中山大学数据科学与计算机学院本科生实验报告

（2019 学年秋季学期）

课程名称：**计算机组成原理实验**　　任课教师：郭雪梅　　助教：汪庭葳、刘洋旗

年级&班级	2018 级计科一班	专业(方向)	计算机类
学号	18340013	姓名	陈琮昊
电话	15734867907	Email	1062080021@qq.com
开始日期	2019.11.6	完成日期	2019.11.11

一、实验题目：

实验 5-运算器实验

二、实验目的：

了解运算器的组成结构、掌握运算器的工作原理、掌握数码管的工作原理与使用方法

三、实验内容

1. 实验步骤：

（1）创建 ALU 模块

（2）创建 top 模块

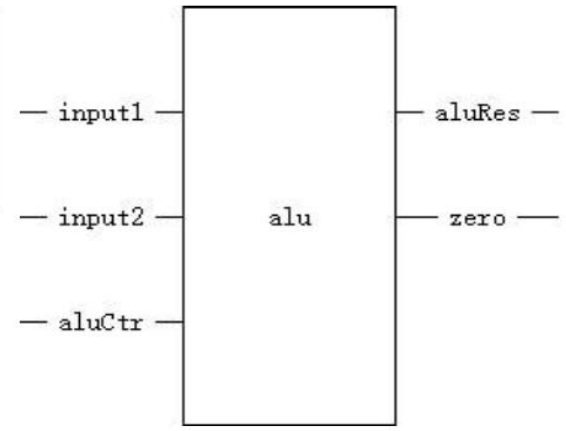
（3）创建源文件

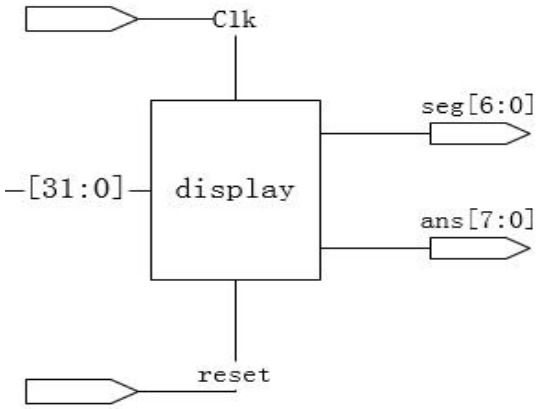
（4）创建仿真文件

（5）运行并连接硬件

2. 实验原理：

input	input1[31:0]
input	input2[31:0]
input	aluCtr[3:0]
output	aluRes[31:0]
output	zero

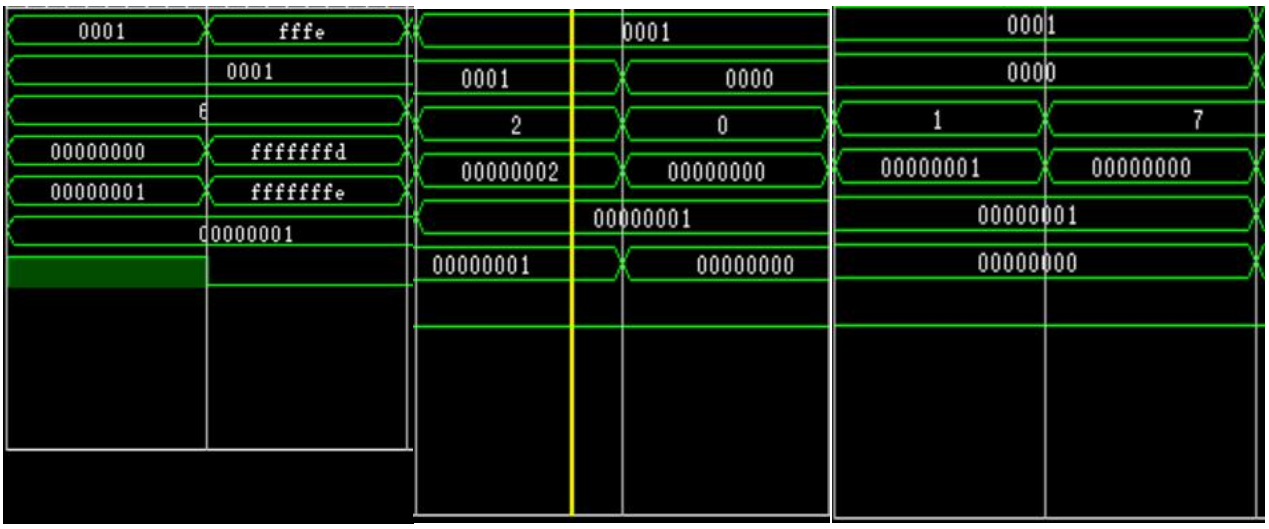




ALU 主要执行 5 种操作：与，或，加，减，小于设置。这五种操作可以使用四位的编码表示：0000，0001，0010，0110，0111。指令不同，则对应的 ALU 运算不同，所以该模块需要根据指令来控制 ALU 进行正确的运算。根据 AluCtr 的不同进行相应的运算操作，将操作结果赋值到输出变量。

四、实验结果：

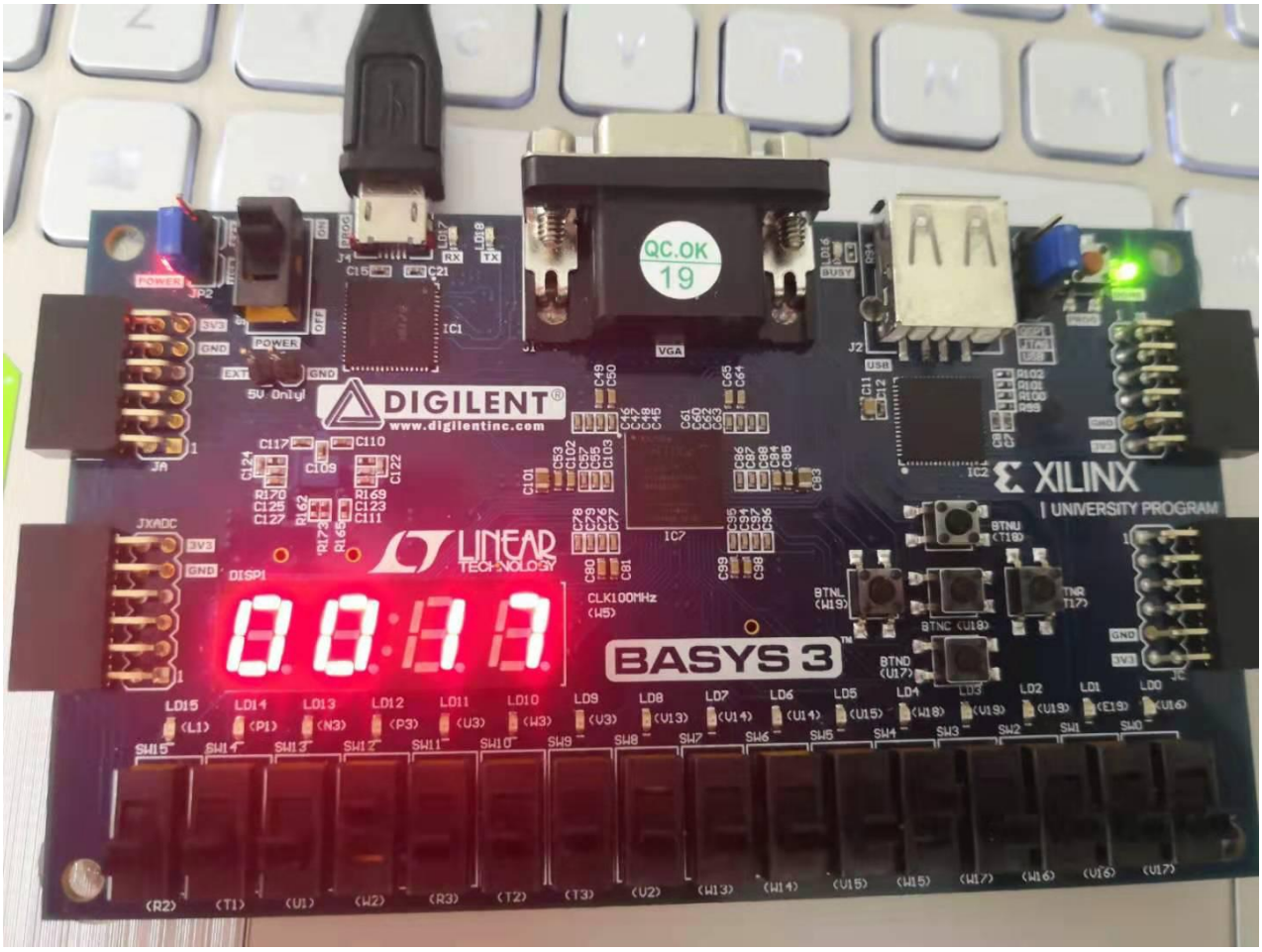
（I）仿真结果如图：

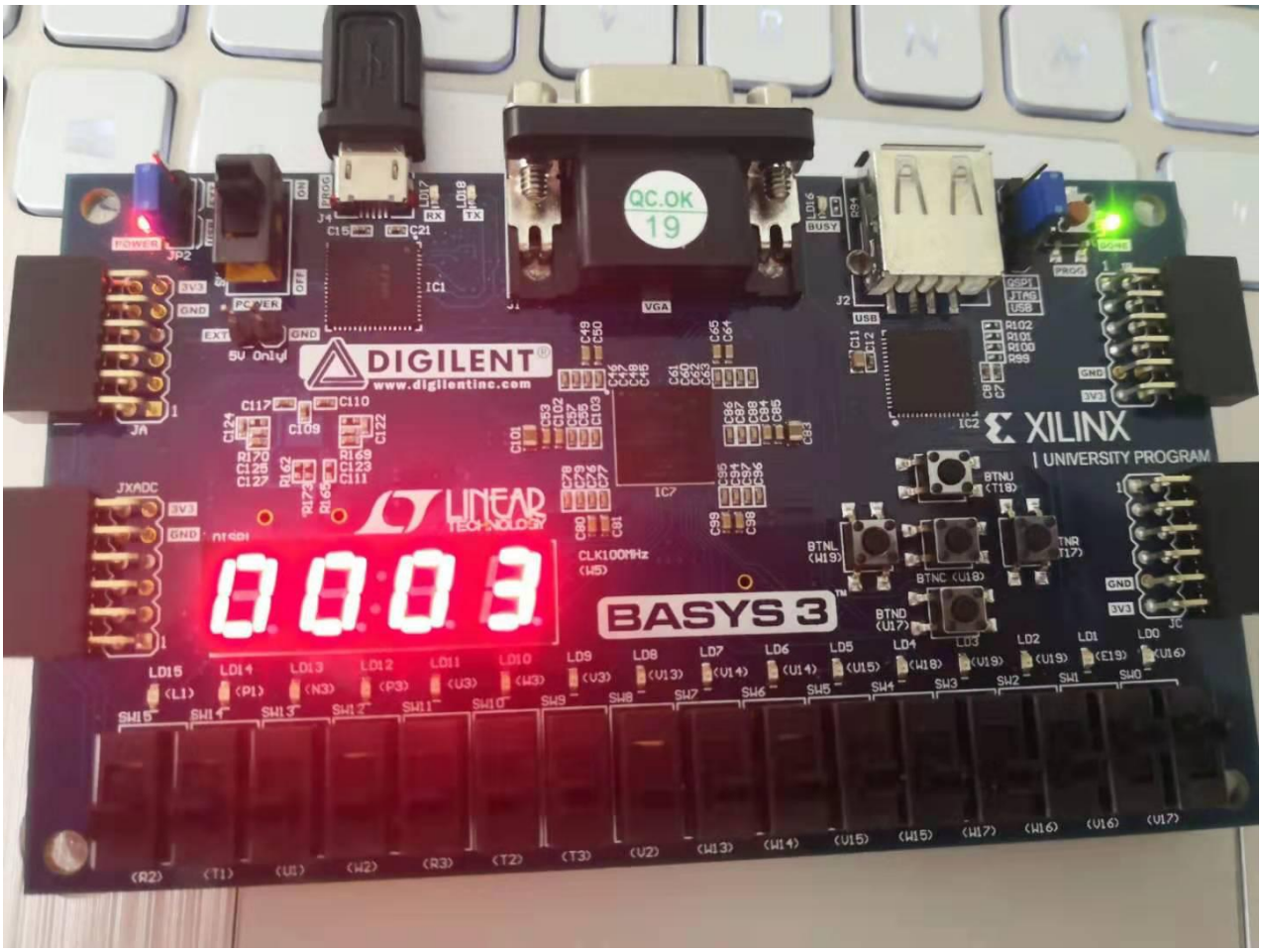


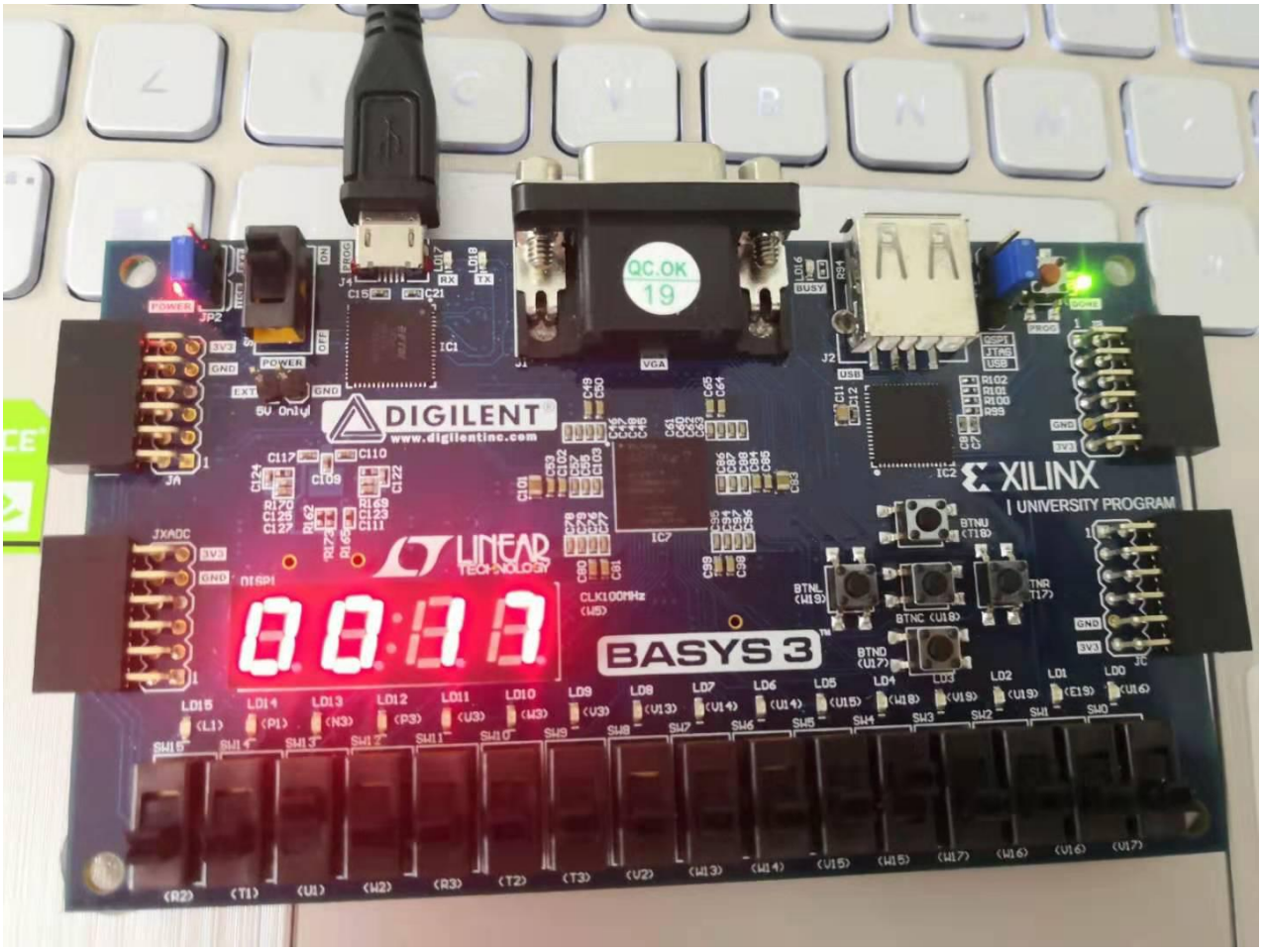
在上图中，第一行为 input1，第二行为 input2，第三行为 AluCtr，第四行为扩展后的结果，第五行为符号扩展后的 input1，第六行为符号扩展后的 input2。按照前面给的 AluCtr 的功能表格可知该结果正确。

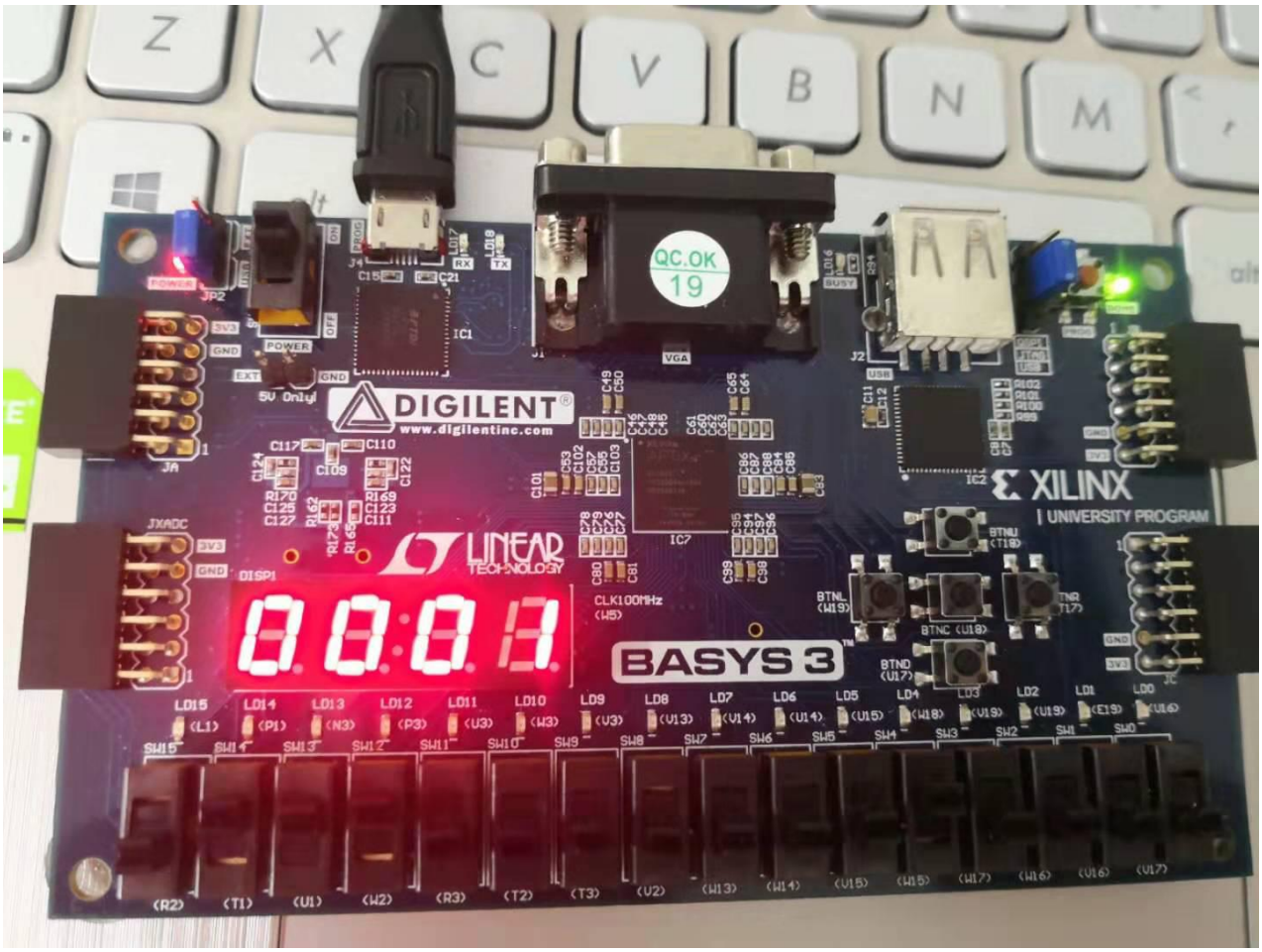
（II）烧板结果如图：

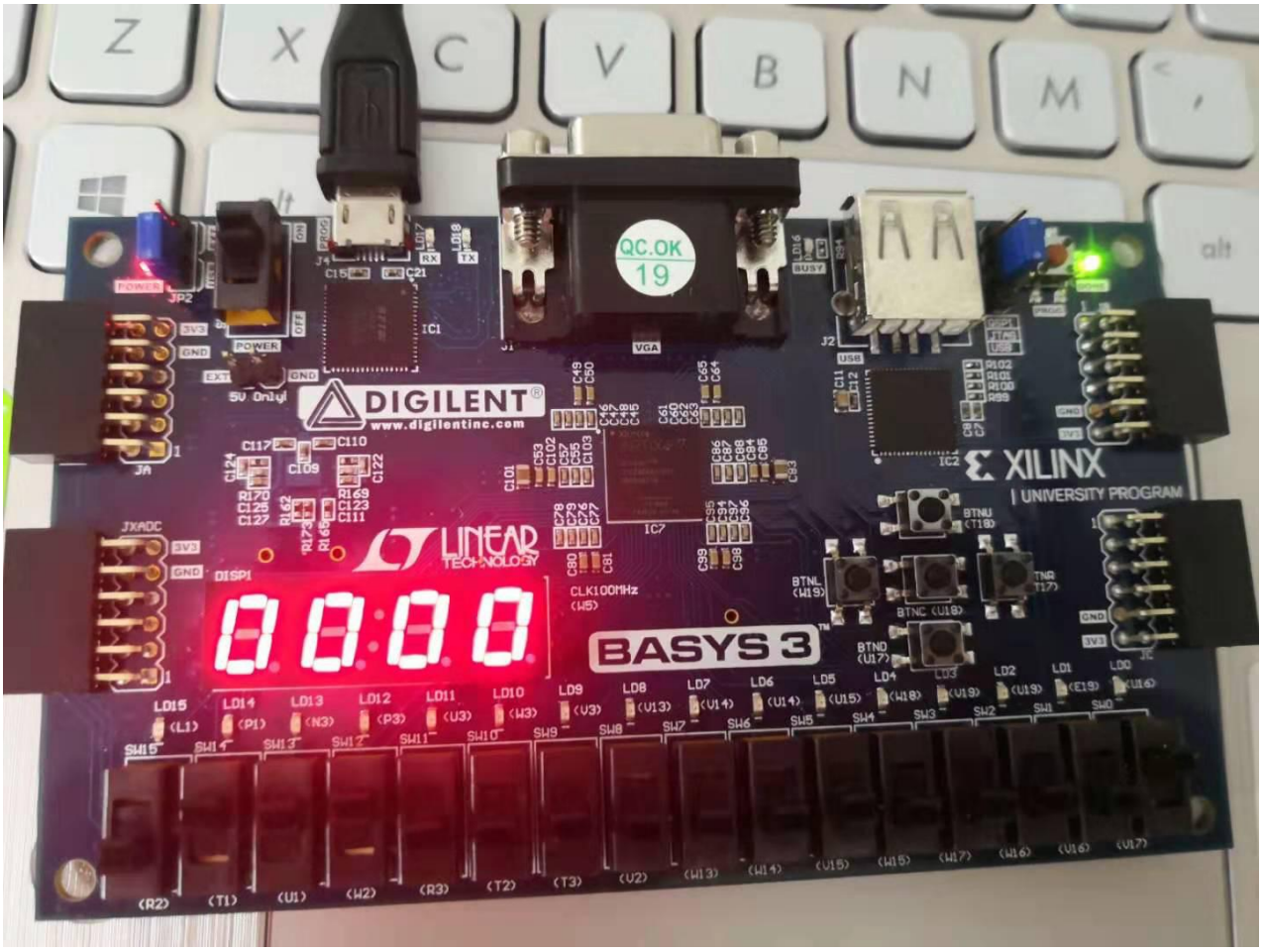
如下面几张图所示，第一张为 0007 或 0010，为 0017；第二张为 0007 与 0003，结果为 0003；第三张为 0007+0010=0017；第四张为 0007<0010 置 1，第五张为 0007>0001 置 0;第六张为 0007-0010=FFF7。其中前四个引脚（R2-W2）控制 AluCtr，后 9 个引脚（V2-V17）为 input2，V2 为符号位，后面 8 个为 input2 从高到低位。

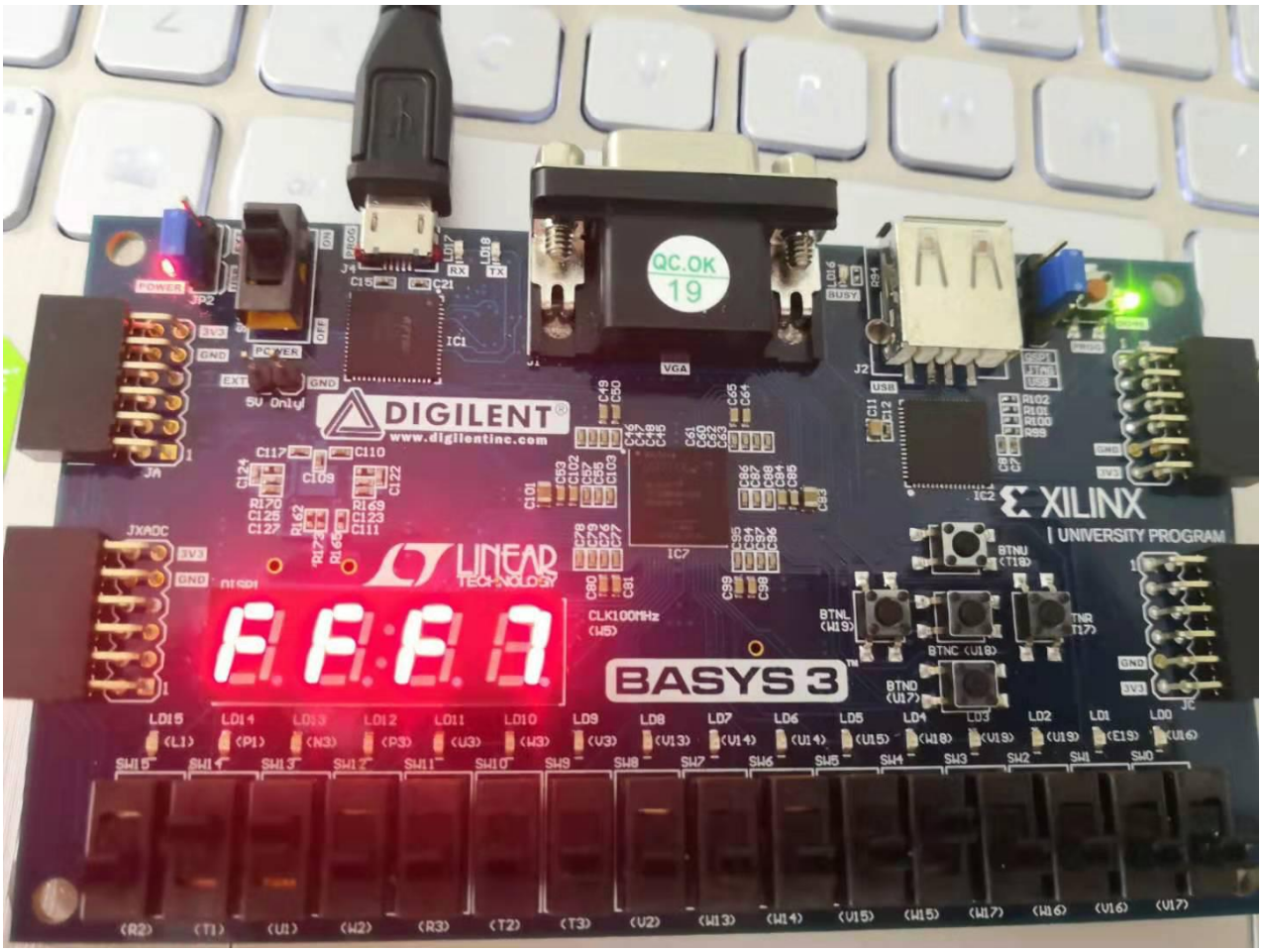

QC.OK
19
DIGILENT
www.digilentinc.com
LINEAR TECHNOLOGY
XILINX
UNIVERSITY PROGRAM
BASYS 3
CLK100MHz
(W5)
0 0:A A
LD15 LD14 LD13 LD12 LD11 LD10 LD9 LD8 LD7 LD6 LD5 LD4 LD3 LD2 LD1 LD0
SW15 SW14 SW13 SW12 SW11 SW10 SW9 SW8 SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0


QC.OK
19
DIGILENT
www.digilentinc.com
LINEAR TECHNOLOGY
XILINX
UNIVERSITY PROGRAM
BASYS 3
CLK100MHz
(W5)
0 0:0 3
LD15 LD14 LD13 LD12 LD11 LD10 LD9 LD8 LD7 LD6 LD5 LD4 LD3 LD2 LD1 LD0
SW15 SW14 SW13 SW12 SW11 SW10 SW9 SW8 SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0


QC.OK
19
DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
00:A3
CLK100MHz
(W5)
BTNU (T18)
BTNL (W19)
BTNC (U18)
BTNR (T17)
BTND (U17)
LD15 LD14 LD13 LD12 LD11 LD10 LD9 LD8 LD7 LD6 LD5 LD4 LD3 LD2 LD1 LD0
SW15 SW14 SW13 SW12 SW11 SW10 SW9 SW8 SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0
(R2) (T1) (U1) (W2) (R3) (T2) (T3) (V2) (W13) (W14) (V15) (W15) (W17) (W16) (V16) (V17)


QC.OK
19
DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
00:03
CLK100MHz
(W5)
BTNU (T18)
BTNL (W19)
BTNC (U18)
BTNR (T17)
BTND (U17)
LD15 LD14 LD13 LD12 LD11 LD10 LD9 LD8 LD7 LD6 LD5 LD4 LD3 LD2 LD1 LD0
SW15 SW14 SW13 SW12 SW11 SW10 SW9 SW8 SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0
(R2) (T1) (U1) (W2) (R3) (T2) (T3) (V2) (W13) (W14) (V15) (W15) (W17) (W16) (V16) (V17)


QC.OK
19
DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
VGA
USB
POWER
5V Only!
CLK100MHz
(W5)
DISP1
0 0:0 0


QC.OK
19
DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
VGA
USB
POWER
5V Only!
CLK100MHz
(W5)
F F:F A

五、实验感想：本次实验主要是实现 ALU。在本次实验中，前面几个步骤都很正常，没有遇到什么麻烦，只不过给的代码里面有一些小错误需要发现改正，这一点并不困难。难点在于最后一步，生成比特流时失败，于是就开始改引脚等等，改完以后发现给的代码中没有符号扩展的相关代码，需要自己写，按照老师给的代码提示需要创建一个文件、写一个子程序然后进行调用。但是尝试了很多次都失败了，这一步过不去后面烧板自然也就不能成功。所以花费了很长时间在添加子程序并调用来成功生成比特流这一步。最后终于成功烧板并显示在了数码管上。这也让我意识到应该学习 Verilog 的相关内容并熟练掌握。

附录（流程图，注释过的代码）：

（1）top. v：

```
module top(
input clk,
input reset, //复位信号（连接一个按键）
input [3:0] aluCtr,
//input [15:0] input1,
//input [15:0] input2,
input [8:0] input3,
output [6:0] seg,//段码
output [3:0] sm_wei//哪个数码管
);
// ALU 信号线
wire zero;
wire[31:0] aluRes;
//wire[15:0] expand;
// ALU 控制信号线
wire [15:0] input1;
wire [15:0] input2;
//  实例化 ALU 模块
assign input1 = 16'h0x7;
assign input2[15:9] = {7{input3[8]}};
assign input2[8:0] = input3[8:0];
wire [31:0] data1;
wire [31:0] data2;
alu alu(.input1(data1),
.input2(data2),
.aluCtr(aluCtr),
.zero(zero),
.aluRes(aluRes));
//  。。。。。。。。。。。。。。。。。。。。。。。。。实例化符号扩展模块
//signext signext(.inst(inst[15:0]), .data(expand));
signext signext0(.inst(input1[15:0]), .data(data1));
signext signext1(.inst(input2[15:0]), .data(data2));
//.....实例化数码管显示模块
```

```
display disp1 (.clk(clk),.sm_wei(sm_wei),.data(aluRes[15:0]),.sm_duan(seg));
endmodule
```

（2）alu.v

```
module alu(
input [31:0] input1,
input [31:0] input2,
input [3:0] aluCtr,
output reg[31:0] aluRes,
output reg zero
);
always @(input1 or input2 or aluCtr) // 运算数或控制码变化时操作
begin
case(aluCtr)
4'b0110: // 减
begin
aluRes = input1 - input2;
if(aluRes == 0)
zero = 1;
else
zero = 0;
end
4'b0010: // 加
aluRes = input1 + input2;
4'b0000: // 与
aluRes = input1 & input2;
4'b0001: // 或
aluRes = input1 | input2;
4'b1100: // 异或
aluRes = input1 ^ input2;
4'b0111: // 小于设置
begin
if(input1<input2)
aluRes = 1;
else
aluRes = 0;
end
default:
aluRes = 0;
endcase
end
endmodule
```

（3）signext.v（该文件是根据 top.v 蓝色所给注释提示添加的子程序）

```
module signext(
input [15:0] inst,
```

```
output [31:0] data
);
assign data = inst[15:15]?{16'hffff,inst}:{16'h0000,inst};
// 如果符号位为 1 ，则在前面添加 ffff；否则添加 0000
endmodule
```

（4）display.v：

```
module display(clk,data,sm_wei,sm_duan);
input clk;
input [15:0] data;
output [3:0] sm_wei;
output [6:0] sm_duan;
//------------------------------------------------------------
//分频
integer clk_cnt;
reg clk_400Hz; always @(posedge clk)
if(clk_cnt==32'd100000)
begin clk_cnt <= 1'b0; clk_400Hz <= ~clk_400Hz;
end else clk_cnt <= clk_cnt + 1'b1;
//------------------------------------------------------------
//位控制
reg [3:0]wei_ctrl=4'b1110; always @(posedge clk_400Hz)
wei_ctrl <= {wei_ctrl[2:0],wei_ctrl[3]};
//段控制
reg [3:0]duan_ctrl;
always @(wei_ctrl, data)
case(wei_ctrl)
4'b1110:duan_ctrl=data[3:0];
4'b1101:duan_ctrl=data[7:4];
4'b1011:duan_ctrl=data[11:8];
4'b0111:duan_ctrl=data[15:12];
default:duan_ctrl=4'hf;
  endcase
//------------------------------------------------------------
//解码模块
reg [7:0]duan;
always @(duan_ctrl)
case(duan_ctrl)
4'h0:duan=7'b100_0000;//0
4'h1:duan=7'b111_1001;//1
4'h2:duan=7'b010_0100;//2
4'h3:duan=7'b011_0000;//3
4'h4:duan=7'b001_1001;//4
4'h5:duan=7'b001_0010;//5
4'h6:duan=7'b000_0010;//6
4'h7:duan=7'b111_1000;//7
```

```
4'h8:duan=7'b000_0000;//8
4'h9:duan=7'b001_0000;//9
4'ha:duan=7'b000_1000;//a
4'hb:duan=7'b000_0011;//b
4'hc:duan=7'b100_0110;//c
4'hd:duan=7'b010_0001;//d
4'he:duan=7'b000_0111;//e
4'hf:duan=7'b000_1110;//f
// 4'hf:duan=7'b111_1111;//不显示
default : duan = 7'b100_0000;//0
endcase
//-------------------------------------------------------
assign sm_wei = wei_ctrl;
assign sm_duan = duan;
endmodule
```

（5）仿真文件：

```
module alusim;
// Inputs
reg [15:0] input1;
reg [15:0] input2;
reg [3:0] aluCtr;
// Outputs
wire [15:0] aluRes;
wire zero;
// Instantiate the Unit Under Test (UUT)
signext uut0 (
.inst(input1),
.data(data1)
);
signext uut1 (
.inst(input2),
.data(data2)
);
alu uut (
.input1(data1),
.input2(data2),
.aluCtr(aluCtr),
.aluRes(aluRes),
.zero(zero)
);
initial begin
// Initialize Inputs
input1 = 1;
input2 = 1;
aluCtr = 4'b0110;
```

```
#100;
input1 = 2;
input2 = 1;
aluCtr = 4'b0110;
#100
input1 = 1;
input2 = 1;
aluCtr = 4'b0010;
#100
input1 = 1;
input2 = 0;
aluCtr = 4'b0000;
#100
input1 = 1;
input2 = 0;
aluCtr = 4'b0001;
#100
input1 = 1;
input2 = 0;
aluCtr = 4'b0111;
#100
input1 = 0;
input2 = 1;
aluCtr = 4'b0111;
end
endmodule
```

（6）约束文件：

```
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports reset]
set_property PACKAGE_PIN U18 [get_ports reset]
set_property IOSTANDARD LVCMOS33 [get_ports {aluCtr[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {aluCtr[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {aluCtr[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {aluCtr[0]}]
set_property PACKAGE_PIN R2 [get_ports {aluCtr[3]}]
set_property PACKAGE_PIN T1 [get_ports {aluCtr[2]}]
set_property PACKAGE_PIN U1 [get_ports {aluCtr[1]}]
set_property PACKAGE_PIN W2 [get_ports {aluCtr[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {input3[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {input3[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {input3[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {input3[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {input3[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {input3[3]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {input3[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {input3[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {input3[0]}]
set_property PACKAGE_PIN V2 [get_ports {input3[8]}]
set_property PACKAGE_PIN W13 [get_ports {input3[7]}]
set_property PACKAGE_PIN W14 [get_ports {input3[6]}]
set_property PACKAGE_PIN V15 [get_ports {input3[5]}]
set_property PACKAGE_PIN W15 [get_ports {input3[4]}]
set_property PACKAGE_PIN W17 [get_ports {input3[3]}]
set_property PACKAGE_PIN W16 [get_ports {input3[2]}]
set_property PACKAGE_PIN V16 [get_ports {input3[1]}]
set_property PACKAGE_PIN V17 [get_ports {input3[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
set_property PACKAGE_PIN U7 [get_ports {seg[6]}]
set_property PACKAGE_PIN V5 [get_ports {seg[5]}]
set_property PACKAGE_PIN U5 [get_ports {seg[4]}]
set_property PACKAGE_PIN V8 [get_ports {seg[3]}]
set_property PACKAGE_PIN U8 [get_ports {seg[2]}]
set_property PACKAGE_PIN W6 [get_ports {seg[1]}]
set_property PACKAGE_PIN W7 [get_ports {seg[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sm_wei[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sm_wei[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sm_wei[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sm_wei[0]}]
set_property PACKAGE_PIN U2 [get_ports {sm_wei[0]}]
set_property PACKAGE_PIN U4 [get_ports {sm_wei[1]}]
set_property PACKAGE_PIN V4 [get_ports {sm_wei[2]}]
set_property PACKAGE_PIN W4 [get_ports {sm_wei[3]}]
```