

数字电路与数字系统实验

实验三 ALU

姓名： 你猜

学号： 你猜

班级： 你猜

邮箱： 你猜

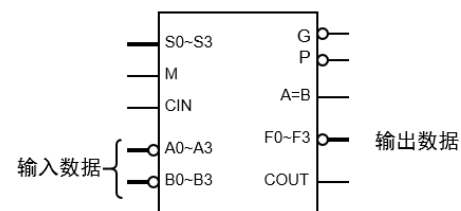
实验时间： 你猜

一、实验目的

学习简单加减法运算器的原理和实现方法，通过 Verilog 语言设计和实现一个带有逻辑运算的简单 ALU。

二、实验原理

一个算术逻辑单元 ALU 能够对 2 个 n 位的操作数进行若干不同的算术和逻辑操作，由一组功能选择输入来指定要执行的操作。现在的 ALU 基本都是用二进制补码的形式来进行操作数的表示和操作。



ALU 的输入包括操作数和指令代码（来指定要执行的操作），输出是运算结果。同时，ALU 也输入或输出 EFLAGS 的一些标志位如溢出、进位、ZF 等。

三、实验环境

Quartus 18.1、FPGA 开发板

四、实验过程

设计思路：

设计三个输入分别为 select、a、b，通过 case 语句进行功能的选择，a 和 b 为两个操作数。对于加法，进位 carry 和结果 res 直接通过拼接得到， $\{carry, res\} = a + b$ 。溢出可以通过

$$Overflow = (in_{x_{n-1}} == in_{y_{n-1}}) \&\& (out_{s_{n-1}} \neq in_{x_{n-1}}) \quad (3-1)$$

其判断原理是：如果两个参加加法运算的变量符号相同，而运算结果的符号与其不相同，则运算结果不准确，产生溢出。即两个正数相加结果为负数，或者得到。判断结果是否为 0 可以通过一元约简运算符 |，对结果的每一位进行或操作，最后取反得到。对于减法，将减数取补码后，其他与加法相同。其他操作如取反、与、或、异或、比较大小等都相对简单，根据要求进行即可。最后将 a、b 和结果分别通过数码管显示出来。

设计代码:

```
module alu(select,a,b,res,carry,overflow,zero,hex0,hex1,hex2,hex3,hex4,hex5,negative);
input [2:0] select;
input [3:0] a,b;

output reg [6:0] hex0, hex1, hex2, hex3, hex4, hex5;
output reg [3:0] res;
output reg carry, overflow, zero, negative;

integer i;
reg [3:0] c, neg_res, neg_a, neg_b;

always @ (select or a or b) begin
    case (select)
        0: begin {carry, res} = a + b;
            overflow = (a[3] == b[3] && a[3] != res[3]);
            zero = ~(|res);
            negative = 0;
            if(res[3]==1) negative=1;end
        1: begin c = ~b + 1;
            {carry, res} = a + c;
            overflow = (a[3] == c[3] && a[3] != res[3]);
            zero = ~(|res);
            negative = 0;
            if(res[3]==1) negative=1;end
        2: begin carry = 0; overflow = 0; negative = 0;
            res = ~a;
            zero = ~(|res);
            if(res[3]==1) negative=1;end
        3: begin carry = 0; overflow = 0; negative = 0;
            res = a & b;
            zero = ~(|res);
            if(res[3]==1) negative=1;end
        4: begin carry = 0; overflow = 0; negative = 0;
            res = a | b;
            zero = ~(|res);
            if(res[3]==1) negative=1;end
        5: begin carry = 0; overflow = 0;
            res = a ^ b;
            zero = ~(|res);
            negative = 0;
            if(res[3]==1) negative=1;end
        6: begin carry = 0; overflow = 0; zero = 0; negative = 0;
            if(a[3]==0 && b[3]==1) res = 1;
            else if(a[3] == 1 && b[3] == 0) res = 0;
            else begin
                if(a>b) res = 1;
                else res = 0;end
            end
        7: begin carry = 0; overflow = 0; zero = 0; negative = 0;
            if(a==b) res = 1;
            else res = 0;end
    endcase
end
```

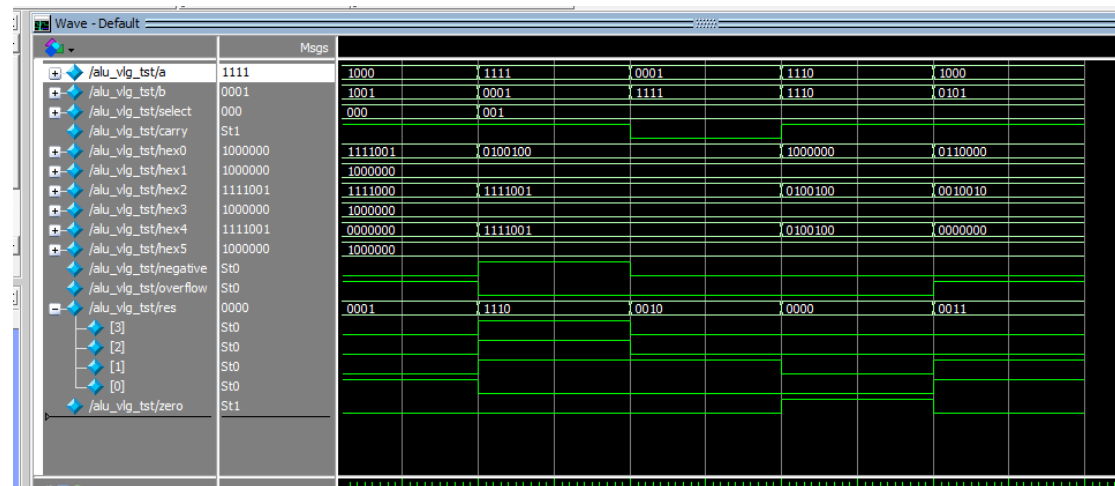
数码管显示部分略...

激励代码：


















加、减法激励代码如下：


















```
61         .negative(negative),
62         .overflow(overflow),
63         .res(res),
64         .select(select),
65         .zero(zero)
66     );
67     initial
68     begin
69         // code that executes only once
70         // insert code here --> begin
71         //add
72         select=3'b000; a=4'b1111; b=4'b0001; #20;
73         a=4'b0111; b=4'b0111; #20;
74         a=4'b1111; b=4'b1110; #20;
75         a=4'b1000; b=4'b1001; #20;
76         //minus
77         select=3'b001; a=4'b1111; b=4'b0001; #20;
78         a=4'b0001; b=4'b1111; #20;
79         a=4'b1110; b=4'b1110; #20;
80         a=4'b1000; b=4'b0101; #20;
81         //not
```

ModelSim 仿真：



引脚分配:

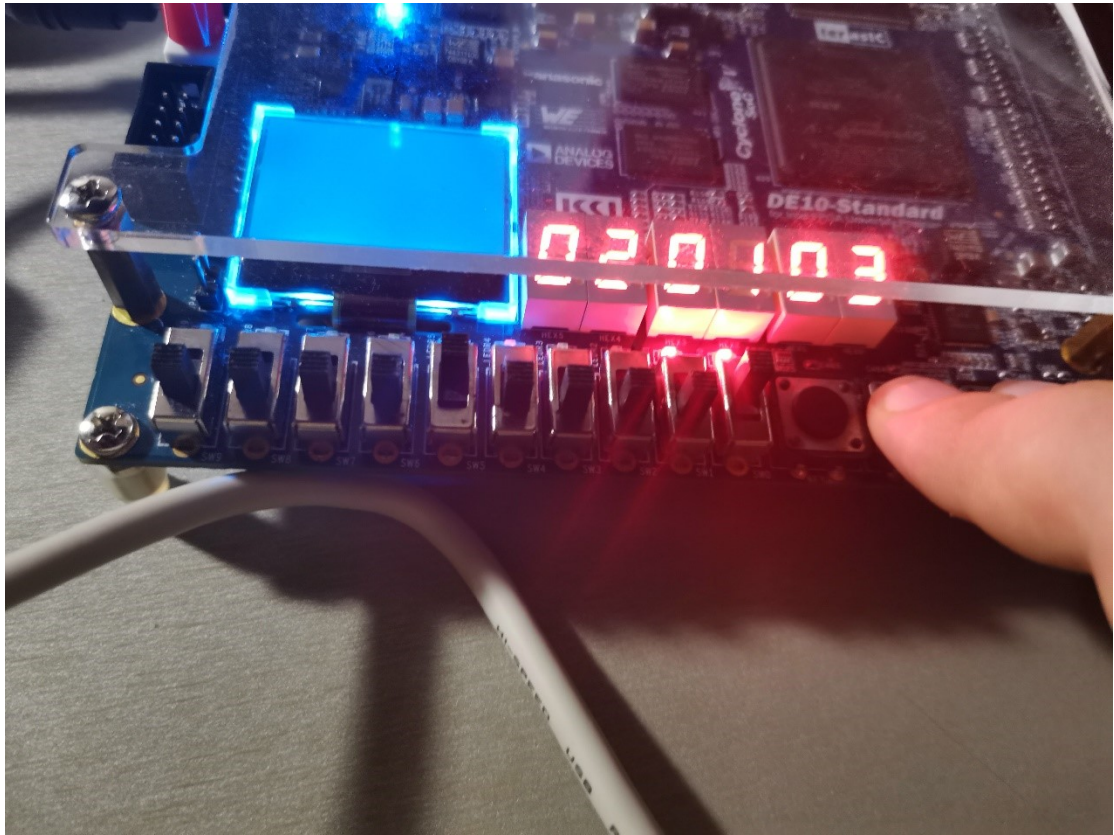
Node Name	Direction	Location	I/O Bank	VREF Group	I/O Sta
 a[3]	Input	PIN_AD30	5B	B5B_N0	2.5 V (def
 a[2]	Input	PIN_AC28	5B	B5B_N0	2.5 V (def
 a[1]	Input	PIN_V25	5B	B5B_N0	2.5 V (def
 a[0]	Input	PIN_W25	5B	B5B_N0	2.5 V (def
 b[3]	Input	PIN_AC30	5B	B5B_N0	2.5 V (def
 b[2]	Input	PIN_AB28	5B	B5B_N0	2.5 V (def
 b[1]	Input	PIN_Y27	5B	B5B_N0	2.5 V (def
 b[0]	Input	PIN_AB30	5B	B5B_N0	2.5 V (def
 carry	Output	PIN_AF24	4A	B4A_N0	2.5 V (def
 hex0[6]	Output	PIN_AH18	4A	B4A_N0	2.5 V (def
 hex0[5]	Output	PIN_AG18	4A	B4A_N0	2.5 V (def
 hex0[4]	Output	PIN_AH17	4A	B4A_N0	2.5 V (def
 hex0[3]	Output	PIN_AG16	4A	B4A_N0	2.5 V (def
 hex0[2]	Output	PIN_AG17	4A	B4A_N0	2.5 V (def
 hex0[1]	Output	PIN_V18	4A	B4A_N0	2.5 V (def
 hex0[0]	Output	PIN_W17	4A	B4A_N0	2.5 V (def
 hex1[6]	Output	PIN_V17	4A	B4A_N0	2.5 V (def

Node Name	Direction	Location	I/O Bank	VREF Group	I
 hex1[5]	Output	PIN_AE17	4A	B4A_N0	2.5
 hex1[4]	Output	PIN_AE18	4A	B4A_N0	2.5
 hex1[3]	Output	PIN_AD17	4A	B4A_N0	2.5
 hex1[2]	Output	PIN_AE16	4A	B4A_N0	2.5
 hex1[1]	Output	PIN_V16	4A	B4A_N0	2.5
 hex1[0]	Output	PIN_AF16	4A	B4A_N0	2.5
 hex2[6]	Output	PIN_W16	4A	B4A_N0	2.5
 hex2[5]	Output	PIN_AF18	4A	B4A_N0	2.5
 hex2[4]	Output	PIN_Y18	4A	B4A_N0	2.5
 hex2[3]	Output	PIN_Y17	4A	B4A_N0	2.5
 hex2[2]	Output	PIN_AA18	4A	B4A_N0	2.5
 hex2[1]	Output	PIN_AB17	4A	B4A_N0	2.5
 hex2[0]	Output	PIN_AA21	4A	B4A_N0	2.5
 hex3[6]	Output	PIN_AD20	4A	B4A_N0	2.5
 hex3[5]	Output	PIN_AA19	4A	B4A_N0	2.5
 hex3[4]	Output	PIN_AC20	4A	B4A_N0	2.5
 hex3[3]	Output	PIN_AA20	4A	B4A_N0	2.5

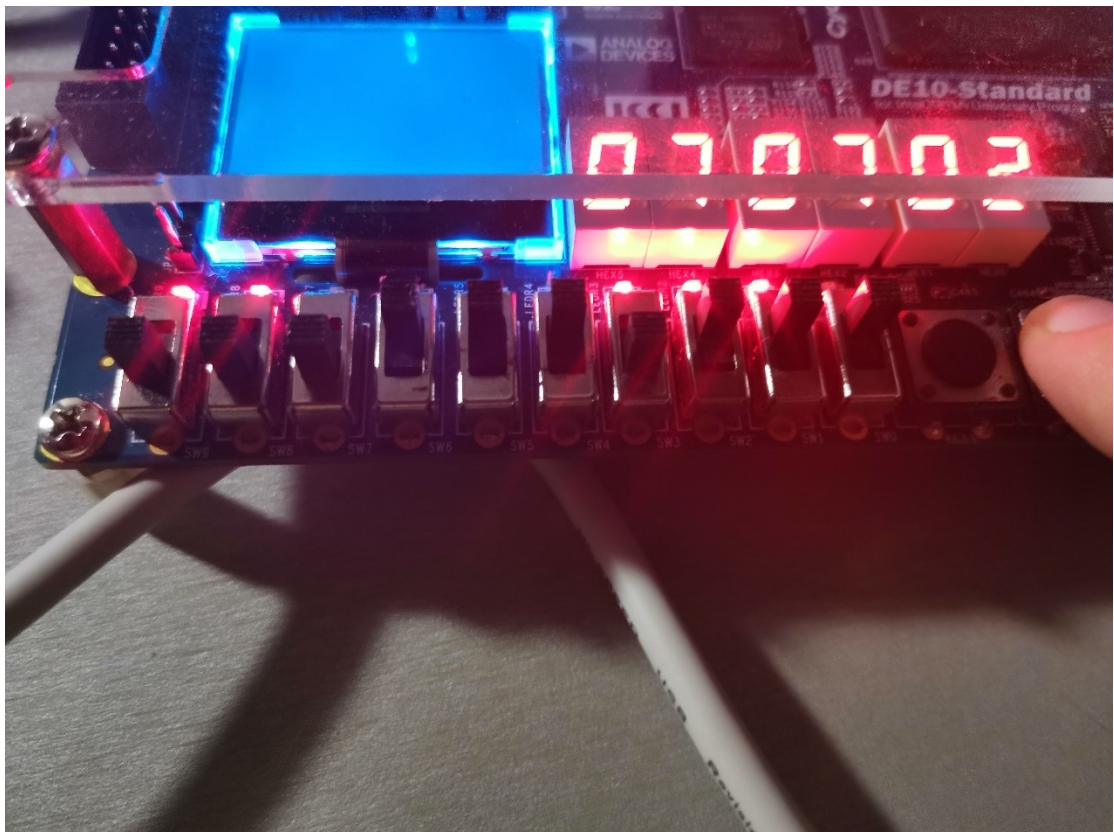
	Node Name	Direction	Location	I/O Bank	VREF Group	I/O
out	hex3[2]	Output	PIN_AD19	4A	B4A_N0	2.5 V
out	hex3[1]	Output	PIN_W19	4A	B4A_N0	2.5 V
out	hex3[0]	Output	PIN_Y19	4A	B4A_N0	2.5 V
out	hex4[6]	Output	PIN_AH22	4A	B4A_N0	2.5 V
out	hex4[5]	Output	PIN_AF23	4A	B4A_N0	2.5 V
out	hex4[4]	Output	PIN_AG23	4A	B4A_N0	2.5 V
out	hex4[3]	Output	PIN_AE23	4A	B4A_N0	2.5 V
out	hex4[2]	Output	PIN_AE22	4A	B4A_N0	2.5 V
out	hex4[1]	Output	PIN_AG22	4A	B4A_N0	2.5 V
out	hex4[0]	Output	PIN_AD21	4A	B4A_N0	2.5 V
out	hex5[6]	Output	PIN_AB21	4A	B4A_N0	2.5 V
out	hex5[5]	Output	PIN_AF19	4A	B4A_N0	2.5 V
out	hex5[4]	Output	PIN_AE19	4A	B4A_N0	2.5 V
out	hex5[3]	Output	PIN_AG20	4A	B4A_N0	2.5 V
out	hex5[2]	Output	PIN_AF20	4A	B4A_N0	2.5 V
out	hex5[1]	Output	PIN_AG21	4A	B4A_N0	2.5 V
out	hex5[0]	Output	PIN_AF21	4A	B4A_N0	2.5 V

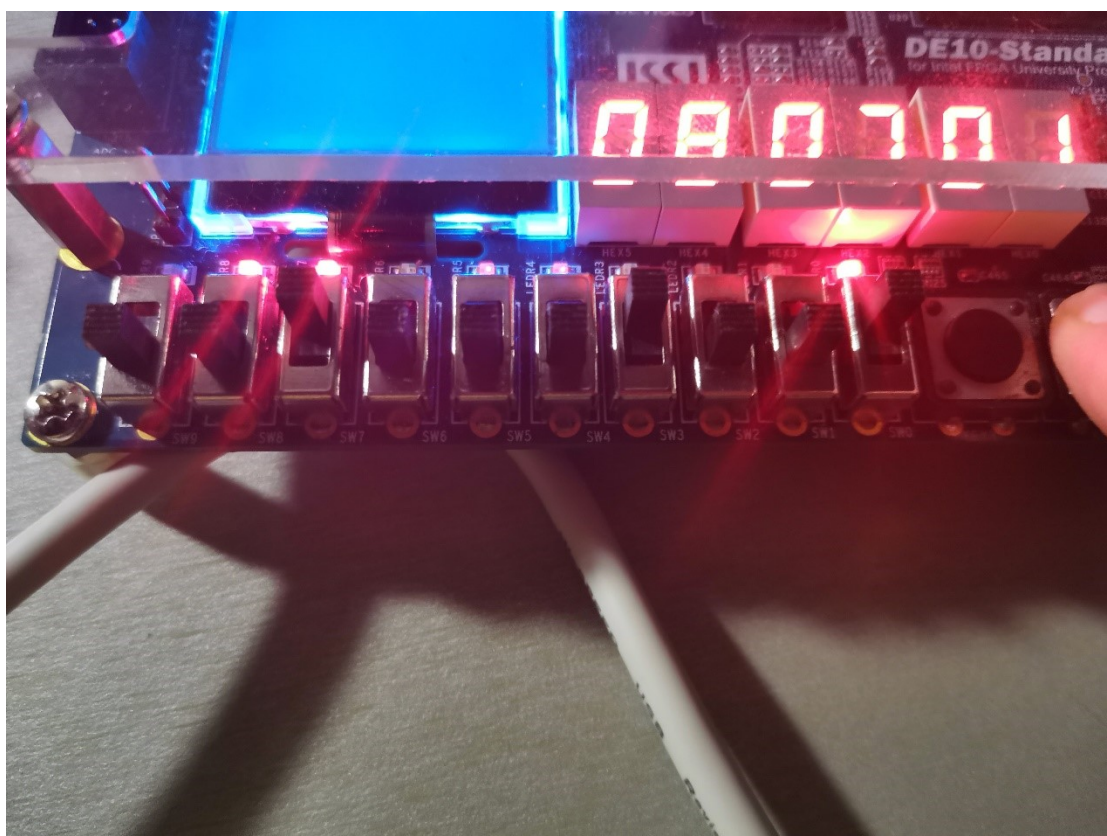
out	negative	Output	PIN_AC22	4A	B4A_N0	2.5
out	overflow	Output	PIN_AB22	5A	B5A_N0	2.5
out	res[3]	Output	PIN_AD24	4A	B4A_N0	2.5
out	res[2]	Output	PIN_AC23	4A	B4A_N0	2.5
out	res[1]	Output	PIN_AB23	5A	B5A_N0	2.5
out	res[0]	Output	PIN_AA24	5A	B5A_N0	2.5
in	select[2]	Input	PIN_AA14	3B	B3B_N0	2.5
in	select[1]	Input	PIN_AK4	3B	B3B_N0	2.5
in	select[0]	Input	PIN_AJ4	3B	B3B_N0	2.5
out	zero	Output	PIN_AE24	4A	B4A_N0	2.5

实验结果：
加法未溢出：

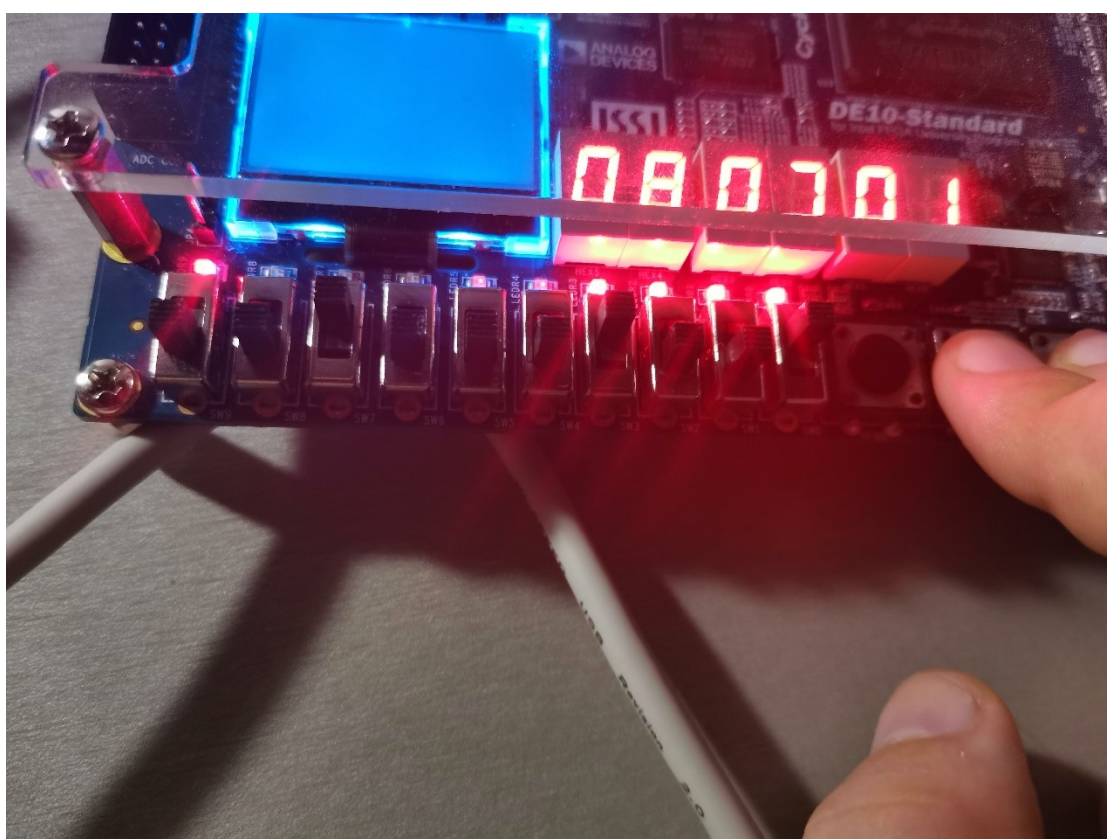


加法溢出：

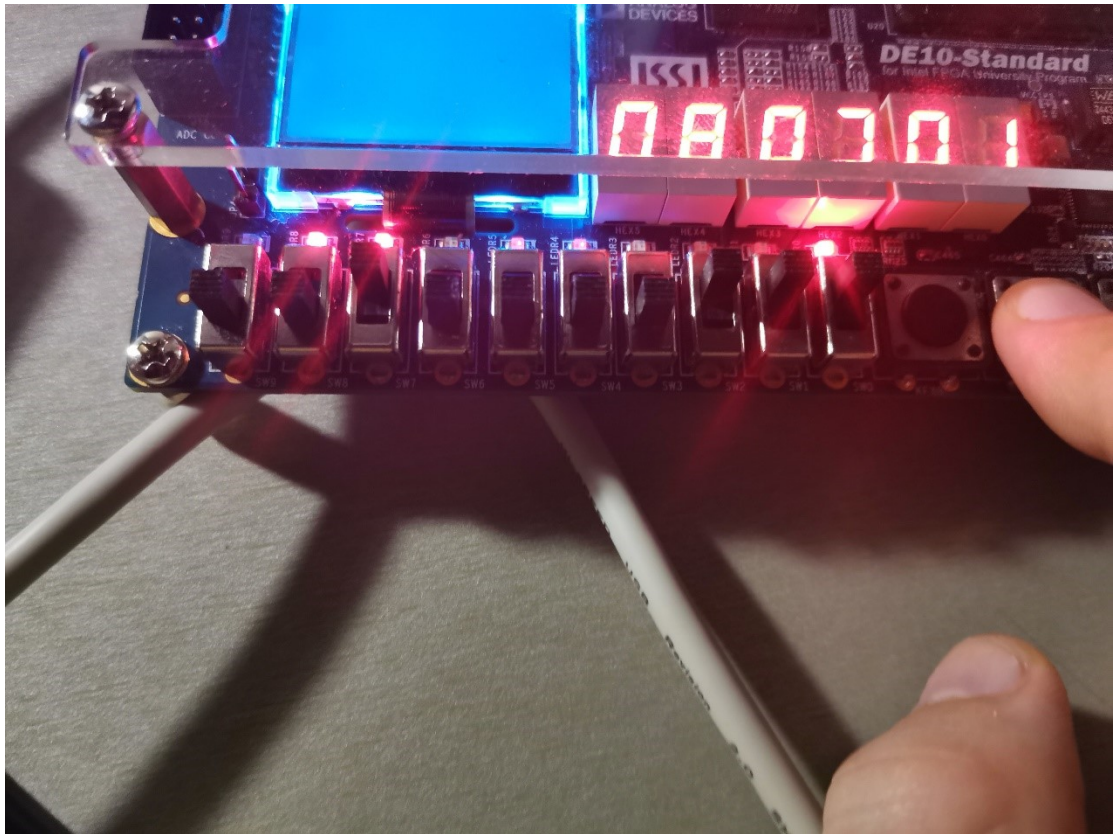




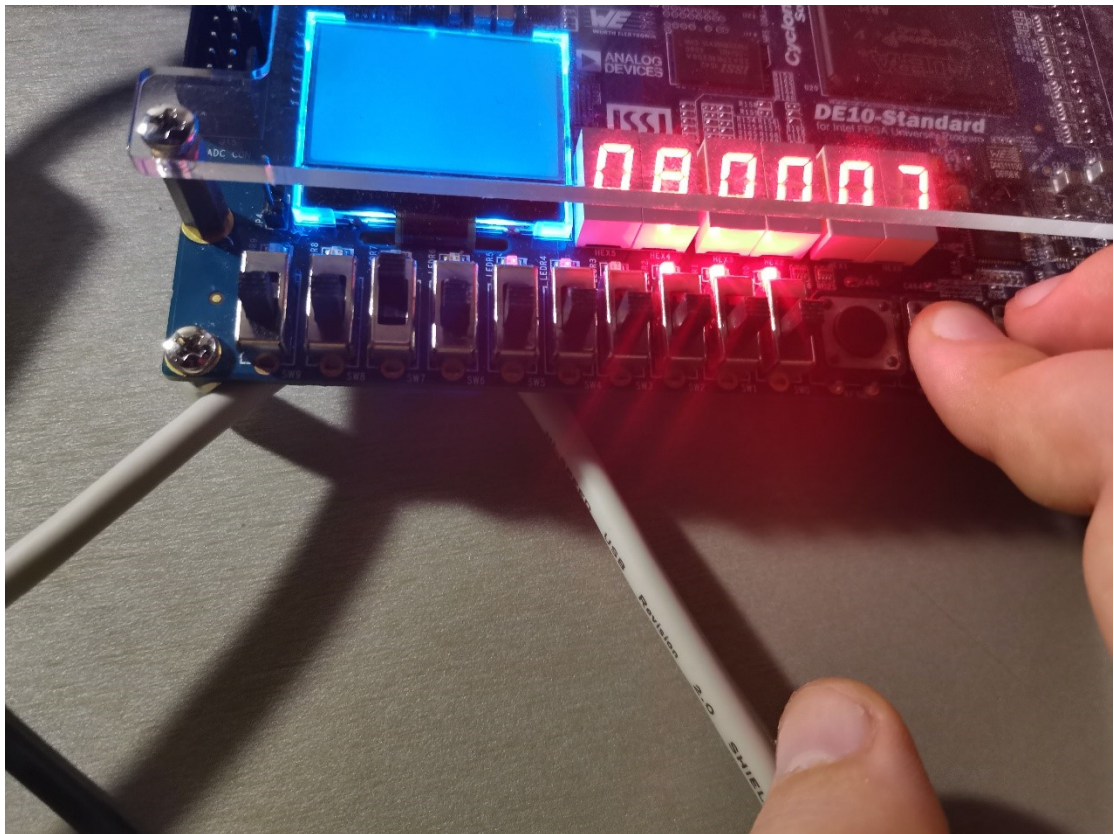
减法未溢出:



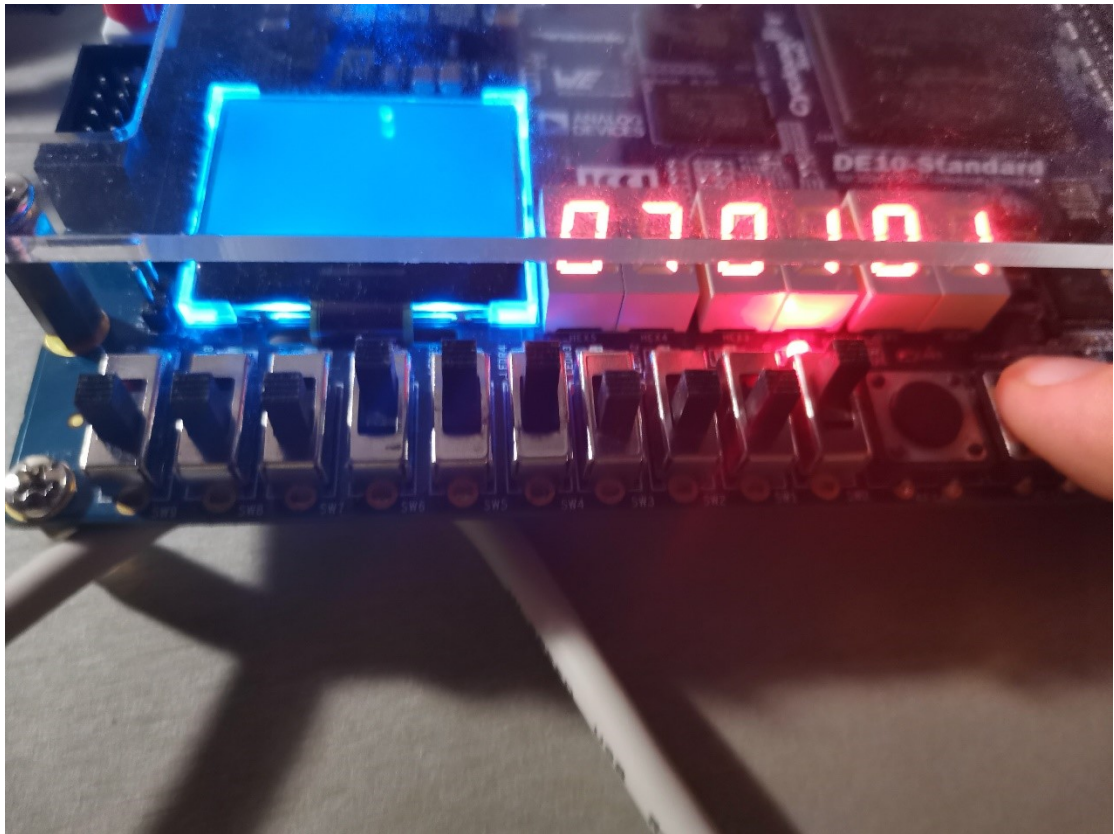
减法溢出：



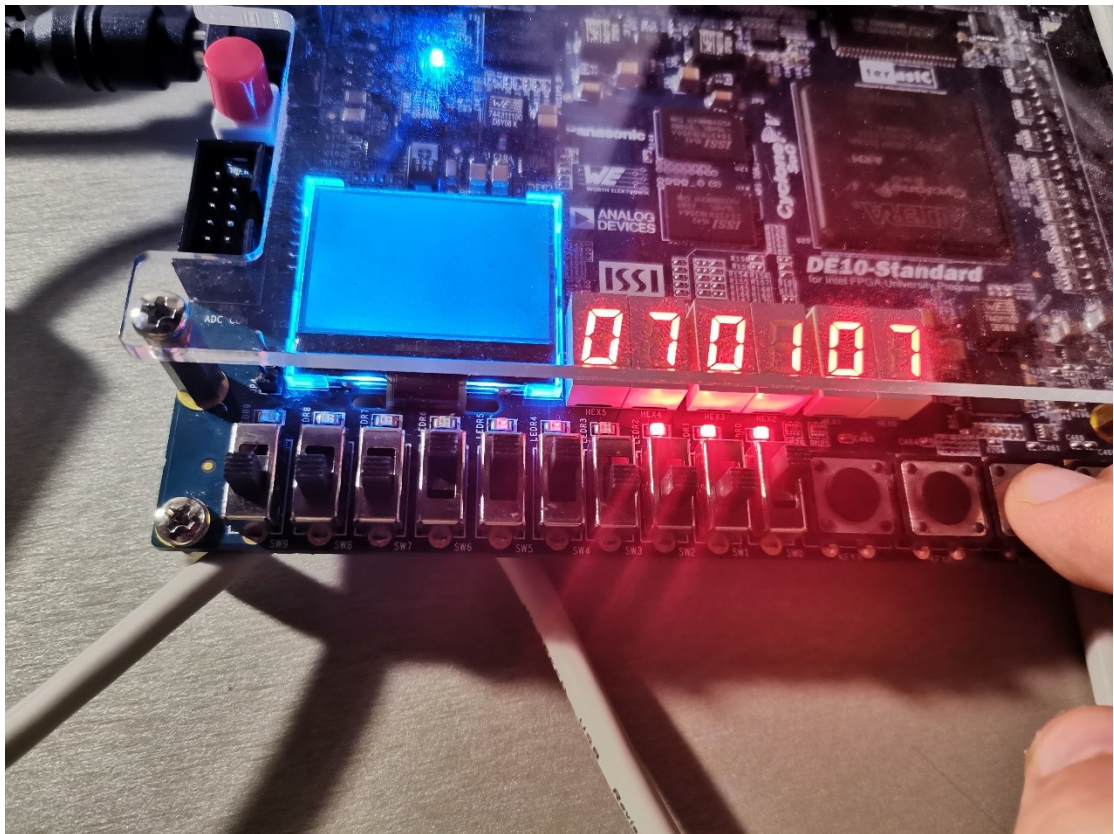
取反：



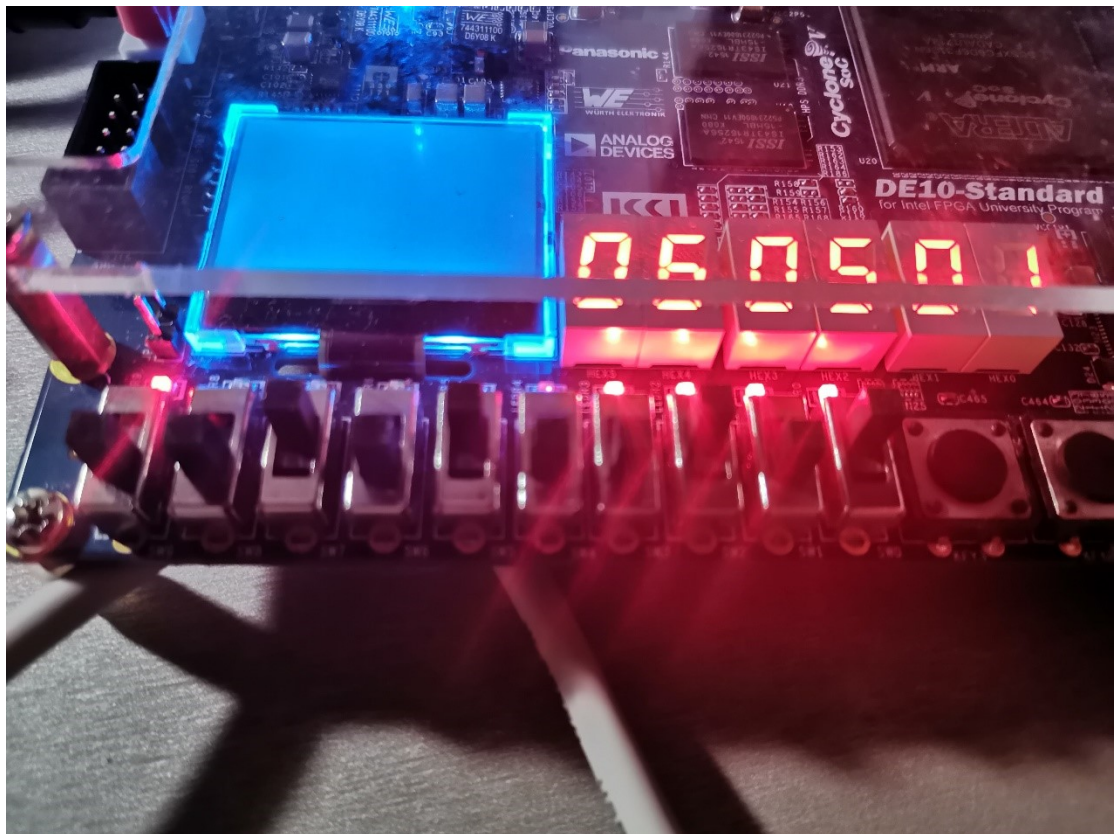
与:



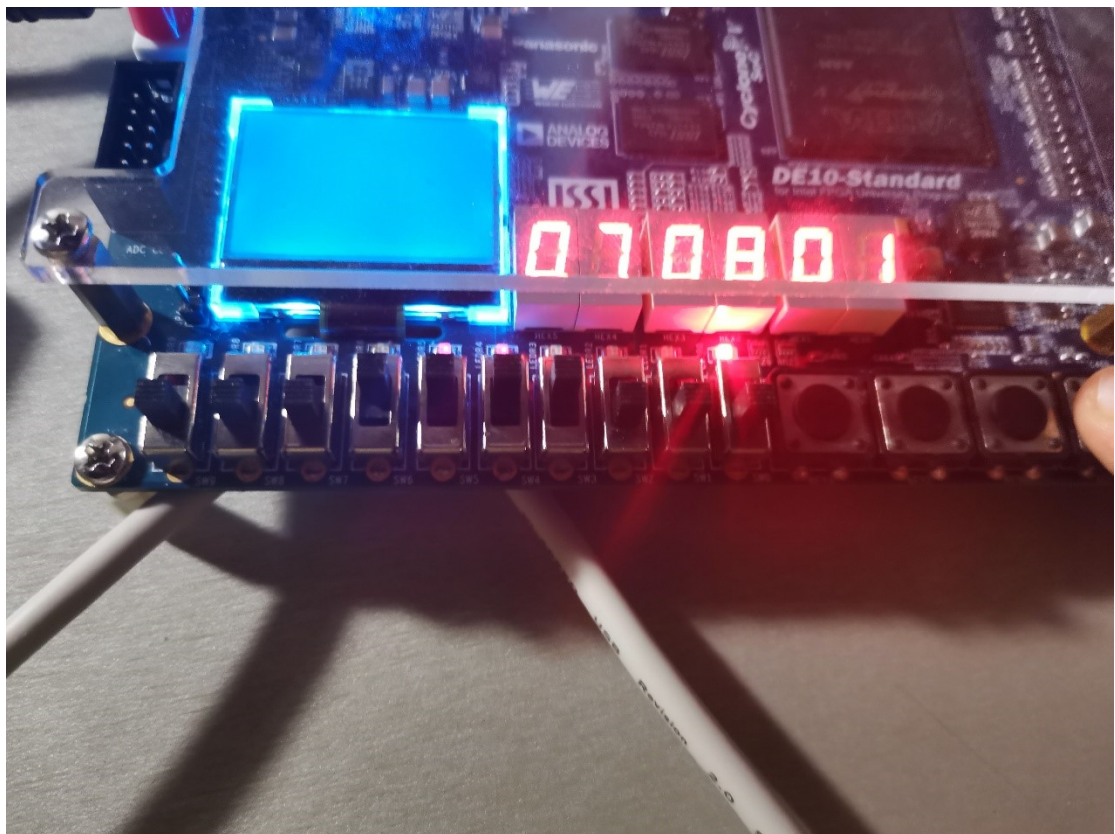
或:



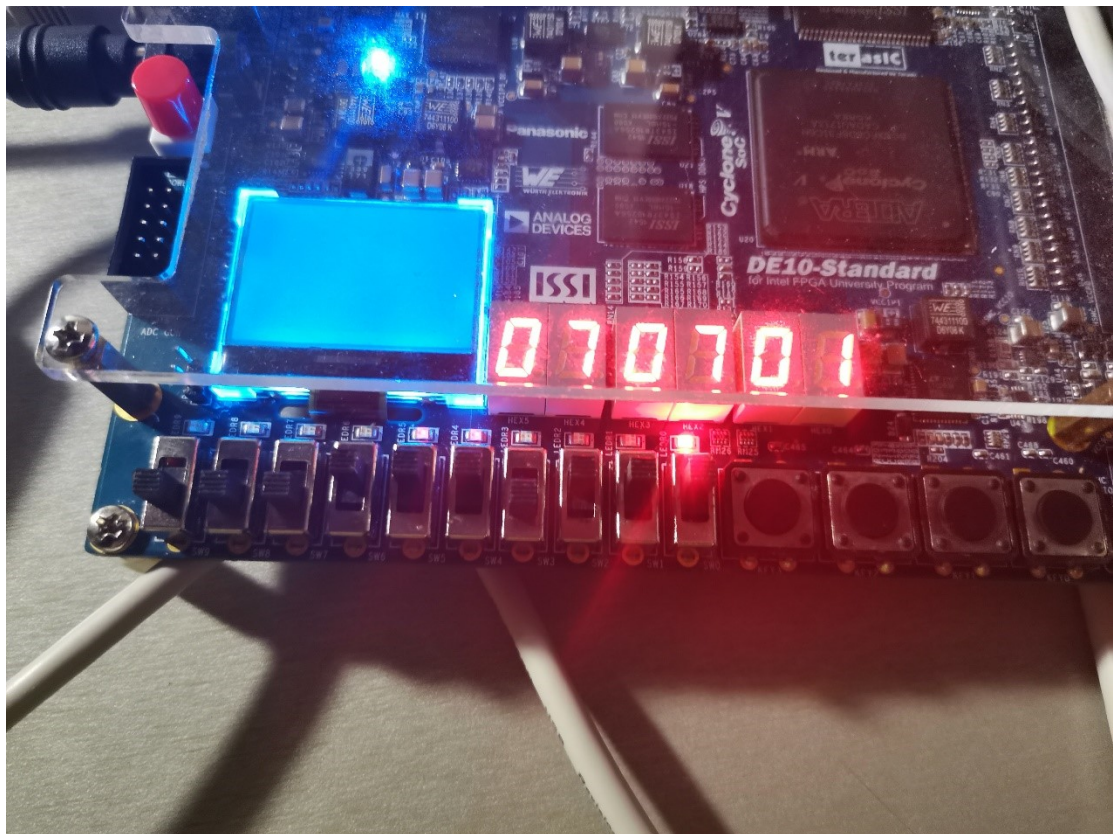
异或：



比较大小：



判断相等：



五、 实验中遇到的问题及解决办法

1. 在最后用七段数码管显示 a、b、结果的时候，都直接按照原码进行显示，如果是负数需要自己转换成对应正数对应的原码。如下图所示：

```
if(res[3] == 1) begin neg_res = ~res + 1;
case (neg_res % 10)
0: hex0 = 7'b1000000;
1: hex0 = 7'b1111001;
```

2. 进行比大小的时候也是同样的问题，都是直接按照原码进行比大小，如果遇到 a、b 两个操作数符号不同的情况，需要特殊讨论，如下图所示：

```
if(res[3]==1) negative=1;end
6: begin carry = 0; overflow = 0; zero = 0; negative = 0;
if(a[3]==0 && b[3]==1) res = 1;
else if(a[3] == 1 && b[3] == 0) res = 0;
else begin
if(a>b) res = 1;
else res = 0;end
end
7: begin carry = 0; overflow = 0; zero = 0; negative = 0;
```

3. 在写代码的时候，老是将一些 c、c++的语法写进去，导致需要写完后要不断的修改，造成进度变慢，非常浪费时间。需要进一步熟悉 verilog 的各种语法。

六、 启示

```
assign zero = ~( | Result );
```

一元约简运算符是单目运算符，其运算规则类似于位运算符中的与、或、非，但其运算过程不同。约简运算符对单个操作数进行运算，最后返回一位数。所以，`|result` 相当于 `result[0] | result[1] | result[2] | result[3]`。若 `result` 为 0 的话，`|result=0`，否则等于 1，最后取反即可。所以，判断结果是否为 0 可以通过 `~(|result)` 这种简便的方法来实现。

七、 意见与建议

虽然之前我并没有上过数字电路这门课，但是实验手册前面的讲解非常的清楚，由浅入深，帮助我学习和完成了这次实验。