浙江大学

本科实验报告

课程名称：	数字逻辑设计
姓　　名：	
学　　院：	竺可桢学院
专　　业：	混合班
指导教师：	董亚波
报告日期：	2025 年 4 月 10 日

浙江大学实验报告

课程名称：数字逻辑设计 实验类型：综合

实验项目名称：全加器、加减法器和 ALU 基本原理与设计

学生姓名：__ 学号：___同组学生姓名：无

实验地点：紫金港东四 509 室 实验日期：2025 年 4 月 9 日

一、 实验目的

- 掌握一位全加器的工作原理和逻辑功能
- 掌握串行进位加法器的工作原理和进位延迟
- 掌握减法器的实现原理
- 掌握加减法器的设计方法
- 掌握 ALU 基本原理及在 CPU 中的作用
- 掌握 ALU 的设计方法

二、操作方法与实验步骤

1. 原理图方式设计 4 位加减法器和 4 位 ALU

实验原理

1 位全加器：

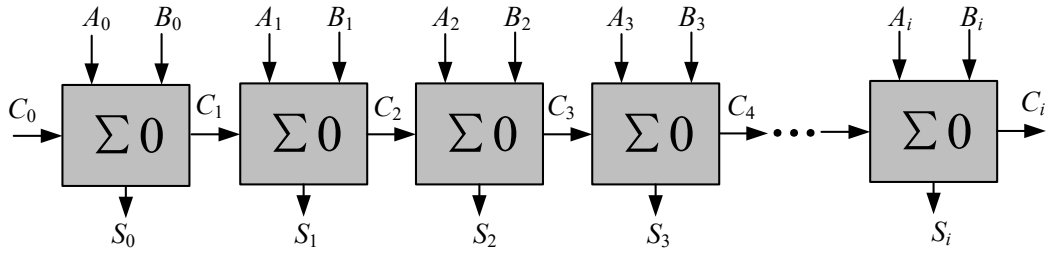
- 三个输入位：数据位 A_i 和 B_i，低位进位输入 C_i
- 二个输出位：全加和 S_i，进位输出 C_{i+1}
- 真值表如下：

A_i	B_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- 逻辑表达式为：$S_i = A_i \oplus B_i \oplus C_i$，$C_{i+1} = A_iB_i + A_iC_i + B_iC_i$

多位串行进位加法器：

- 由一位全加器将进位串接构成
- 低位进位 C_0 为 0，C_i 为高位进位输出
- 原理图如下：

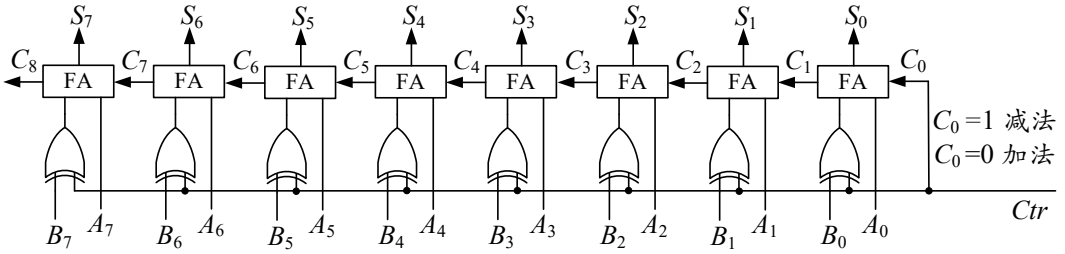


1 位加减法器：

- 用负数补码加法实现，减数当作负数求补码
- 共用加法器
- 用“异或”门控制求反，低位进位 C_0 为 1

多位串行进位全减器：

- 用负数补码加法实现，减数当作负数求补码
- 共用加法器
- 用“异或”门控制求反，低位进位 C0 为 1
- 原理图如下：



4 位 ALU 功能定义：

- 两个 4 位操作数 A(3:0)，B(3:0)
- S(1:0)是 ALU 的功能选择引脚，分别选择选择加、减、与、或操作，即
  S(1:0) = 00：C = A + B
  S(1:0) = 01：C = A – B
  S(1:0) = 10：C = A & B
  S(1:0) = 11：C = A | B
- ALU 计算得到进位 Co 和结果 C(3:0)
- 需要用 4 位 2 输入与门和 4 位 2 输入或门

实验步骤

- 根据一位全加器的输入输出关系，得到电路图，命名为 Adder1b
- 将一位全加器串接，得到四位全加器电路图，命名为 Adder4b，并进行测试
- 根据减法的原理，得到一位全加减器电路图，命名为 AddSub1b
- 将一位全加减器串接，得到四位全加减器电路图，命名为 AddSub4b，并进行测试

- 根据 4 位 ALU 原理图，得到电路图，命名为 MyALU，对其功能进行测试，并导出 MyALU.v，需要用到 Mux4to1、Mux4to1b4

2. 设计 4 位 ALU 应用工程，在实验板上下载验证

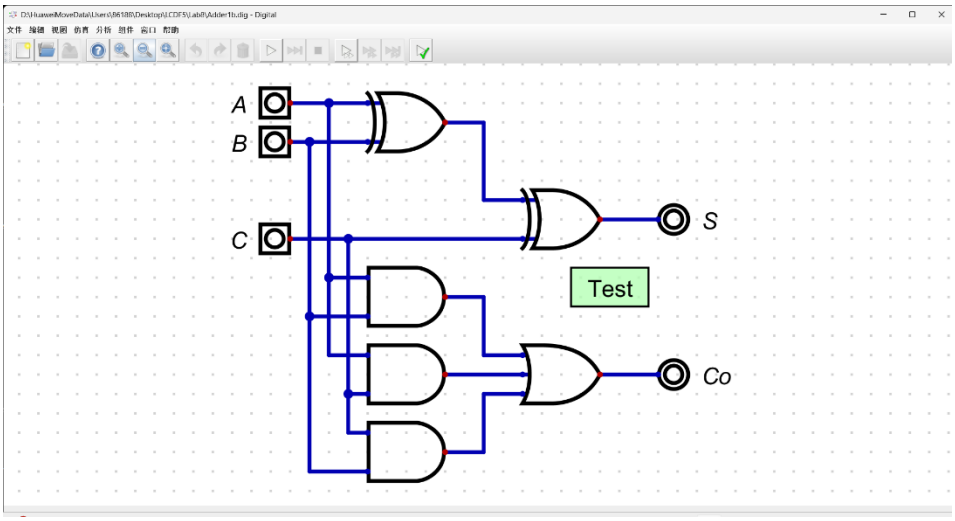
实验步骤

- 在 Vivado 中新建工程 MyALU,添加源文件 MyALU.v(上一个任务),clkdiv.v，DispNum.v，CreateNumber.v（上一次实验）以及新建源文件 Top.v
- 在实验 7 基础上，更新 CreateNumber 模块
- 在 Top.v 中用行为描述进行设计
- 对 Top.v 进行行为仿真
- 将按键进行去抖动处理，然后上板进一步验证。

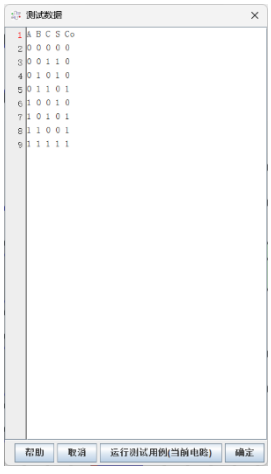
三、实验结果与分析

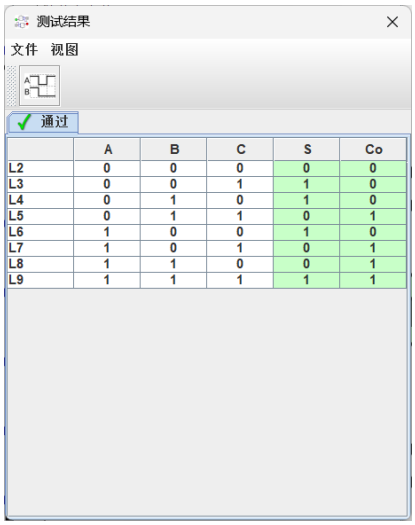
1. 原理图方式设计 4 位加减法器和 4 位 ALU

按照原理图搭建的电路 Adder1b 如图所示：

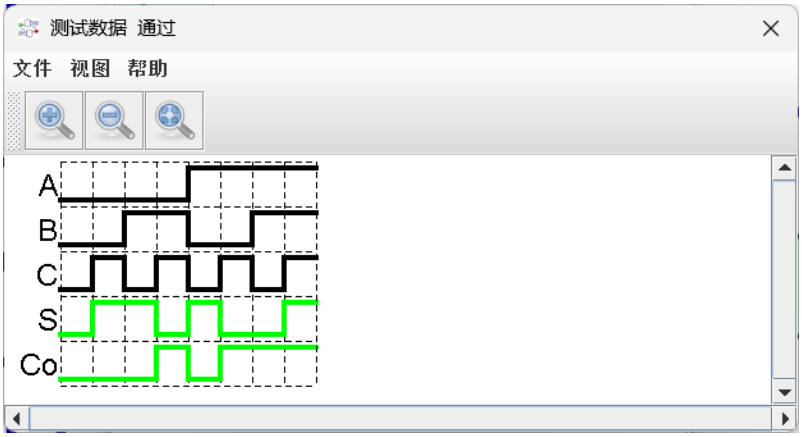


按照真值表对该电路图进行正确性检验，结果如下（已加入预期结果）：



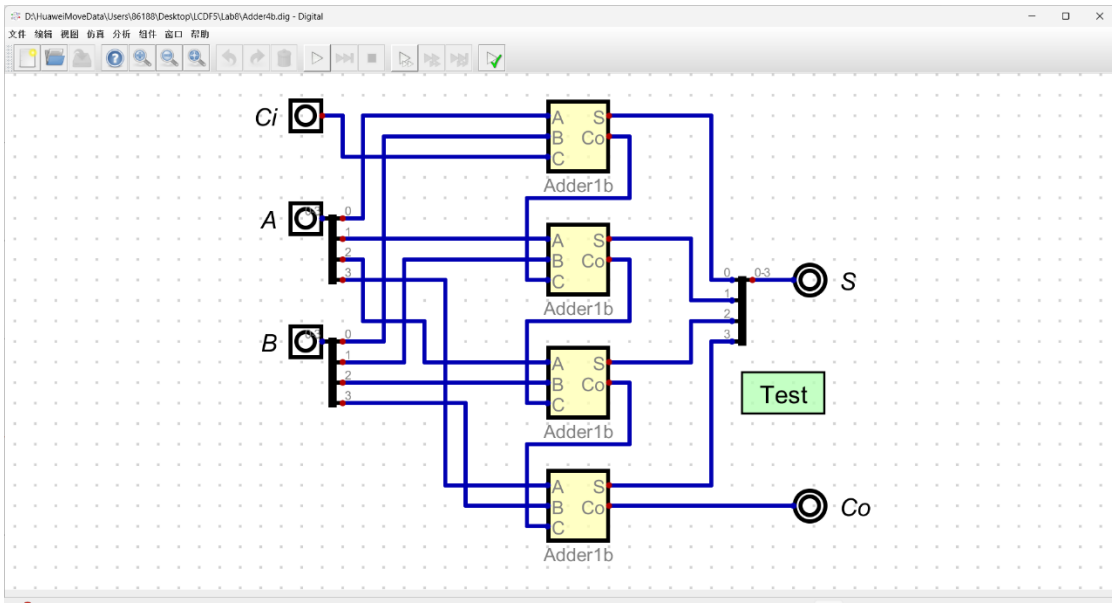


	A	B	C	S	Co
L2	0	0	0	0	0
L3	0	0	1	1	0
L4	0	1	0	1	0
L5	0	1	1	0	1
L6	1	0	0	1	0
L7	1	0	1	0	1
L8	1	1	0	0	1
L9	1	1	1	1	1



其中，当 A,B,C 中出现奇数个 1 的时候，S 的值应当为 1，否则为 0；当 A,B,C 中至少出现两个 1 时，说明存在进位，Co 的值为 1，经检验，该输出符合预期。

接下来将 Adder1b 串接得到电路 Adder4b：



对该电路图进行正确性验证（已加入预期结果）：

测试数据

```
A B Ci S Co
loop(a,16)
  loop(b,16)
    (a) (b) 0 ((a+b)&15) ((a+b)>>4)
    (a) (b) 1 ((a+b+1)&15) ((a+b+1)>>4)
  end loop
end loop
```

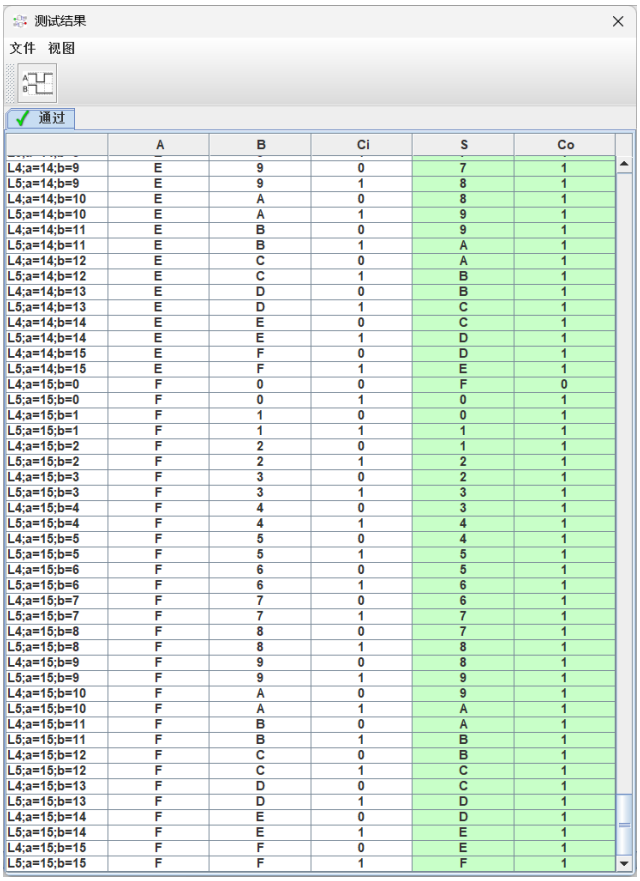
帮助 | 取消 | 运行测试用例(当前电路) | 确定

测试结果

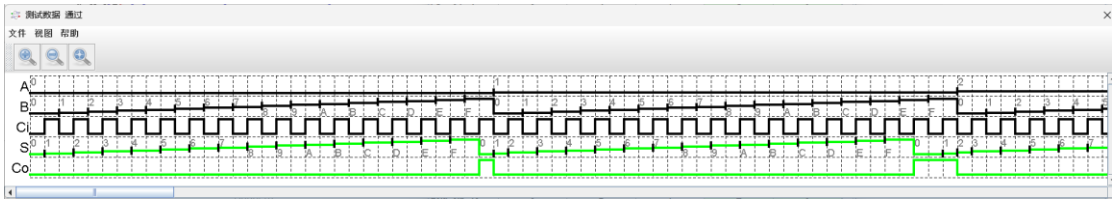
文件 视图

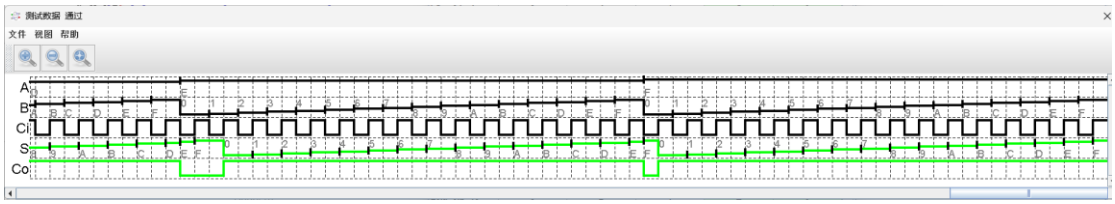
通过

	A	B	Ci	S	Co
L4;a=0;b=0	0	0	0	0	0
L5;a=0;b=0	0	0	1	1	0
L4;a=0;b=1	0	1	0	1	0
L5;a=0;b=1	0	1	1	2	0
L4;a=0;b=2	0	2	0	2	0
L5;a=0;b=2	0	2	1	3	0
L4;a=0;b=3	0	3	0	3	0
L5;a=0;b=3	0	3	1	4	0
L4;a=0;b=4	0	4	0	4	0
L5;a=0;b=4	0	4	1	5	0
L4;a=0;b=5	0	5	0	5	0
L5;a=0;b=5	0	5	1	6	0
L4;a=0;b=6	0	6	0	6	0
L5;a=0;b=6	0	6	1	7	0
L4;a=0;b=7	0	7	0	7	0
L5;a=0;b=7	0	7	1	8	0
L4;a=0;b=8	0	8	0	8	0
L5;a=0;b=8	0	8	1	9	0
L4;a=0;b=9	0	9	0	9	0
L5;a=0;b=9	0	9	1	A	0
L4;a=0;b=10	0	A	0	A	0
L5;a=0;b=10	0	A	1	B	0
L4;a=0;b=11	0	B	0	B	0
L5;a=0;b=11	0	B	1	C	0
L4;a=0;b=12	0	C	0	C	0
L5;a=0;b=12	0	C	1	D	0
L4;a=0;b=13	0	D	0	D	0
L5;a=0;b=13	0	D	1	E	0
L4;a=0;b=14	0	E	0	E	0
L5;a=0;b=14	0	E	1	F	0
L4;a=0;b=15	0	F	0	F	0
L5;a=0;b=15	0	F	1	0	1
L4;a=1;b=0	1	0	0	1	0
L5;a=1;b=0	1	0	1	2	0
L4;a=1;b=1	1	1	0	2	0
L5;a=1;b=1	1	1	1	3	0
L4;a=1;b=2	1	2	0	3	0
L5;a=1;b=2	1	2	1	4	0
L4;a=1;b=3	1	3	0	4	0
L5;a=1;b=3	1	3	1	5	0
L4;a=1;b=4	1	4	0	5	0
L5;a=1;b=4	1	4	1	6	0
L4;a=1;b=5	1	5	0	6	0
L5;a=1;b=5	1	5	1	7	0
L4;a=1;b=6	1	6	0	7	0
L5;a=1;b=6	1	6	1	8	0



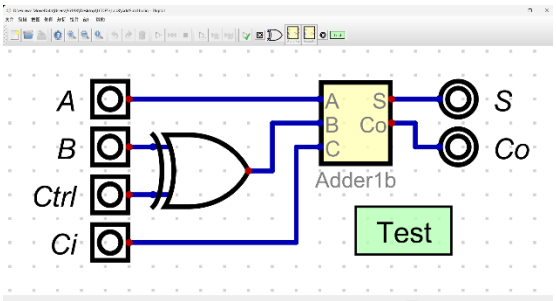
	A	B	Ci	S	Co
L4;a=14;b=9	E	9	0	7	1
L5;a=14;b=9	E	9	1	8	1
L4;a=14;b=10	E	A	0	8	1
L5;a=14;b=10	E	A	1	9	1
L4;a=14;b=11	E	B	0	9	1
L5;a=14;b=11	E	B	1	A	1
L4;a=14;b=12	E	C	0	A	1
L5;a=14;b=12	E	C	1	B	1
L4;a=14;b=13	E	D	0	B	1
L5;a=14;b=13	E	D	1	C	1
L4;a=14;b=14	E	E	0	C	1
L5;a=14;b=14	E	E	1	D	1
L4;a=14;b=15	E	F	0	D	1
L5;a=14;b=15	E	F	1	E	1
L4;a=15;b=0	F	0	0	F	0
L5;a=15;b=0	F	0	1	0	1
L4;a=15;b=1	F	1	0	0	1
L5;a=15;b=1	F	1	1	1	1
L4;a=15;b=2	F	2	0	1	1
L5;a=15;b=2	F	2	1	2	1
L4;a=15;b=3	F	3	0	2	1
L5;a=15;b=3	F	3	1	3	1
L4;a=15;b=4	F	4	0	3	1
L5;a=15;b=4	F	4	1	4	1
L4;a=15;b=5	F	5	0	4	1
L5;a=15;b=5	F	5	1	5	1
L4;a=15;b=6	F	6	0	5	1
L5;a=15;b=6	F	6	1	6	1
L4;a=15;b=7	F	7	0	6	1
L5;a=15;b=7	F	7	1	7	1
L4;a=15;b=8	F	8	0	7	1
L5;a=15;b=8	F	8	1	8	1
L4;a=15;b=9	F	9	0	8	1
L5;a=15;b=9	F	9	1	9	1
L4;a=15;b=10	F	A	0	9	1
L5;a=15;b=10	F	A	1	A	1
L4;a=15;b=11	F	B	0	A	1
L5;a=15;b=11	F	B	1	B	1
L4;a=15;b=12	F	C	0	B	1
L5;a=15;b=12	F	C	1	C	1
L4;a=15;b=13	F	D	0	C	1
L5;a=15;b=13	F	D	1	D	1
L4;a=15;b=14	F	E	0	D	1
L5;a=15;b=14	F	E	1	E	1
L4;a=15;b=15	F	F	0	E	1
L5;a=15;b=15	F	F	1	F	1



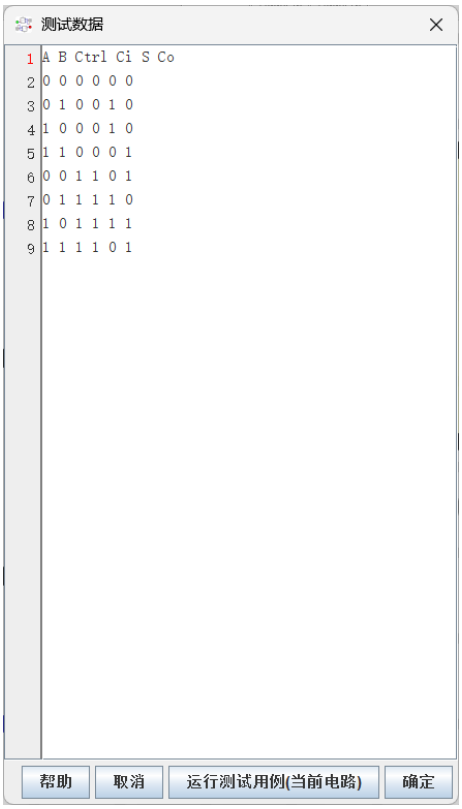


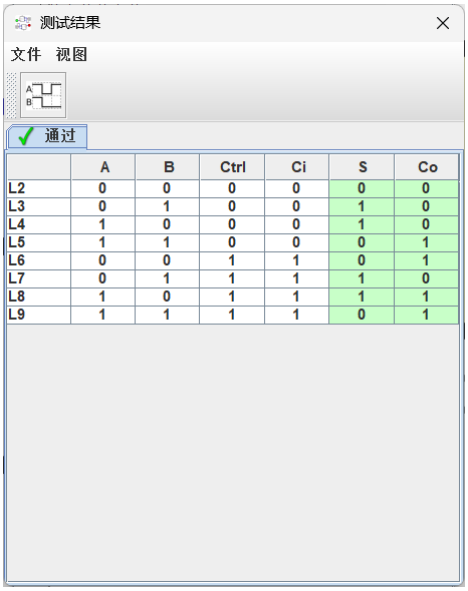
由于输出结果过多(共 512 组),这里只选取了部分结果说明已进行正确性验证。在测试用例中，((a+b)&15)表示 a+b 的结果取低 4 位（即预期的 S），((a+b)>>4)表示 a+b 的结果取高 1 位（即预期的进位 C）。由于该电路通过了所有的测试用例，因此该输出符合预期，电路无误。

再按照原理图搭建电路 AddSub1b：

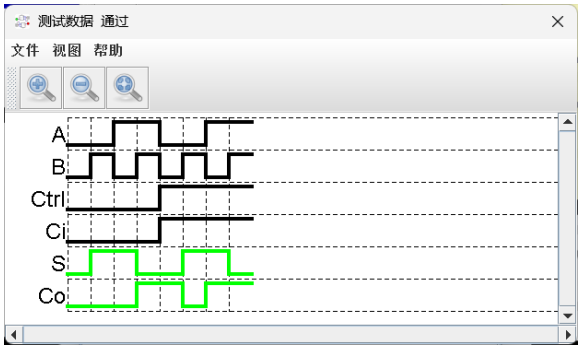


根据真值表对该电路进行正确性验证：



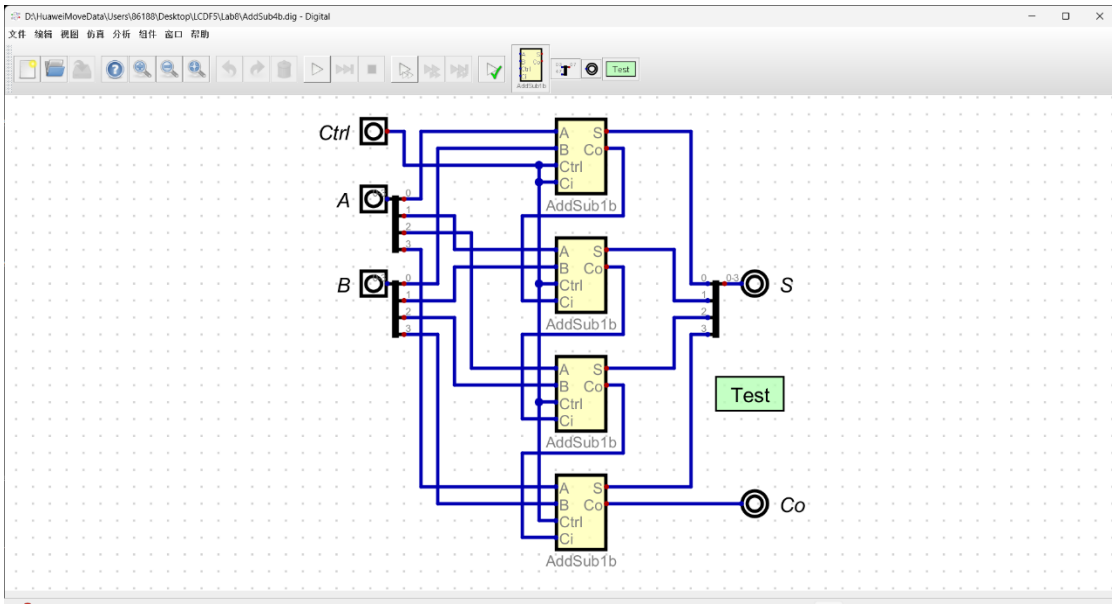


	A	B	Ctrl	Ci	S	Co
L2	0	0	0	0	0	0
L3	0	1	0	0	1	0
L4	1	0	0	0	1	0
L5	1	1	0	0	0	1
L6	0	0	1	1	0	1
L7	0	1	1	1	1	0
L8	1	0	1	1	1	1
L9	1	1	1	1	0	1

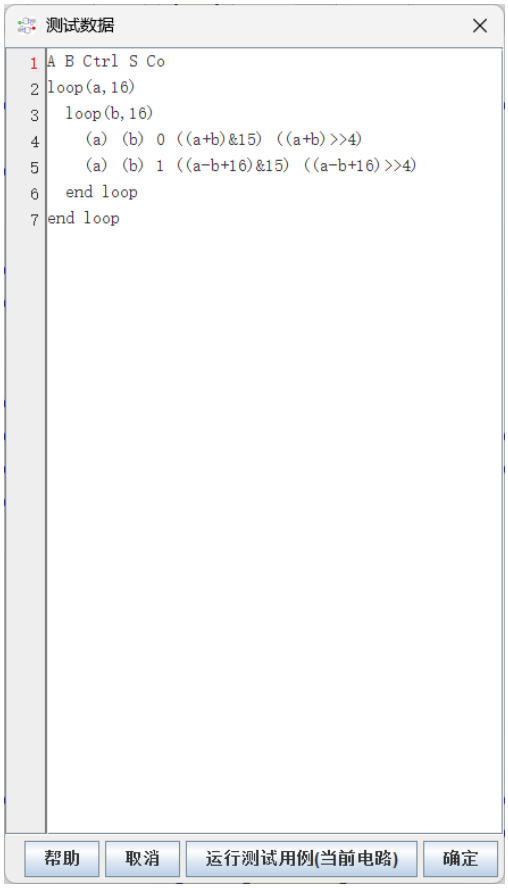


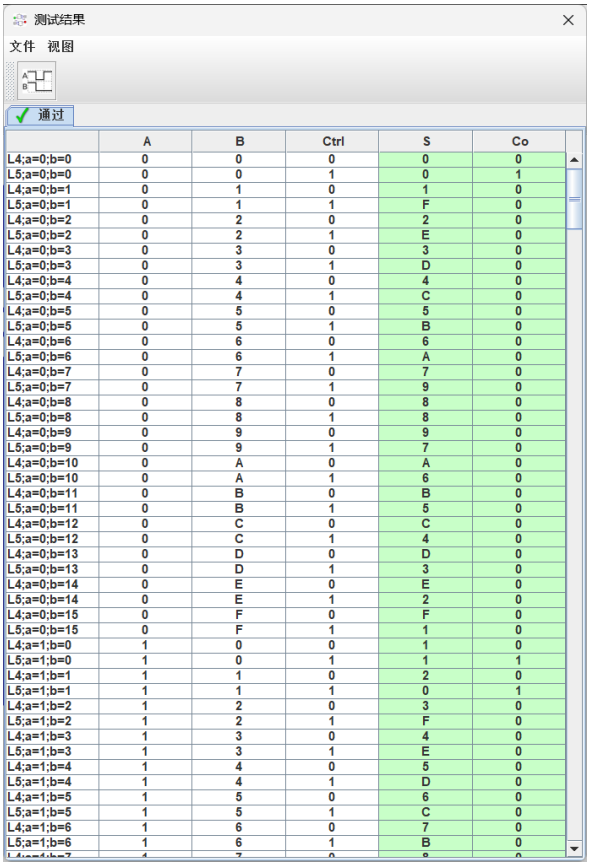
当 Ctrl 为 0 时，表示加减法器执行加法操作；当 Ctrl 为 1 时，表示加减法器执行减法操作。加法操作与 Adder1b 操作一致，只需检查是否在 Ctrl 为 0 时能正常执行加法即可；对于减法操作，Co 为 1 表示没有借位，Co 为 0 表示有借位，其中有借位的情况只有在 A=0,B=1 时发生，此时 S=1,Co=0，其余情况均为 S=A-B,Co=1。因此该电路无误。

将 AddSub1b 进行串接，得到电路 AddSub4b：

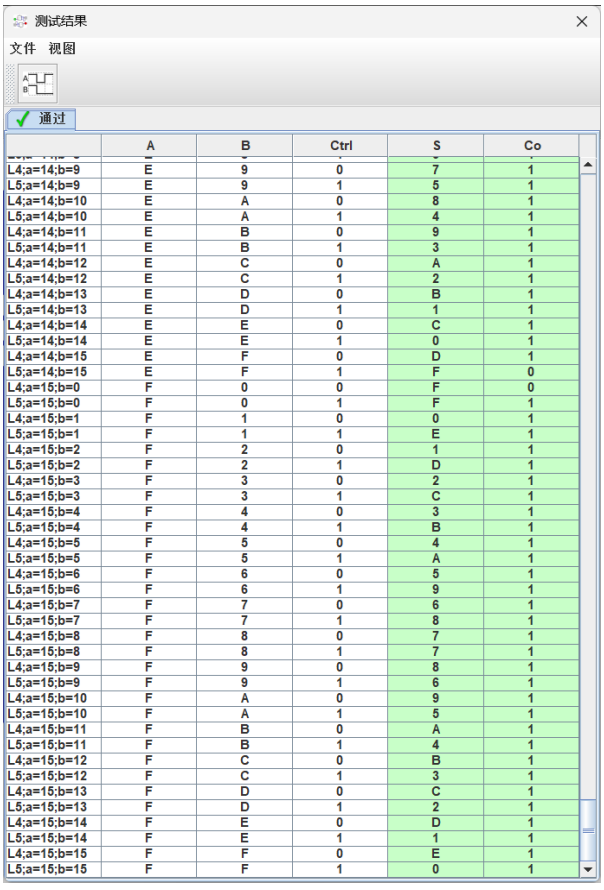


对该电路进行正确性验证（已加入预期结果）：

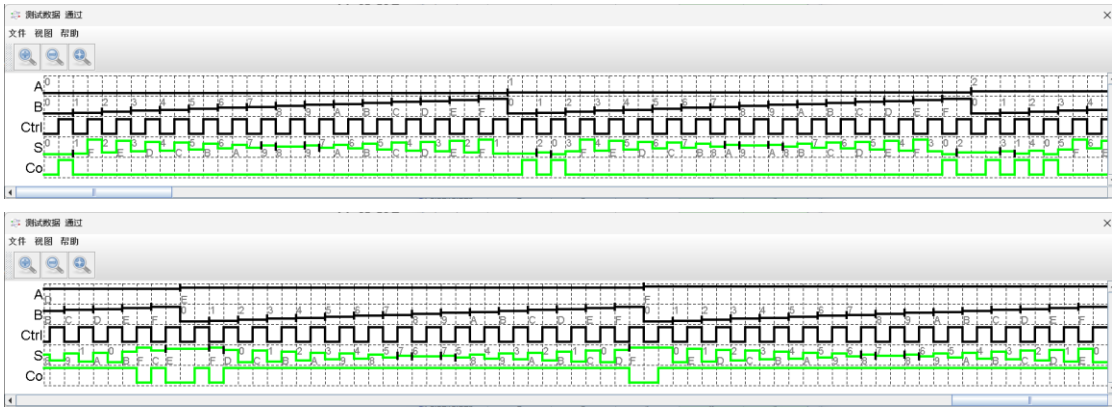




	A	B	Ctrl	S	Co
L4;a=0;b=0	0	0	0	0	0
L5;a=0;b=0	0	0	1	0	1
L4;a=0;b=1	0	1	0	1	0
L5;a=0;b=1	0	1	1	F	0
L4;a=0;b=2	0	2	0	2	0
L5;a=0;b=2	0	2	1	E	0
L4;a=0;b=3	0	3	0	3	0
L5;a=0;b=3	0	3	1	D	0
L4;a=0;b=4	0	4	0	4	0
L5;a=0;b=4	0	4	1	C	0
L4;a=0;b=5	0	5	0	5	0
L5;a=0;b=5	0	5	1	B	0
L4;a=0;b=6	0	6	0	6	0
L5;a=0;b=6	0	6	1	A	0
L4;a=0;b=7	0	7	0	7	0
L5;a=0;b=7	0	7	1	9	0
L4;a=0;b=8	0	8	0	8	0
L5;a=0;b=8	0	8	1	8	0
L4;a=0;b=9	0	9	0	9	0
L5;a=0;b=9	0	9	1	7	0
L4;a=0;b=10	0	A	0	A	0
L5;a=0;b=10	0	A	1	6	0
L4;a=0;b=11	0	B	0	B	0
L5;a=0;b=11	0	B	1	5	0
L4;a=0;b=12	0	C	0	C	0
L5;a=0;b=12	0	C	1	4	0
L4;a=0;b=13	0	D	0	D	0
L5;a=0;b=13	0	D	1	3	0
L4;a=0;b=14	0	E	0	E	0
L5;a=0;b=14	0	E	1	2	0
L4;a=0;b=15	0	F	0	F	0
L5;a=0;b=15	0	F	1	1	0
L4;a=1;b=0	1	0	0	1	0
L5;a=1;b=0	1	0	1	1	1
L4;a=1;b=1	1	1	0	2	0
L5;a=1;b=1	1	1	1	0	1
L4;a=1;b=2	1	2	0	3	0
L5;a=1;b=2	1	2	1	F	0
L4;a=1;b=3	1	3	0	4	0
L5;a=1;b=3	1	3	1	E	0
L4;a=1;b=4	1	4	0	5	0
L5;a=1;b=4	1	4	1	D	0
L4;a=1;b=5	1	5	0	6	0
L5;a=1;b=5	1	5	1	C	0
L4;a=1;b=6	1	6	0	7	0
L5;a=1;b=6	1	6	1	B	0

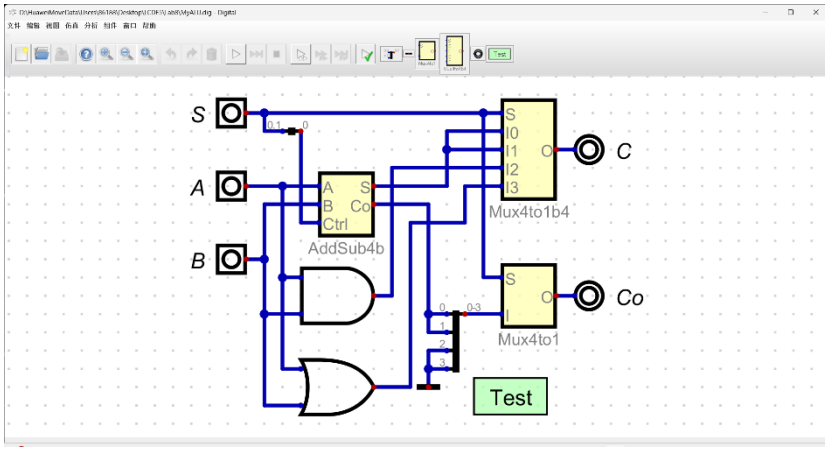


	A	B	Ctrl	S	Co
L4;a=14;b=9	E	9	0	7	1
L5;a=14;b=9	E	9	1	5	1
L4;a=14;b=10	E	A	0	8	1
L5;a=14;b=10	E	A	1	4	1
L4;a=14;b=11	E	B	0	9	1
L5;a=14;b=11	E	B	1	3	1
L4;a=14;b=12	E	C	0	A	1
L5;a=14;b=12	E	C	1	2	1
L4;a=14;b=13	E	D	0	B	1
L5;a=14;b=13	E	D	1	1	1
L4;a=14;b=14	E	E	0	C	1
L5;a=14;b=14	E	E	1	0	1
L4;a=14;b=15	E	F	0	D	1
L5;a=14;b=15	E	F	1	F	0
L4;a=15;b=0	F	0	0	F	0
L5;a=15;b=0	F	0	1	F	1
L4;a=15;b=1	F	1	0	0	1
L5;a=15;b=1	F	1	1	E	1
L4;a=15;b=2	F	2	0	1	1
L5;a=15;b=2	F	2	1	D	1
L4;a=15;b=3	F	3	0	2	1
L5;a=15;b=3	F	3	1	C	1
L4;a=15;b=4	F	4	0	3	1
L5;a=15;b=4	F	4	1	B	1
L4;a=15;b=5	F	5	0	4	1
L5;a=15;b=5	F	5	1	A	1
L4;a=15;b=6	F	6	0	5	1
L5;a=15;b=6	F	6	1	9	1
L4;a=15;b=7	F	7	0	6	1
L5;a=15;b=7	F	7	1	8	1
L4;a=15;b=8	F	8	0	7	1
L5;a=15;b=8	F	8	1	7	1
L4;a=15;b=9	F	9	0	8	1
L5;a=15;b=9	F	9	1	6	1
L4;a=15;b=10	F	A	0	9	1
L5;a=15;b=10	F	A	1	5	1
L4;a=15;b=11	F	B	0	A	1
L5;a=15;b=11	F	B	1	4	1
L4;a=15;b=12	F	C	0	B	1
L5;a=15;b=12	F	C	1	3	1
L4;a=15;b=13	F	D	0	C	1
L5;a=15;b=13	F	D	1	2	1
L4;a=15;b=14	F	E	0	D	1
L5;a=15;b=14	F	E	1	1	1
L4;a=15;b=15	F	F	0	E	1
L5;a=15;b=15	F	F	1	0	1

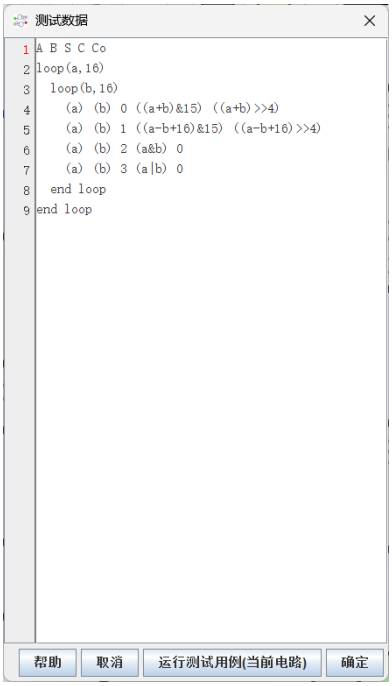


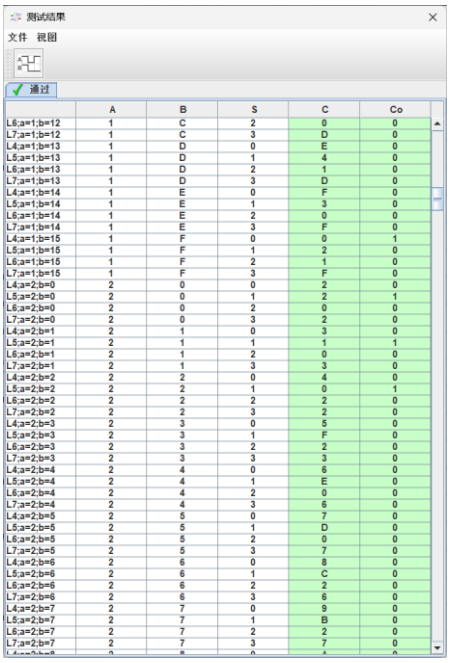
由于输出结果过多（共 512 组），这里只选取了部分结构说明已进行正确性验证。在该测试用例中，((a-b+16)&15)表示 a-b 结果的低 4 位（因为 16-b 为 4 位二进制下 b 的补码），((a-b+16)>>1)表示 Co 的结果（为 1 说明 a>=b，即够减）。由于该电路通过了所有的测试用例，因此该电路无误。

最后根据 4 位 ALU 的工作原理，搭建 MyALU 电路：

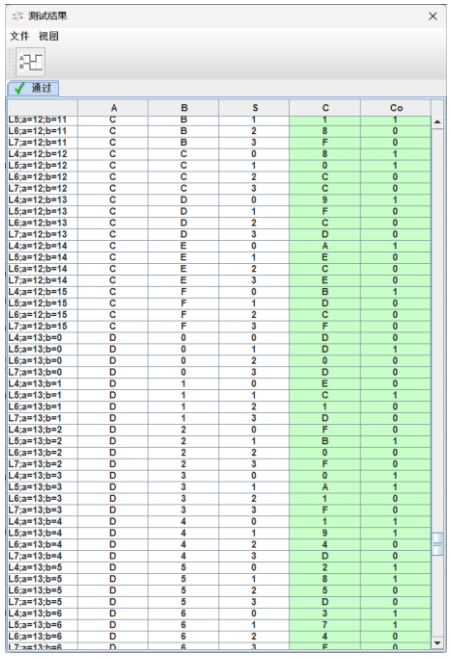


对该电路进行正确性验证：

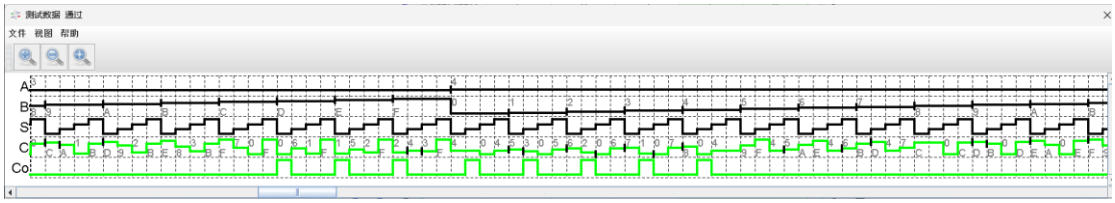


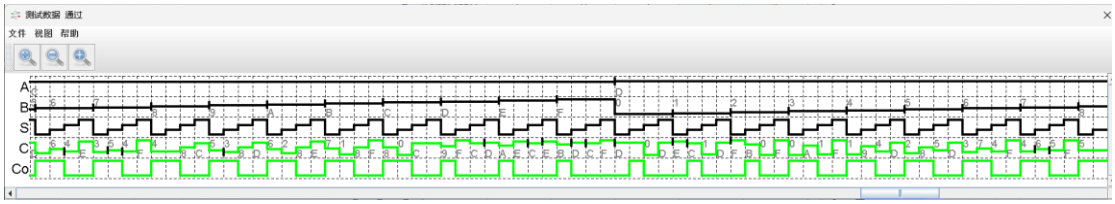


	A	B	S	C	Co
L6;a=1;b=12	1	C	2	0	0
L7;a=1;b=12	1	C	3	D	0
L4;a=1;b=13	1	D	0	E	0
L5;a=1;b=13	1	D	1	4	0
L6;a=1;b=13	1	D	2	1	0
L7;a=1;b=13	1	D	3	D	0
L4;a=1;b=14	1	E	0	F	0
L5;a=1;b=14	1	E	1	3	0
L6;a=1;b=14	1	E	2	0	0
L7;a=1;b=14	1	E	3	F	0
L4;a=1;b=15	1	F	0	0	1
L5;a=1;b=15	1	F	1	2	0
L6;a=1;b=15	1	F	2	1	0
L7;a=1;b=15	1	F	3	F	0
L4;a=2;b=0	2	0	0	2	0
L5;a=2;b=0	2	0	1	2	1
L6;a=2;b=0	2	0	2	0	0
L7;a=2;b=0	2	0	3	2	0
L4;a=2;b=1	2	1	0	3	0
L5;a=2;b=1	2	1	1	1	1
L6;a=2;b=1	2	1	2	0	0
L7;a=2;b=1	2	1	3	3	0
L4;a=2;b=2	2	2	0	4	0
L5;a=2;b=2	2	2	1	0	1
L6;a=2;b=2	2	2	2	2	0
L7;a=2;b=2	2	2	3	2	0
L4;a=2;b=3	2	3	0	5	0
L5;a=2;b=3	2	3	1	F	0
L6;a=2;b=3	2	3	2	2	0
L7;a=2;b=3	2	3	3	3	0
L4;a=2;b=4	2	4	0	6	0
L5;a=2;b=4	2	4	1	E	0
L6;a=2;b=4	2	4	2	0	0
L7;a=2;b=4	2	4	3	6	0
L4;a=2;b=5	2	5	0	7	0
L5;a=2;b=5	2	5	1	D	0
L6;a=2;b=5	2	5	2	0	0
L7;a=2;b=5	2	5	3	7	0
L4;a=2;b=6	2	6	0	8	0
L5;a=2;b=6	2	6	1	C	0
L6;a=2;b=6	2	6	2	2	0
L7;a=2;b=6	2	6	3	6	0
L4;a=2;b=7	2	7	0	9	0
L5;a=2;b=7	2	7	1	B	0
L6;a=2;b=7	2	7	2	2	0
L7;a=2;b=7	2	7	3	7	0



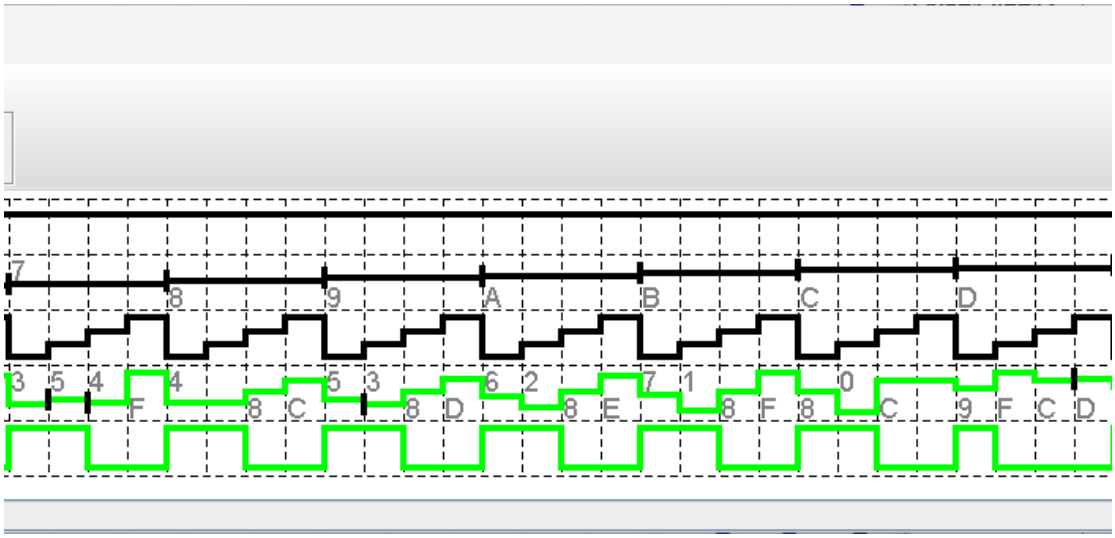
	A	B	S	C	Co
L5;a=12;b=11	C	B	1	1	1
L6;a=12;b=11	C	B	2	8	0
L7;a=12;b=11	C	B	3	F	0
L4;a=12;b=12	C	C	0	8	1
L5;a=12;b=12	C	C	1	0	1
L6;a=12;b=12	C	C	2	C	0
L7;a=12;b=12	C	C	3	C	0
L4;a=12;b=13	C	D	0	9	1
L5;a=12;b=13	C	D	1	F	0
L6;a=12;b=13	C	D	2	C	0
L7;a=12;b=13	C	D	3	D	0
L4;a=12;b=14	C	E	0	A	1
L5;a=12;b=14	C	E	1	E	0
L6;a=12;b=14	C	E	2	C	0
L7;a=12;b=14	C	E	3	E	0
L4;a=12;b=15	C	F	0	B	1
L5;a=12;b=15	C	F	1	D	0
L6;a=12;b=15	C	F	2	C	0
L7;a=12;b=15	C	F	3	F	0
L4;a=13;b=0	D	0	0	D	0
L5;a=13;b=0	D	0	1	D	1
L6;a=13;b=0	D	0	2	0	0
L7;a=13;b=0	D	0	3	D	0
L4;a=13;b=1	D	1	0	E	0
L5;a=13;b=1	D	1	1	C	1
L6;a=13;b=1	D	1	2	1	0
L7;a=13;b=1	D	1	3	D	0
L4;a=13;b=2	D	2	0	F	0
L5;a=13;b=2	D	2	1	B	1
L6;a=13;b=2	D	2	2	0	0
L7;a=13;b=2	D	2	3	F	0
L4;a=13;b=3	D	3	0	0	1
L5;a=13;b=3	D	3	1	A	1
L6;a=13;b=3	D	3	2	1	0
L7;a=13;b=3	D	3	3	F	0
L4;a=13;b=4	D	4	0	1	1
L5;a=13;b=4	D	4	1	9	1
L6;a=13;b=4	D	4	2	4	0
L7;a=13;b=4	D	4	3	D	0
L4;a=13;b=5	D	5	0	2	1
L5;a=13;b=5	D	5	1	8	1
L6;a=13;b=5	D	5	2	5	0
L7;a=13;b=5	D	5	3	D	0
L4;a=13;b=6	D	6	0	3	1
L5;a=13;b=6	D	6	1	7	1
L6;a=13;b=6	D	6	2	4	0
L7;a=13;b=6	D	6	3	F	0





由于测试数据过多（共 1024 组），这里只对测试用例及典型结果进行分析。(a&b),(a|b)即为 a,b 之间位运算的结果。

接下来对这一段波形图进行分析：



第一行为 A（始终为 12）；第二行为 B（从 7 到 13）；第三行为 S（0 表示加法，1 表示减法，2 表示按位与，3 表示按位或）；第四行为 C，表示运算结果的低 4 位；第五行为 Co，表示进位（位运算始终为 0）。

则这些结果分别表示：

- 12+7=3(+16),12-7=5,12&7=4,12|7=15
- 12+8=4(+16),12-8=4,12&8=8,12|8=12
- 12+9=5(+16),12-9=3,12&9=8,12|9=13
- 12+10=6(+16),12-10=2,12&10=8,12|10=14
- 12+11=7(+16),12-11=1,12&11=8,12|11=15
- 12+12=8(+16),12-12=0,12&12=12,12|12=12
- 12+13=9(+16),12-13=15(-16),12&13=12,12|13=13

其他波形可同理进行分析。由于该电路已通过所有的测试用例，因此该电路无误，可进行将其直接导出为 MyALU.v，为任务 2 做铺垫：

```
/*
 * Generated by Digital. Don't modify this file!
 * Any changes will be lost if this file is regenerated.
 */

module Adder1b (
  input A,
  input B,
  input C,
  output S,
  output Co
);
  assign S = ((A ^ B) ^ C);
  assign Co = ((A & B) | (A & C) | (C & B));
endmodule

module AddSub1b (
```

```
  input A,
  input B,
  input Ctrl,
  input Ci,
  output S,
  output Co
);
  wire s0;
  assign s0 = (B ^ Ctrl);
  Adder1b Adder1b_i0 (
    .A( A ),
    .B( s0 ),
    .C( Ci ),
    .S( S ),
    .Co( Co )
  );
endmodule

module AddSub4b (
  input [3:0] A,
  input [3:0] B,
  input Ctrl,
  output [3:0] S,
  output Co
);
  wire s0;
  wire s1;
  wire s2;
  wire s3;
  wire s4;
  wire s5;
  wire s6;
  wire s7;
  wire s8;
  wire s9;
  wire s10;
  wire s11;
  wire s12;
  wire s13;
  wire s14;
  assign s0 = A[0];
  assign s1 = A[1];
  assign s2 = A[2];
  assign s3 = A[3];
```

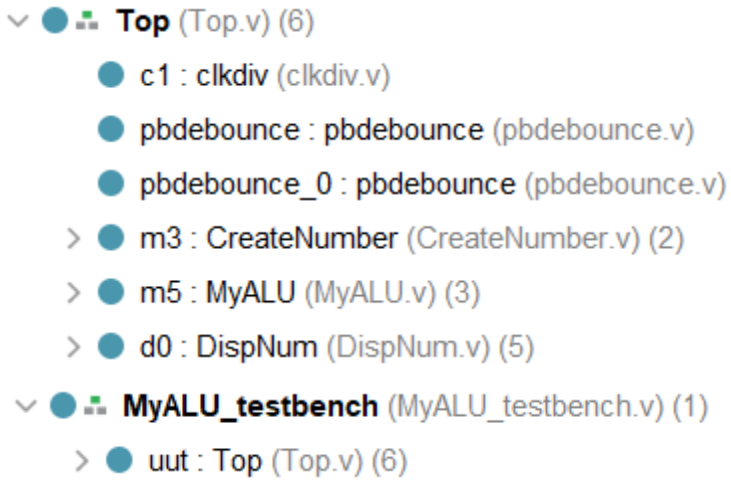
```
  assign s4 = B[0];
  assign s5 = B[1];
  assign s6 = B[2];
  assign s7 = B[3];
  AddSub1b AddSub1b_i0 (
    .A( s0 ),
    .B( s4 ),
    .Ctrl( Ctrl ),
    .Ci( Ctrl ),
    .S( s8 ),
    .Co( s9 )
  );
  AddSub1b AddSub1b_i1 (
    .A( s1 ),
    .B( s5 ),
    .Ctrl( Ctrl ),
    .Ci( s9 ),
    .S( s10 ),
    .Co( s11 )
  );
  AddSub1b AddSub1b_i2 (
    .A( s2 ),
    .B( s6 ),
    .Ctrl( Ctrl ),
    .Ci( s11 ),
    .S( s12 ),
    .Co( s13 )
  );
  AddSub1b AddSub1b_i3 (
    .A( s3 ),
    .B( s7 ),
    .Ctrl( Ctrl ),
    .Ci( s13 ),
    .S( s14 ),
    .Co( Co )
  );
  assign S[0] = s8;
  assign S[1] = s10;
  assign S[2] = s12;
  assign S[3] = s14;
endmodule

module Mux4to1b4 (
  input [1:0] S,
```

```
  input [3:0] I0,
  input [3:0] I1,
  input [3:0] I2,
  input [3:0] I3,
  output [3:0] O
);
  wire s0;
  wire s1;
  wire s2;
  wire s3;
  wire s4;
  wire s5;
  wire s6;
  wire s7;
  assign s0 = S[0];
  assign s1 = S[1];
  assign s2 = ~ s0;
  assign s3 = ~ s1;
  assign s7 = (s1 & s0);
  assign s4 = (s2 & s3);
  assign s5 = (s0 & s3);
  assign s6 = (s2 & s1);
  assign O[0] = ((s4 & I0[0]) | (s5 & I1[0]) | (s6 & I2[0]) | (s7 &
I3[0]));
  assign O[1] = ((s4 & I0[1]) | (s5 & I1[1]) | (s6 & I2[1]) | (s7 &
I3[1]));
  assign O[2] = ((s4 & I0[2]) | (s5 & I1[2]) | (s6 & I2[2]) | (s7 &
I3[2]));
  assign O[3] = ((s4 & I0[3]) | (s5 & I1[3]) | (s6 & I2[3]) | (s7 &
I3[3]));
endmodule

module Mux4to1 (
  input [1:0] S,
  input [3:0] I,
  output O
);
  assign O = (((~ S[0] & ~ S[1]) & I[0]) | ((S[0] & ~ S[1]) & I[1]) |
((~ S[0] & S[1]) & I[2]) | ((S[1] & S[0]) & I[3]));
endmodule

module MyALU (
  input [3:0] A,
  input [3:0] B,
```

```
  input [1:0] S,
  output [3:0] C,
  output Co
);
  wire s0;
  wire [3:0] s1;
  wire s2;
  wire [3:0] s3;
  wire [3:0] s4;
  wire [3:0] s5;
  assign s3 = (A & B);
  assign s4 = (A | B);
  assign s0 = S[0];
  AddSub4b AddSub4b_i0 (
    .A( A ),
    .B( B ),
    .Ctrl( s0 ),
    .S( s1 ),
    .Co( s2 )
  );
  assign s5[0] = s2;
  assign s5[1] = s2;
  assign s5[2] = 1'b0;
  assign s5[3] = 1'b0;
  Mux4to1b4 Mux4to1b4_i1 (
    .S( S ),
    .I0( s1 ),
    .I1( s1 ),
    .I2( s3 ),
    .I3( s4 ),
    .O( C )
  );
  Mux4to1 Mux4to1_i2 (
    .S( S ),
    .I( s5 ),
    .O( Co )
  );
endmodule
```

2. 设计 4 位 ALU 应用工程，在实验板上下载验证

按要求创建 MyALU 工程，其结构如图所示：



MyALU.v 为导入文件，代码已展示，故不再重复展示。
clkdiv.v 的代码与 DispNum.v 的代码与实验 7 完全一致，故不再重复展示。
CreateNumber.v 代码如下：

```
module CreateNumber(
    input wire [3:0] btn,
    input wire [3:0] sw,
    output reg [7:0] num
    );
    wire [3:0] A,B;

    initial num <= 16'hDD;

    AddSub4b a1 (.A(num[3:0]), .B(4'b1), .Ctrl(sw[1]), .S(A), .Co());
    AddSub4b a2 (.A(num[7:4]), .B(4'b1), .Ctrl(sw[0]), .S(B), .Co());

    always@(posedge btn[3]) num[3:0]<=A;
    always@(posedge btn[2]) num[7:4]<=B;

endmodule
```

pbdebounce.v 代码如下（用于防抖）：

```
module pbdebounce(
    input wire clk_1ms,
    input wire button,
    output reg pbreg
    );

    reg [7:0] pbshift;

    always@(posedge clk_1ms) begin
        pbshift=pbshift<<1;
        pbshift[0]=button;
        if (pbshift==8'b0)
```

```
            pbreg=0;
        if (pbshift==8'hFF)
            pbreg=1;
    end
endmodule
```

Top.v 代码如下（防抖版本）：

```
`timescale 1ns / 1ps
module Top(
    input wire clk,
    input wire [15:0] SW,
    input wire [3:0] BTN,
    output wire [3:0] AN,
    output wire [7:0] SEGMENT,
    output wire BTNX4
);

    wire [15:0] num;
    wire [1:0] btn_out;
    wire [3:0] C;
    wire Co;
    wire [31:0] clk_div;
    wire [15:0] disp_hexs;
    assign disp_hexs[15:12] = num[3:0]; //A
    assign disp_hexs[11:8] = num[7:4];  //B
    assign disp_hexs[7:4] = {3'b000, Co};
    assign disp_hexs[3:0] = C[3:0];

    clkdiv c1(.clk(clk), .rst(1'b0), .clk_div(clk_div));
    wire [3:0] btn;
    assign btn[1:0]=BTN[1:0];
    pbdebounce(.clk_1ms(clk_div[17]),.button(BTN[3]),.pbreg(btn[3]));
    pbdebounce(.clk_1ms(clk_div[17]),.button(BTN[2]),.pbreg(btn[2]));
    CreateNumber m3(.btn(btn[3:0]),.sw(SW[3:0]),.num(num));
    MyALU
m5(.A(disp_hexs[15:12]),.B(disp_hexs[11:8]),.S(SW[15:14]),.C(disp_hexs[
3:0]),.Co(disp_hexs[4]));
    DispNum d0(.scan(clk_div[18:17]), .HEXS(disp_hexs),
     .LES(4'b0), .point(4'b0), .AN(AN), .SEGMENT(SEGMENT));
    assign BTNX4 = 1'b0;    //Enable button inputs
endmodule
```

MyALU_testbench.v 如下：

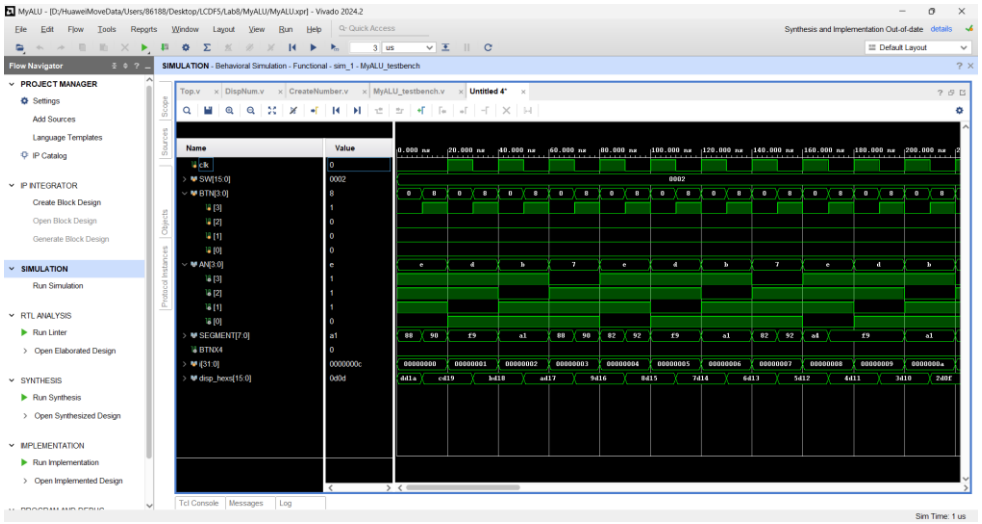
```
module MyALU_testbench();
    reg clk;
    reg [15:0] SW;
```

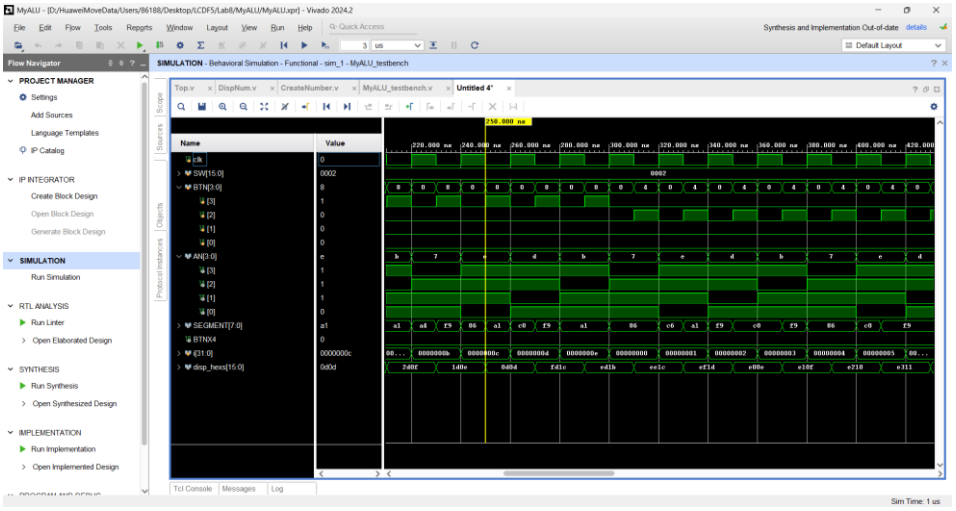
```
reg [3:0] BTN;
wire [3:0] AN;
wire [7:0] SEGMENT;
wire BTNX4;
Top uut(
   .clk(clk),
   .SW(SW),
   .BTN(BTN),
   .AN(AN),
   .SEGMENT(SEGMENT),
   .BTNX4(BTNX4)
);
integer i;
initial begin
   clk=0;
   SW[15:0]=16'b0000_0000_0000_0010;//SW[1]=1,SW[0]=0
   BTN[3:0]=4'b0000;
   for(i=0;i<15;i=i+1) begin
   #10
   BTN[3]=1;
   #10
   BTN[3]=0;
   end
   for(i=0;i<13;i=i+1) begin
   #10
   BTN[2]=1;
   #10
   BTN[2]=0;
   end
   #80
   SW[14]=0;
   #80
   SW[14]=1;
   #80
   SW[15]=1;
   SW[14]=0;
   #80
   SW[15]=1;
   SW[14]=1;
end
always begin
    #10 clk = 0;
    #10 clk = 1;
end
```

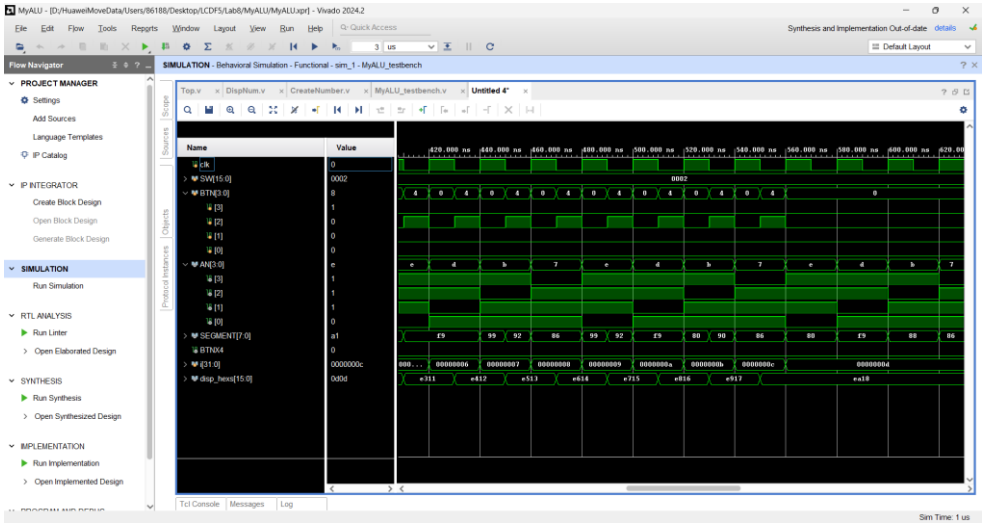
```
endmodule
```

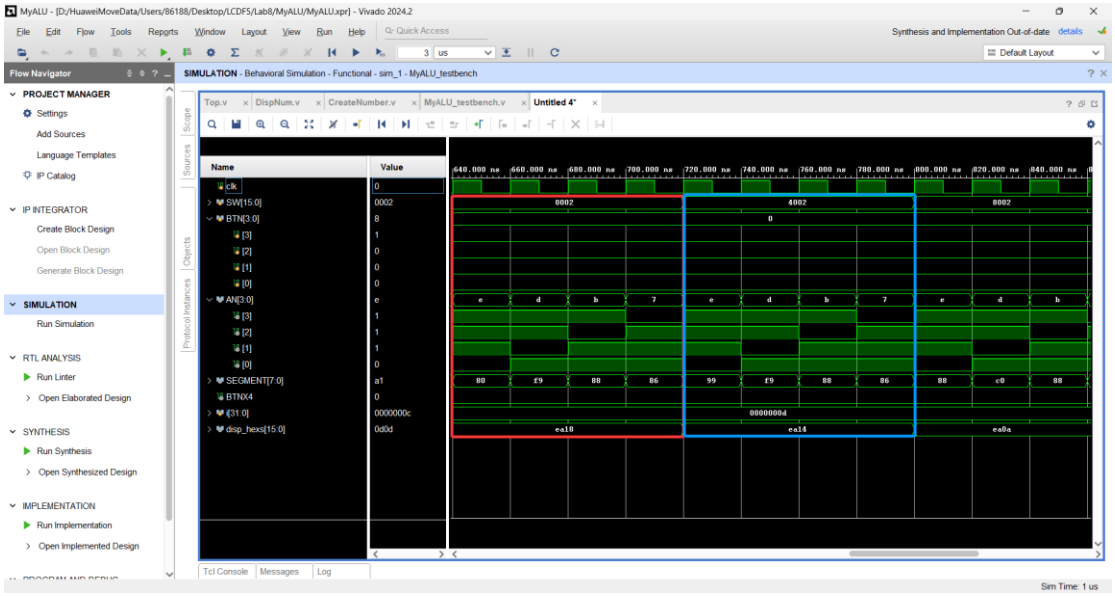
注：在仿真中 clk_div[18:17]应改为 clk_div[1:0]，且无防抖。其仿真为将 A 从 D 开始一直自减，直到减到 E（D->C->B->…->0->F->E，共 15 次），将 B 从 D 开始一直自增，直到增到 A（D->E->F->0->1->…->A）。

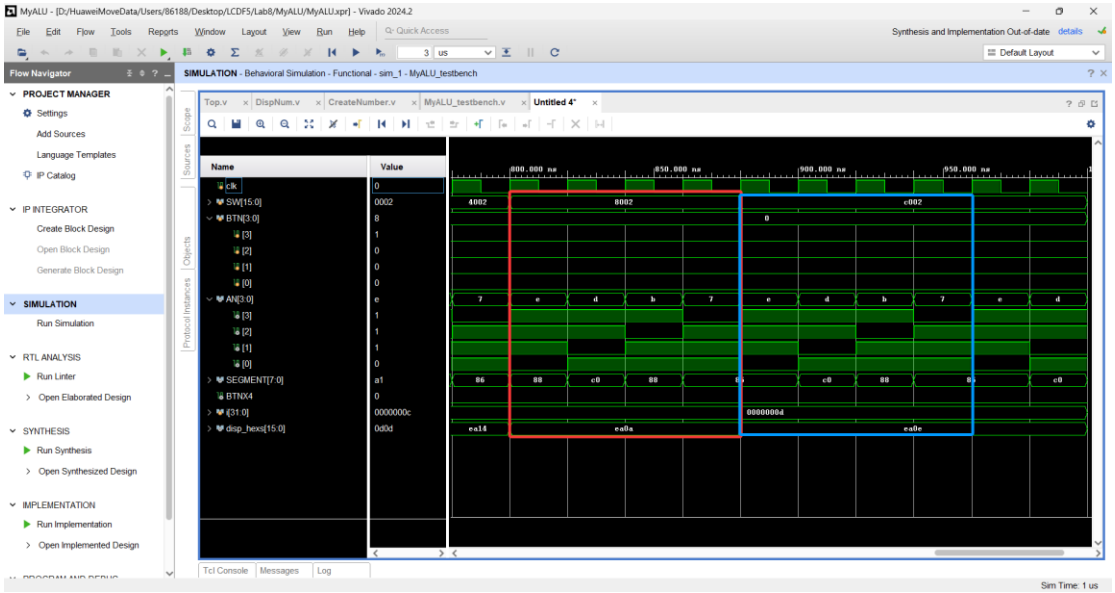
仿真结果如下图所示：











这里对 A=E,B=A 的情况进行分析：

- E+A=8(+16)，即 Co=1,C=8
- E-A=4，即 Co=1,C=4
- E&A=A，即 Co=0,C=A
- E|A=E，即 Co=0,C=E

与仿真结果完全一致，说明整个工程无误。

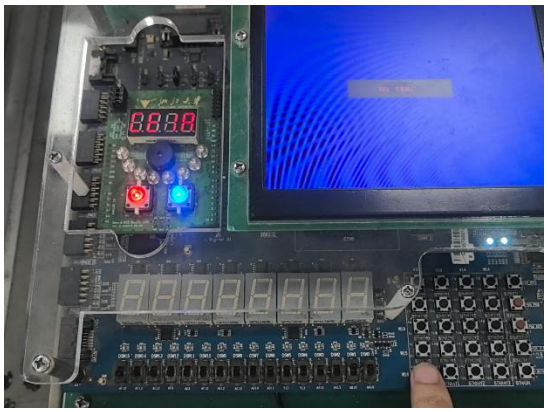
引脚约束代码如下：

```
set_property PACKAGE_PIN AA10 [get_ports {SW[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[0]}]
set_property PACKAGE_PIN AB10 [get_ports {SW[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[1]}]
set_property PACKAGE_PIN AA13 [get_ports {SW[2]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[2]}]
set_property PACKAGE_PIN AA12 [get_ports {SW[3]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[3]}]
set_property PACKAGE_PIN Y13 [get_ports {SW[4]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[4]}]
set_property PACKAGE_PIN Y12 [get_ports {SW[5]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[5]}]
set_property PACKAGE_PIN AD11 [get_ports {SW[6]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[6]}]
set_property PACKAGE_PIN AD10 [get_ports {SW[7]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[7]}]
set_property PACKAGE_PIN AE10 [get_ports {SW[8]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[8]}]
set_property PACKAGE_PIN AE12 [get_ports {SW[9]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[9]}]
set_property PACKAGE_PIN AF12 [get_ports {SW[10]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[10]}]
set_property PACKAGE_PIN AE8 [get_ports {SW[11]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[11]}]
set_property PACKAGE_PIN AF8 [get_ports {SW[12]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[12]}]
set_property PACKAGE_PIN AE13 [get_ports {SW[13]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[13]}]
set_property PACKAGE_PIN AF13 [get_ports {SW[14]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[14]}]
set_property PACKAGE_PIN AF10 [get_ports {SW[15]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[15]}]
```

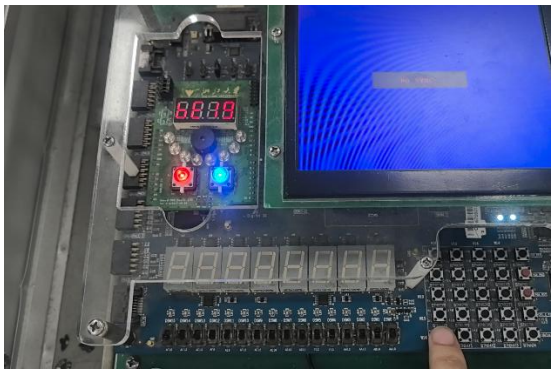
```
set_property PACKAGE_PIN AB22 [get_ports {SEGMENT[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[0]}]
set_property PACKAGE_PIN AD24 [get_ports {SEGMENT[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[1]}]
set_property PACKAGE_PIN AD23 [get_ports {SEGMENT[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[2]}]
set_property PACKAGE_PIN Y21 [get_ports {SEGMENT[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[3]}]
set_property PACKAGE_PIN W20 [get_ports {SEGMENT[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[4]}]
set_property PACKAGE_PIN AC24 [get_ports {SEGMENT[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[5]}]
set_property PACKAGE_PIN AC23 [get_ports {SEGMENT[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[6]}]
set_property PACKAGE_PIN AA22 [get_ports {SEGMENT[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[7]}]
set_property PACKAGE_PIN AD21 [get_ports {AN[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[0]}]
set_property PACKAGE_PIN AC21 [get_ports {AN[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[1]}]
set_property PACKAGE_PIN AB21 [get_ports {AN[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[2]}]
set_property PACKAGE_PIN AC22 [get_ports {AN[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[3]}]
create_clock -name clk100MHZ -period 10.0 [get_ports clk]
set_property PACKAGE_PIN AC18 [get_ports clk]
set_property IOSTANDARD LVCMOS18 [get_ports clk]
set_property PACKAGE_PIN W14 [get_ports BTN[0]]
set_property IOSTANDARD LVCMOS18 [get_ports BTN[0]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets BTN_IBUF[0]]
set_property PACKAGE_PIN V14 [get_ports BTN[1]]
set_property IOSTANDARD LVCMOS18 [get_ports BTN[1]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets BTN_IBUF[1]]
set_property PACKAGE_PIN V19 [get_ports BTN[2]]
set_property IOSTANDARD LVCMOS18 [get_ports BTN[2]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets BTN_IBUF[2]]
set_property PACKAGE_PIN V18 [get_ports BTN[3]]
set_property IOSTANDARD LVCMOS18 [get_ports BTN[3]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets BTN_IBUF[3]]
set_property PACKAGE_PIN W16 [get_ports BTNX4]
set_property IOSTANDARD LVCMOS18 [get ports BTNX4]
```

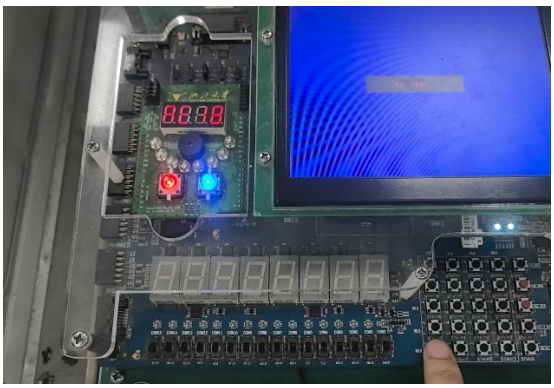
将其下载到板上进行验证，结果如下：

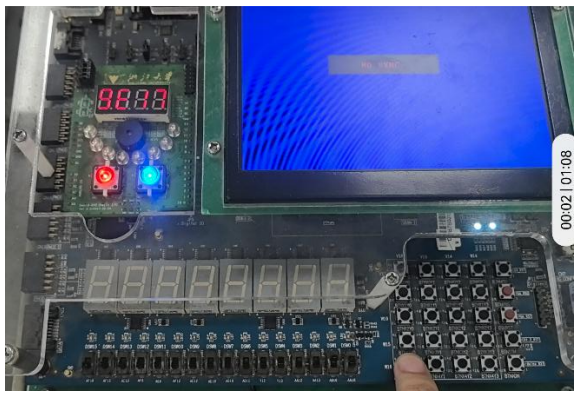
下载完成：

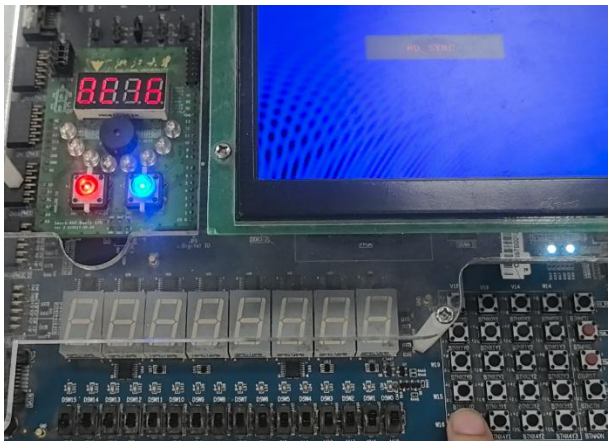


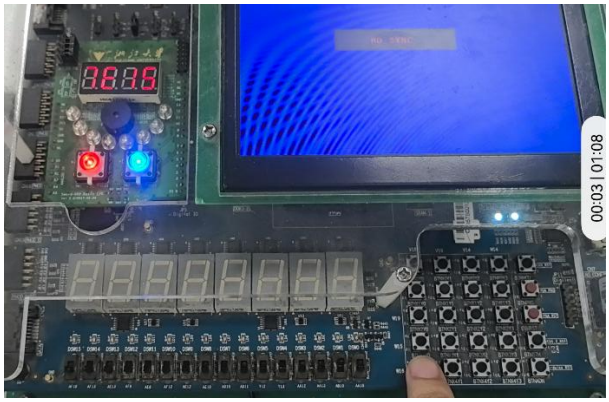
将 A 调整至 E（减法）：

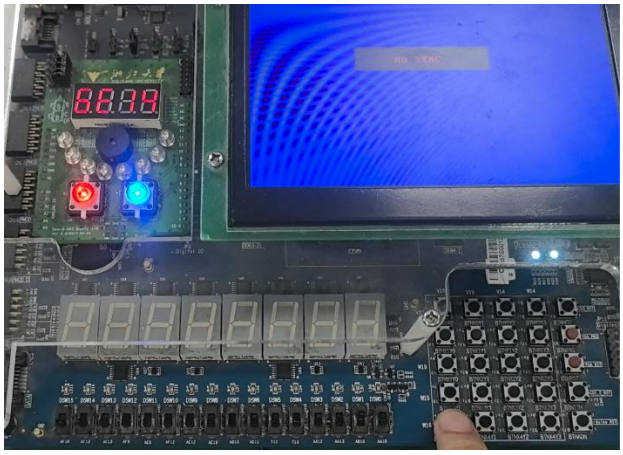


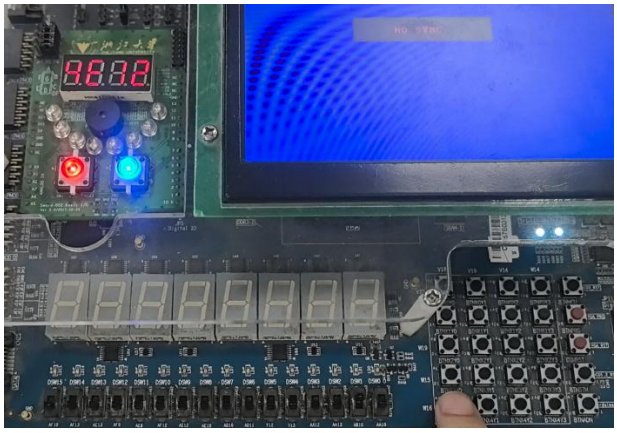


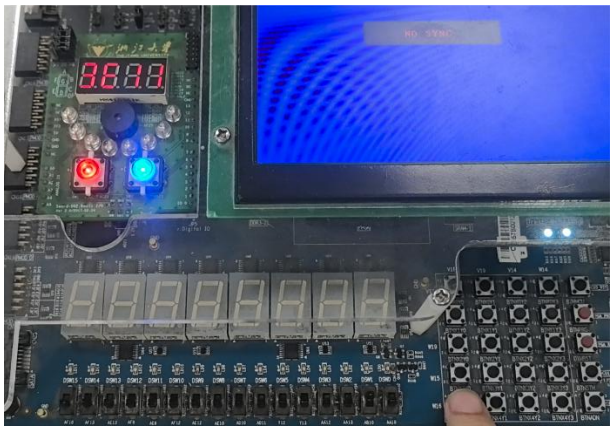


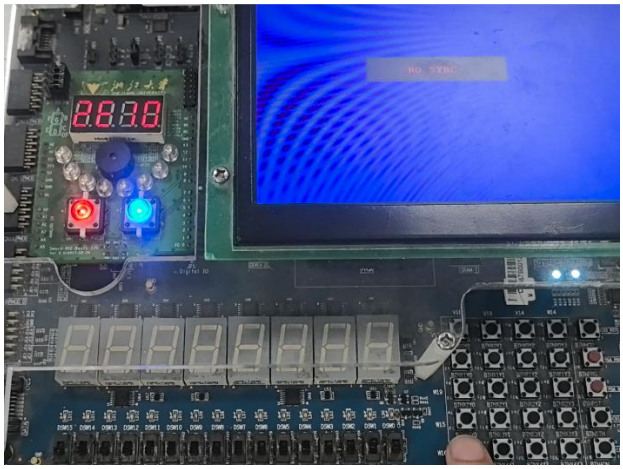

NO SYNC

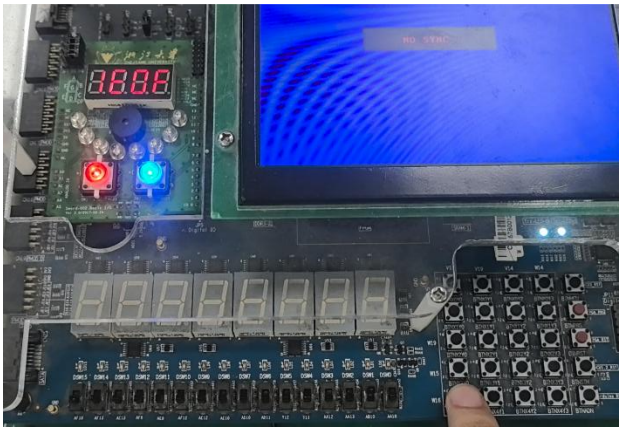

NO SYNC
00:03 | 01:08

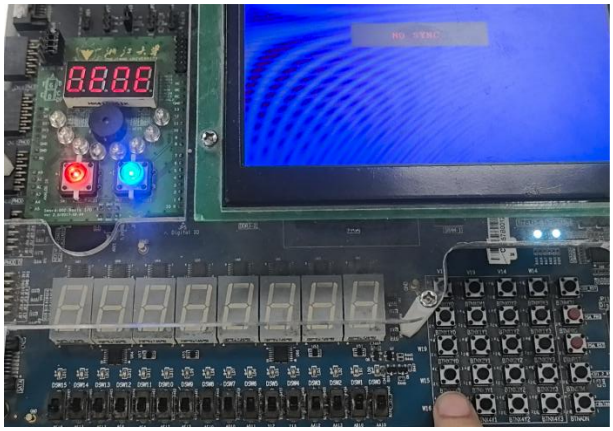

NO SYNC

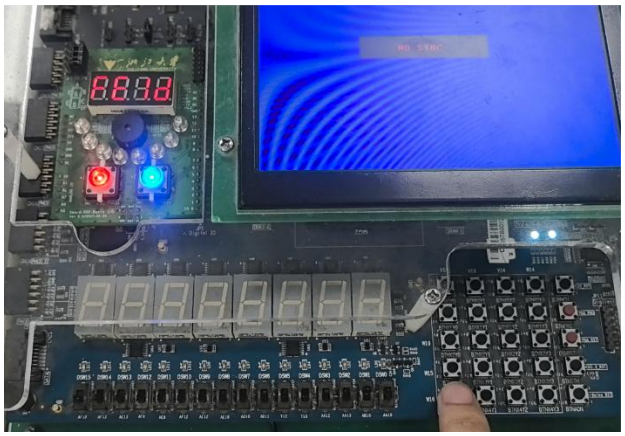

NO SYNC

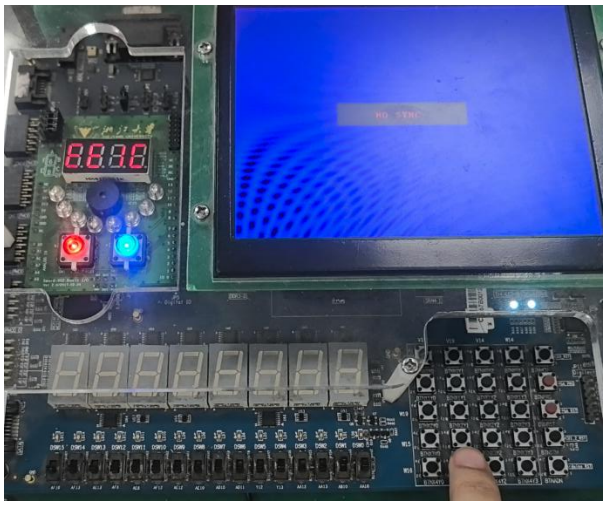

NO SYNC

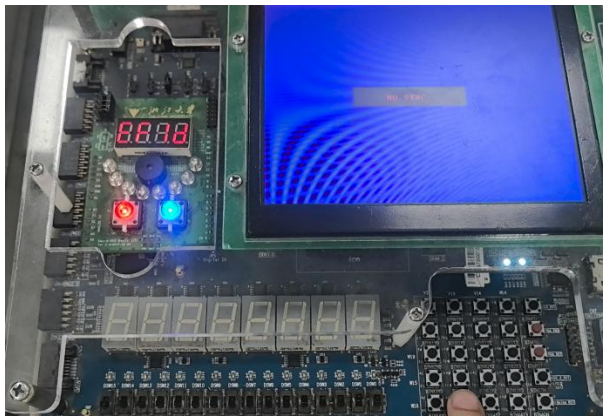

NO SYNC

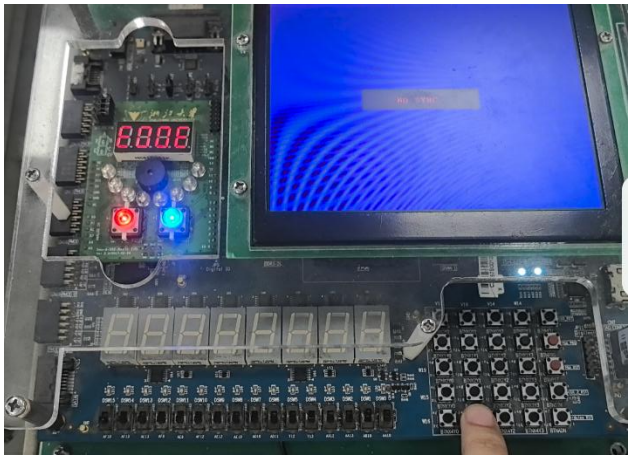

NO SYNC

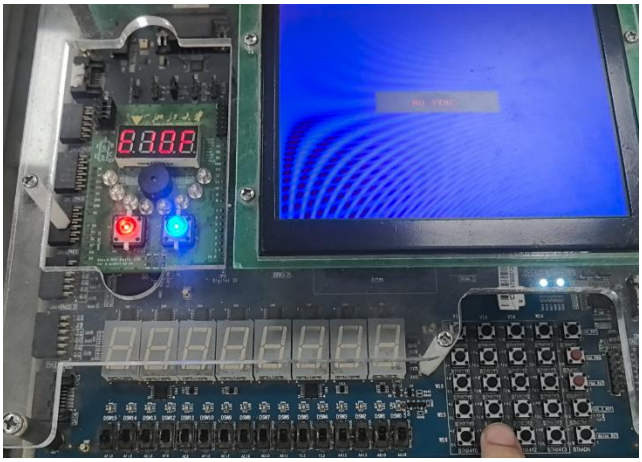

NO SYNC

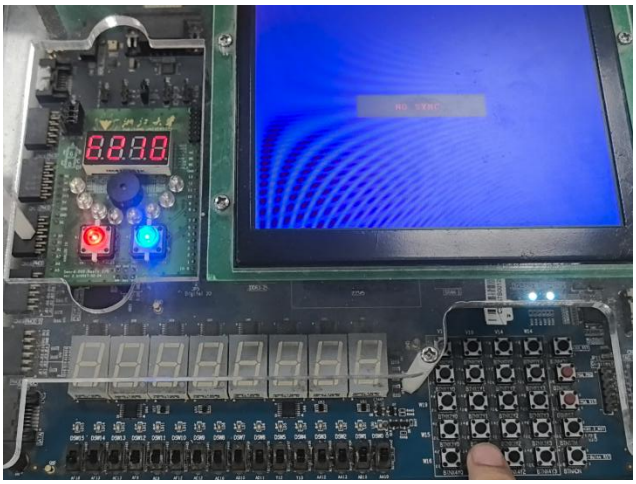

NO SYNC

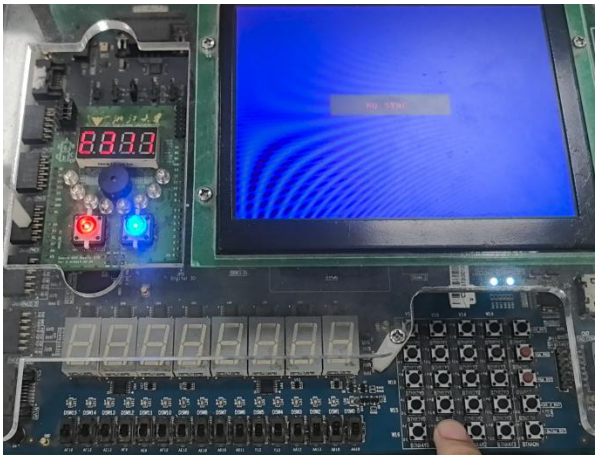

NO SYNC

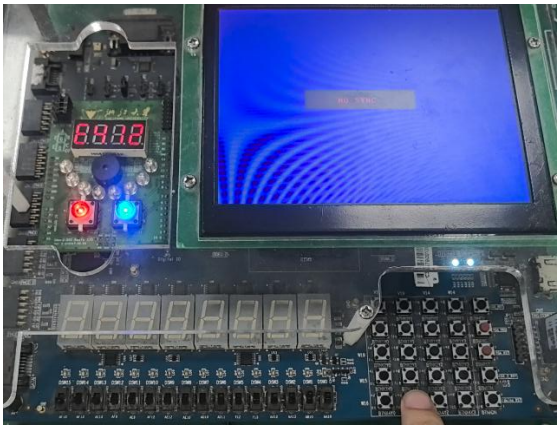

NO SYNC

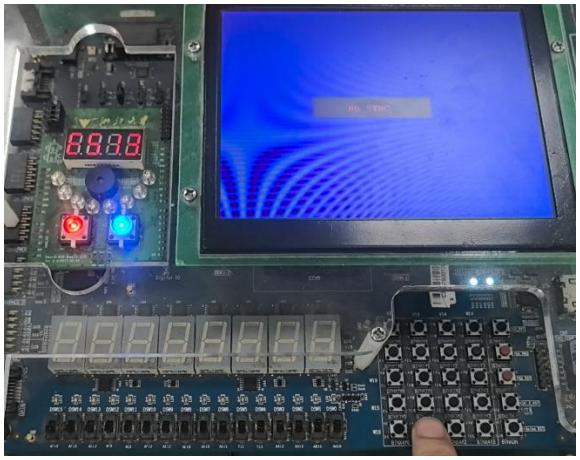

NO SYNC

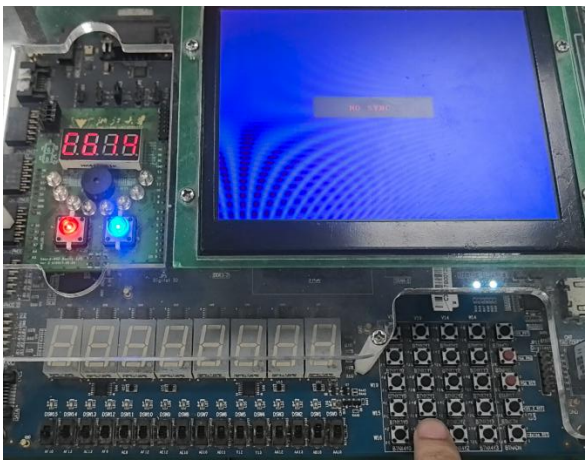

NO SYNC

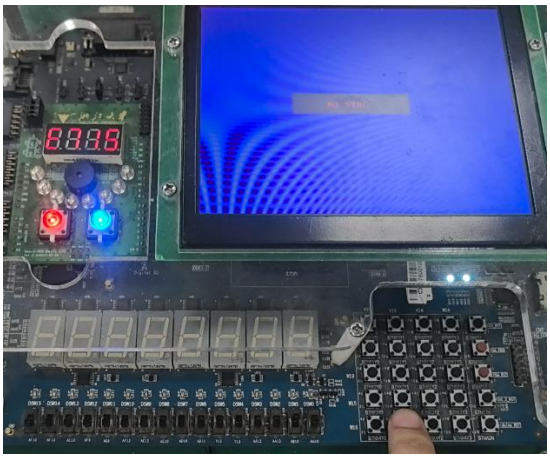

NO SYNC

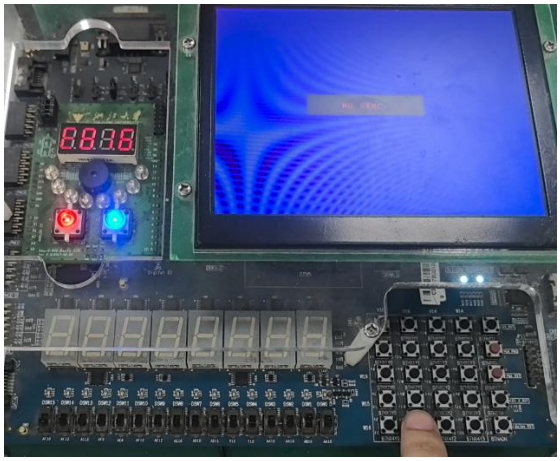

NO SYNC

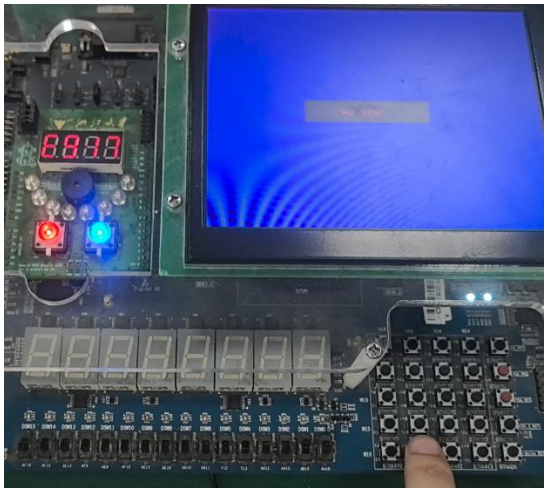

NO SYNC

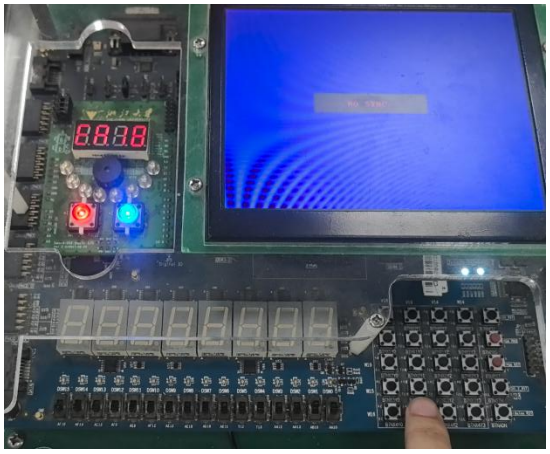




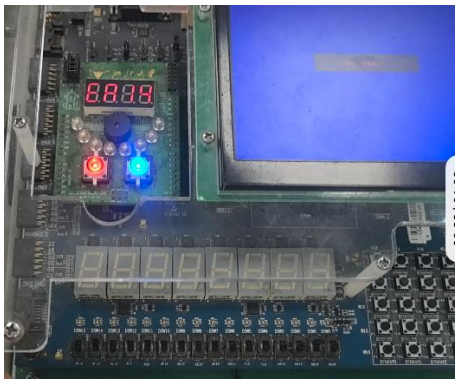


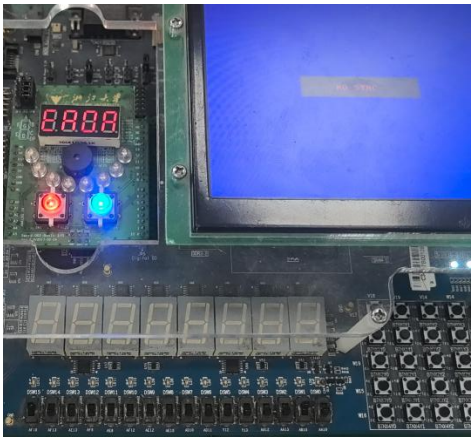


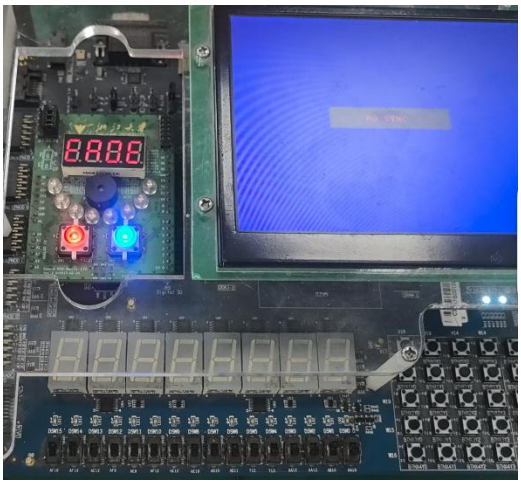




调整完成后（加法已展示），调整 SW[15]与 SW[14]来改变运算状态（顺序为减、与、或）：

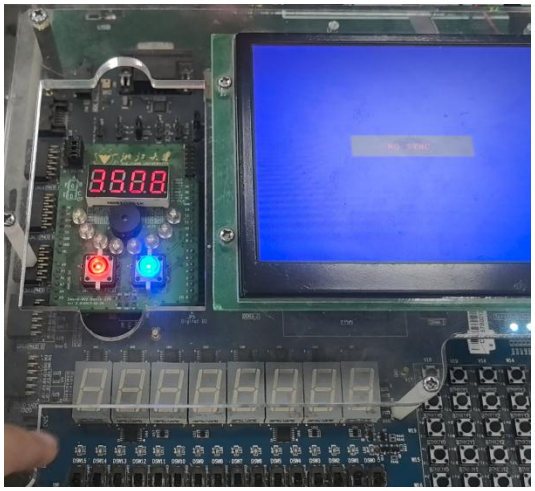


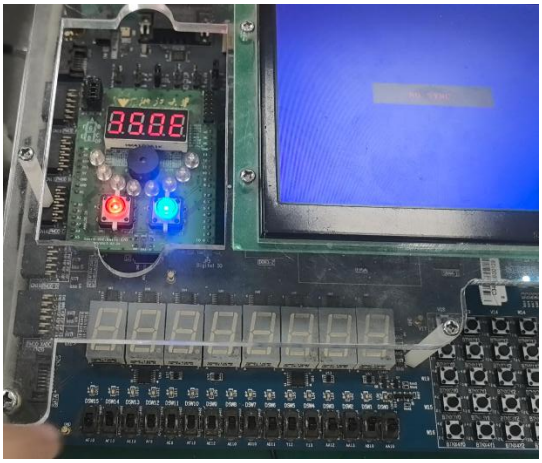


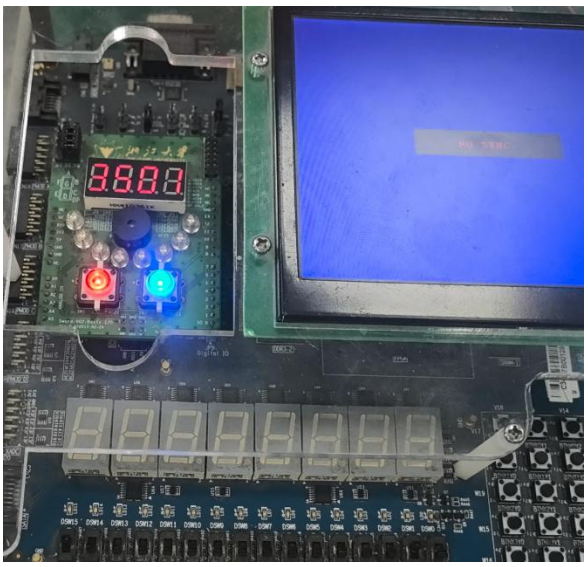


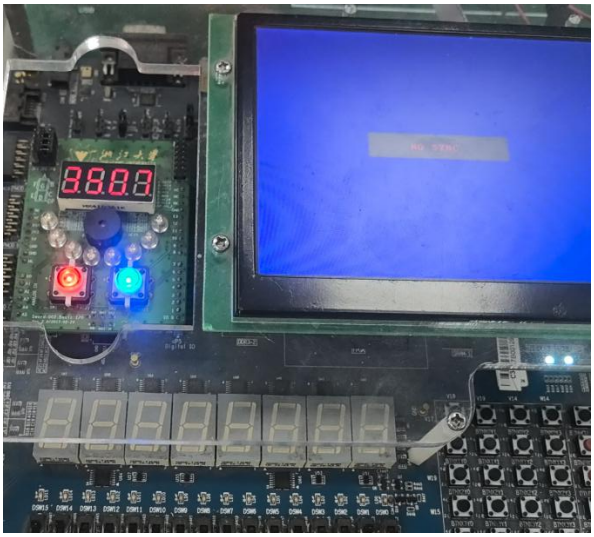
为了更好展现一般性，我还令 A=3,B=5，进行运算（顺序为加、减、与、或）：
其中：

- 3+5=8，故 Co=0,C=8
- 3-5=14(-16)，故 Co=0,C=E（借位）
- 3&5=1，故 Co=0,C=1
- 3|5=7，故 Co=0,C=7









注意到加了去抖动后，每按下一次按键，基本上都只会改变一个数字。照片偶然出现的变化 2 的情况应该属于抖动周期较大所致。

若想在仿真中做防抖动，可以将 clk[18:17]改为 clk[5:4]（不能是[1:0]），同时将 clk_1ms 的 clk[0]改为 clk[1]，并在仿真代码中人为制造短周期的抖动进行检验。

四、讨论、心得

这一次实验中，我吸取了上次实验的教训，提前将激励代码完成，从而在实验课上能够仅仅对激励代码微调即可通过验收。