山东大学 计算机科学与技术 学院

计算机组成与设计 课程实验报告

学号：202300130073	姓名：陈玉澍	班级：计科 23 级 1 班

实验题目：

实验 3 补码加法器

实验学时：2	实验日期：2025.3.11

实验目的：

采用一位全加器和异或门设计一个四位二进制补码加法器，操作数和运算结果均采用补码表示，该加法器要求可做补码加法，也可做补码减法运算。

实验软件和硬件环境：

软件环境：

Vivado 软件、FPGA 实验平台

硬件环境：

1. 实验室台式机

2. FPGA 服务器，PYNQ-Z2 开发板

实验原理和方法：

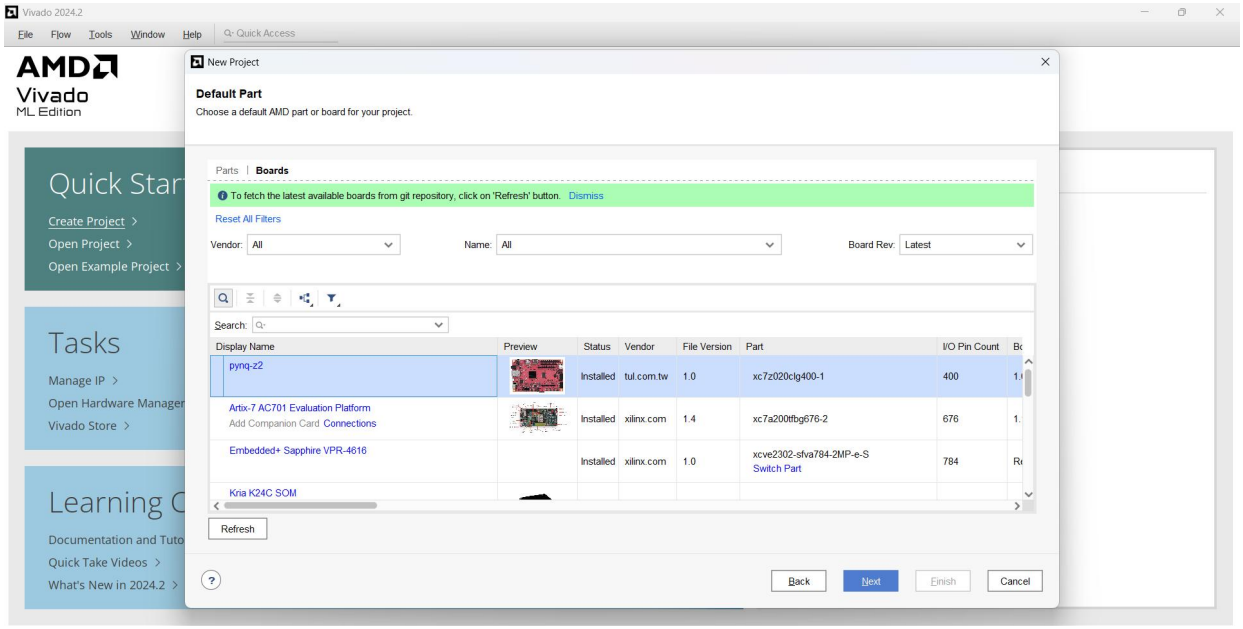
一、补码运算：

在补码运算中：（X)补 十 (Y)补 = (X 十 Y)补

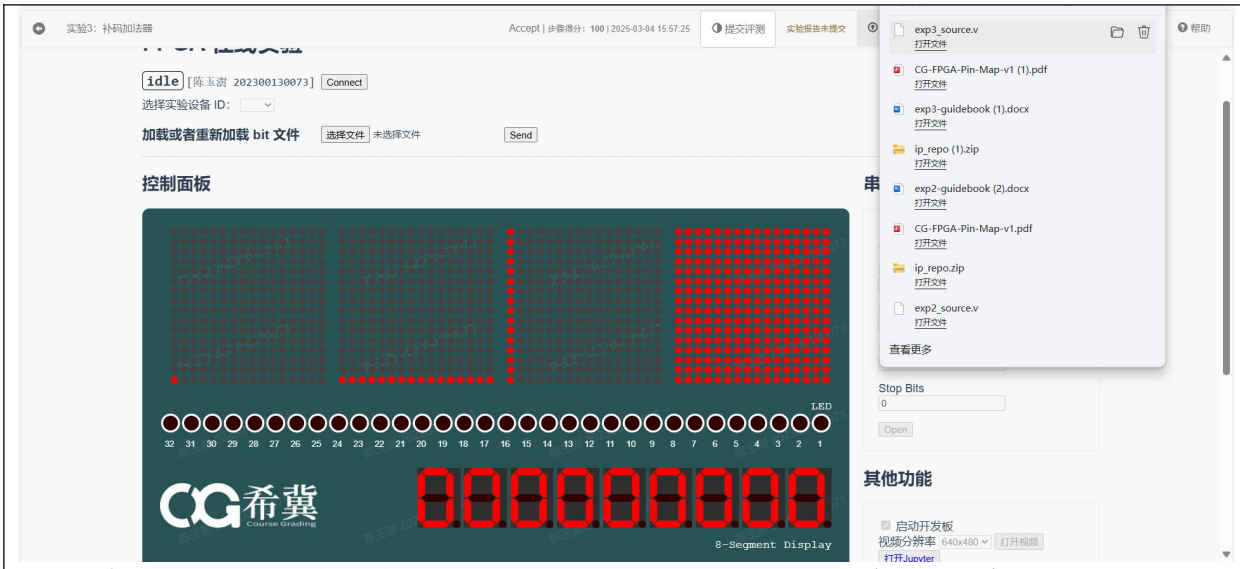
(X)补 - (Y)补 = (X)补 十 (-Y)补

实验步骤：

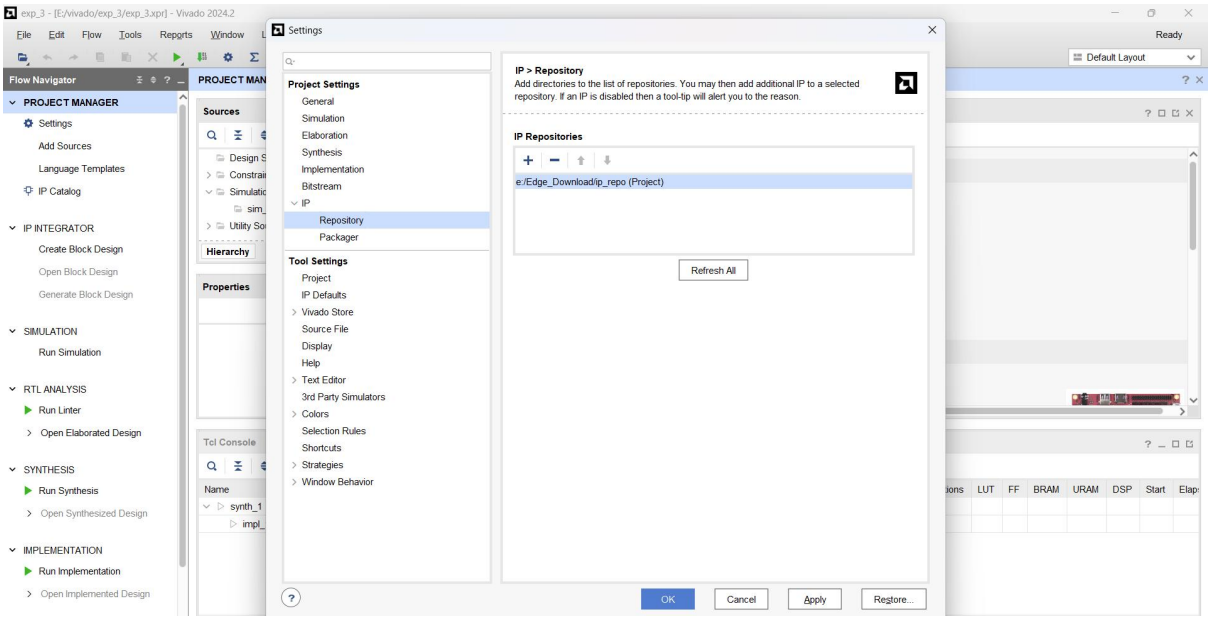
（1）创建工程：打开本地安装的 Vivado 2022.2，新建项目，选择 pynq-z2 器件。



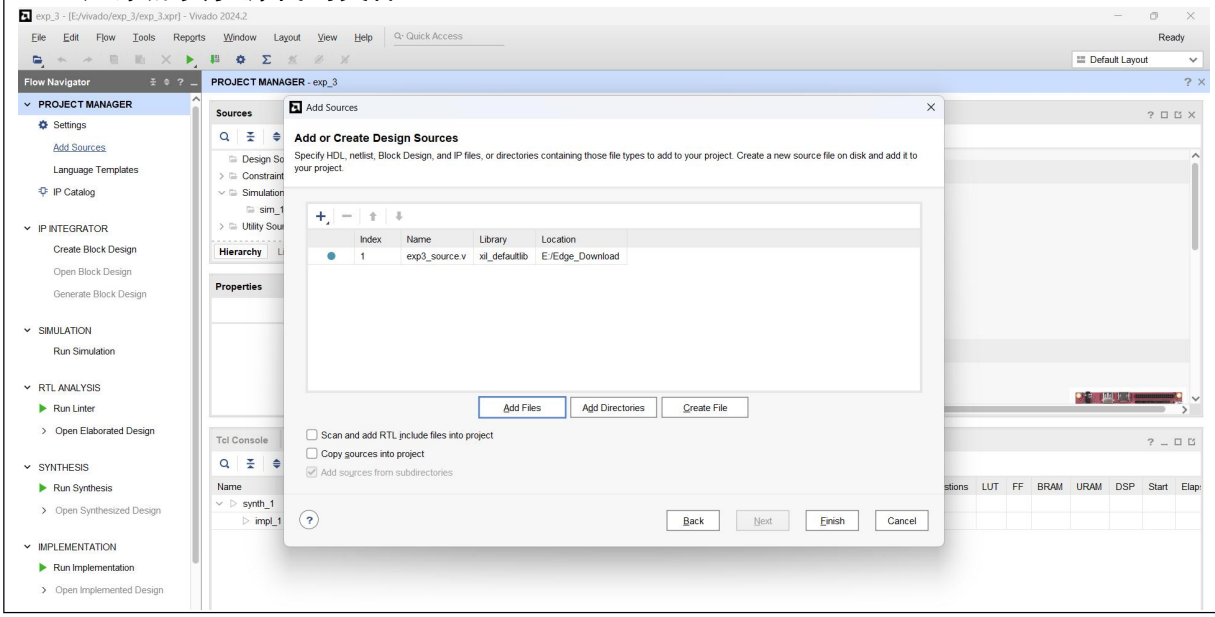
（2）添加实验环境：进入 FPGA 在线实验环境，点击右上角项目材料下载实验源代码和希冀 ip 核到本地并解压。



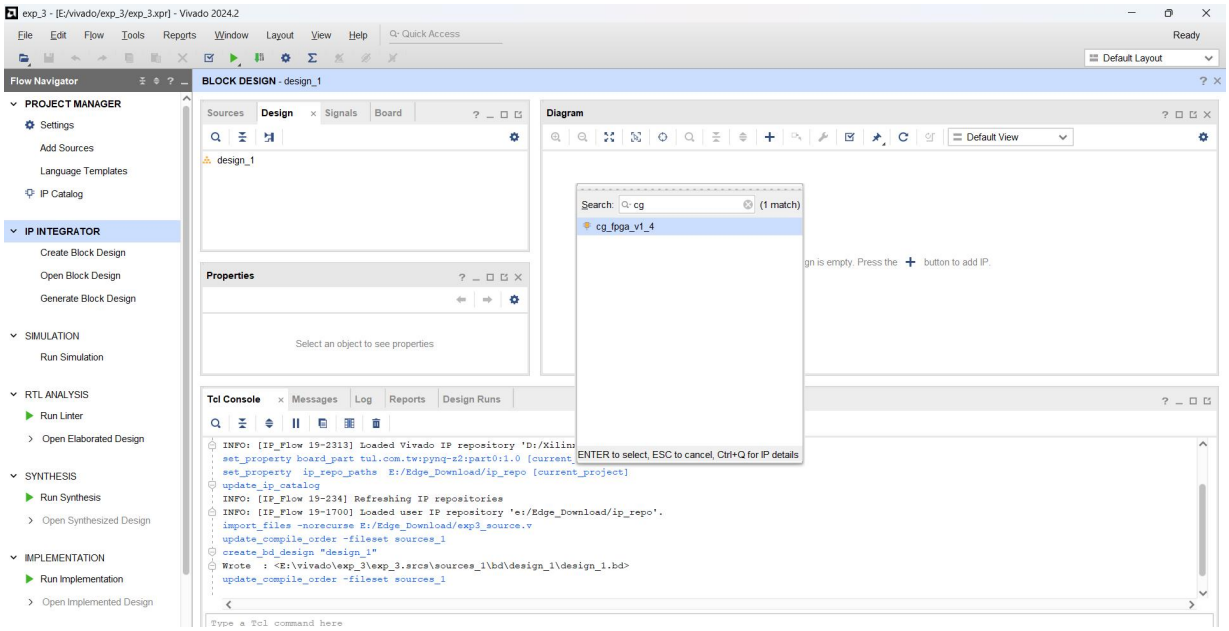
（3）在 Vivado 项目中，点击 Settings→IP→Repository，将上一步解压后的 ip_repo 文件夹的位置添加进 IP 搜索目录。



（4）点击 Sources 窗口中的+，选择 Add or create design sources → Next → Add File，添加实验源代码文件。

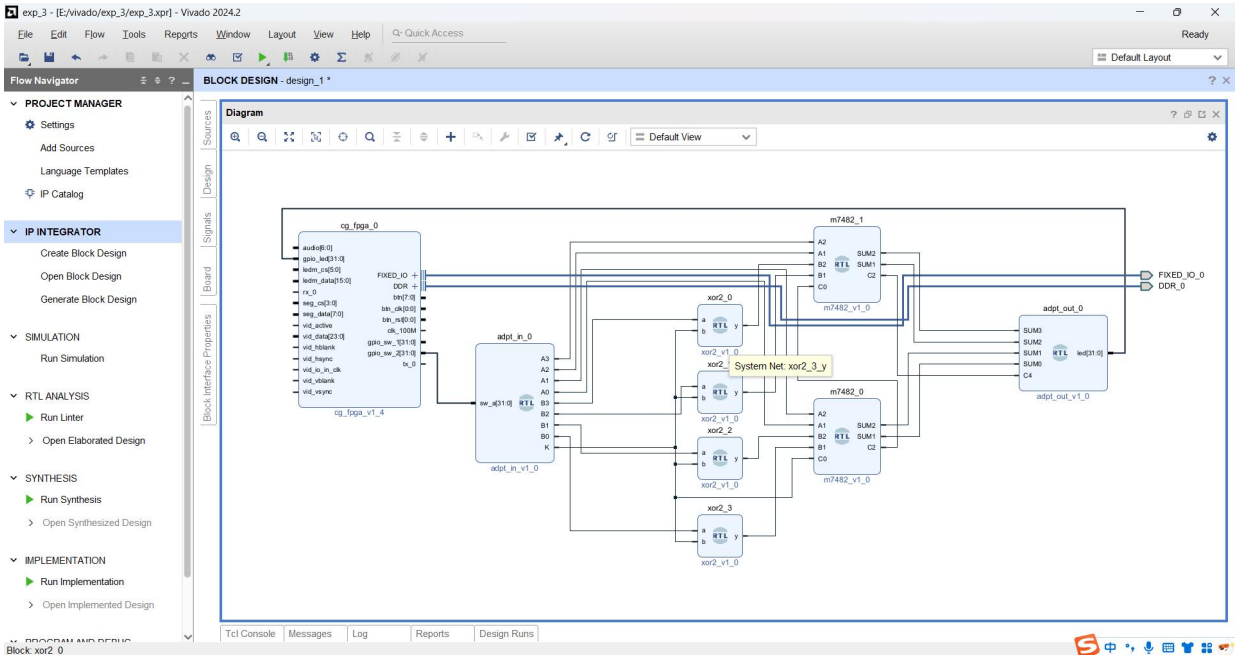


（5）点击 Create Block Design 创建一个新的顶层设计，随后点击添加 IP 核按钮，添加 cg_fpga IP.

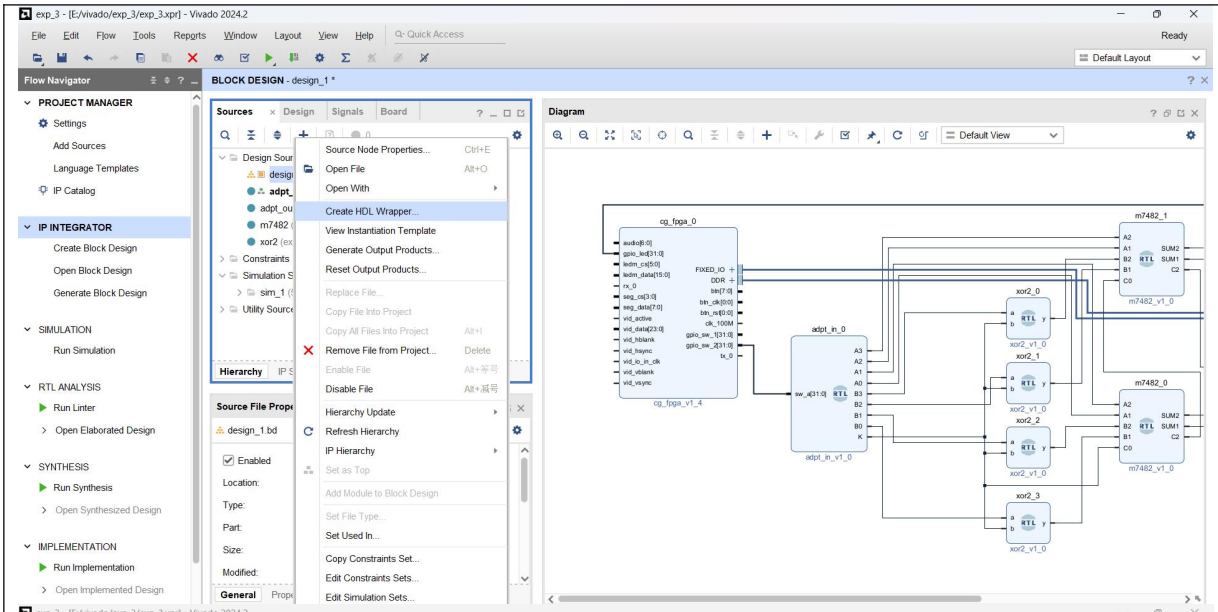


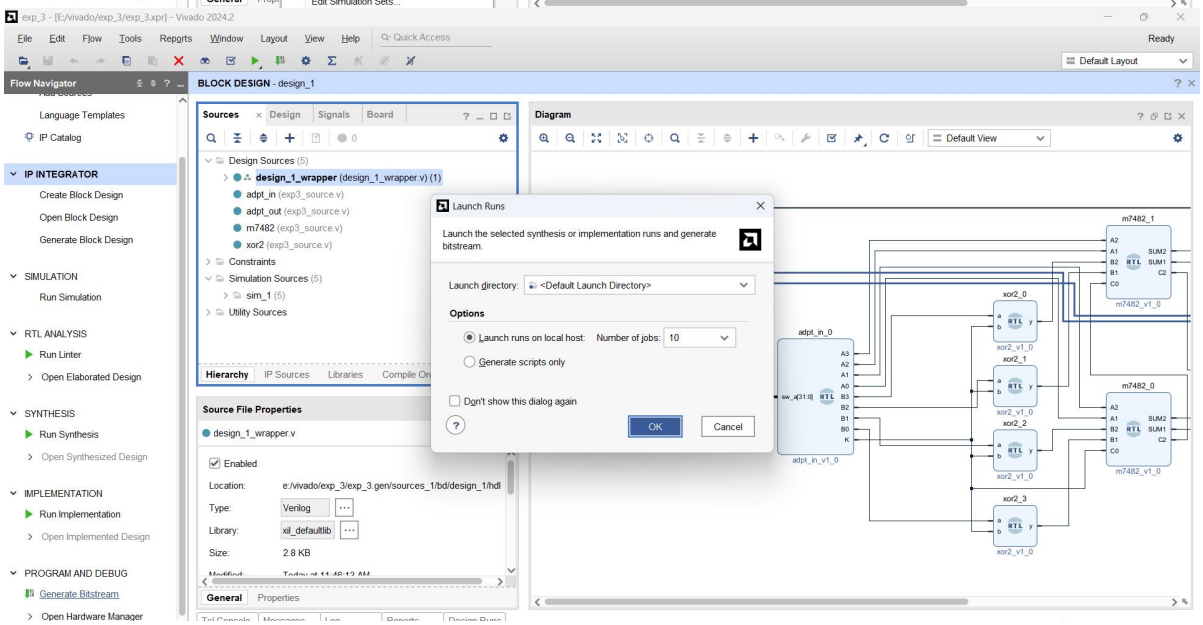
（6）在 Sources 窗口下的 Design sources 中，根据上面的电路图拖拽相应模块，完成原理图的输入。

（7）将输入的模块与 cg_fpga 如下图连接，并选择 cg_fpga 模块上的 FIXED_IO 和 DDR，点击右键→Make External. 这里为了保证补码减法器运算正确，可直接将 CO 与输入 K 连接。



（8）右击 Sources 下顶层设计图标→Create HDL Wrapper，待 Wrapper 正确生成后，点击左下方 Generate Bitstream，开始综合并生成 bit 文件。注意：综合前 wrapper 模块应被设置为顶层（加粗表示），若自动设置错误，需右击 wrapper 图标点击 Set as Top 手动设置。



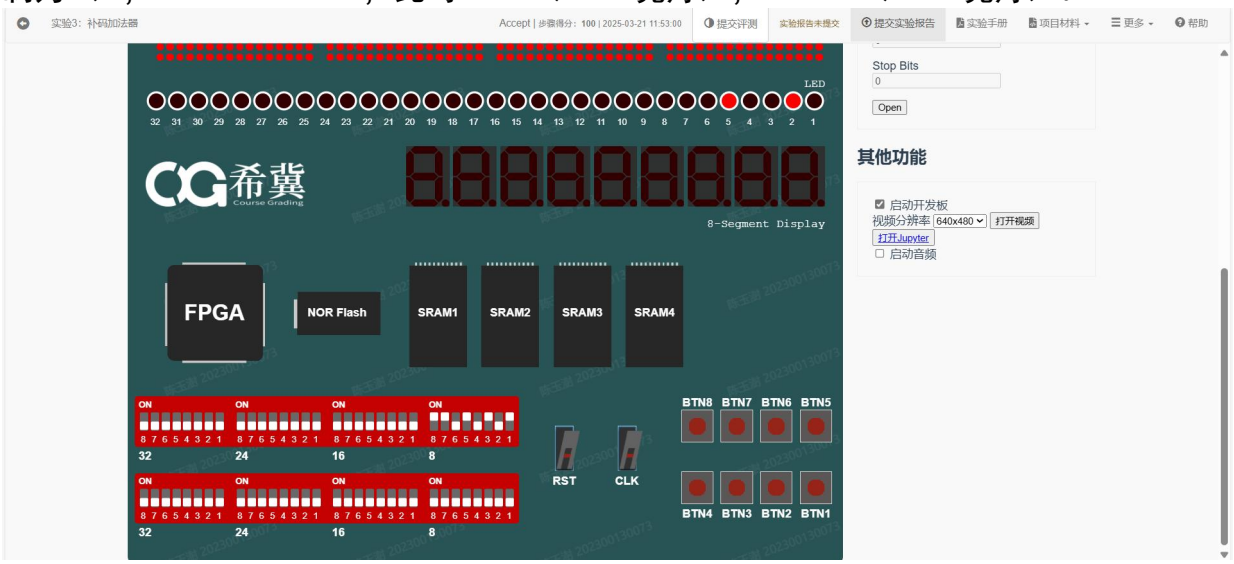


（9）通过 FPGA 云实验平台，可在线分配远程 FPGA 硬件开发板。首先点击 connect 按钮，然后在下拉菜单中选择任意空闲的开发板，并点击 Choose File 中选择上一步生成的 *.bit 文件，后点击 send，即可将本地 bit 文件烧写至希冀远程 FPGA.

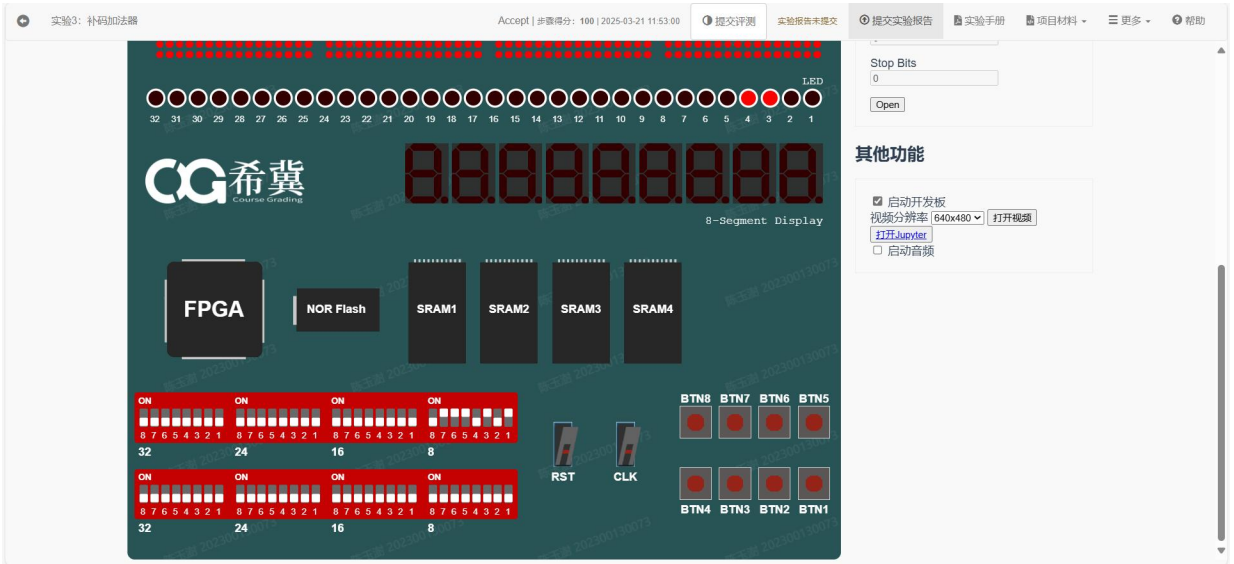
利用输入开关键改变 K、C0 和 A、B 操作数的值，看 LED 指示灯显示的结果是否正确并记录结果。

A3-A0 管脚分别绑定在上排拨码开关 4-1，B3-B0 管脚分别绑定在上排拨码开关 8-5，K 管脚绑定在上排拨码开关 9 上，SUM3-SUM0 管脚绑定在 LED4-1 上，C4 管脚绑定在 LED5 上。

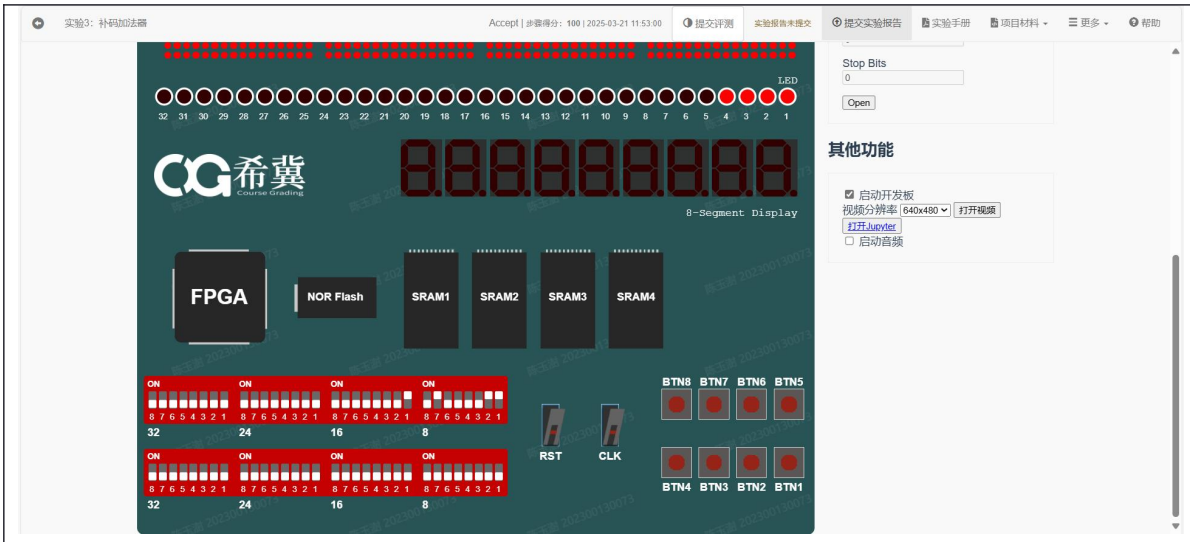
首先！输入为 011010101 时，即 K=0（加运算），A=1101（十进制为 13），B=0101（十进制为 5），13+5=18=10010，此时 C4=1（LED5 亮灯），SUM1=1（LED2 亮灯）。



下面是输入为 001110101 时，即 K=0（加运算），A=0111（十进制为 7），B=0101（十进制为 5），7+5=12=01100，此时 C4=0（LED5 不亮灯），SUM3=SUM2=1（LED4/3 亮灯）。



还有当输入为 101000011 时，即 K=1（减运算），B=0100（十进制为 4），A=0011（十进制为 3），3-4=-1=01111，此时 C4=0（LED5 不亮灯），SUM3/2/1/0=1（LED4/3/2/1 亮灯）。



选做：利用四位补码加法运算的结果实现两个 8 位二进制数 A（a7a6a5a4a3a2a1a0）和 B(b7b6b5b4b3b2b1b0)的逻辑运算并生成元件符号。使平台工作于模式 5，当按键开关不足时，可使用平台上红色的拨码开关。

此时与必做的不同在于，实现补码运算的两个数都变成了 8 位，因此需要对元件 adapt_in 和 adapt_out 进行改装，使其也满足 8 位数。

方法是点击元件，右键“Go to Source”，对代码进行修改。修改完成后，点击“Refresh Module”刷新元件即可。

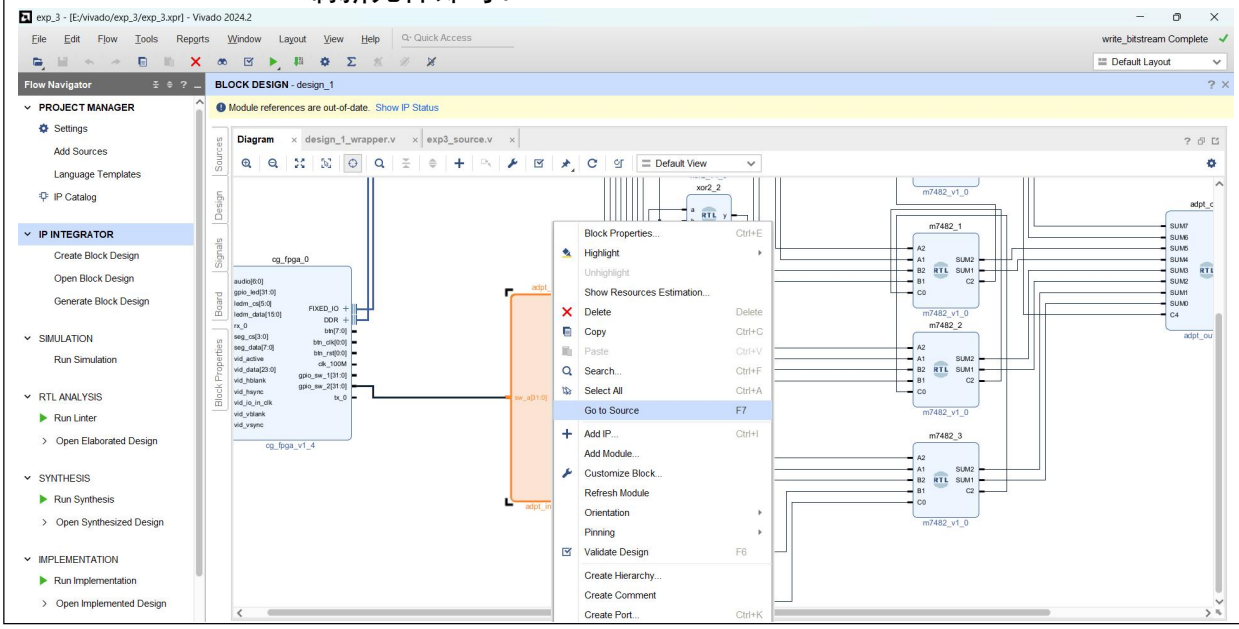
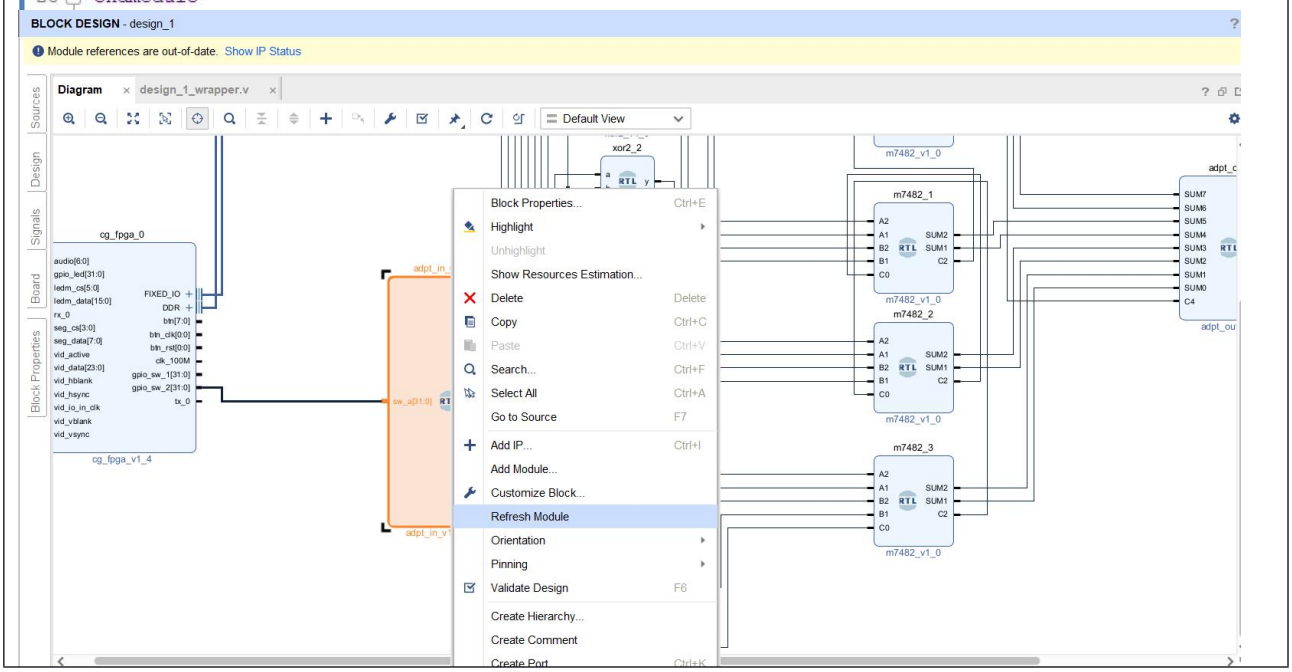
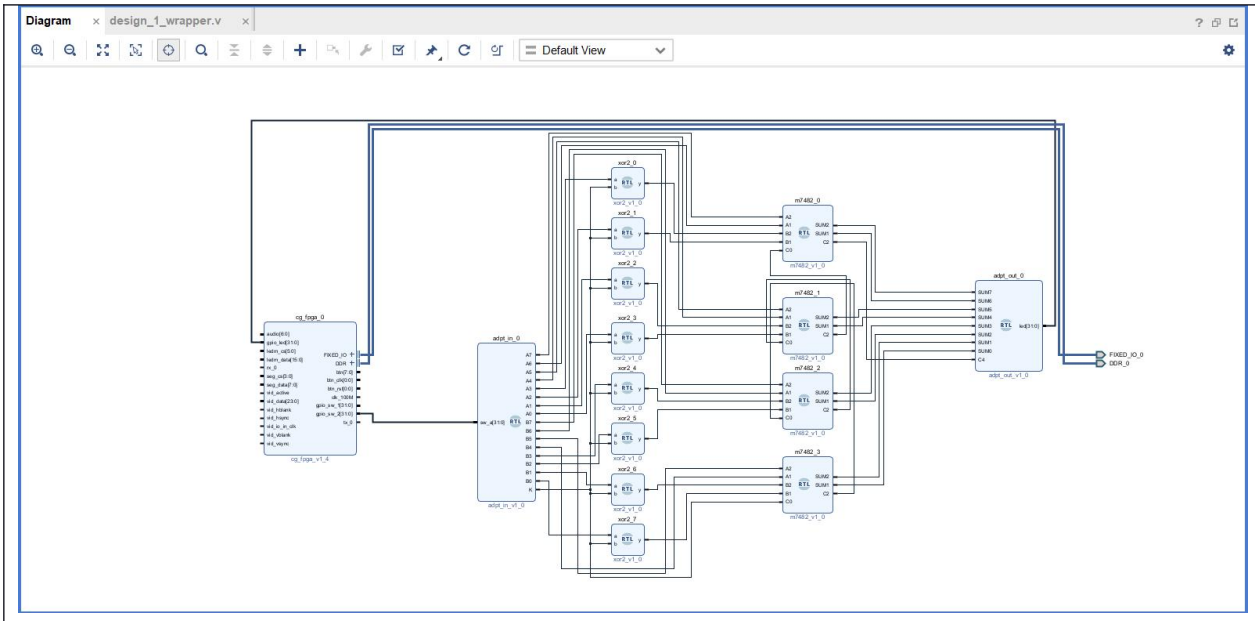


Diagram | design_1_wrapper.v | exp3_source.v

E:/vivado/exp_3/exp_3.srcs/sources_1/imports/Edge_Download/exp3_source.v

```
1  module adpt_in
2  (
3      input  [31:0]   sw_a,
4      output          A7,
5      output          A6,
6      output          A5,
7      output          A4,
8      output          A3,
9      output          A2,
10     output          A1,
11     output          A0,
12     output          B7,
13     output          B6,
14     output          B5,
15     output          B4,
16     output          B3,
17     output          B2,
18     output          B1,
19     output          B0,
20     output          K
21 );
22
23 assign {K, B7, B6, B5, B4, B3, B2, B1, B0, A7, A6, A5, A4,A3, A2, A1, A0} = ~sw_a[16:0];
24
25 endmodule
```

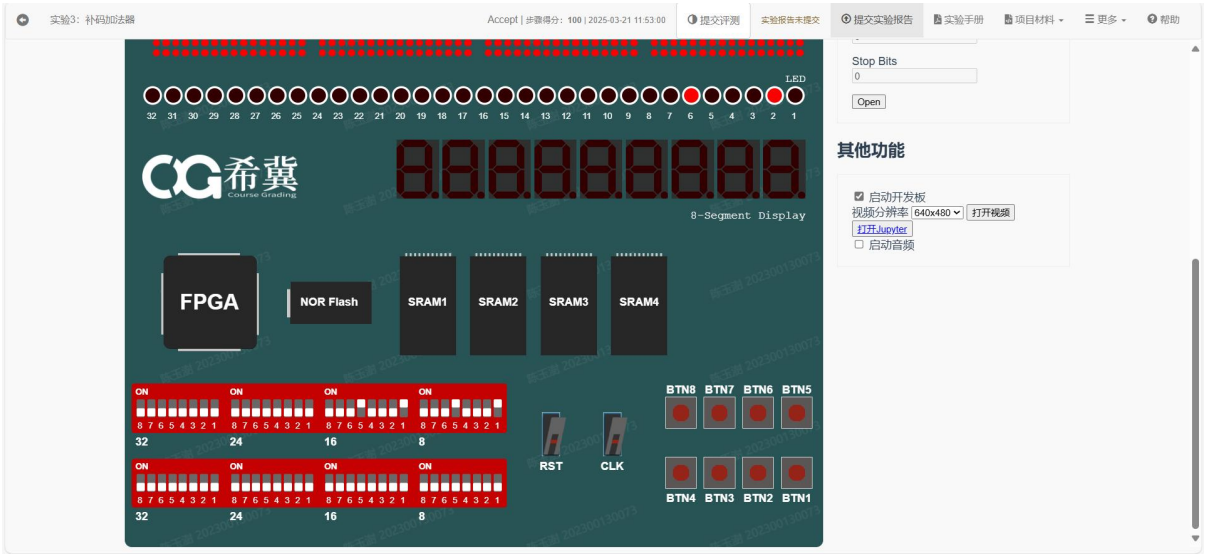




利用输入开关及发光二极管 LD 测试逻辑运算部件的功能并记录测试结果。

K 管脚绑定在上排第三组拨码开关 1，a7-a4、a3-a0 管脚分别绑定在上排拨码开关第一、二组的 4-1 上，b7-b4、b3-b0 管脚分别绑定在上排拨码开关第一、二组的 8-5 上，SUM7-SUM0 管脚绑定在 LED8-1 上，C4 管脚绑定在 LED9 上。

当输入为 00001000100010001 时，A=00010001，B=00010001，K=0（加运算），得到的结果 SUM=00100010，即 LED6 和 LED2 亮，其余灯不亮。



结论分析与体会：

这次实验使我提高了对于 Vivado 软件的应用熟练度，对全加器和异或门有了具体的实际应用，对补码加法器电路有了更加深入、深刻的认识和掌握。同时，选做实验锻炼了我举一反三的能力，通过更改元件代码以实现调整接口等操作，可以实现自由选择位数进行补码加减运算，十分的便捷、高效。

就元件代码修改后，电路图中迟迟未更新问题处理的：

代码修改后，电路图中的元件一般不会自动刷新其格式，需要手动右键元件，选择“Refresh Module”，或者点击“Run”按钮编译运行，才可以刷新电路图中元件的样式。