《ＥＤＡ技术》试验

实验五层次化电路设计与实现

专业：　　计算机科学与技术

姓名：　　高星杰

学号：　　2021307220712

班级：　　计科 2102

报告上交时间：2023 年　4 月 3 日

一、实验目的

1、层次化电路设计与实现

二、实验任务及要求

任务一：4 位累加器的设计、仿真与实现

要求：用层次化的电路设计方法；累加器要求用按键输入数据，用七段数码管显示结果、且要求用 signatap 抓取感兴趣的信号；

提示：可参考以下 8 位累加器的设计与实现方法来做

任务二：设计与实现一个 4 位的算术逻辑运算单元 ALU（要求实现加法、减法、与、或、异或、非、逻辑左移、逻辑右移、算术右移）。要求：（1）要求用层次化的电路设计方法；如加法器、减法器等分别做一个单独的模块，其它运算也可以做单独的模块。

学有余力的同学可选做：

任务三：某三层楼房的楼梯通道共用一盏灯，每层楼都安装了一只开关并能独立控制该灯，请设计楼道灯的控制电路。要求：采用图形输入逻辑功能描述。

任务四：在任务 1 的基础上，增加控制要求，灯打开后，延时若干秒自动关闭，请重新设计楼道灯的控制电路。要求：采用 Verilog 代码输入逻辑功能描述。

三、实验原理与步骤

任务一：4 位累加器的设计、仿真与实现

原理：分成层次设计：显示模块、加法器、分频器。然后最后结合在一起，每个模块我们都已经实现过了，再结合一起就可以了。

Verilog 代码：

```
module task1(in,out,clk,clear,cin,cout);
(*chip_pin="Y23,Y24,AA22,AA23"*)input [3:0]in;
(*chip_pin="AA14,AG18,AF17,AH17,AG17,AE17,AD17"*)output [6:0]out;
(*chip_pin="Y2"*)input clk;
(*chip_pin="AA24"*)input clear;
(*chip_pin="AB23"*)input cin;
reg[35:0] num;
wire [3:0] data;
wire [3:0] sum;
output cout;
reg flags;
reg4(data,sum,clk,clear);
always@(posedge clk)
begin
    num<=(num+1)%50000000;
    if(num==1)begin
       flags<=1;
    end
    else flags<=0;
end
add4(flags,sum,cout,in,data,cin);
show(sum,out);
endmodule
```

```
module show(in1,out);
   input [3:0] in1;
   output reg[6:0] out;
   always@(in1)
   begin
      case(in1)
      4'b0000:out<=7'b1000000;
      4'b0001:out<=7'b1111001;
      4'b0010:out<=7'b0100100;
      4'b0011:out<=7'b0110000;
      4'b0100:out<=7'b0011001;
      4'b0101:out<=7'b0010010;
      4'b0110:out<=7'b0000010;
      4'b0111:out<=7'b1111000;
      4'b1000:out<=7'b0000000;
      4'b1001:out<=7'b0010000;
      4'b1010:out<=7'b0001000;
      4'b1011:out<=7'b0000011;
      4'b1100:out<=7'b1000110;
      4'b1101:out<=7'b0100001;
      4'b1110:out<=7'b0000110;
      4'b1111:out<=7'b0001110;
      default:out<=7'b1111111;
      endcase;
   end
endmodule
```

```
module add4(flags,sum,cout,b,a,cin);
output reg [3:0]sum;
input flags;
output reg cout;
input [3:0]a;
input [3:0]b;
input cin;
always@(flags)
begin
   if(flags==1){cout,sum}=a+b+cin;
end
endmodule
```

任务二：设计与实现一个 4 位的算术逻辑运算单元 ALU

实验原理：同样是结构化设计将具体要实现的功能分层设计编写包括：加法、减法、与、或、异或、非、逻辑左移、逻辑右移、算术右移等模块，还包括显示模块。

Verilog 程序：

```
wire [3:0]subres;
reg [3:0] res;
add4 addit(aluc,addres,a,b);
sub4(aluc,subres,a,b);
yu4(aluc,yures,a,b);
huo4(aluc,huores,a,b);
yihuo4(aluc,yihuo,a,b);
fei(aluc,feires,a,b);
youyi(aluc,youyires,a,b);
zuoyi(aluc,zuoyires,a,b);
youyisuan(aluc,youyisuanres,a,b);
always@(aluc)
begin
   case(aluc)
   4'd0:res<=addres;
   4'd1:res<=subres;
   4'd2:res<=yures;
   4'd3:res<=huores;
   4'd4:res<=yihuo;
   4'd5:res<=feires;
   4'd6:res<=youyires;
   4'd7:res<=zuoyires;
   4'd8:res<=youyisuanres;
   endcase;
end
show(res,out);
endmodule
```

显示模块：

```
module show(in1,out);
   input [3:0] in1;
   output reg[6:0] out;
   always@(in1)
   begin
      case(in1)
      4'b0000:out<=7'b1000000;
      4'b0001:out<=7'b1111001;
      4'b0010:out<=7'b0100100;
      4'b0011:out<=7'b0110000;
      4'b0100:out<=7'b0011001;
      4'b0101:out<=7'b0010010;
      4'b0110:out<=7'b0000010;
      4'b0111:out<=7'b1111000;
      4'b1000:out<=7'b0000000;
      4'b1001:out<=7'b0010000;
      4'b1010:out<=7'b0001000;
      4'b1011:out<=7'b0000011;
      4'b1100:out<=7'b1000110;
      4'b1101:out<=7'b0100001;
      4'b1110:out<=7'b0000110;
      4'b1111:out<=7'b0001110;
      default:out<=7'b1111111;
      endcase;
   end
endmodule
```

```
module yihuo4(flags,sum,a,b);
input [3:0]flags;
output reg [3:0]sum;
input [3:0]a;
input [3:0]b;
always@(flags)
begin
    if(flags==4)sum=a^b;
end
endmodule
```

```
module youyi(flags,sum,a,b);
input [3:0]flags;
output reg [3:0]sum;
input [3:0]a;
input [3:0]b;
always@(flags)
begin
    if(flags==6)sum=a<<b;
end
endmodule
```

```
module zuoyi(flags,sum,a,b);
input [3:0]flags;
output reg [3:0]sum;
input [3:0]a;
input [3:0]b;
always@(flags)
begin
    if(flags==7)sum=a>>b;
end
endmodule
```

```
module huo4(flags,sum,a,b);
input [3:0]flags;
output reg [3:0]sum;
input [3:0]a;
input [3:0]b;
always@(flags)
begin
    if(flags==3)sum=a|b;
end
endmodule
```

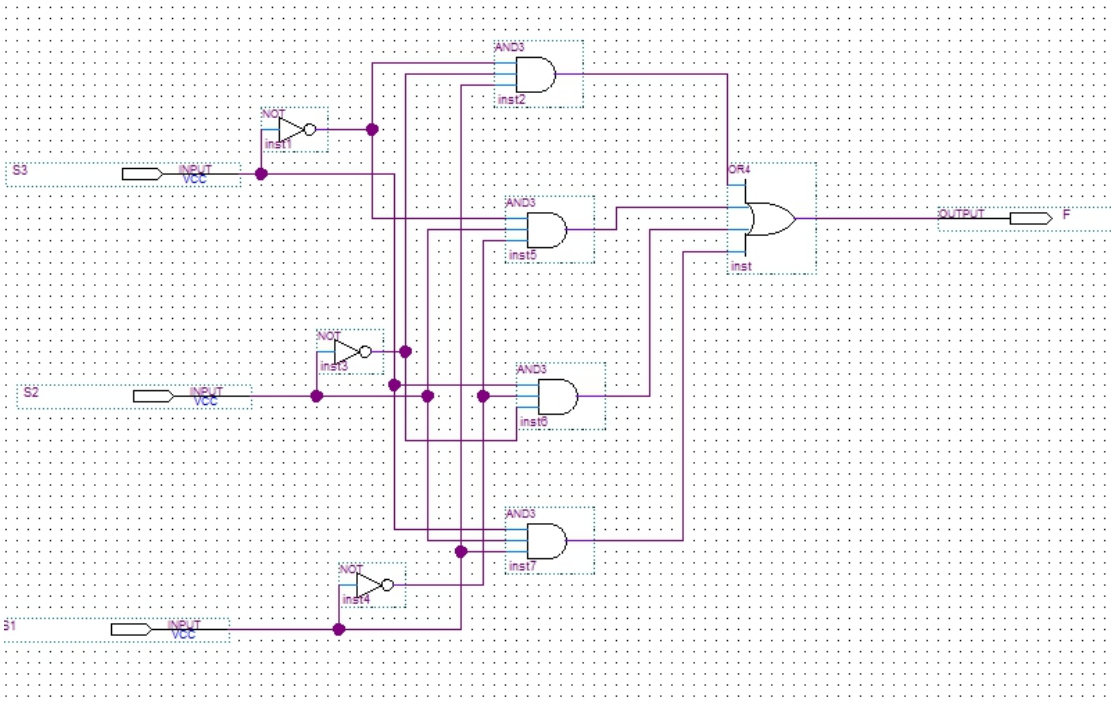
```
module youyisuan(flags,sum,a,b);
input [3:0]flags;
output reg [3:0]sum;
input [3:0]a;
input [3:0]b;
always@(flags)
begin
    if(flags==8)sum=a>>>b;
end
endmodule
```

模块较多就不一一展示了，因为都大同小异。

附加题：任务三：楼道灯的控制电路

实验原理：由于本任务要求要用电路图设计所以，根据功能写出来真值表，然后根据真值表写出来逻辑表达式，然后根据逻辑表达式画出电路图就可以实现。

电路图：



任务四：楼道灯控制电路延时若干秒自动关闭

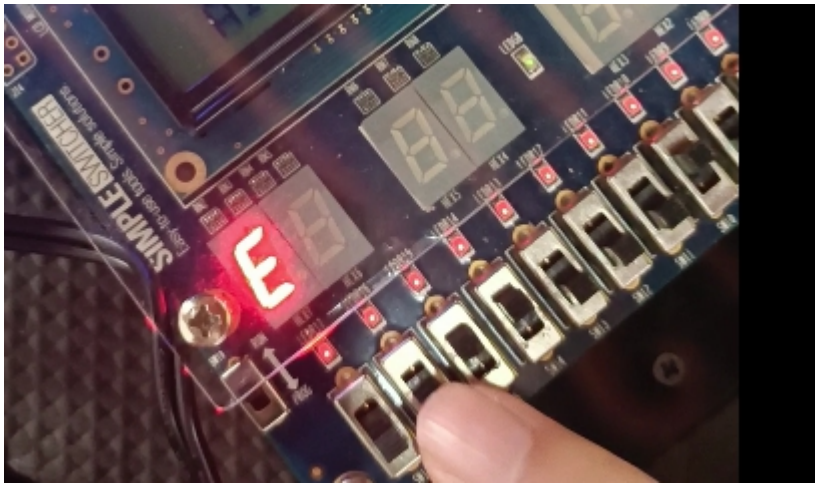
实现原理就是：同样是在实验三的基础上增加一个分频器，在亮了之后计时我设置的分频器是 1s 后灭。

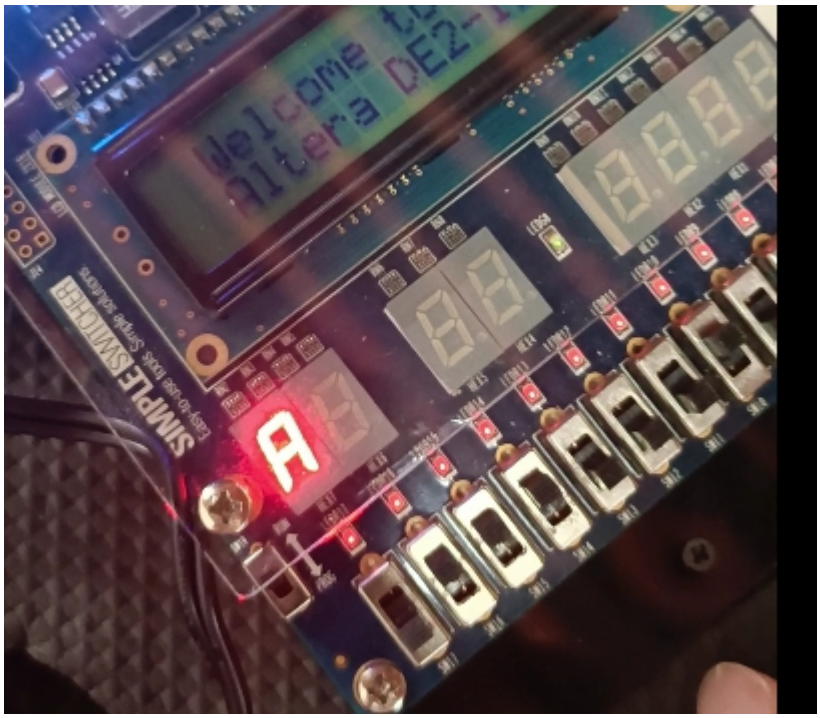
Verilog 程序：

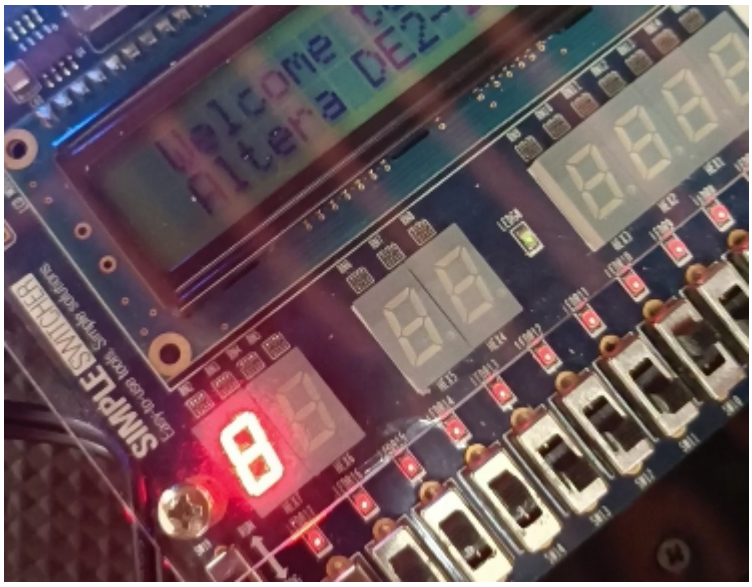
```
module task4(s1,s2,s3,f,clk);
(*chip_pin="Y23,Y24,AA22"*)input s1,s2,s3;
(*chip_pin="Y2"*)input clk;
(*chip_pin="H15"*)output reg f;
reg [35:0]num;
always@(posedge clk)
begin
   if(f==1)begin
      num=(num+1)%50000000;
   end
   if(num==50000000-1)begin
      num<=0;
      f<=0;
   end
   case({s3,s2,s1})
      3'b111:f<=1;
      3'b100:f<=1;
      3'b010:f<=1;
      3'b001:f<=1;
      default:f<=0;
      endcase
end
endmodule
```

四、实验结果与分析

任务一：（因为当时是拍的视频所以图片是在视频中截图的，可能尺寸不太规范）



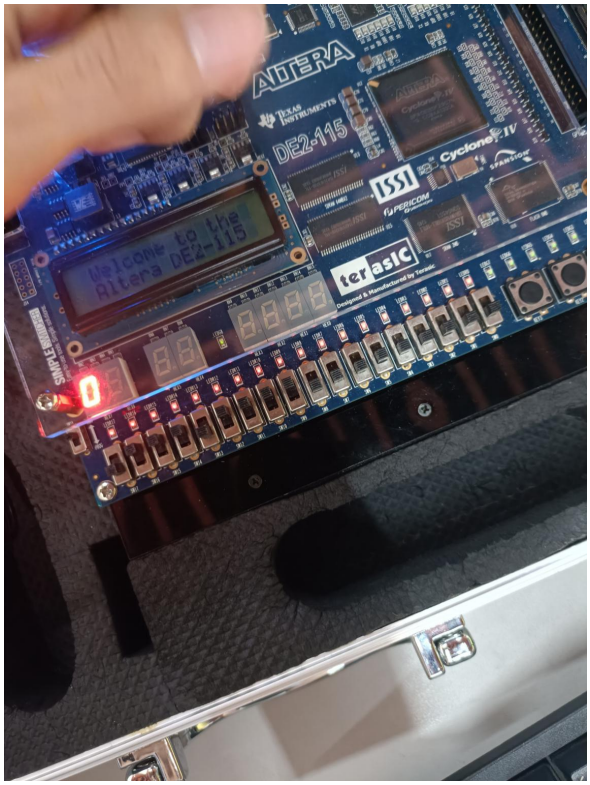


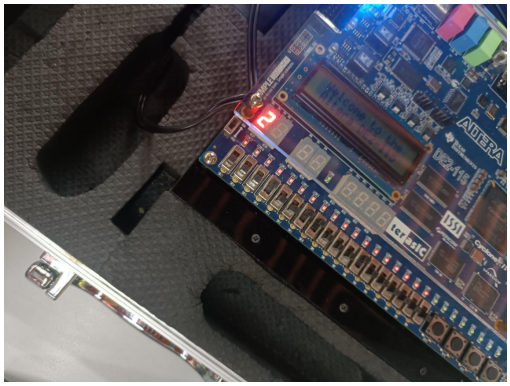


可以看出是输入是 2 开始了以 2 为单位进行的累加，一秒加一次。8，10，中间改为了以 4 递增就变成了 14。所以认为是正确的。

任务二：

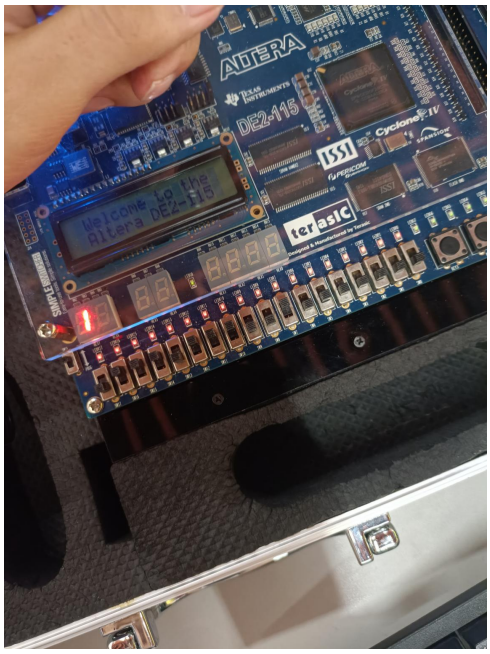
相减：7-7



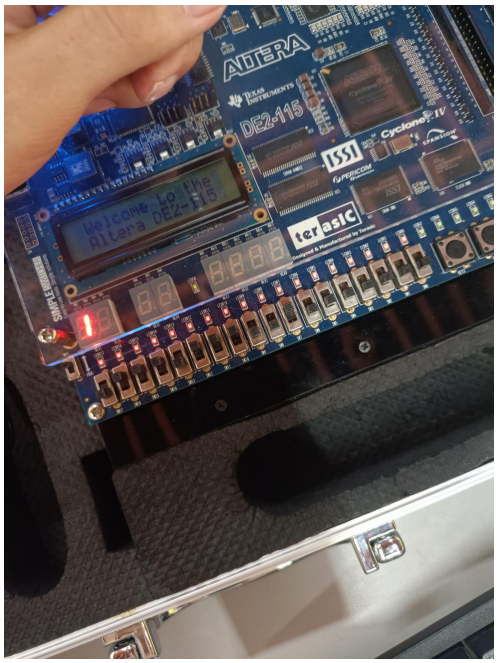


（3-1=2）

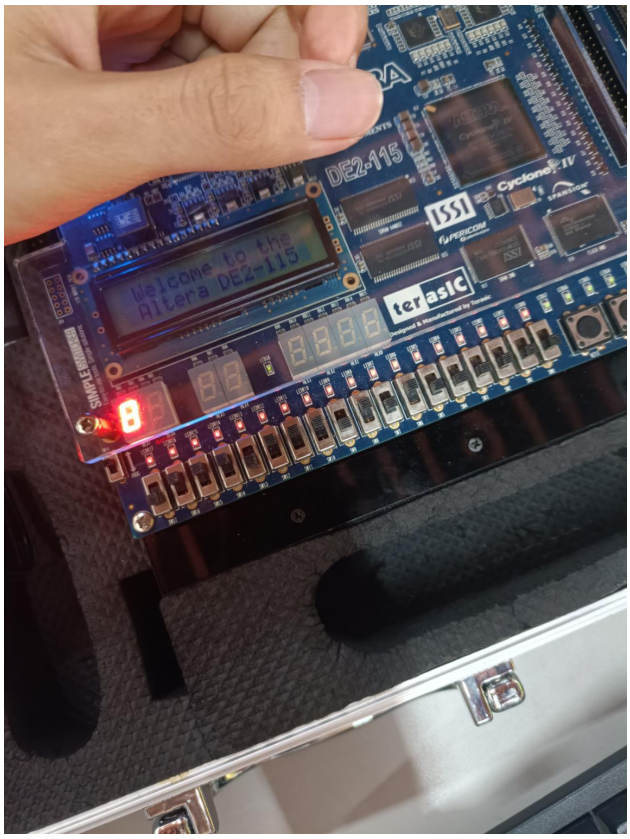
与：7 与 7



算数右移：



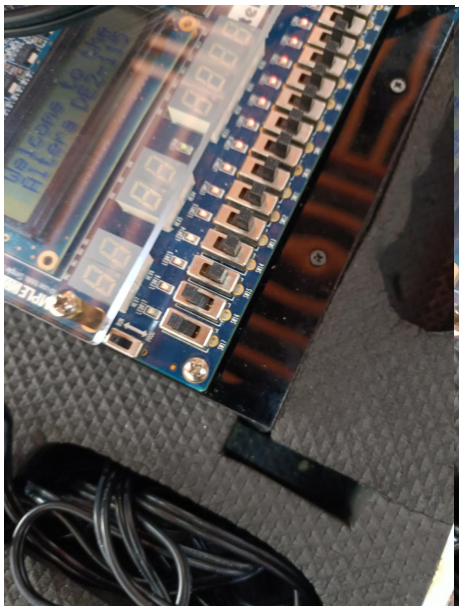
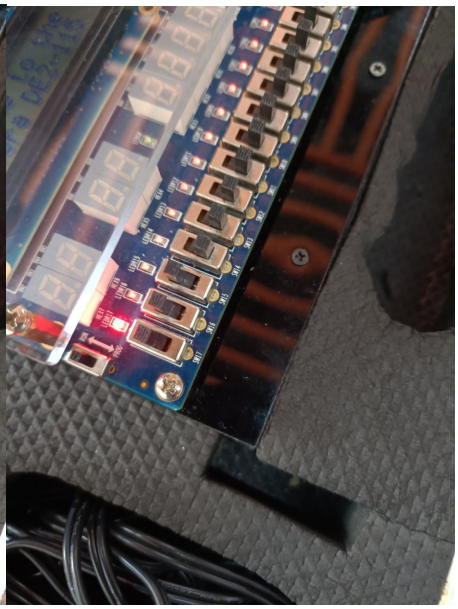
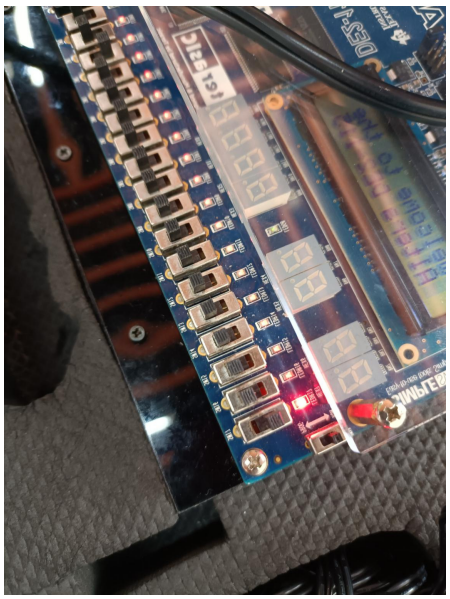
非运算：



剩下的功能由于图片数量太多未展示。

根据实验结果可以认为实验结果正确。

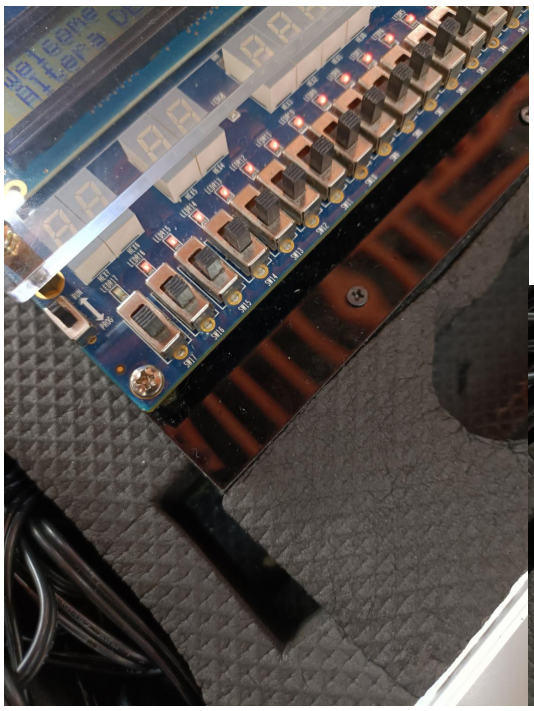
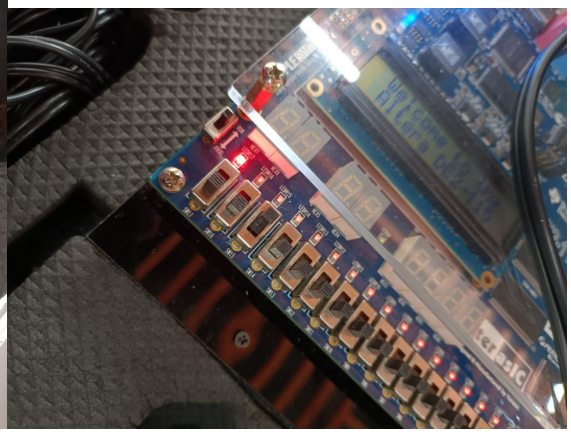
任务三：

可以根据真值表看在 111 时候亮 100 时候亮所以可以认为任务完成

（其他结果的图片数量太多所以没有展示）

任务四：

灯在亮了一秒后就灭了，所以可以认为实现了功能。

四、实验体会与讨论

本次实验主要卡在了任务一中的对加法器的设计，要设计一个参数来实现只有分频器过了 1s 再进行加，出了许多奇奇怪怪的 bug，真的是不写不知道，一写就出问题，本来以为很简单的事情，结果还是出了几次 bug，改就花费了很多时间

本次实验又用了多次层次化设计方法，将原本大的功能分为小的功能再来实验和测试就方便很多。也是在一次次的 bug 中找到了技巧感觉逐渐了解到了 verilog 程序的思想了，经过检查也是不断增加了自信，不断发现了其中的乐趣。