**北京科技大学数字逻辑实验报告**

学院：计算机与通信工程学院 专业： 物联网工程 班级： 物联201

姓名： 赵方程 学号： 42024137 实验日期： 2021 年 11 月 24 日

**1、实验名称：**实验二 组合逻辑实验

**2、实验目的：**

(1) 学习用verilog设计较复杂的组合逻辑电路

(2) 进一步熟悉vivado工具

**3、实验内容：**

1. **实验2.1——多路选择器**
   1. 新建工程，用3种描述方法分别实现1位2选1选择器，并撰写测试激励，对比3个选择器的输出是否一致；
   2. 用1位2选1搭建1位4选1选择器；并仿真测试；
   3. 用两种描述方法实现4位二选一选择器，并撰写测试激励，对比输出是否一致。
2. **实验2.2——7段数码管**

4个拨码开关控制1个数码管，选择板子上的4个拨码开关作为数据输入，选择板子上的1个数码管作为输出，将输入的4位二进制转换为16进制数输出显示到数码管。

1. **实验2.3——加法器实现及性能比较**

实现32位逐位进位加法器、32位选择进位加法器，并进行性能比较。

* 1. 学习课件中逐位进位加法器、选择进位加法器的原理；
  2. 新建工程，完成两种加法器的Verilog描述，其中**一位全加器**已经提供（**module add1**在add32\_tb.v文件中已定义）直接使用即可，两种加法器请都使用add1作为基本模块开始搭建；模块名称及端口定义如下：

module csadd32 (a,b,cin,s,cout); //选择进位加法器

module rcadd32 (a,b,cin,s,cout); //逐位进位加法器

* 1. 编写测试激励：可更改add32\_tb.v文件中激励产生代码，按照自己的思路产生测试数据，通过仿真验证加法器功能，并观察两种加法器的性能情况。

**实验步骤：**

4、**实验2.1（多路选择器）的实现及仿真验证**

（1）3种描述方法分别实现1位2选1选择器Verilog代码

module mux1 (

    input wire [1:0] d,

    input wire a,

    output wire y

);

  assign y = a == 1'b0 ? d[0]:d[1];

endmodule

数据流描述 version 1

module mux2(

    input wire [1:0] d,

    input wire a,

    output wire y

);

not(na,a);

and(d0,d[0],na);

and(d1,d[1],a);

or(y,d0,d1);

endmodule

结构化描述 version 2

module mux3(

    input wire [1:0] d,

    input wire a,

    output wire y

)

begin

    if(a == 1'b1)

        y = d[0];

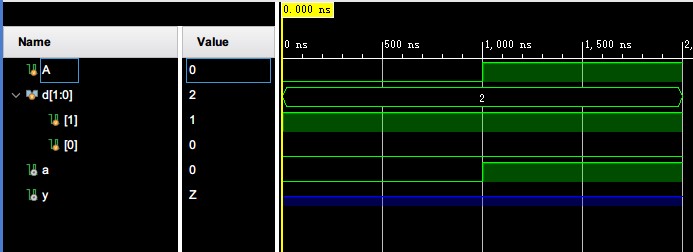
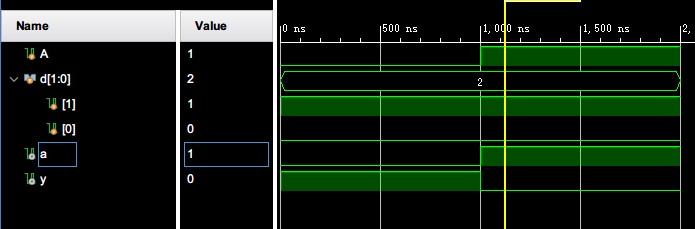
    else

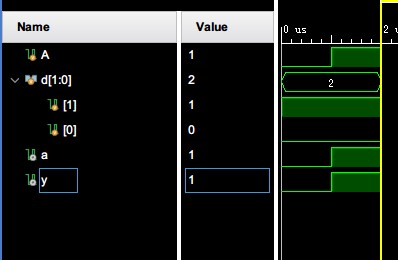
        y = d[1];

end

endmodule

行为级描述 version 3

（2）（1）的仿真波形及对比说明



A信号随时间分别为0，1，对应选择的数据为D[0] D[1]

（3）用1位2选1搭建1位4选1选择器的Verilog代码

module mux4\_1(

        input  wire [1:0] d1,d2,

        input  wire [1:0] a,

        output wire       y

    );

    reg[1:0] d;

    always @(\*) begin

        if (a[0]) begin

          assign d = d1;

        end else begin

          assign d = d2;

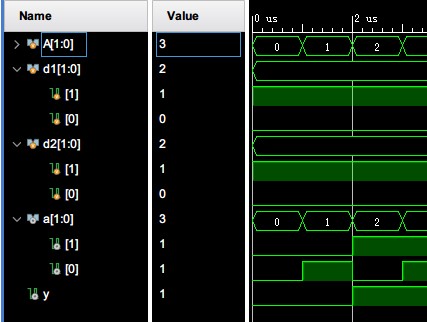
        end

    end

    mux1 mux2\_inst(.d(d),.a(a[1]),.y(y));

endmodule

（4）（3）的仿真波形及说明



A信号随时间分别为00，01，10，11时对应选择的数据为D0[0] D1[0]，D0[1],D1[1]

（5）用两种描述方法实现4位二选一选择器的Verilog代码

module mux4bit2to1(

    input [3:0] a,

    input [3:0] b,

    input s,

    output [3:0]y

)

    assign y = s==1 ? a : b;

endmodule

version 1

module mux4bit2to1(

    input [3:0] a,

    input [3:0] b,

    input s,

    output[3:0] y

);

    reg [3:0] o;

    always @(\*)begin

        if (s)begin

            o = a;

        end else begin

            o = b;

        end

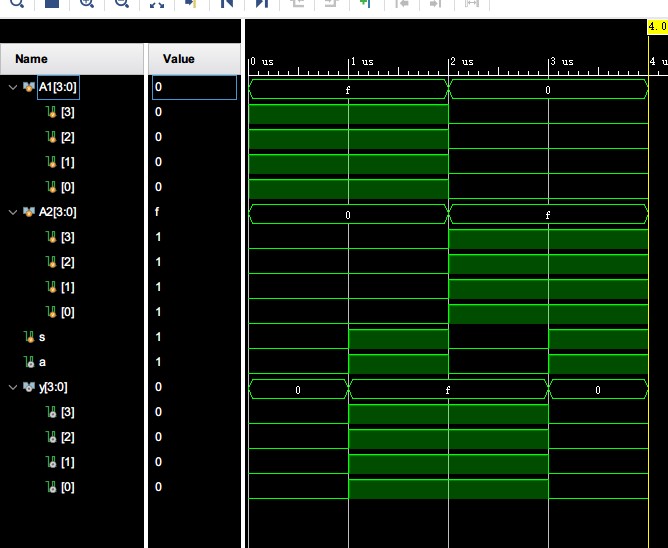
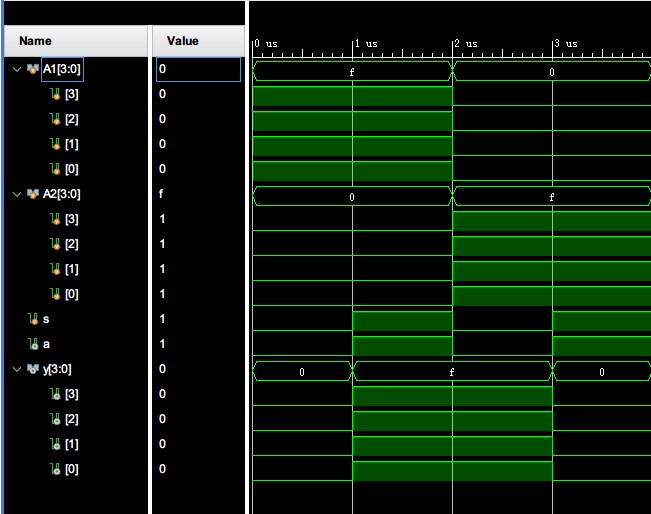
    end

    assign y = o;

endmodule

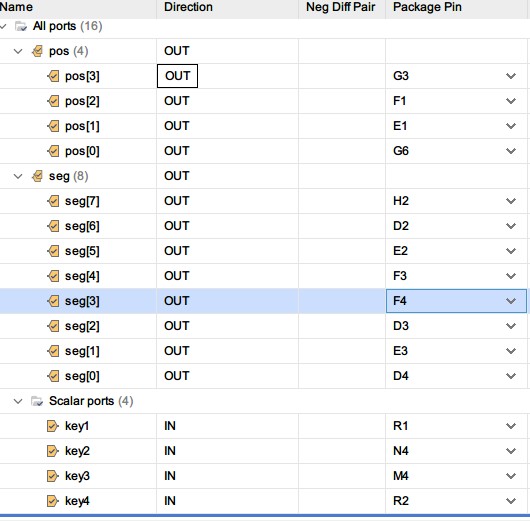
version 2

（6）（5）的仿真波形及对比说明

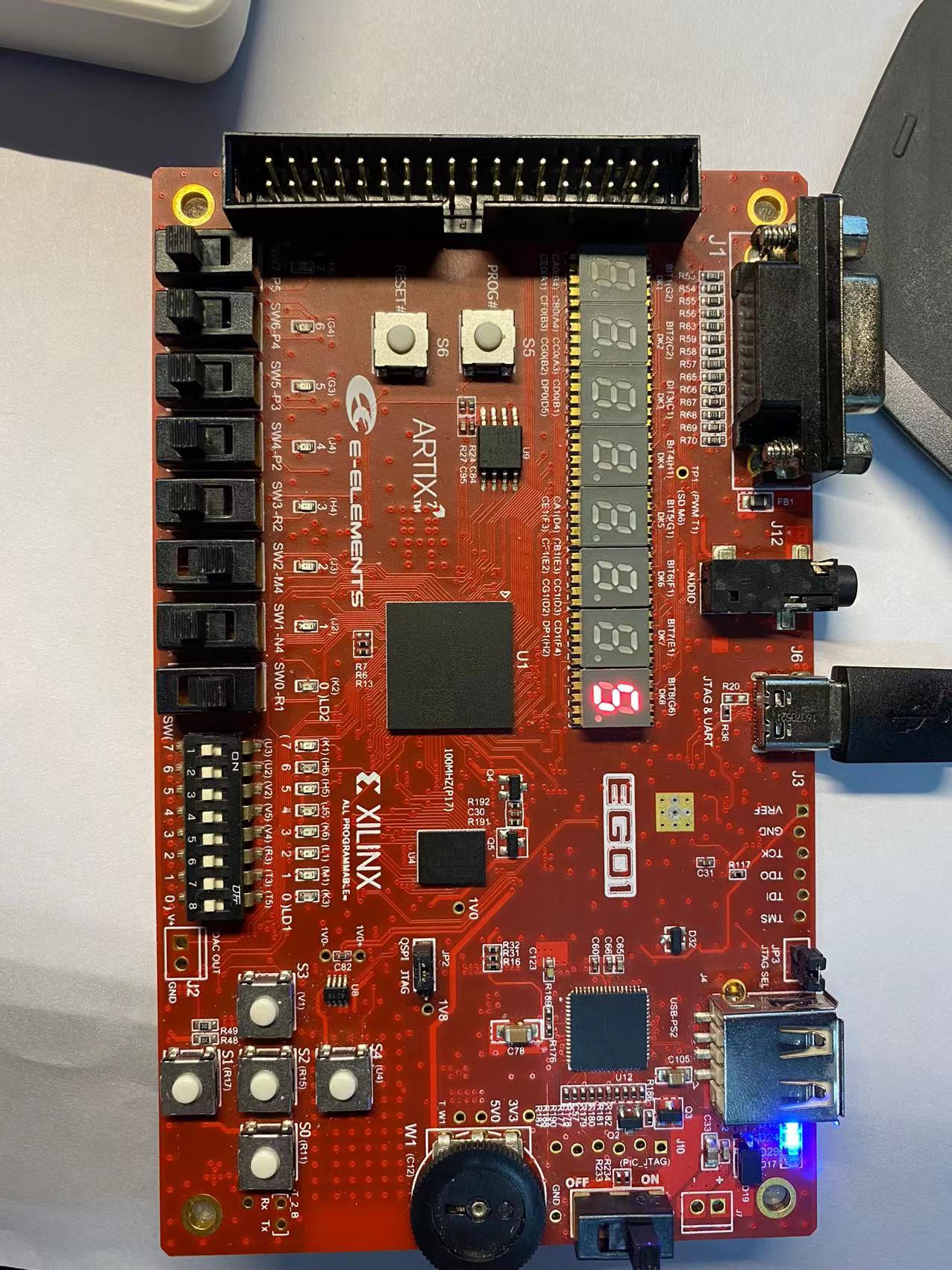
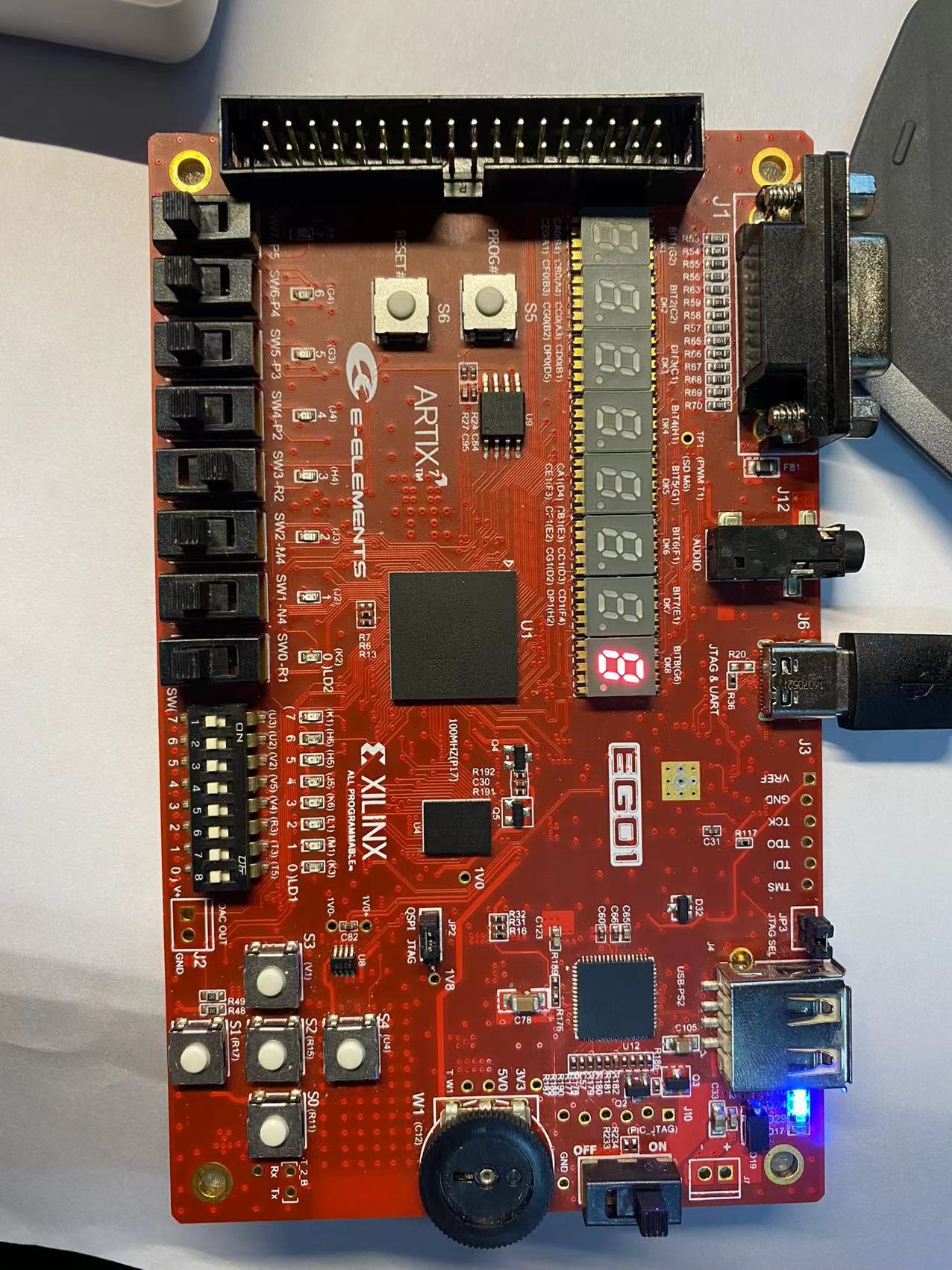
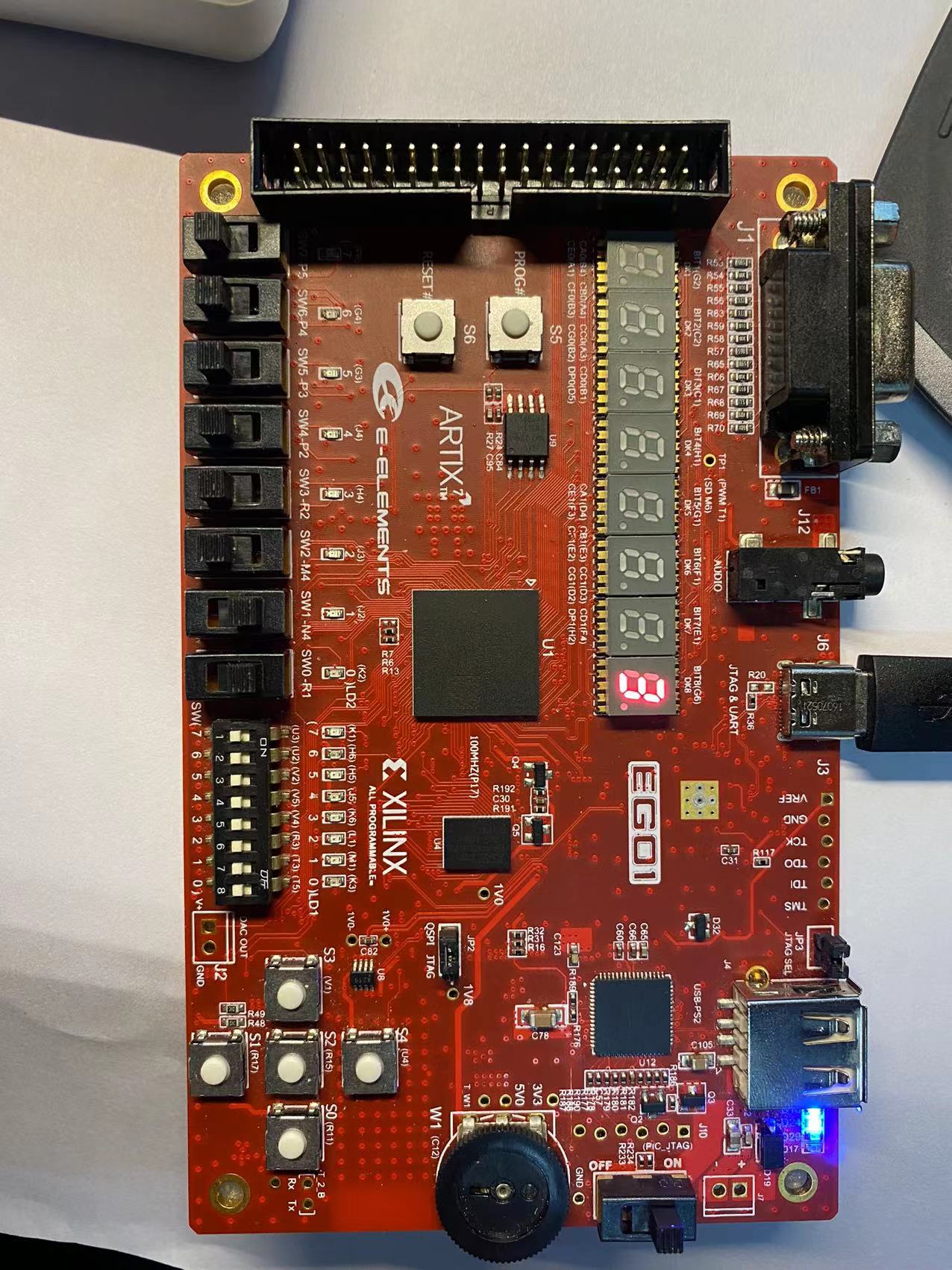
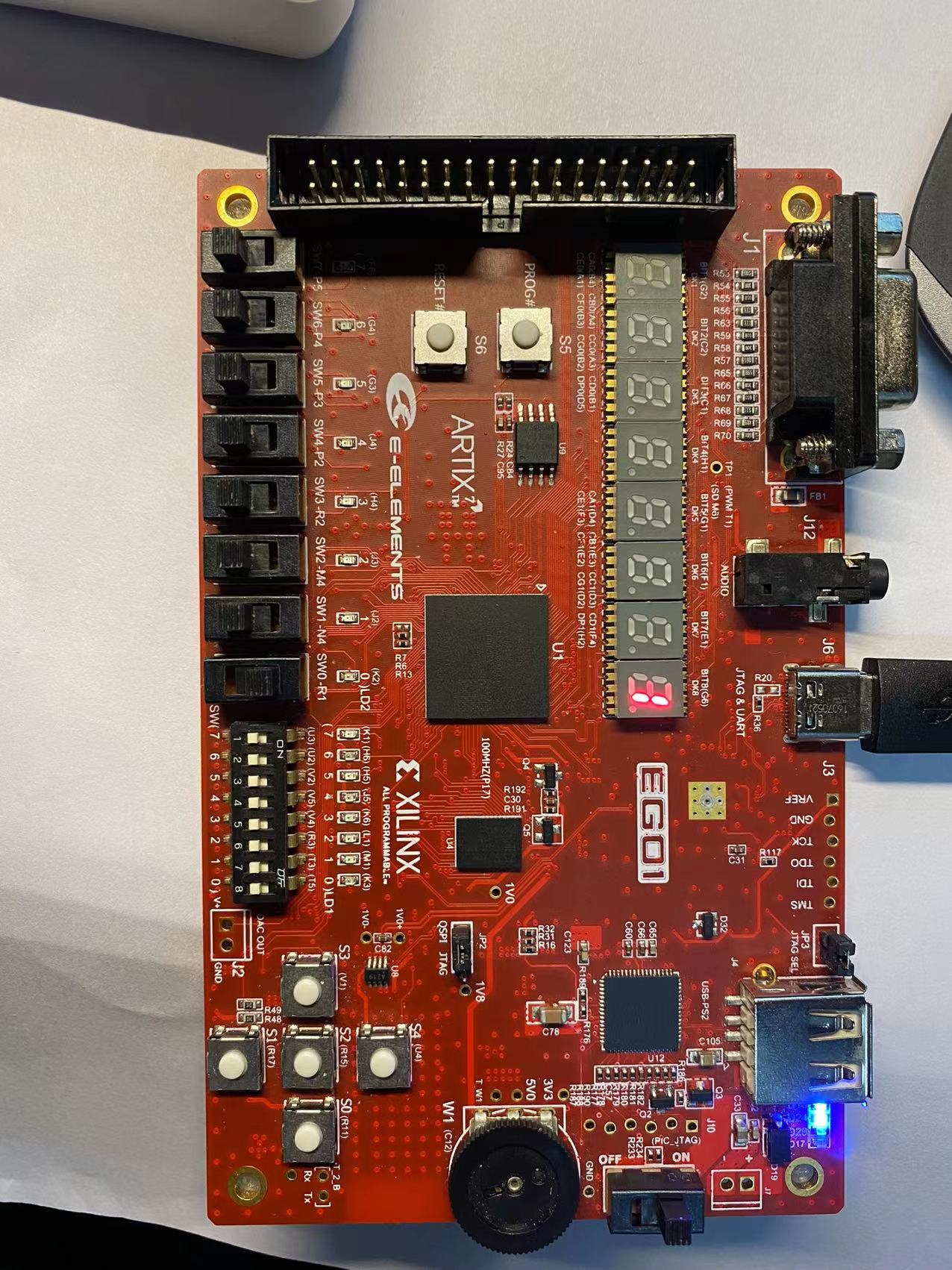
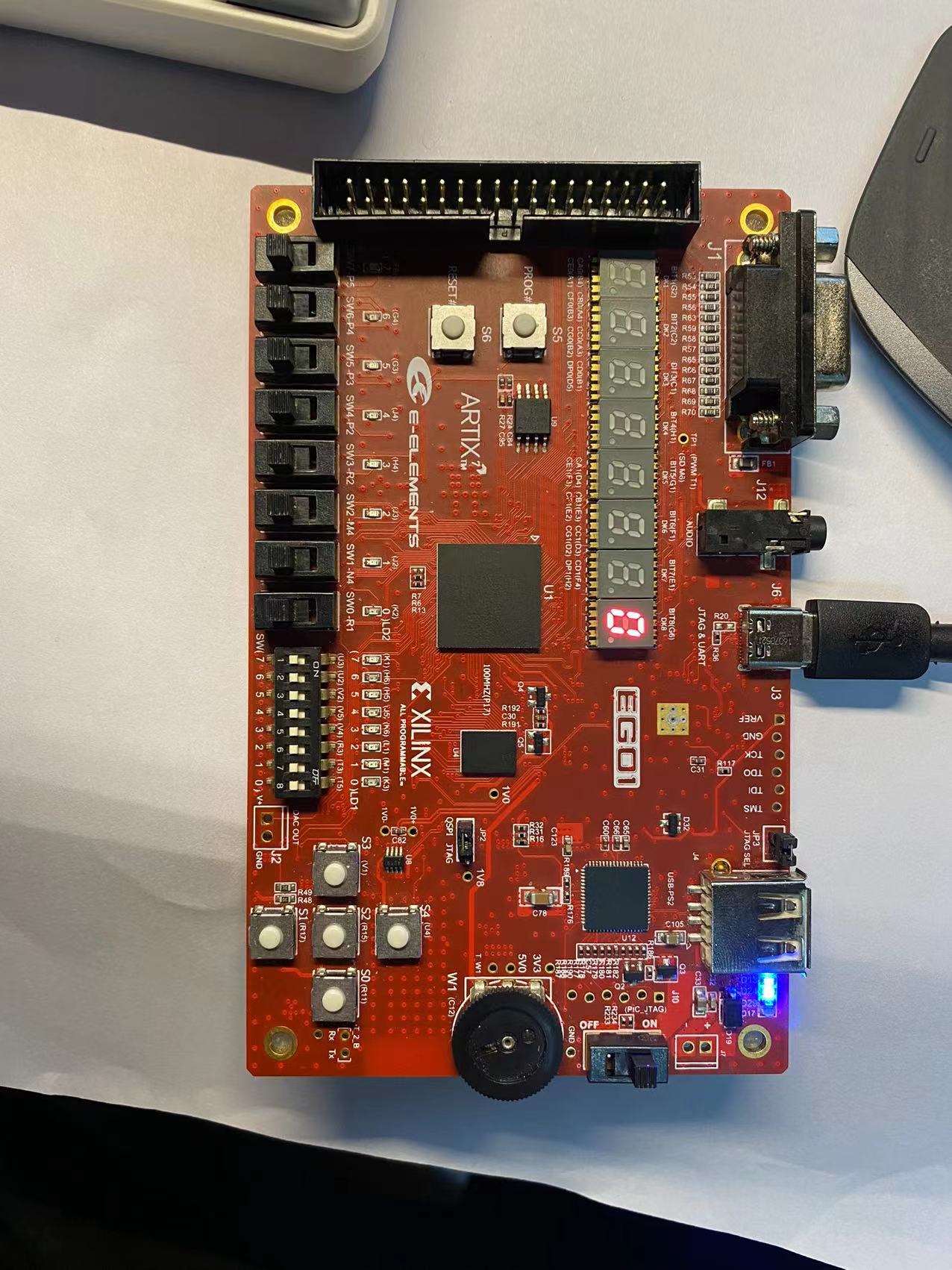
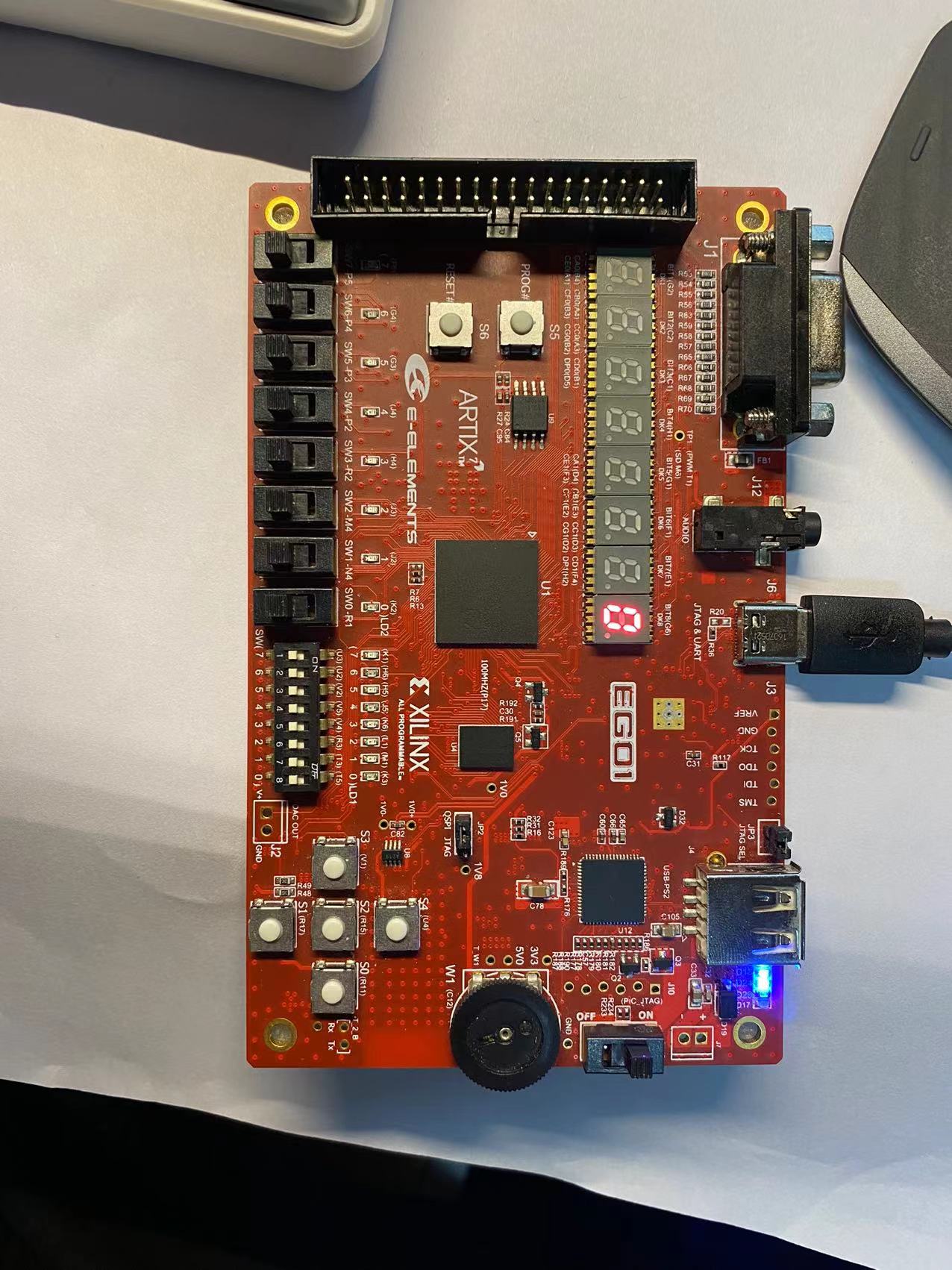


5、**实验2.2（7段数码管）的实现及验证**

（1）系统输入输出信号定义

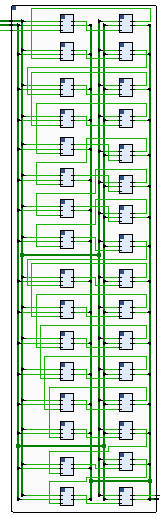


（2）板级测试验证

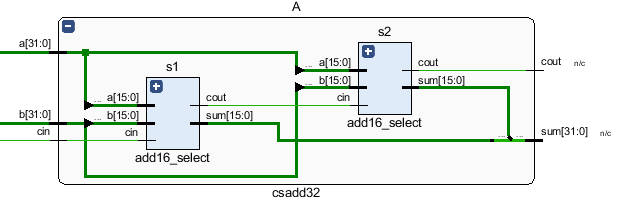


6、**实验2.3（加法器及性能比较）的实现及仿真验证**

（1）32位逐位/选择进位加法器模块RTL分析结构图及说明



逐位进位加法器



选择进位加法器

（3）测试激励的设计

`timescale 1ns / 1ps

module add32\_tb();

    reg [31:0]a;

    reg [31:0]b;

    reg cin;

    reg clk;

    wire [31:0]s0,s1,s2;

    wire cout0,cout1,cout2;

    initial begin

        a = 4'bxxxx;

        b = 4'bxxxx;

        cin = 1'bx;

        clk = 0;

    end

    always #100 clk = ~clk;

    always@(posedge clk)begin

        a={$random}%2\*\*30;

        b={$random}%2\*\*30;

        cin={$random}%2;

        #150;

        cin=1'bx;

    end

    csadd32 A(a,b,cin,s0,cout0);

    rcadd32 B(a,b,cin,s1,cout1);

endmodule

*//一位全加器*

module add1(

    input a,

    input b,

    input cin,

    output sum,

    output cout

);

    assign #4 sum = a ^ b ^ cin;

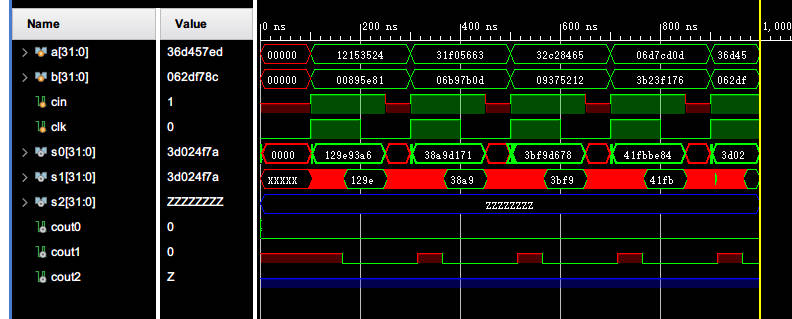
    assign #2 cout =  (cin==1)|(cin==0)?(a & cin) | (b & cin)| (a & b):1'bx;

*// assign #3 cout = (a & b) | (a & cin) | (b & cin);*

endmodule

只需要对比每次计算消耗的时间即可比较性能

（4）仿真波形及说明



观察可知s0输出结果总是比s1快，可见选择进位加法器在速度上比逐位进位加法器快

**7、实验中遇到的问题、现象及解决方法**

问题1：7段数码管res 和 msg定义

现象：I/O ports 绑定时数量不对

问题原因：检查代码发现res和msg写反了

解决方法：更正代码

**9、本次实验心得体会**

加深了对选择进位加法器的理解，以及简单分析其消耗时间的计算。

**10、附录：实现的各种32位加法器Verilog代码 + 测试激励代码**

逐位进位加法器

module rcadd32(

    input [31:0] a,

    input [31:0] b,

    input cin,

    output [31:0] sum,

    output cout

);

    wire [31:0]carry;

    add1 add1 (.a(a[0]) ,.b(b[0]) ,.cin(cin)      ,.sum(sum[0]) ,.cout(carry[0])) ;

    add1 add2 (.a(a[1]) ,.b(b[1]) ,.cin(carry[0]) ,.sum(sum[1]) ,.cout(carry[1])) ;

    add1 add3 (.a(a[2]) ,.b(b[2]) ,.cin(carry[1]) ,.sum(sum[2]) ,.cout(carry[2])) ;

    add1 add4 (.a(a[3]) ,.b(b[3]) ,.cin(carry[2]) ,.sum(sum[3]) ,.cout(carry[3])) ;

    add1 add5 (.a(a[4]) ,.b(b[4]) ,.cin(carry[3]) ,.sum(sum[4]) ,.cout(carry[4])) ;

    add1 add6 (.a(a[5]) ,.b(b[5]) ,.cin(carry[4]) ,.sum(sum[5]) ,.cout(carry[5])) ;

    add1 add7 (.a(a[6]) ,.b(b[6]) ,.cin(carry[5]) ,.sum(sum[6]) ,.cout(carry[6])) ;

    add1 add8 (.a(a[7]) ,.b(b[7]) ,.cin(carry[6]) ,.sum(sum[7]) ,.cout(carry[7])) ;

    add1 add9 (.a(a[8]) ,.b(b[8]) ,.cin(carry[7]) ,.sum(sum[8]) ,.cout(carry[8])) ;

    add1 add10(.a(a[9]) ,.b(b[9]) ,.cin(carry[8]) ,.sum(sum[9]) ,.cout(carry[9])) ;

    add1 add11(.a(a[10]),.b(b[10]),.cin(carry[9]) ,.sum(sum[10]),.cout(carry[10]));

    add1 add12(.a(a[11]),.b(b[11]),.cin(carry[10]),.sum(sum[11]),.cout(carry[11]));

    add1 add13(.a(a[12]),.b(b[12]),.cin(carry[11]),.sum(sum[12]),.cout(carry[12]));

    add1 add14(.a(a[13]),.b(b[13]),.cin(carry[12]),.sum(sum[13]),.cout(carry[13]));

    add1 add15(.a(a[14]),.b(b[14]),.cin(carry[13]),.sum(sum[14]),.cout(carry[14]));

    add1 add16(.a(a[15]),.b(b[15]),.cin(carry[14]),.sum(sum[15]),.cout(carry[15]));

    add1 add17(.a(a[16]),.b(b[16]),.cin(carry[15]),.sum(sum[16]),.cout(carry[16]));

    add1 add18(.a(a[17]),.b(b[17]),.cin(carry[16]),.sum(sum[17]),.cout(carry[17]));

    add1 add19(.a(a[18]),.b(b[18]),.cin(carry[17]),.sum(sum[18]),.cout(carry[18]));

    add1 add20(.a(a[19]),.b(b[19]),.cin(carry[18]),.sum(sum[19]),.cout(carry[19]));

    add1 add21(.a(a[20]),.b(b[20]),.cin(carry[19]),.sum(sum[20]),.cout(carry[20]));

    add1 add22(.a(a[21]),.b(b[21]),.cin(carry[20]),.sum(sum[21]),.cout(carry[21]));

    add1 add23(.a(a[22]),.b(b[22]),.cin(carry[21]),.sum(sum[22]),.cout(carry[22]));

    add1 add24(.a(a[23]),.b(b[23]),.cin(carry[22]),.sum(sum[23]),.cout(carry[23]));

    add1 add25(.a(a[24]),.b(b[24]),.cin(carry[23]),.sum(sum[24]),.cout(carry[24]));

    add1 add26(.a(a[25]),.b(b[25]),.cin(carry[24]),.sum(sum[25]),.cout(carry[25]));

    add1 add27(.a(a[26]),.b(b[26]),.cin(carry[25]),.sum(sum[26]),.cout(carry[26]));

    add1 add28(.a(a[27]),.b(b[27]),.cin(carry[26]),.sum(sum[27]),.cout(carry[27]));

    add1 add29(.a(a[28]),.b(b[28]),.cin(carry[27]),.sum(sum[28]),.cout(carry[28]));

    add1 add30(.a(a[29]),.b(b[29]),.cin(carry[28]),.sum(sum[29]),.cout(carry[29]));

    add1 add31(.a(a[30]),.b(b[30]),.cin(carry[29]),.sum(sum[30]),.cout(carry[30]));

    add1 add32(.a(a[31]),.b(b[31]),.cin(carry[30]),.sum(sum[31]),.cout(cout))     ;

endmodule

选择进位加法器

module csadd32(

        input    [31:0]a,

        input    [31:0]b,

        input     cin,

        output   [31:0]sum,

        output cout

);

        wire c;

        add16s s1(.a(a[15:0]),.b(b[15:0]),.cin(cin),.sum(sum[15:0]),.cout(c));

        add16s s2(.a(a[31:16]),.b(b[31:16]),.cin(c),.sum(sum[31:16]),.cout(cout));

endmodule

module add16s(

        input [15:0]a,

        input  [15:0]b,

        input cin,

        output  [15:0]sum,

        output cout

);

        wire[2:0] c;

        add4s s1(.a(a[3:0]),.b(b[3:0]),.cin(cin),.sum(sum[3:0]),.cout(c[0]));

        add4s s2(.a(a[7:4]),.b(b[7:4]),.cin(c[0]),.sum(sum[7:4]),.cout(c[1]));

        add4s s3(.a(a[11:8]),.b(b[11:8]),.cin(c[1]),.sum(sum[11:8]),.cout(c[2]));

        add4s s4(.a(a[15:12]),.b(b[15:12]),.cin(c[2]),.sum(sum[15:12]),.cout(cout));

endmodule

module add4s(

        input   [3:0]a,

        input   [3:0]b,

        input   cin,

        output  [3:0]sum,

        output  cout

);

        wire[2:0] c;

        add1s s1(.a(a[0]),.b(b[0]),.cin(cin ),.sum(sum[0]),.cout(c[0]));

        add1s s2(.a(a[1]),.b(b[1]),.cin(c[0]),.sum(sum[1]),.cout(c[1]));

        add1s s3(.a(a[2]),.b(b[2]),.cin(c[1]),.sum(sum[2]),.cout(c[2]));

        add1s s4(.a(a[3]),.b(b[3]),.cin(c[2]),.sum(sum[3]),.cout(cout));

endmodule

module add1s(

        input  a,

        input  b,

        input  cin,

        output sum,

        output cout

);

        wire s1,s2,c1,c2;

        add1 sel1( .a(a),.b(b),.cin(0),.sum(s1),.cout(c1));

        add1 sel2( .a(a),.b(b),.cin(1),.sum(s2),.cout(c2));

        assign sum=cin?s2:s1;

        assign cout=cin?c2:c1;

endmodule

测试激励

`timescale 1ns / 1ps

module add32\_tb();

    reg [31:0]a;

    reg [31:0]b;

    reg cin;

    reg clk;

    wire [31:0]s0,s1,s2;

    wire cout0,cout1,cout2;

    initial begin

        a = 4'bxxxx;

        b = 4'bxxxx;

        cin = 1'bx;

        clk = 0;

    end

    always #100 clk = ~clk;

    always@(posedge clk)begin

        a={$random}%2\*\*30;

        b={$random}%2\*\*30;

        cin={$random}%2;

        #150;

        cin=1'bx;

    end

    csadd32 A(a,b,cin,s0,cout0);

    rcadd32 B(a,b,cin,s1,cout1);

endmodule

*//一位全加器*

module add1(

    input a,

    input b,

    input cin,

    output sum,

    output cout

);

    assign #4 sum = a ^ b ^ cin;

    assign #2 cout =  (cin==1)|(cin==0)?(a & cin) | (b & cin)| (a & b):1'bx;

*// assign #3 cout = (a & b) | (a & cin) | (b & cin);*

endmodule