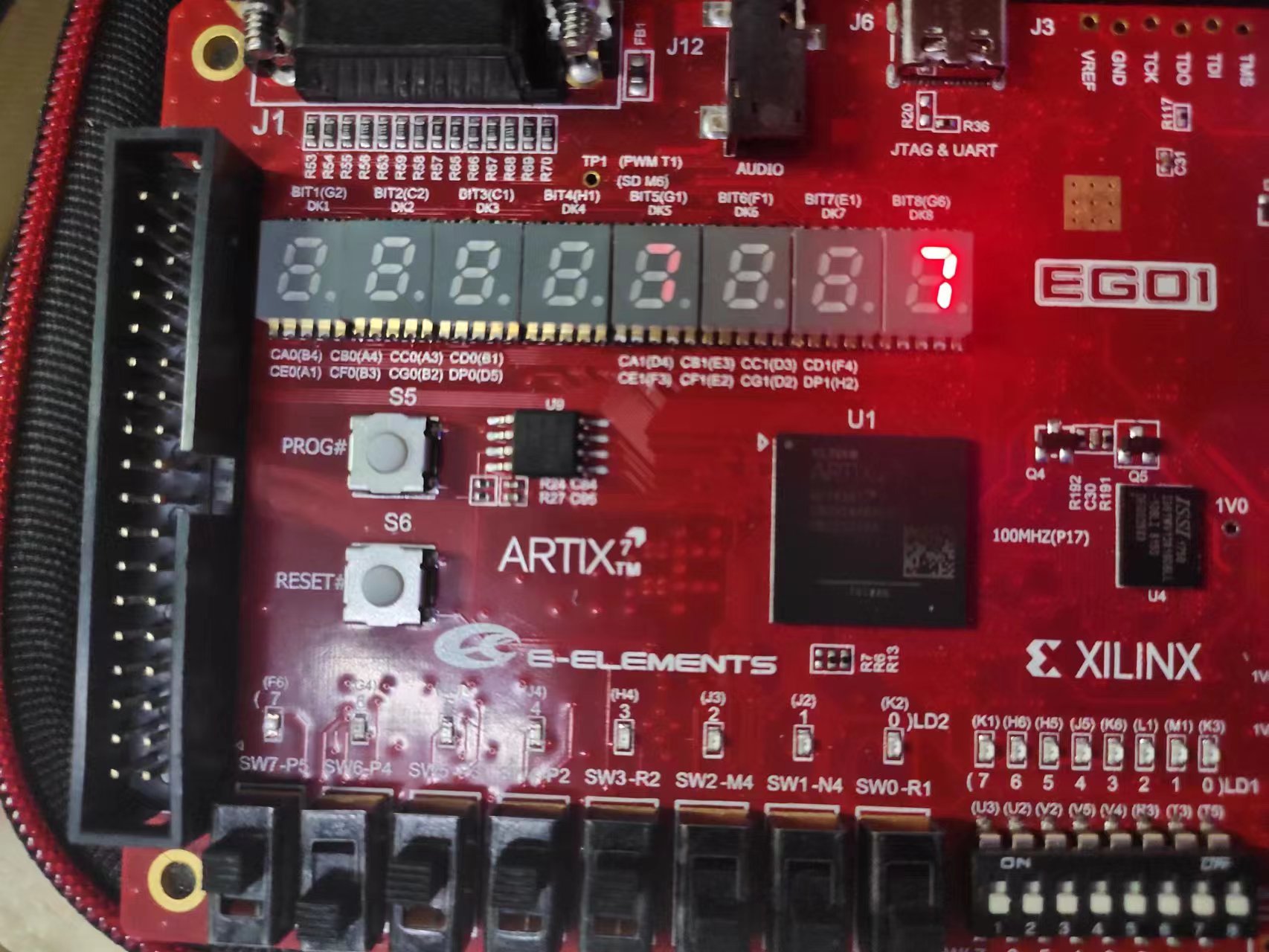
        .led(led)

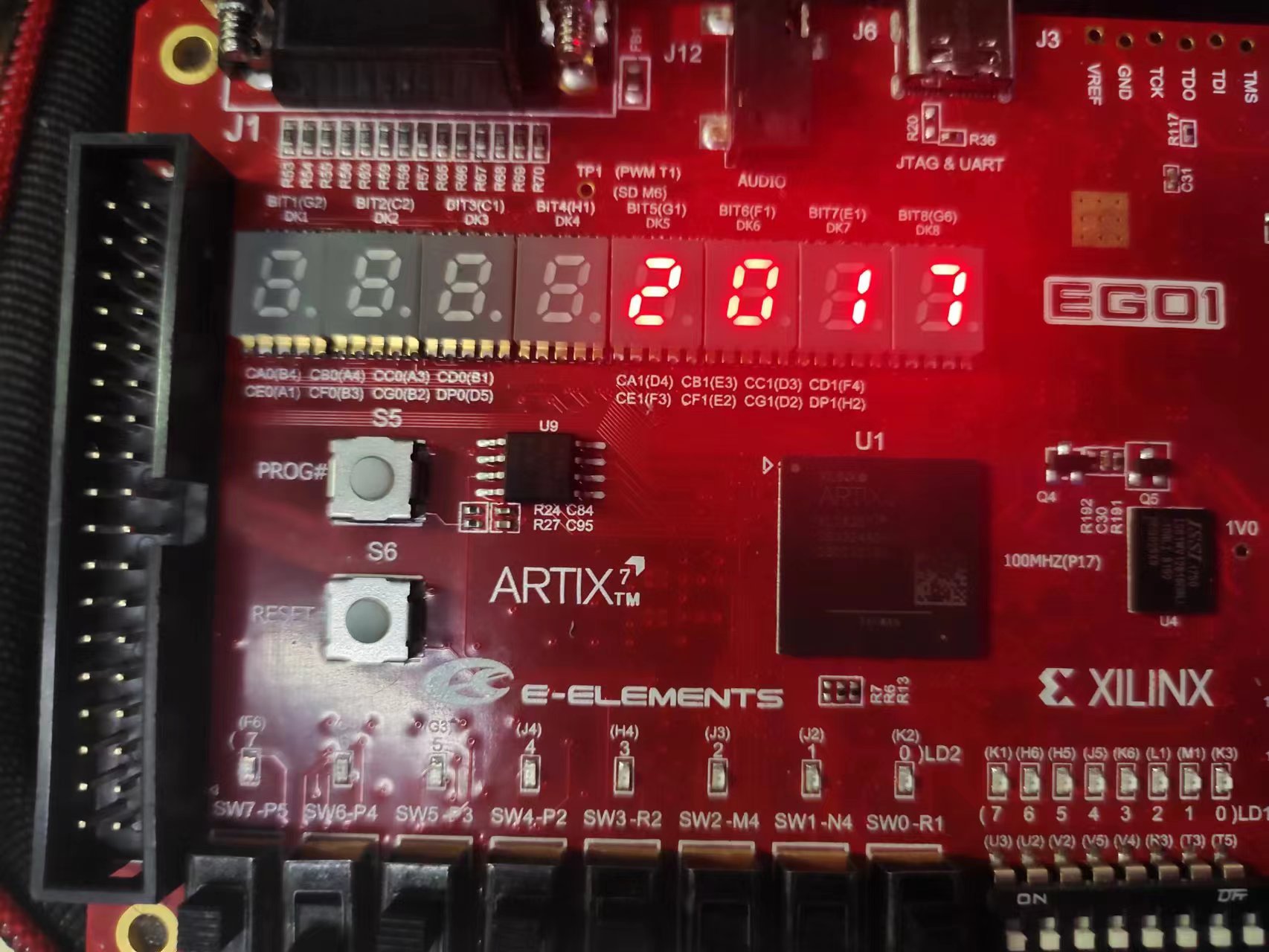
           );

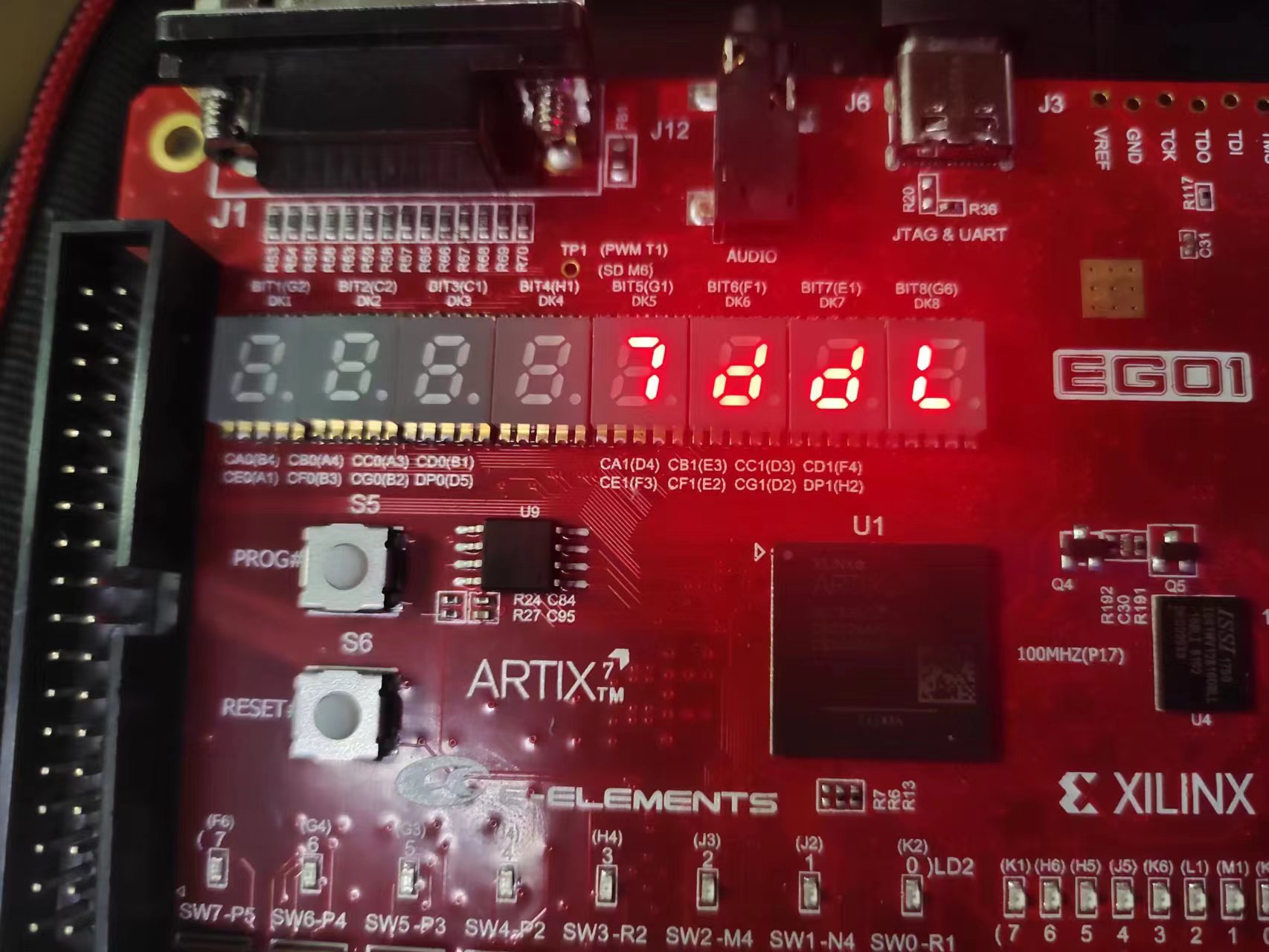
endmodule

**5 实验结果和分析**=

测试效果如下功能正常







**6 实验结论**

（基于实验结果和分析，得出结论）

成功使用两种方式编写Verilog代码，实现了lab所要求的功能。