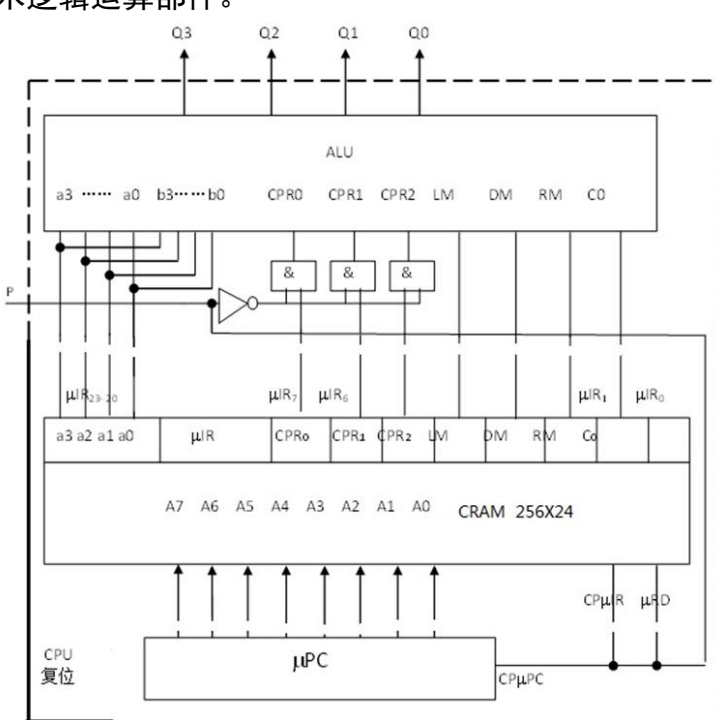
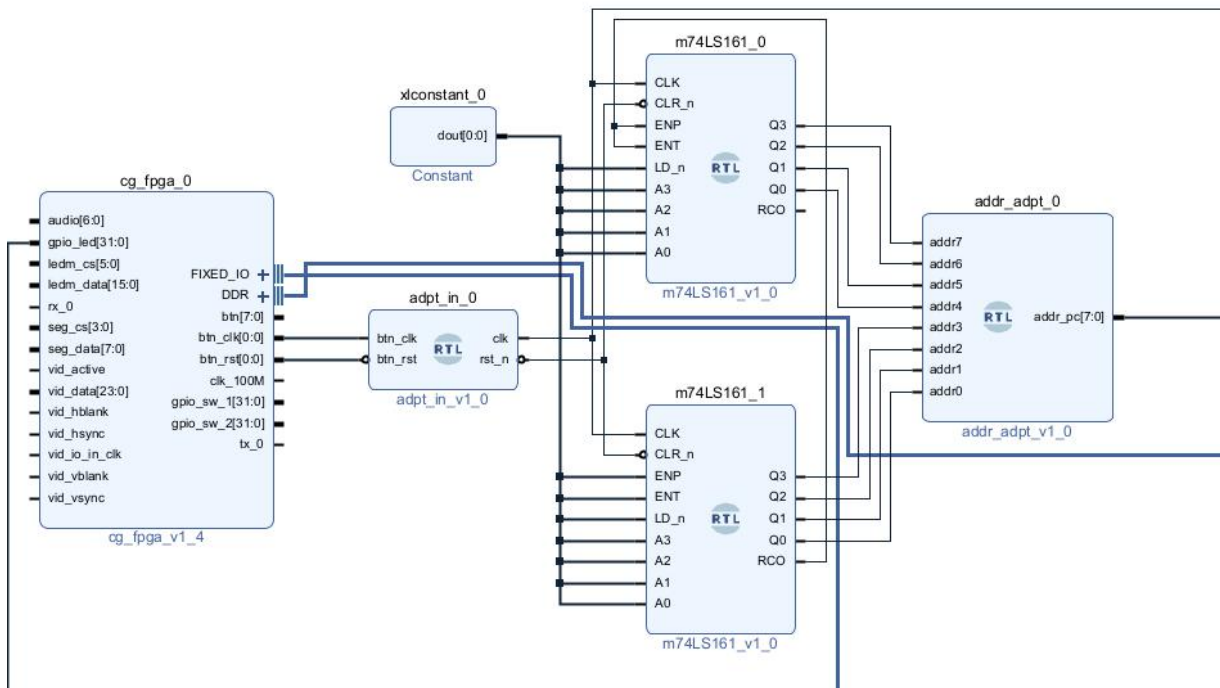


计算机组成与设计

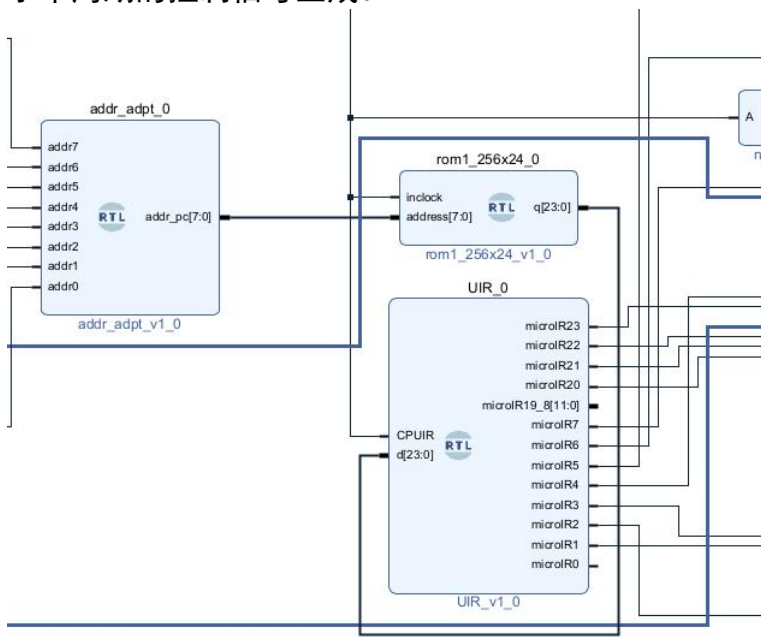
课程实验报告

学号：202300130073	姓名：陈玉澍	班级：计科 23 级 1 班
实验题目： 实验 14 综合实验		
实验学时：2	实验日期：2025. 5. 20	
实验目的： 学会并且掌握 CPU 综合实验电路，包括运算器电路和控制器电路。参照 CPU 综合实验结构框图。		
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、实验说明 运算器由三个寄存器 R0、R1、R2、移位器、加法器等构成，并组装在一起构成 ALU 算术逻辑运算部件。		
<div></div>		
二、实验电路设计思路 1、首先设计微程序计数器 uPC，在微指令控制器中用来逐条读取微指令。微程序计数器（uPC）用于逐条读取微指令 ROM（256×24）。为实现对 256 条指令的寻址，设计中采用		

两个 4 位同步二进制计数器 74LS161 进行级联，构成一个 8 位计数器 ( $2^8=256$ )。级联的方法是：低位计数器 (LS161\_0) 的 ENP/ENT 和 CLK 配合工作，使在每个时钟周期自增 1，将进位 RCO 连接到高位计数器的 ENT，ENP 始终保持高电平，使只有当低位溢出时，高位才+1，实现完整 8 位加法。将产生的两个 4 位数通过 addr\_adapt\_0 组件拼接转换为一个完整的 8 位地址。

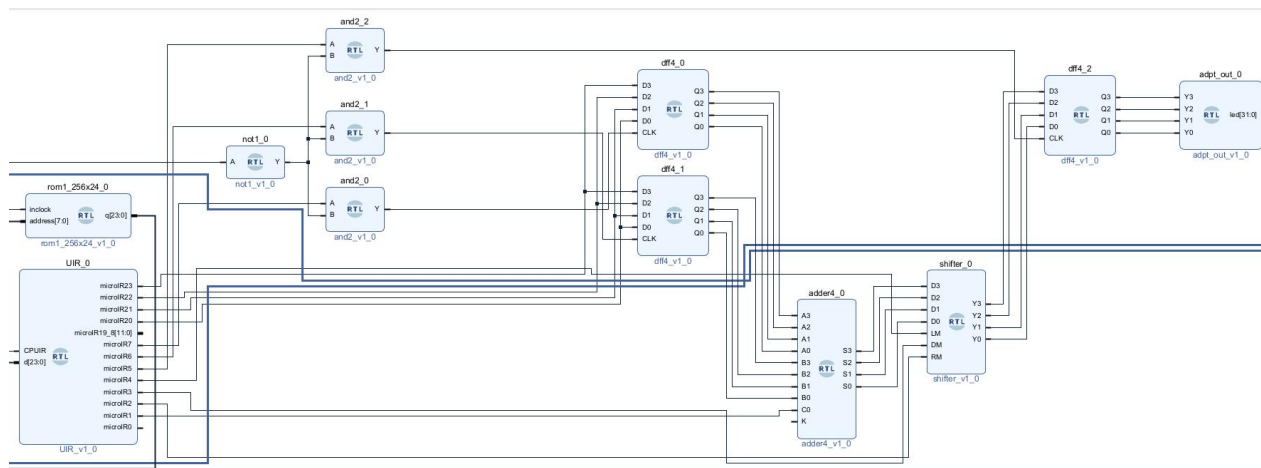


2、随后，微程序计数器 (uPC) 输出的 8 位地址被送入微指令存储器 ( $256 \times 24$  ROM)，从中读取对应地址处的微指令 (共 24 位)，并将该微指令加载到微指令寄存器 (uIR) 中，用于本周期的控制信号生成。



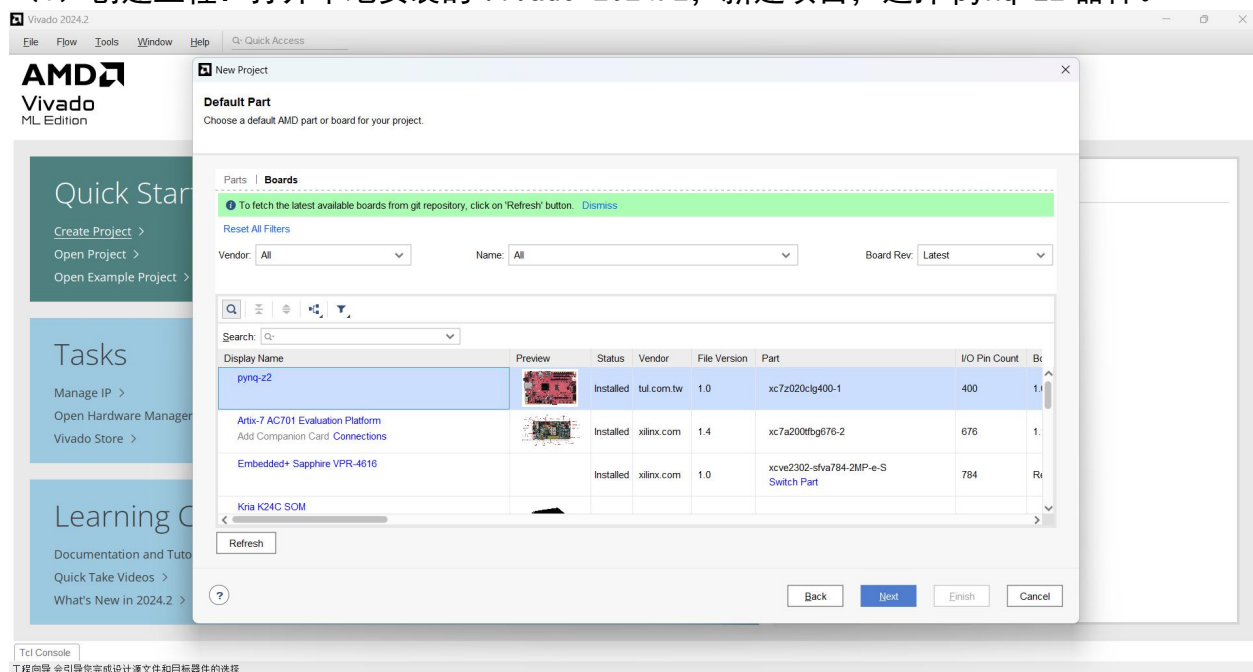
3、uIR 将得到的 24 位微指令拆分，R0 控制微程序计数器 uPC 是否复位 (如重启微程序计数器)，R1 控制 ALU 的进位输入 C0，R2 输出 RM 信号 (右移控制位，用于控制移位器是否右移)，R3 输出 DM 信号 (移位方向控制位，与 RM/LM 联合决定移位方向)，R4 输出 LM 信号

（左移控制位，用于控制移位器是否左移），R5、R6、R7 分别作为 CPR0、CPR1、CPR2，用于选择参与 ALU 运算的寄存器，R20-R23 输出立即数数据，根据 CPR0-CPR2 的控制选择，选择寄存器 diff\_0 或 diff\_1 作为 ALU 运算的输入源。接着 adder 加法器分别读入 diff\_0 和 diff\_1 寄存器中的 4 位数，将它们和来自低位的进位 C0 一同计算，运算结果输出至移位器 shifter，根据 LM/DM/RM 对 ALU 输出的数据进行处理，然后将结果送至 diff\_2 寄存器中，最后通过 adapt\_out\_0 模块输出。



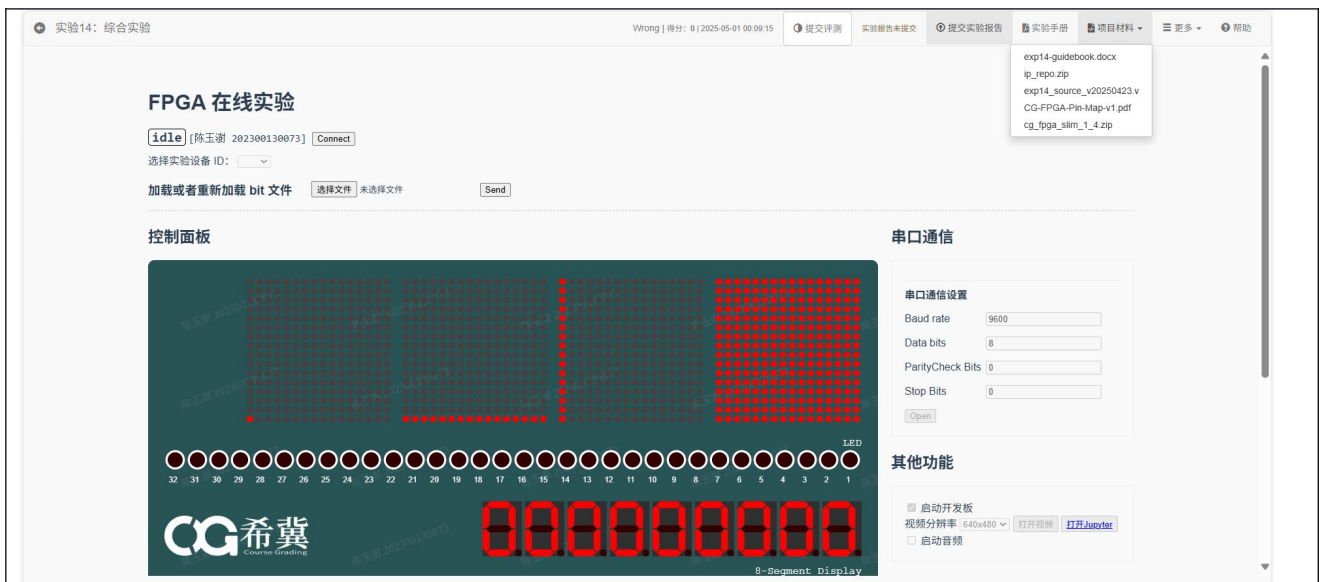
## 实验步骤：

（1）创建工程：打开本地安装的 Vivado 2024.2，新建项目，选择 pynq-z2 器件。

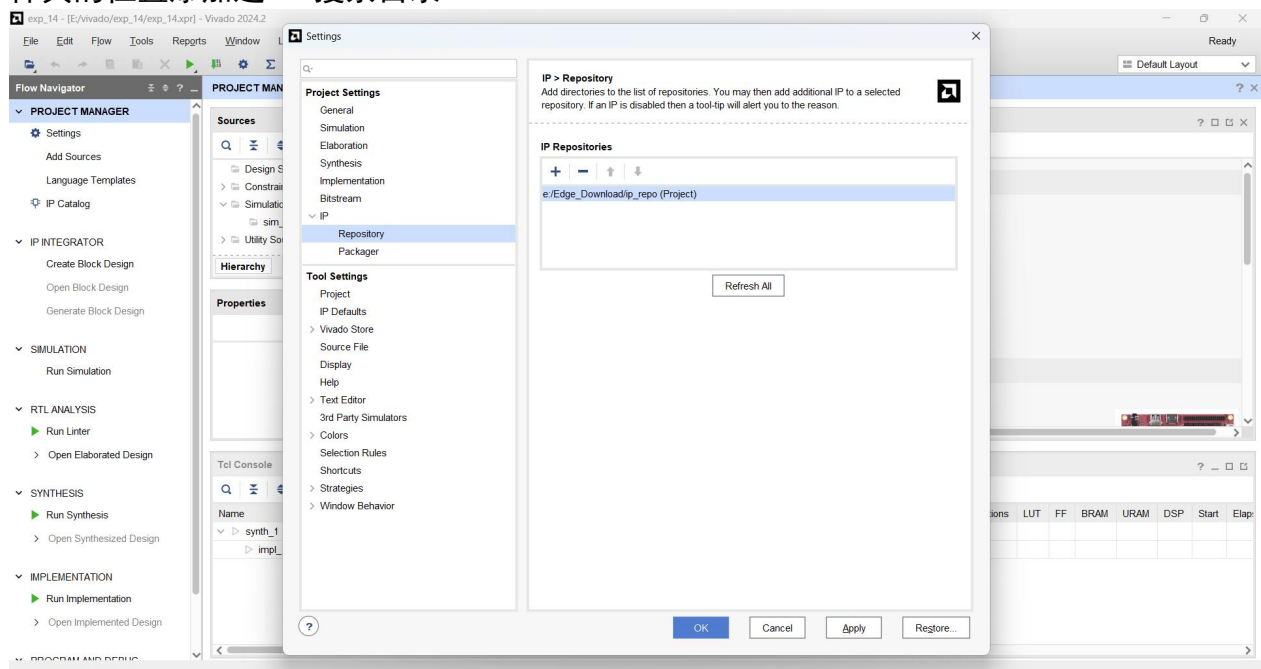


Tcl Console  
工程向导 会引导您完成设计源文件和目标器件的选择

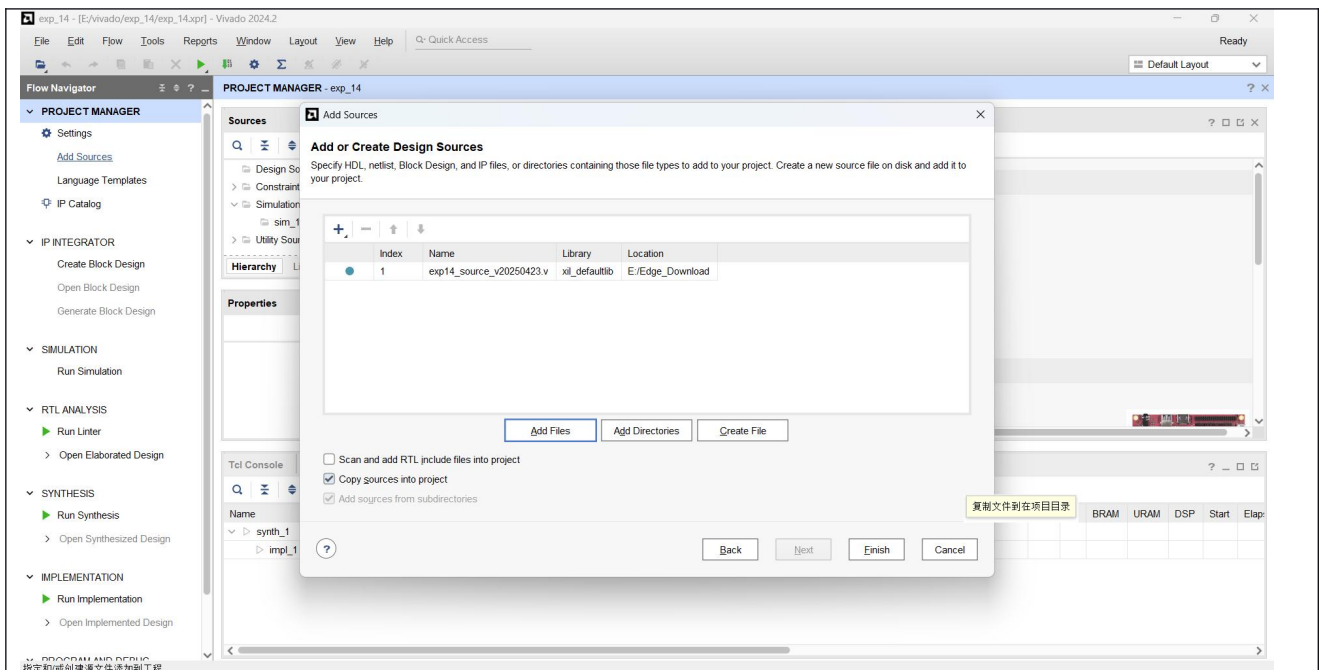
（2）添加实验环境：进入 FPGA 在线实验环境，点击右上角项目材料下载实验源代码和希冀 ip 核到本地并解压。



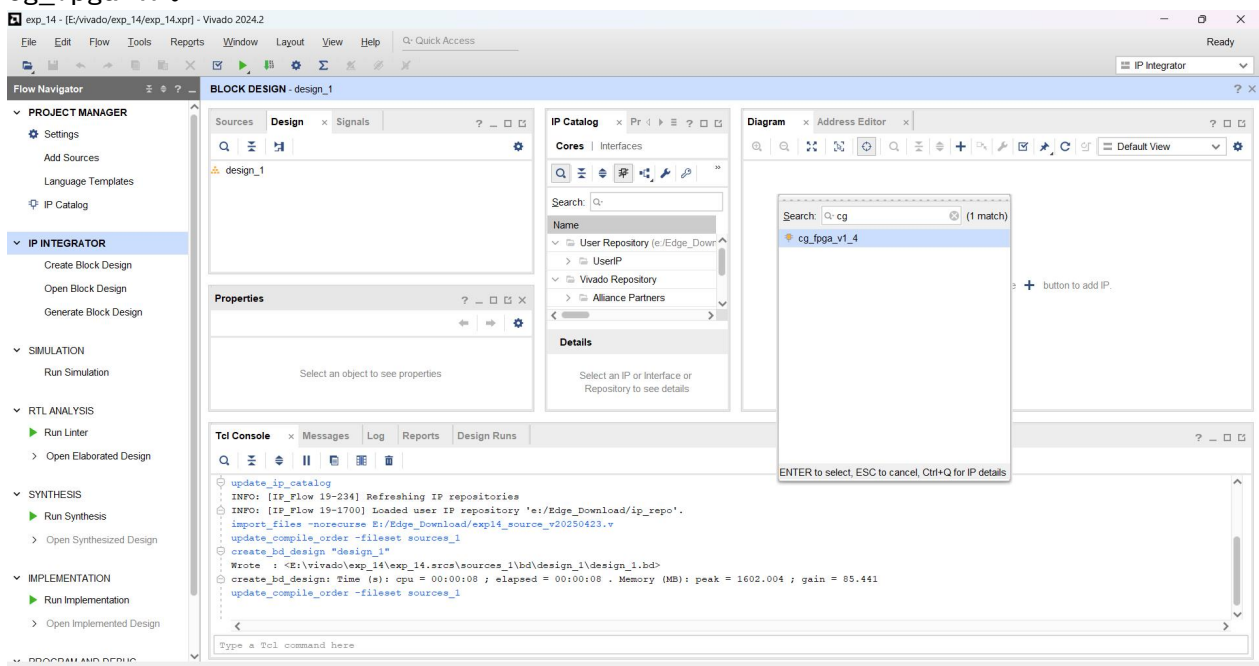
(3) 在 Vivado 项目中，点击 Settings→IP→Repository，将上一步解压后的 ip\_repo 文件夹的位置添加进 IP 搜索目录。



(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。

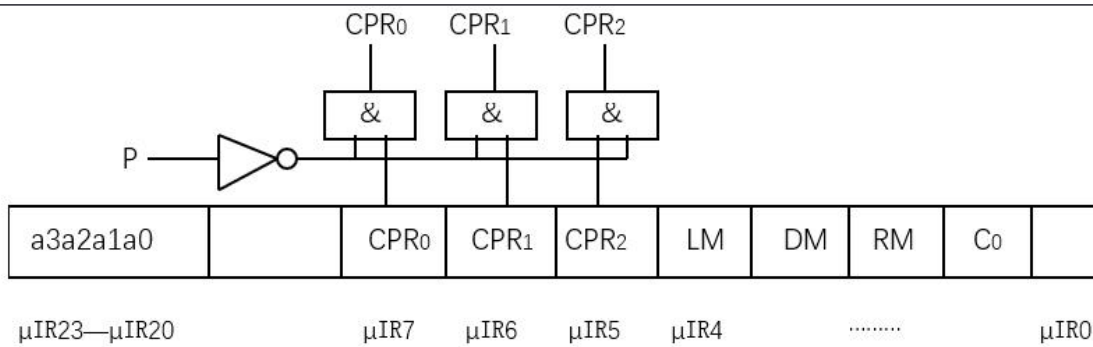


(5) 点击 Create Block Design 创建一个新的顶层设计，随后点击添加 IP 核按钮，添加 cg\_fpga IP。



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

- 1) 搭建 ALU 模块、 $\mu$ PC 模块及门电路按 CPU 综合实验结构框图完成连线
- 2) ALU 的输入数据 a3-a0 依次锁定在  $\mu$ IR23- $\mu$ IR420 上，CPR0、CPR1、CPR2 依次锁定在 mIR7-mIR5 上，LM、DM、RM、C0 依次锁定在  $\mu$ IR4- $\mu$ IR1 上。
- 3) 适配、下载
- 4) 编制微程序：- 将微指令格式分为两部分：前面部分  $\mu$ IR23~ $\mu$ IR20 可设置数据，后面部分  $\mu$ IR7~ $\mu$ IR0 可确定微命令，例：需要 CPR0 脉冲，该位为 1，否则为 0；备用位填 0。



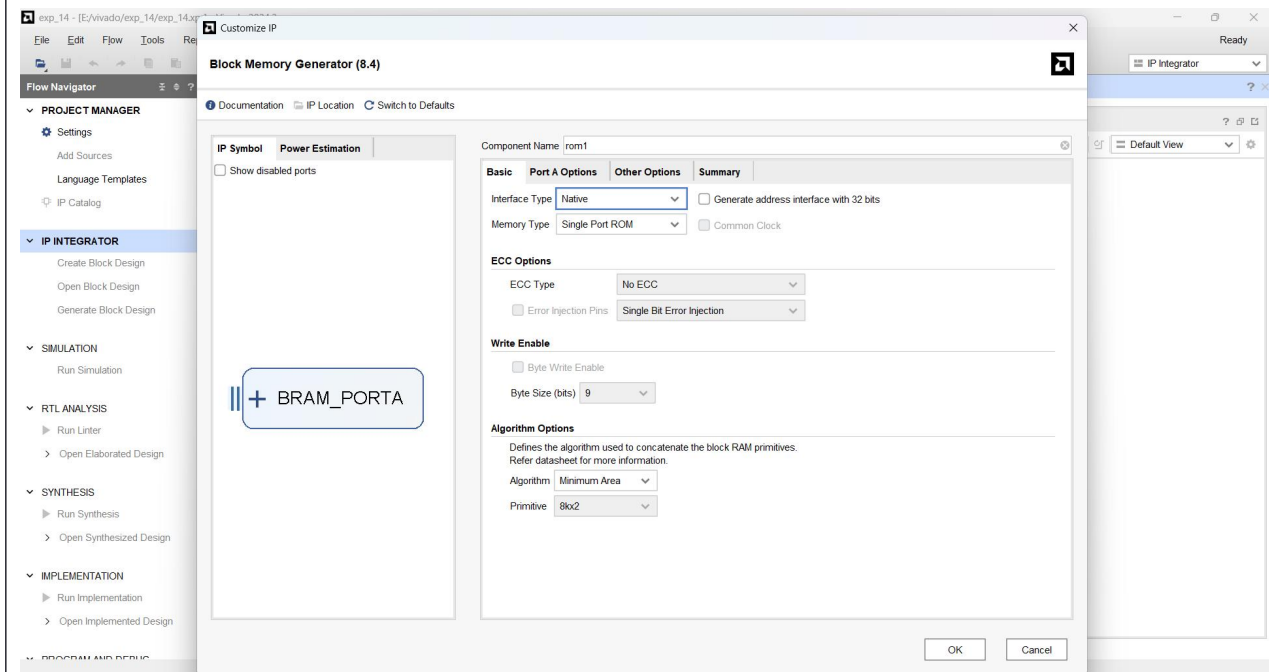
## 5) 功能检查

此外，对 rom 的配置如下：

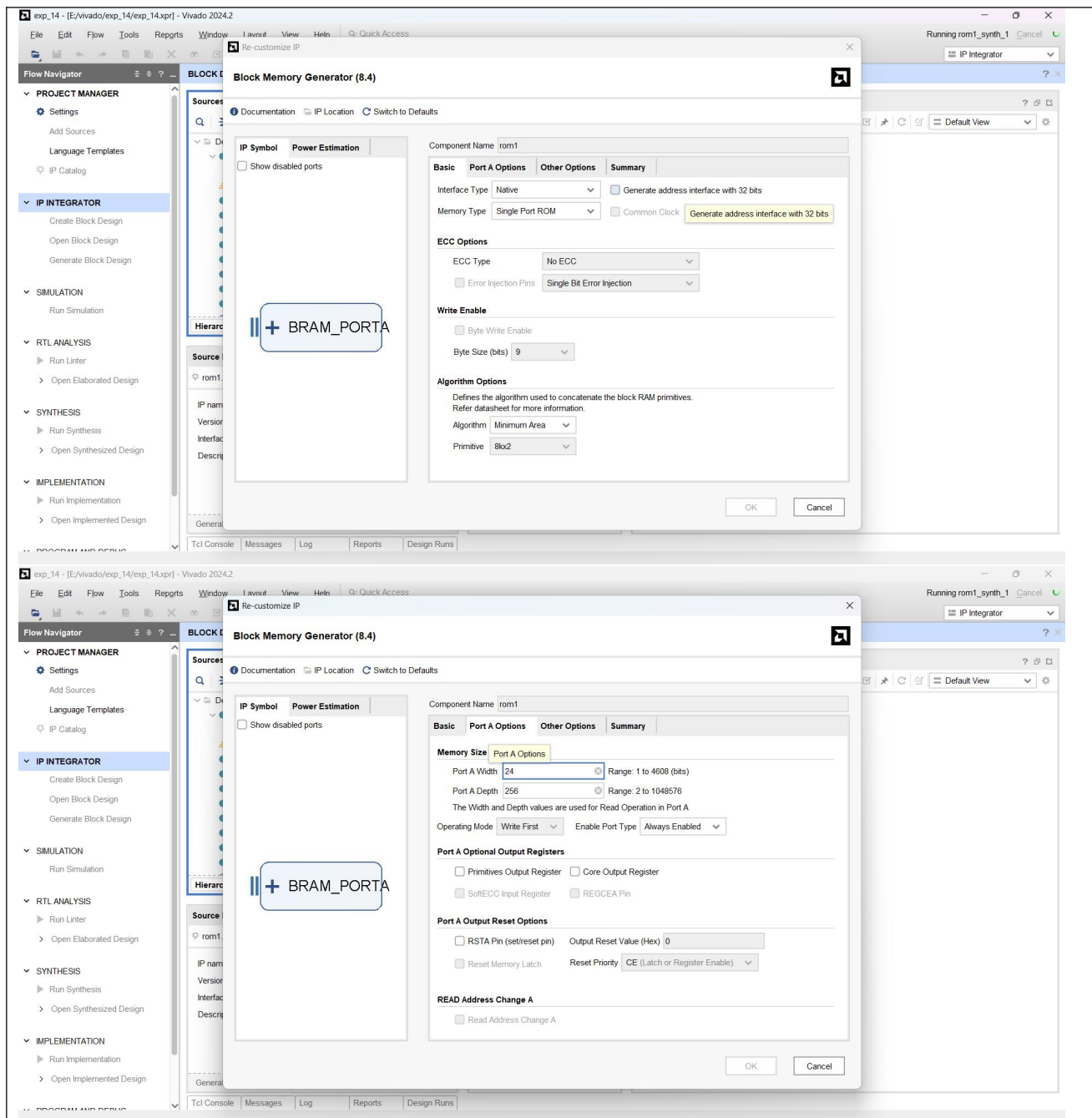
首先，找到 IP Catalog 中的 Block Memory Generator，按照如下方法定制单口 ROM，并命名为 rom1。

按照框图中的数据配置规格为 256\*24 的 ROM，同时将初始化数据加载到 Xilinx FPGA 的内置存储器中。即手动添加 .coe 文件，内容为：

```
600080, 800040, 000028, 03FFFE, 07FFFC, 0FFFC, 1FFE38, 3FF000, 3FC001, 7F8000, 7F000
0, FF0000, FE0000, FE0000, FE0000, FC0000, FC0000, FE0000, FE0000, FE0000, FE0000, 7F0
000, 7F8000, 3FC000, 3FE001, 1FF800, 0FFFF8, 0FFFF8, 07FFFC, 01FFFC, 00FFFC, 003FFC, 0
003E0
00000111111111111111111100
```







The image displays two screenshots from the Vivado 2024.2 IDE. The top screenshot shows the 'Block Memory Generator (8.4)' configuration window. The 'IP Symbol' tab is active, showing the 'rom1' component. A 'COE File Editor - rom1.coe' window is open, displaying the following content:

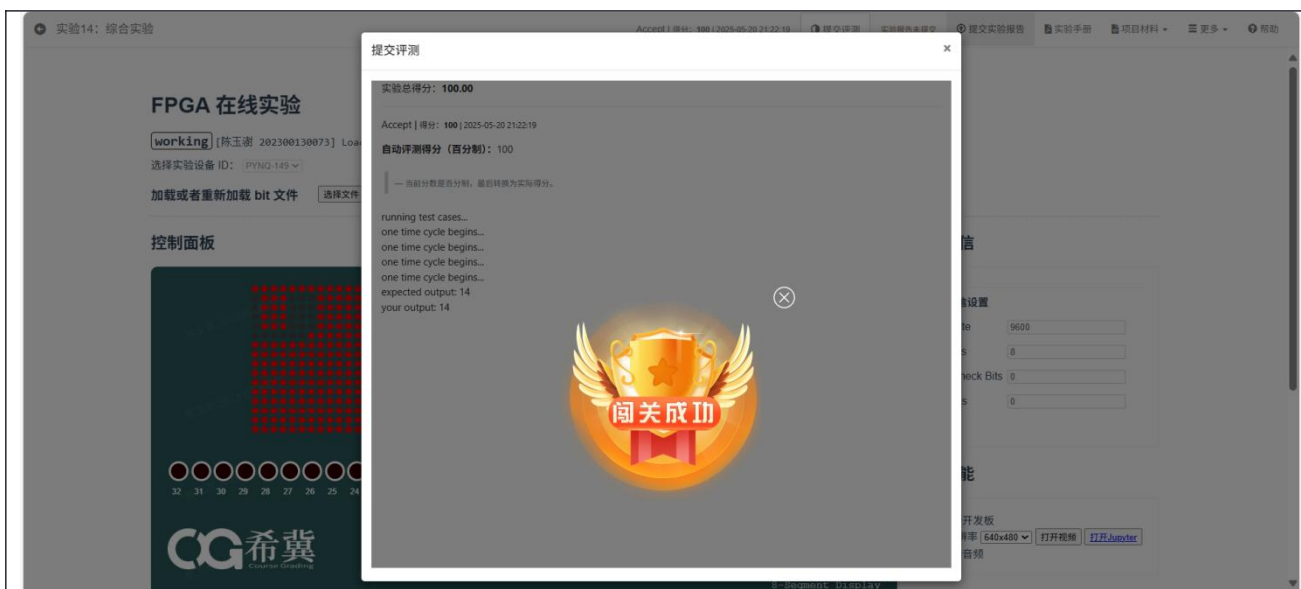
Key	Value
memory_initialization_radix	16
memory_initialization_vector	600080 800040 000028 03FFFE 07FFF...

The bottom screenshot shows the 'Diagram' view of the design. It illustrates the RTL implementation of the Block Memory Generator, featuring a complex network of multiplexers (MUX), adders, and memory blocks (ROM) connected to the system bus. The design is labeled 'exp\_14 - [E:/vivado/exp\_14/exp\_14.xpr] - Vivado 2024.2'.

(7) 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。

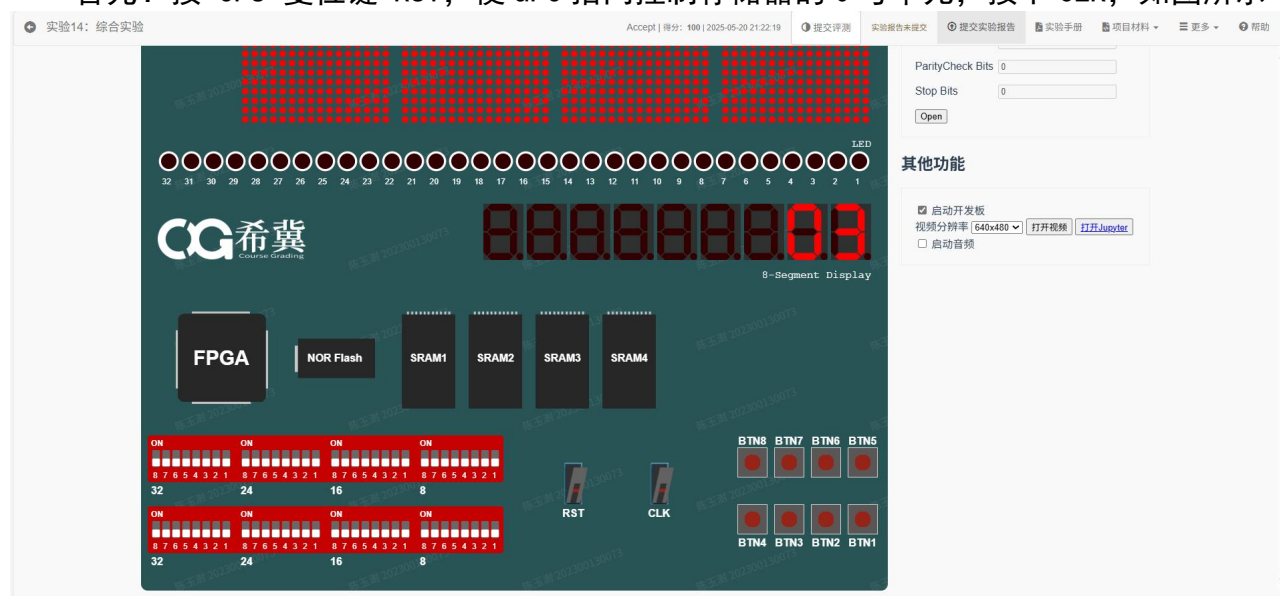




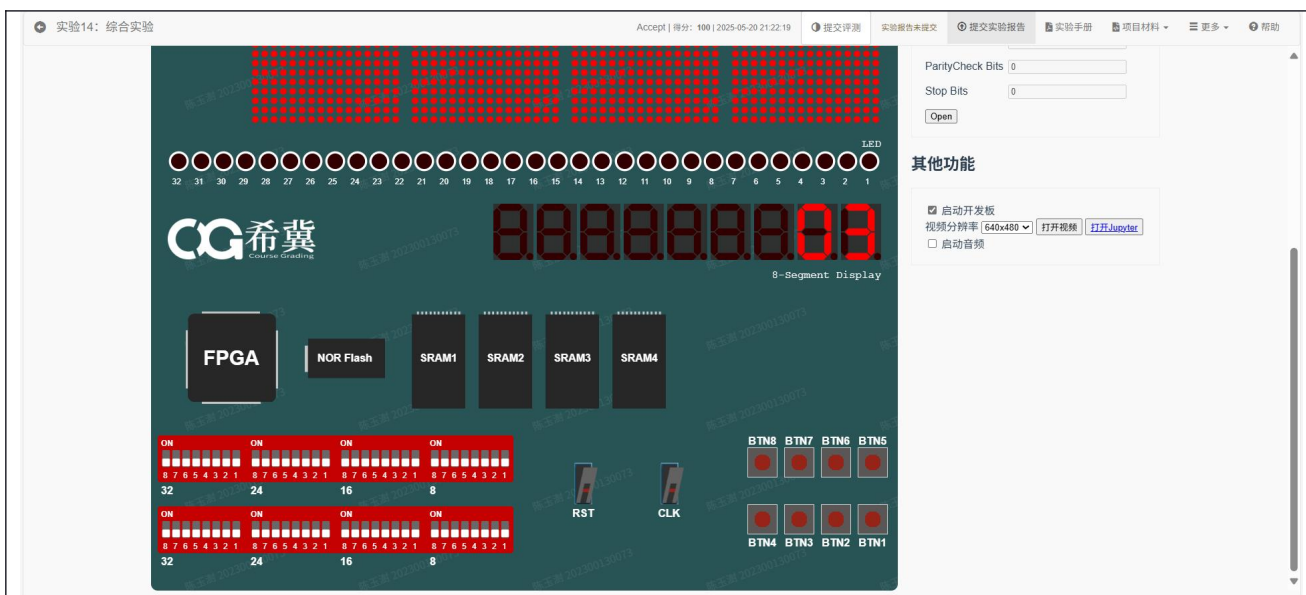


clk 信号的管脚绑定在 btn\_clk 上, rst\_n 信号的管脚绑定在上排拨码开关 btn\_rst 上, Y3, Y2, Y1, Y0 信号的管脚绑定在 LED 4-1 上。

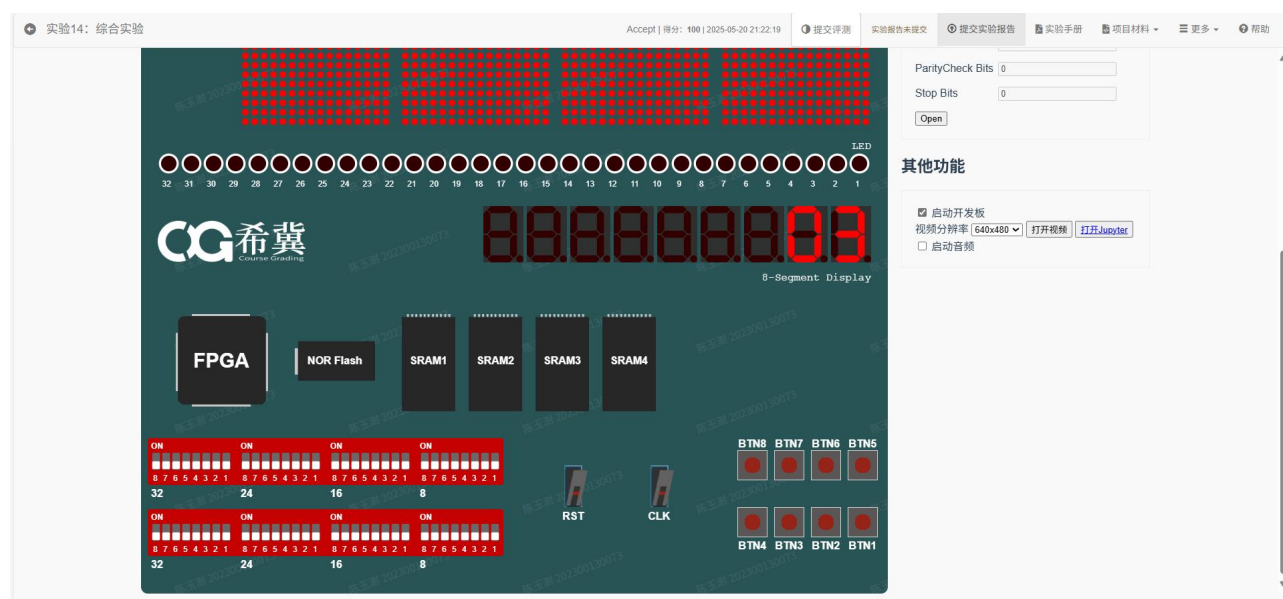
首先！按 CPU 复位键 RST，使 uPC 指向控制存储器的 0 号单元，按下 CLK，如图所示



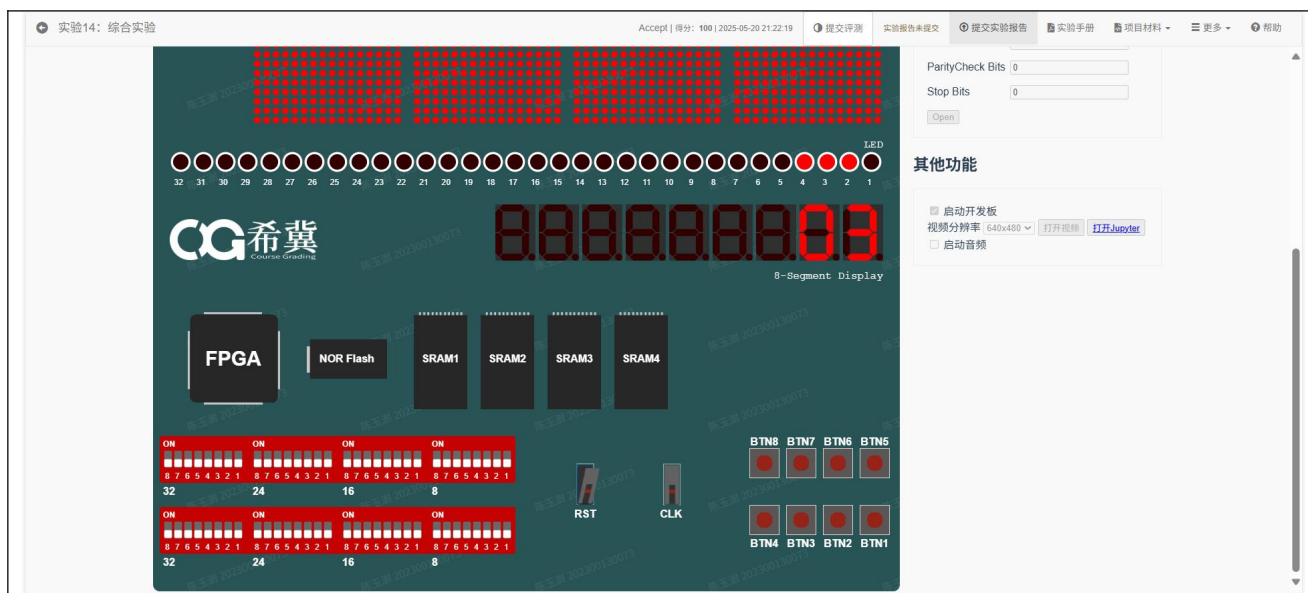
再次按下 CLK，此时第一条微指令（011000000000000010000000）执行完毕，R7、R21、R22 输出高电位，R0 执行写入操作，0110 存入寄存器 R0 中，不过由于 diff\_2 的时钟信号未发生改变，因此没有输出。



然后，再次按下 CLK，此时第二条微指令（10000000000000001000000）执行完毕，此时 R7、R23 输出高电位，R1 执行写入操作，1000 被存入寄存器 R1 中，不过由于 diff\_2 的时钟信号未发生改变，因此没有输出。



最后，再次按下 CLK，此时第三条微指令（00101000）执行完毕，此时 R3、R5 输出高电位，分别执行写直达和输出操作，将结果 1110（14）输出。



综上，经验证实验结果符合预期。

### 结论分析与体会：

这次的实验让我对于 CPU 综合实验电路有了一个深入的理解，我对 CPU 的基本结构和微程序控制原理有了更加深入的理解。实验中通过构建微程序计数器（uPC）、微指令寄存器（uIR）、ROM、ALU、移位器（shifter）等模块，清晰地展现了微指令驱动 CPU 工作的全过程。特别是通过查看各个控制信号（如 R0-R23）在电路中的功能分配，使我认识到微指令的位控制策略对于 CPU 指令执行流程的重要性。此外，在调试中分析寄存器数据流、控制逻辑以及组合逻辑的作用，也增强了我对数字电路系统级联设计的理解和动手能力。此次实验为后续深入学习计算机组成原理与 CPU 结构打下了坚实基础。

### 就加法器运算结果错误问题处理的：

- 1、问题：发现是进位信号未接入导致的 ALU 输出结果异常。
- 2、解决：重新梳理了 uIR 每个位的作用，将 C0 与正确的微指令位 R1 相连，成功解决了问题。