实验 4：加法器和 ALU 设计

姜帅 221220115

一、实验目的

1. 掌握先行进位加法器 CLA 和先行进位部件 CLU 的设计方法。
2. 掌握 32 位先行进位加法器及相关标志位的实现方法。
3. 掌握 ALU 的设计方法，根据指令要求实现 6 种操作的 ALU 器件。

二、实验环境

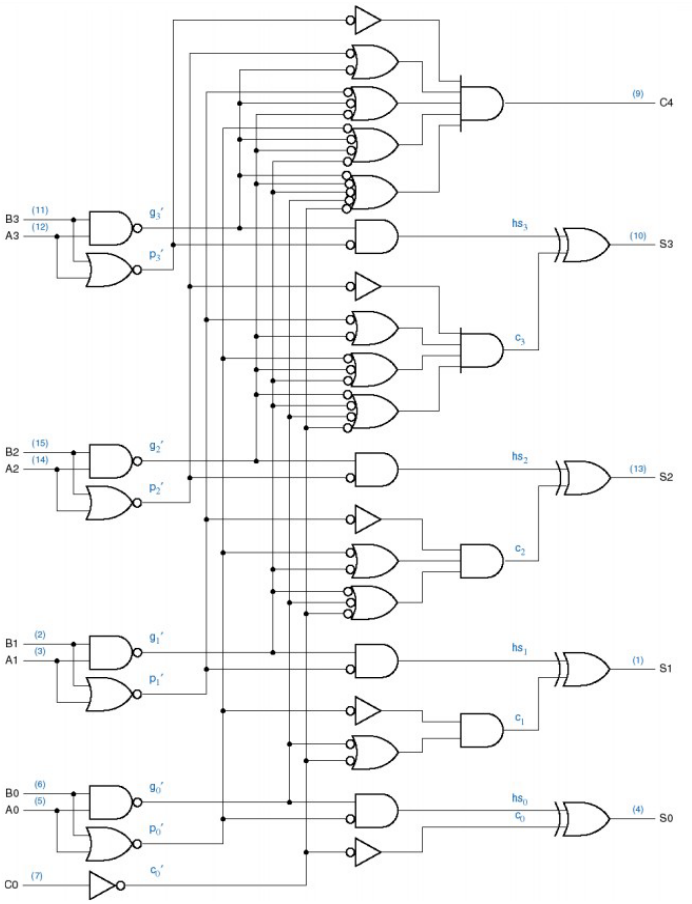
Logisim：https://github.com/Logisim-Ita/Logisim

三、实验内容

1. 4 位先行进位加法器 CLA 实验

根据给出的 4 位 CLA 电路原理图（参照其他原理图亦可），实现并验证 4 位先行进位加法器 CLA。

原理图：



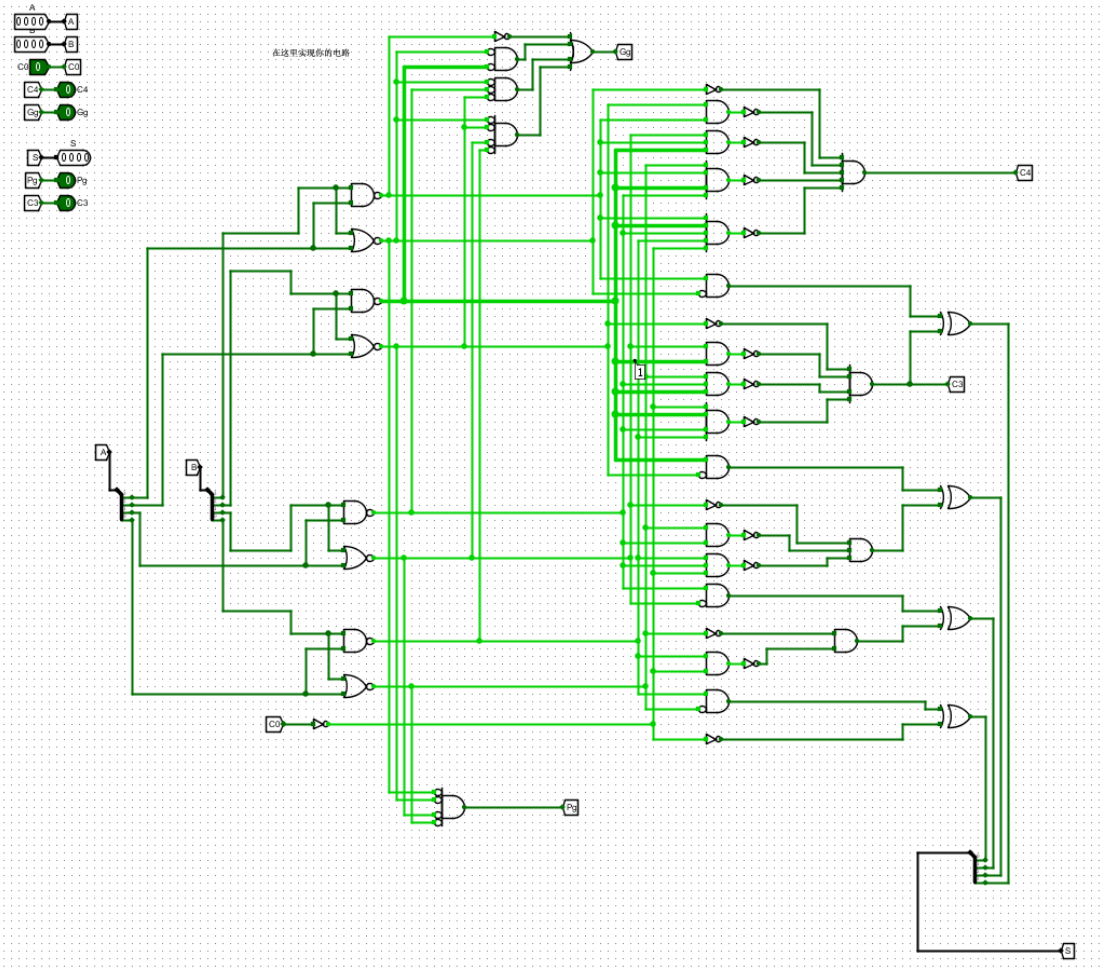
实验步骤如下：

1）创建子电路。在工程中添加一个名为“4 位先行进位部件 CLU”的子电路，并双击该子电路，在右侧工作区中构建相应电路。

2）设计子电路并进行功能测试。

检验：设置输入为 C0=0，A=1011，B=0111，则输出 S=0010=2，C4=1。

电路图：



2. 4 位先行进位部件 CLU 实验

根据以下 4 个并行进位 C*i* 的逻辑表达式，构建 4 位 CLU 电路，并进行功能验证。

$C_1 = G_0 + P_0C_0$

$C_2 = G_1 + P_2G_0 + P_1P_0C_0$

$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$

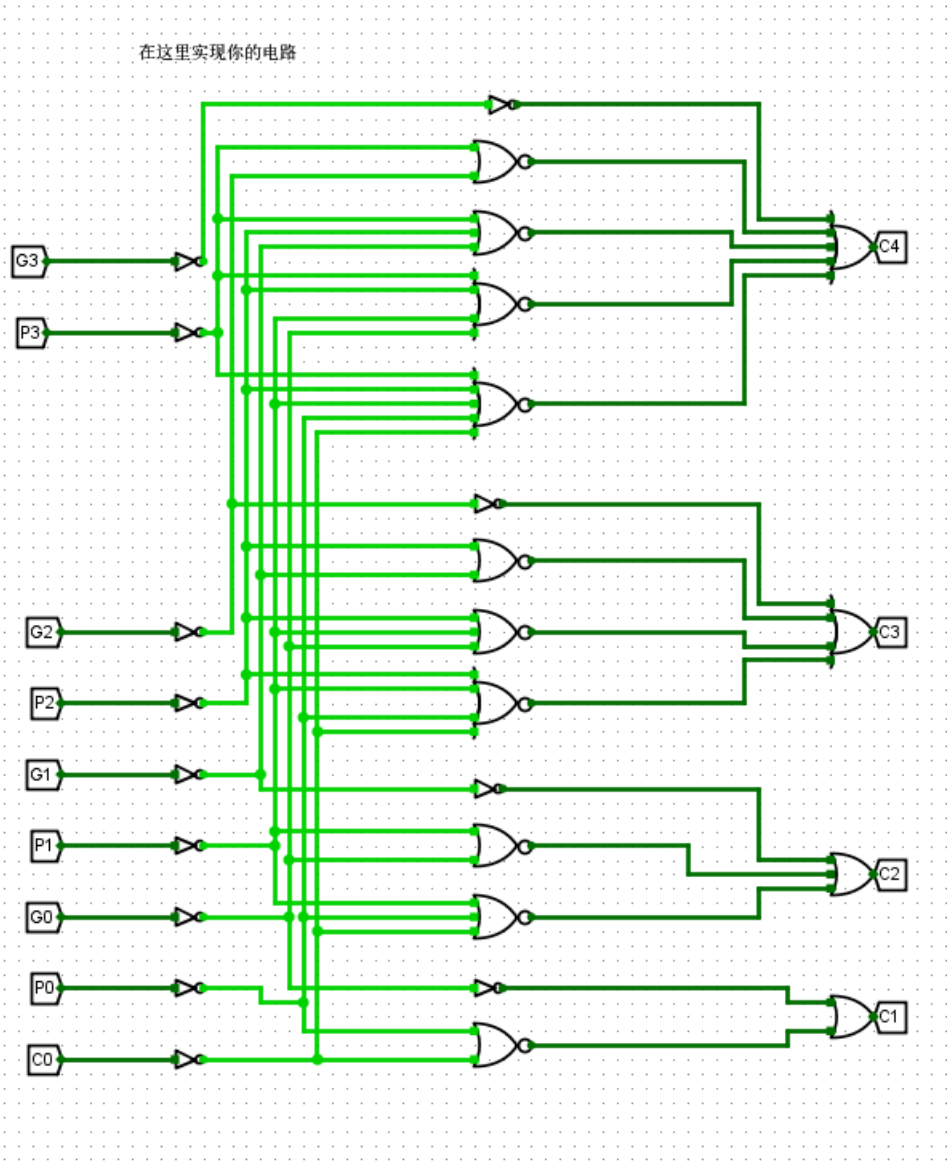
$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$

实验步骤如下：

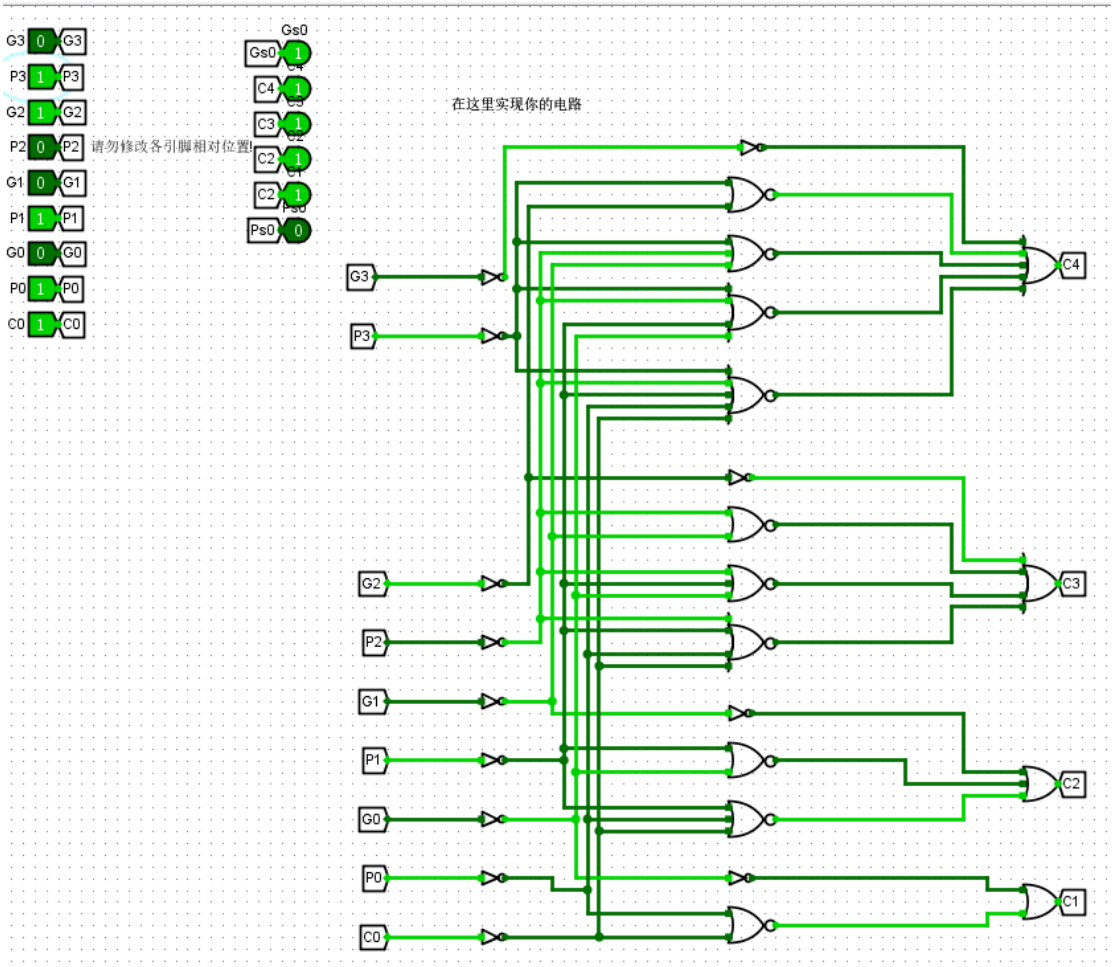
1）创建子电路。

2）设计子电路并进行功能测试。

检验：设置输入 P=1011，G=0100，C0=1，则输出 C=111


在这里实现你的电路
G3
P3
G2
P2
G1
P1
G0
P0
C0
C4
C3
C2
C1

仿真测试：



3. 16 位两级先行进位加法器实验

若将式(11-1)中进位 C_4 的逻辑表达式改写成为 $C_4=Gg+PgC_0$,则 Pg、Gg 分别表示 4 位加法器的组间进位传递和组间进位生成输出变量，其逻辑表达式分别如下：

$Pg= P_3P_2P_1P_0$

$Gg= G_3+P_3G_2+P_3P_2G_1+P_3P_2P_1G_0$

对于一个 16 位加法器，可以分成 4 组，每组用一个 4 位 CLA 实现，C_4、C_8、C_{12}、C_{16} 分别为每一组向前一组的进位，即组间进位。将 Pg、Gg 用于生成 4 个组间进位，则有如下逻辑表达式：

$C_4= Gg_0 + Pg_0C_0$

$C_8= Gg_1 + Pg_1C_4 = Gg_1 + Pg_1Gg_0 + Pg_1Pg_0C_0$

$C_{12}= Gg_2 + Pg_2C_8= Gg_2 + Pg_2Gg_1 + Pg_2Pg_1Gg_0 + Pg_2Pg_1Pg_0C_0$

$C_{16}= Gg_3 + Pg_3C_{12} = Gg_3 + Pg_3Gg_2 + Pg_3Pg_2Gg_1 + Pg_3Pg_2Pg_1Gg_0+ Pg_3Pg_2Pg_1Pg_0C_0$

16 位两级先行进位加法器原理图：

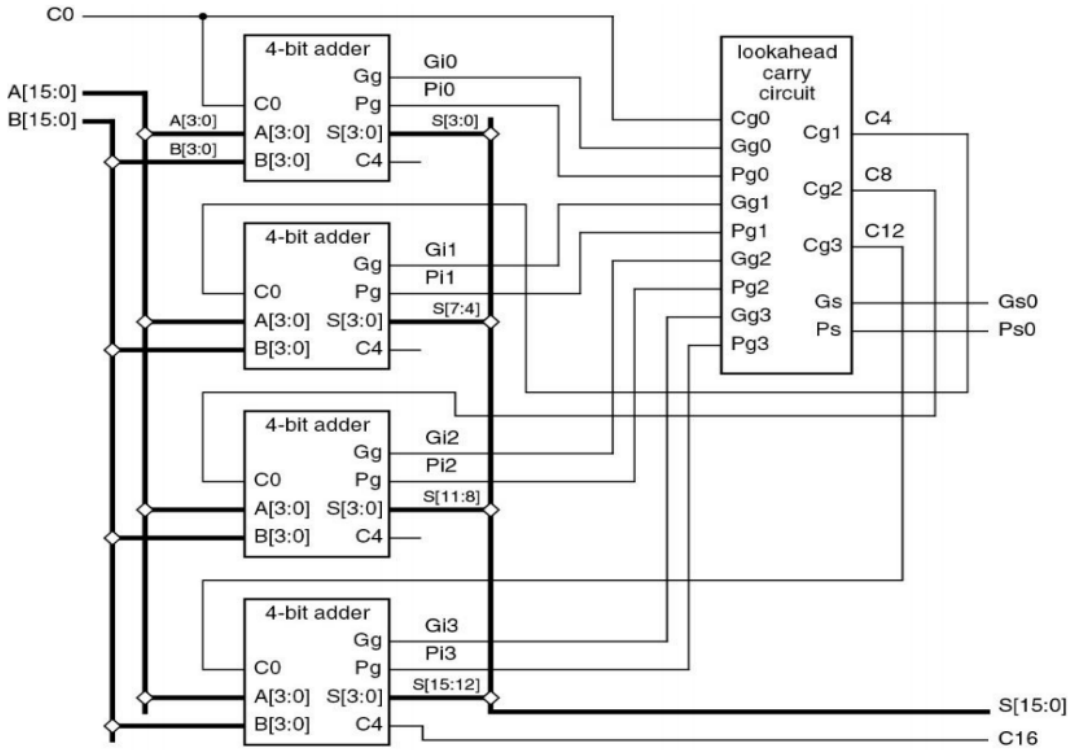


图 11.72　16 位两级先行进位加法器原理图

实验步骤如下。

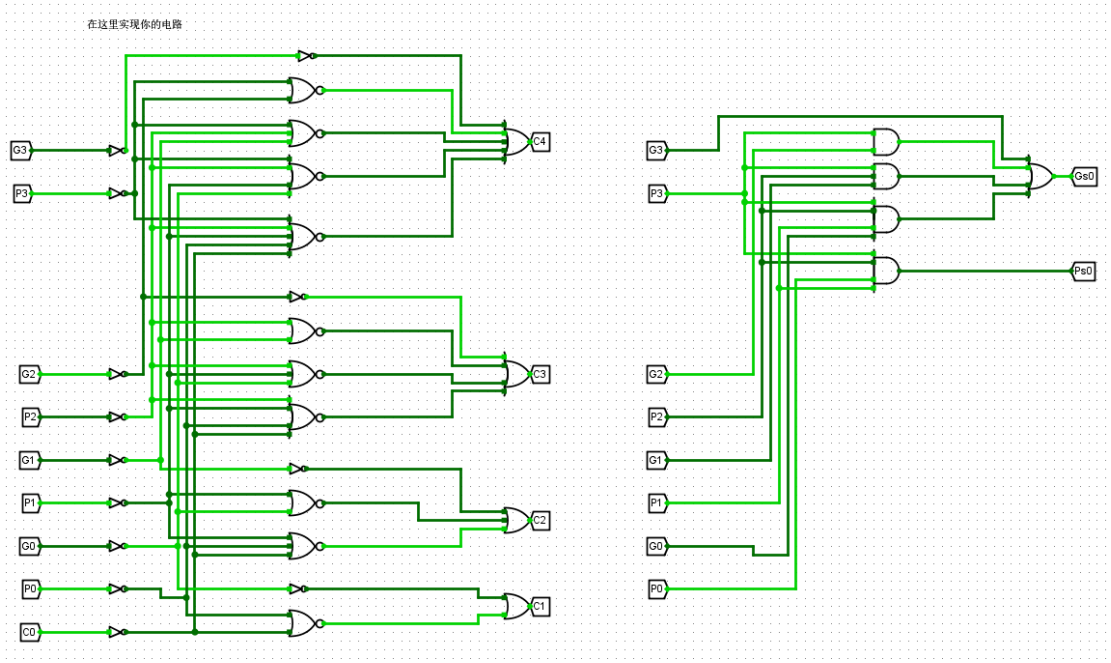
1）创建子电路。

2）设计子电路并进行功能测试。

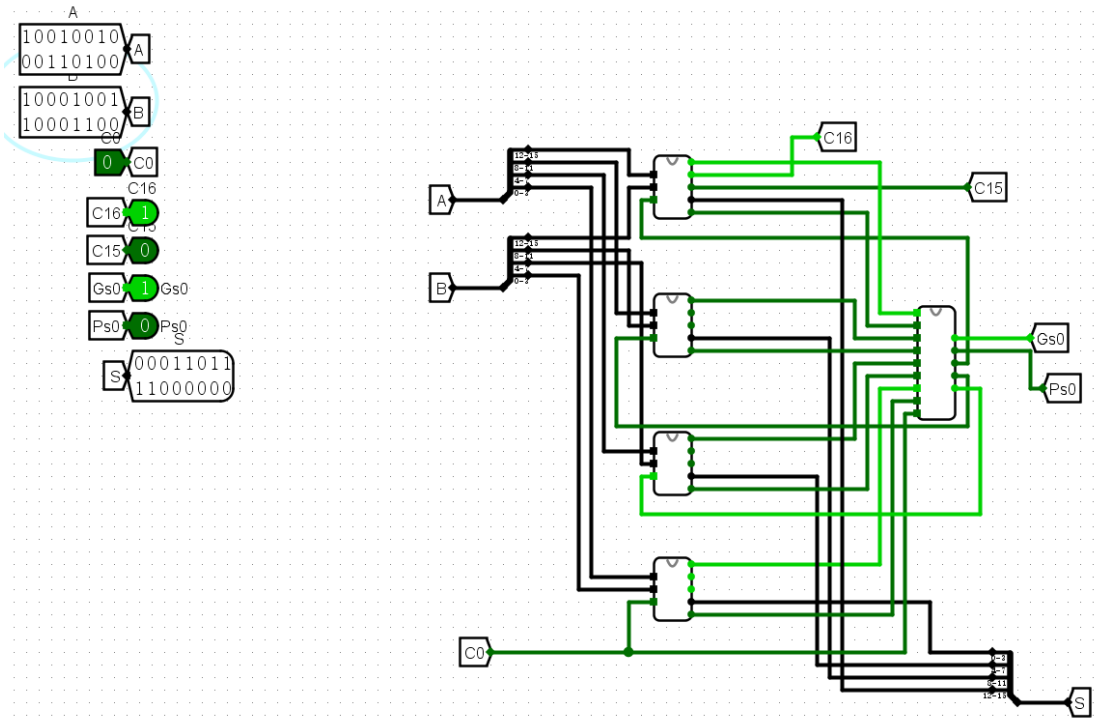
根据图 11.72 中的原理图可知，实现 16 位两级先行加法器需要包括以下部件：4 个带组间进位传递和组间进位生成输出变量 Pg、Gg 的 4 位 CLA、1 个 4 位组间 CLU、分线器、输入/输出引脚等。因此需要将上述第 1 个和第 2 个实验实现的 4 位 CLA、4 位 CLU 进行适当修改，以支持组间进位传递和组间进位生成。修改后的 4 位 CLA 电路如图 11.73 所示。为了方便地将其用于构建 16 位两级先行进位加法器，需要对其进行封装。考虑到在以后实现算术逻辑部件（ALU）时需要在加法器中生成溢出标志（OF）等各种标志位，在图 11.73 中增加了次高位进位 C_3 和最高位进位 C_4 两个输出端口，以便在后续的电路中可以方便地使用 C_3 和 C_4 生成溢出标志位 OF，因为 $OF= C_3 \oplus C_4$。

带 Pg，Gg 的 4 位 CLA 电路（在第一个实验完成）

4 位组间 CLU 电路图（右侧为 Pg，Gg）



3）检验：当输入 A=0x9234，B=0x89AC 时，则输出 S=0x1BE0，Ps0=0，Gs0=1，C16=1。这里，Ps0、Gs0 为 16 位加法器向高位组（每组 16 位时）的组间进位传递和组间进位生成输出变量，C16 为 16 位加法器向高位组（每组 16 位时）的组间进位输出变量



4. 32 位加法器构建实验

通过将两个 16 位两级先行进位加法器串行级联构建一个 32 位加法器，并根据给出的标志位生成电路，在 32 位加法器中生成 CF、SF、OF、ZF 标志位。

原理图：

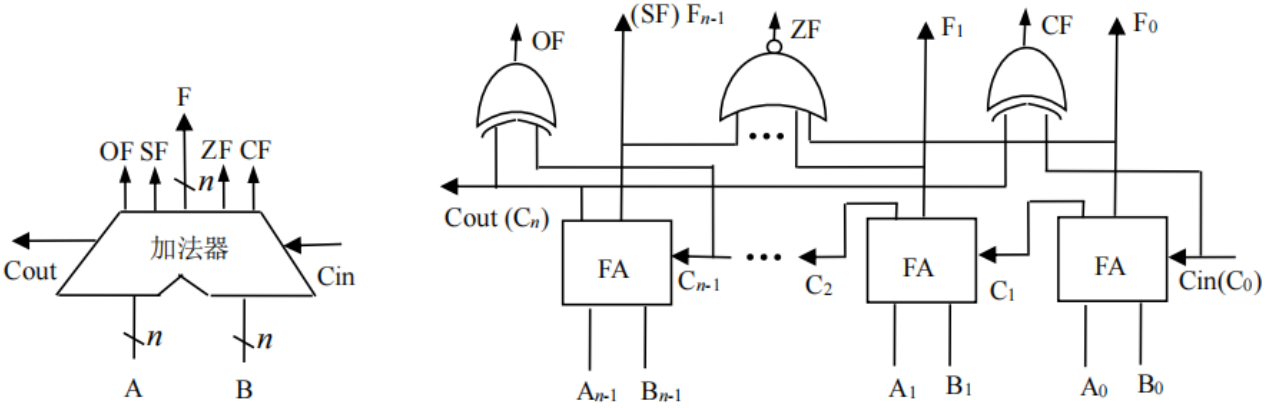


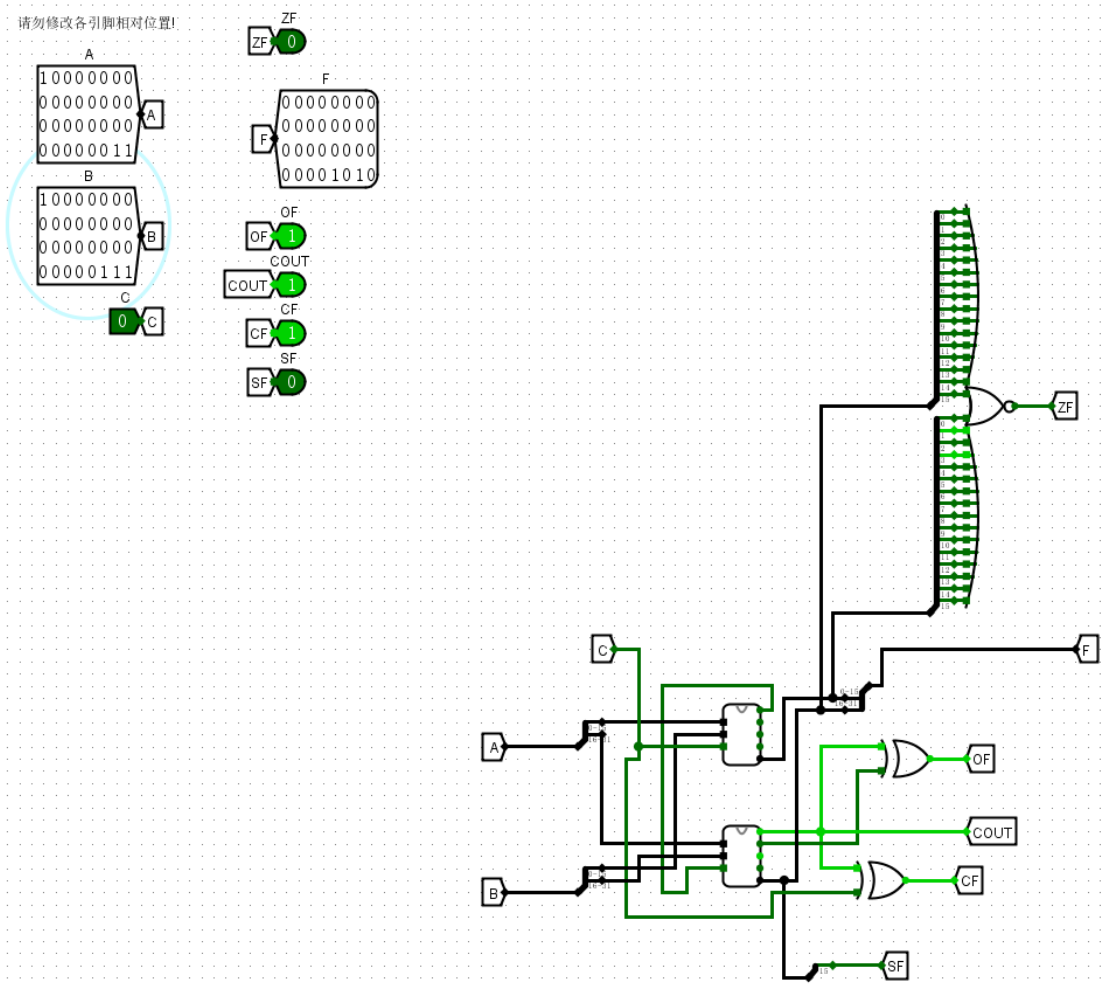
图 11.76 加法器标志位生成电路图

实验步骤如下。

1）创建子电路。

2）设计子电路并进行功能测试。

3）检验：输入 A 为 10000000000000000000000000000011B,B 为 10000000000000000000000000000111，cin 为 0，预期输出 OF=1，COUT=1，CF=1，s=00000000000000000000000000001010



5. ALU 设计实验

根据给出的电路原理图和 ALU 引脚定义，设计一个支持 9 条 RV32I 指令所包含的 6 种操作（add、or、slt、sltu、srcB、sub）的 32 位 ALU，并对 ALU 的功能进行验证。支持 9 条目标指令的 ALU 原理图如图 11.78 所示，输入为两个 32 位操作数 A 和 B，核心部件是加法器，加法器的输出除了两数之和 Add-Result 以外，还有进位标志 Add-carry、零标志 Zero、溢出标志 Add- Overflow 和符号标志 Add-Sign。在操作控制端 ALUctr 的控制下，在 ALU 中执行“加法”、“按位或”、“操作数 B 直接输出”、“带符号整数比较小于置 1”和“无符号数比较小于置 1”等运算，Result 作为 ALU 运算的结果被输出，同时，零标志 Zero 被作为 ALU 的结果标志信息输出。

原理图：

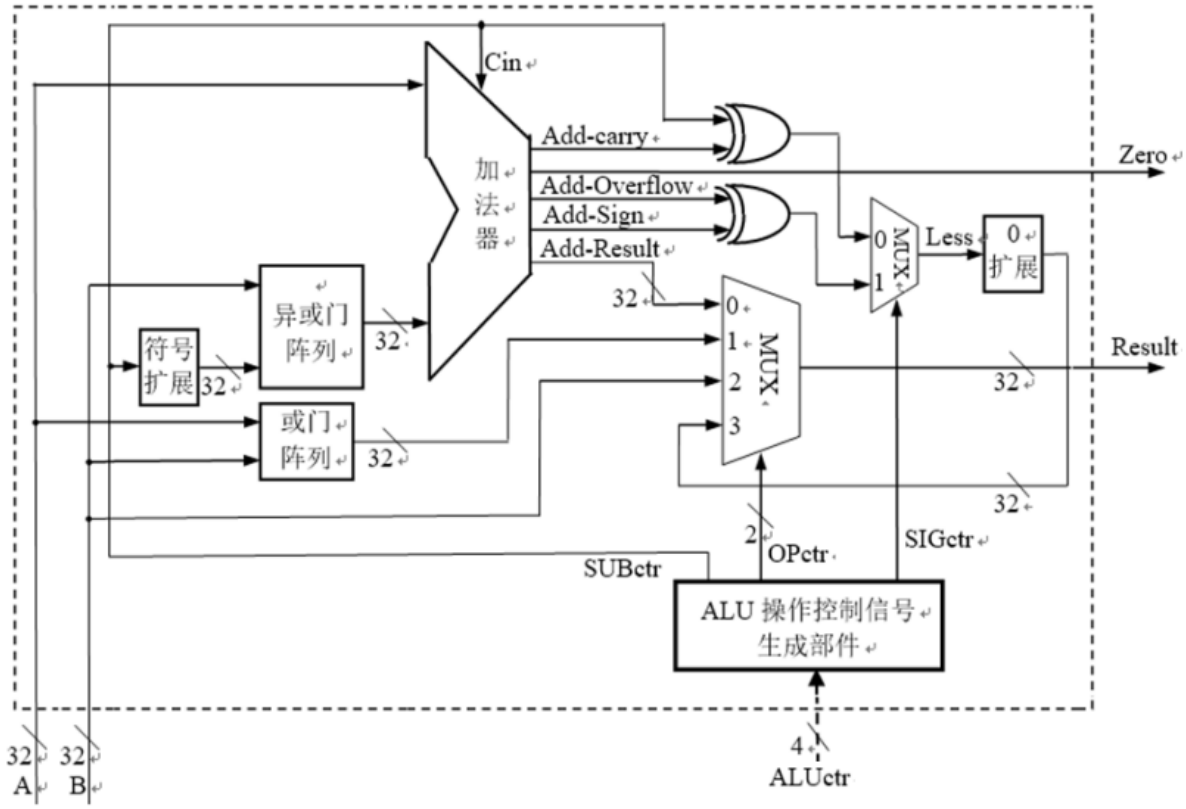


图 11.78　支持 9 条目标指令的 ALU 原理图

ALU 设计实验步骤如下。

1） 设计 ALU 操作控制部件并封装成子电路 ALU 控制信号转换。根据表 11.11 中的 ALUctr 编码方案，得到控制信号 SUBctr、SIGctr、Opctr[0:1]对应的逻辑表达式如下：

SUBctr = (~ALUctr<3> & ~ALUctr<2> & ALUctr<1>) | ALUctr<3>

SIGctr = ~ALUctr<0>

OPctr<1> = (~ALUctr<3> & ~ALUctr<2> & ALUctr<1>) |

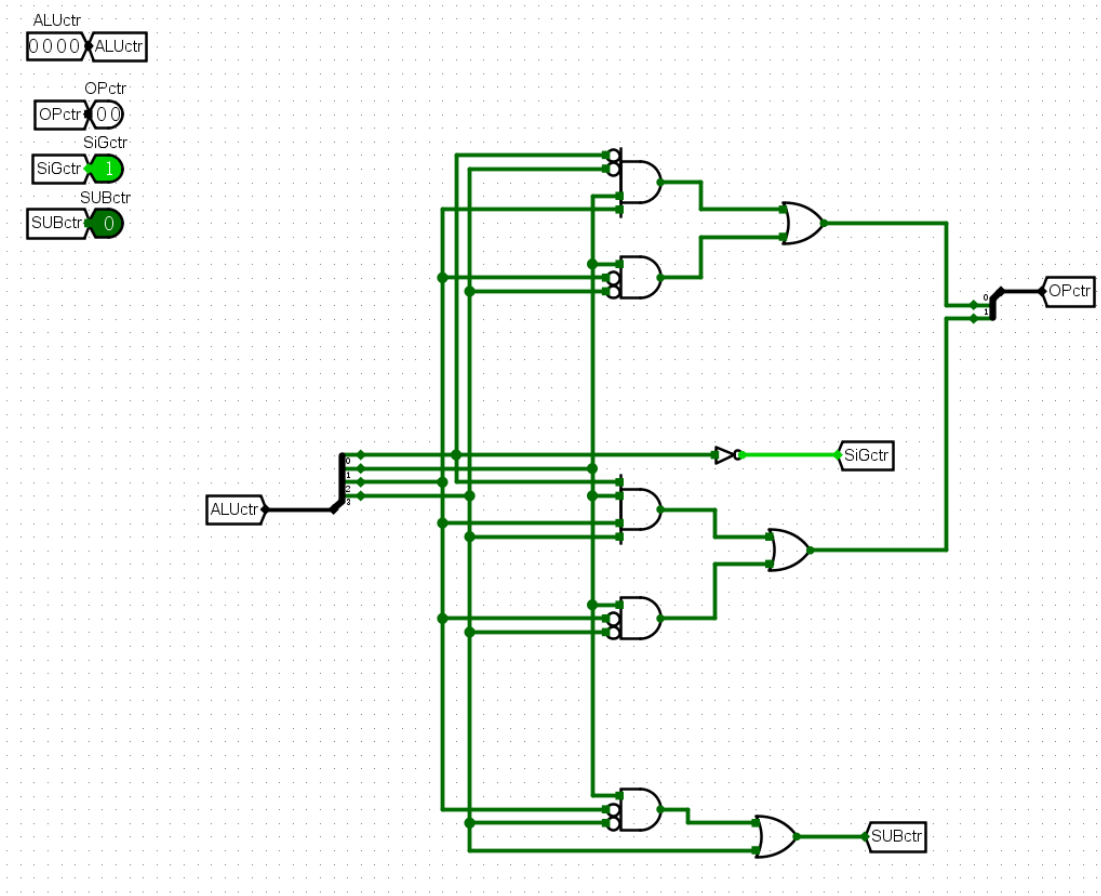
(ALUctr<3> & ALUctr<2> & ALUctr<1> & ALUctr<0>)

OPctr<0> = (~ALUctr<3> & ~ALUctr<2> & ALUctr<1>) |

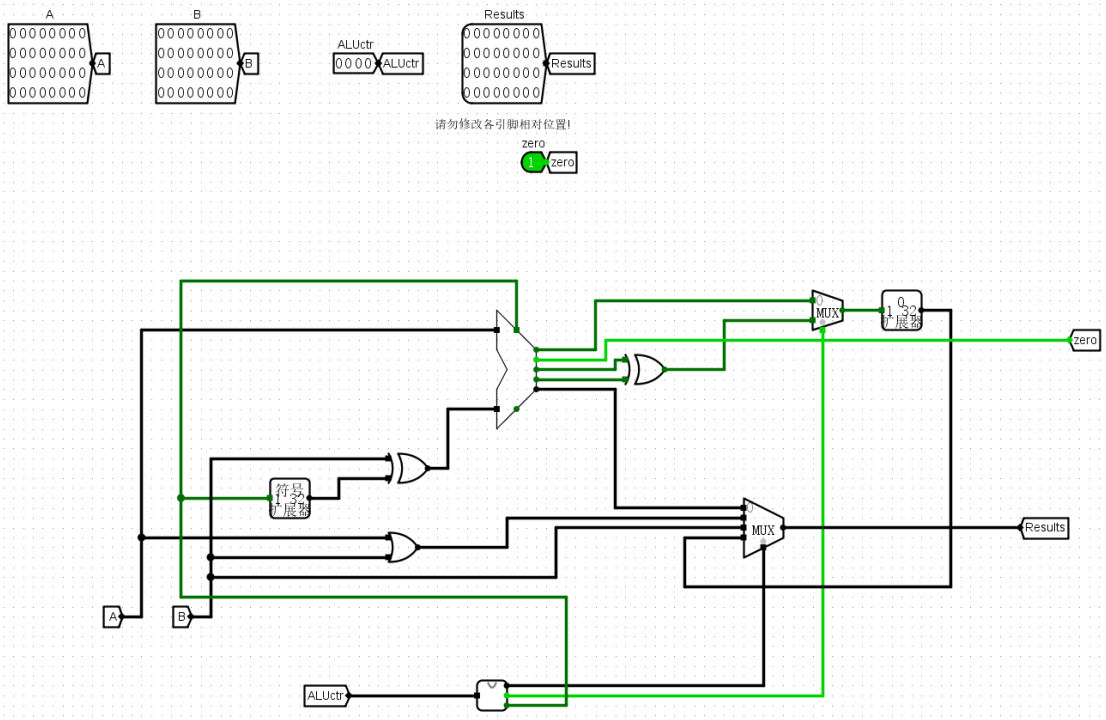
(~ALUctr<3> & ALUctr<2> & ALUctr<1> & ~ALUctr<0>)

根据上述逻辑表达式设计 ALU 操作控制部件，对应电路如图 11.80 所示，将其封装成子电路 ALU 控制信号转换。

电路图：

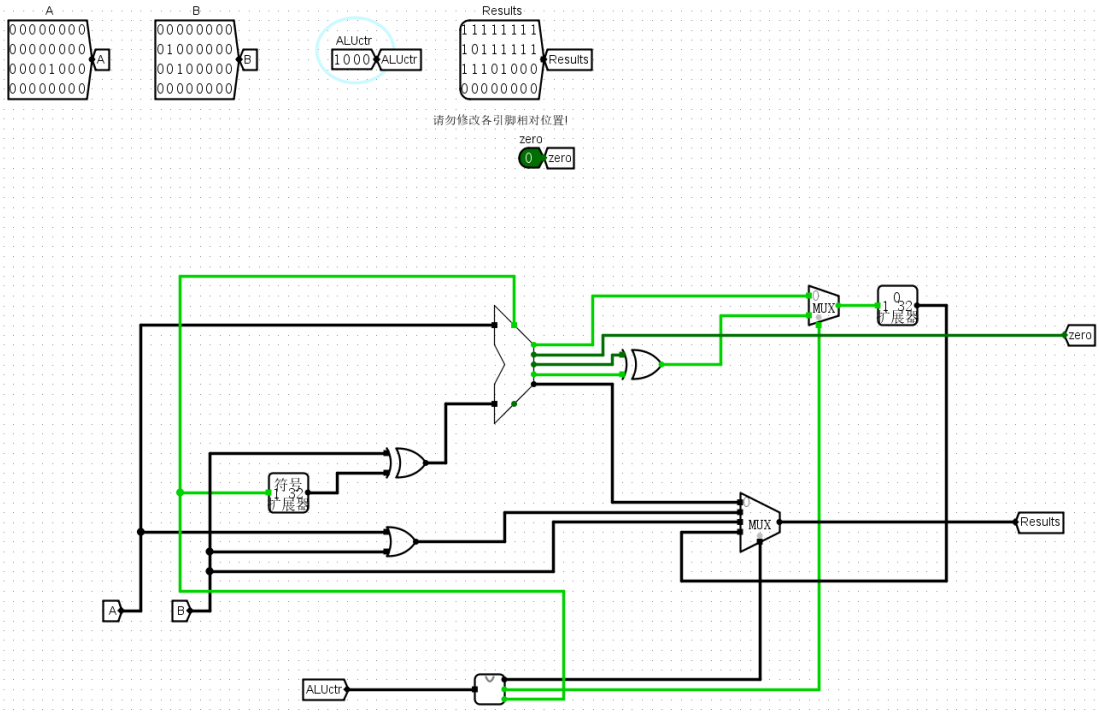


2） 设计实现一个 32 位 ALU。在工程中添加一个名为“32 位 ALU”的子电路，双击该子电路名称，在右侧工作区中构建相应电路。如图 11.81 所示，首先在工作区中添加所需的 1 个 32 位加法器、1 个 ALU 操作控制部件、1 个 2 选 1 多路选择器、1 个 4 选 1 多路选择器、1 个符号扩展器、1 个零扩展器、逻辑门以及输入/输出引脚等，然后进行线路连接,添加标识符和电路功能描述信息。提示:由于 32 位加法器直接输出了 CF 标志位，因此不需要将 CIN 和 Adder-carry 异或来生成 CF，而是直接将加法器的 CF 输出端连到 2 选 1 多路选择器的输入端。

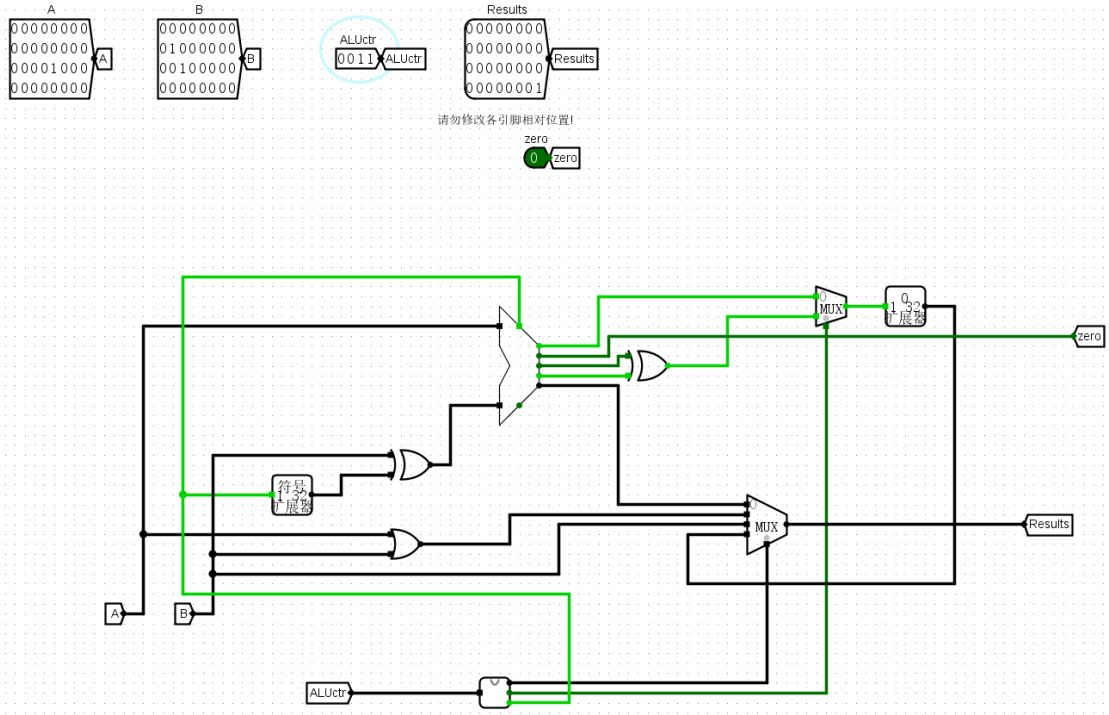


3）检验：

示例 1：sub 操作验证。如图 11.82 所示，将 ALUctr 设置为 1000，此时，OPctr[1:0]为 00，因此，ALU 输出的结果 Results 为 4 选 1 多路选择器中第 0 路（最上面）的输入值，即为加法器的输出 Add-Result。此时，加法器在控制信号 SUBctr=CIN=1 的控制下，其输出结果为 A 和 B 之间的差。对 A 和 B 分别设置不同的输入值，以验证在不同的操作数情况下结果的正确性。图 11.82 给出了 A=0x0000 0800、B=0x0040 2000 时的执行结果为 Results=0xffbf e800，显然运算结果正确。



示例 2：sltu 操作验证。如图 11.83 所示，将 ALUctr 设置为 0011，此时，OPctr[1:0]为 11，因此，ALU 输出的结果 Results 为 4 选 1 多路选择器中第 3 路（最下面）的输入值，即为 0 扩展器的输出。此时，2 选 1 多路选择器在控制信号 SIGctr=0 的控制下，其输出结果为加法器生成的 CF 标志。对 A 和 B 分别设置不同的输入值，以验证在不同的操作数情况下结果的正确性。图 11.83 给出了 A=0x0000 0800、B=0x0040 2000 时的执行结果为 Results=0x0000 0001，即 A<B，显然运算结果正确。

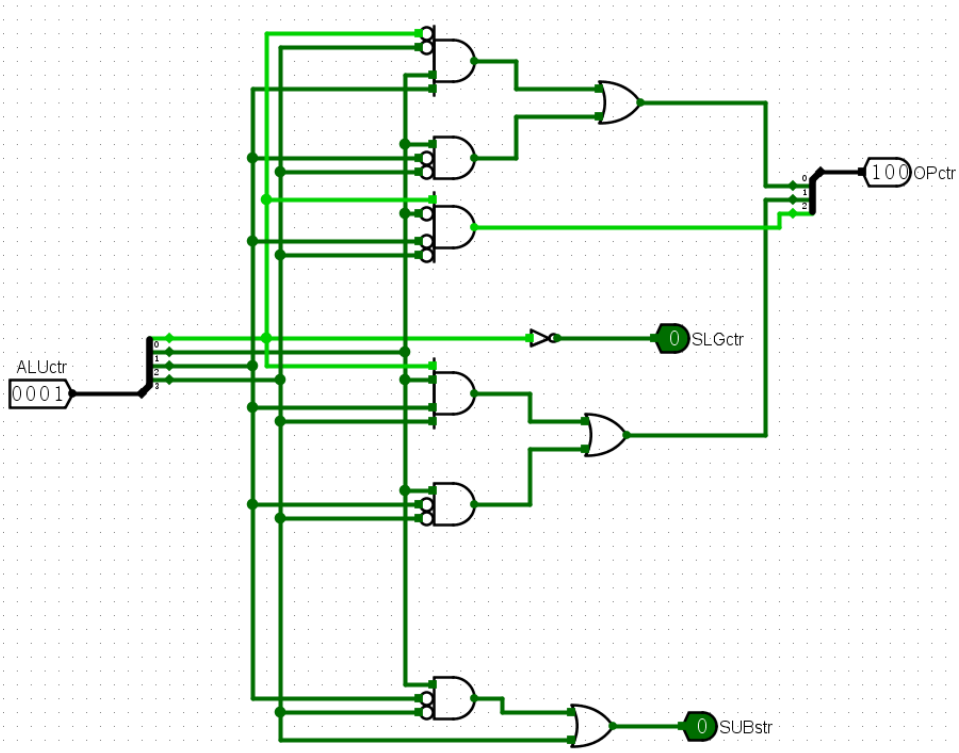


四、思考题

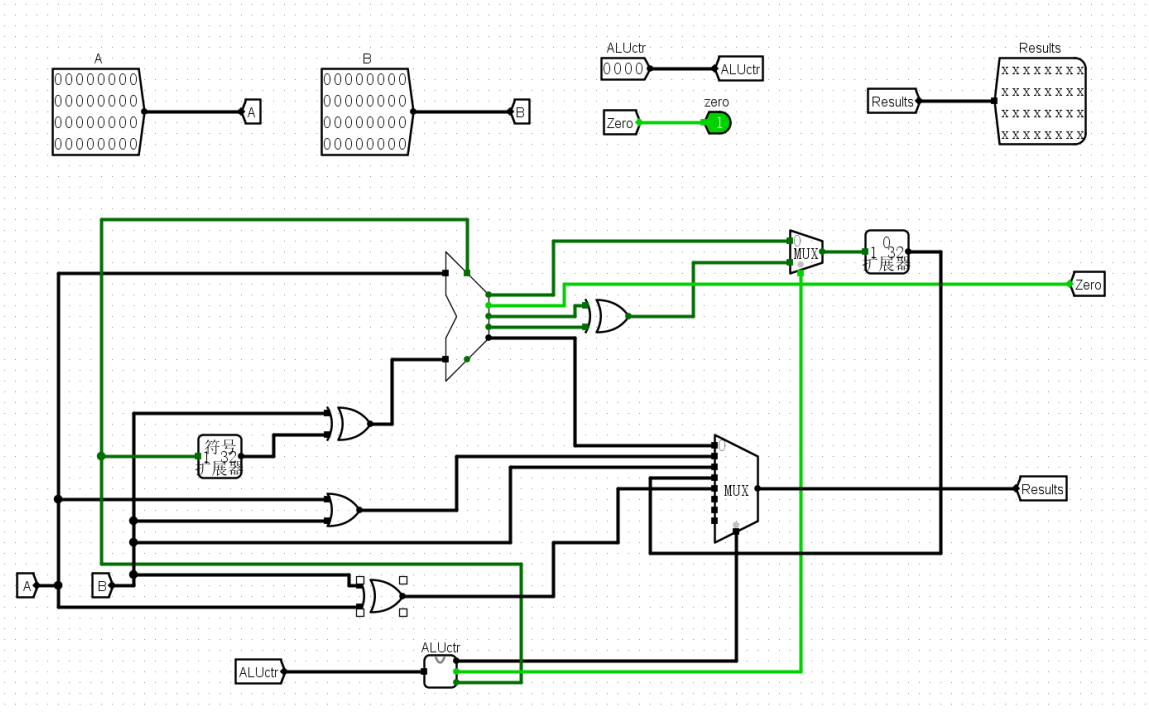
1. 若需要增加一条“xor rd，rs1，rs2”指令，则在所设计的 32 位 ALU 中要做哪些修改？

首先要增加一条 ALUctr 的有效信息，不妨取第一个空信息 0001 为新的 ALUctr，再增加 OPctr 位数，因为两位只能选择四种结果，而现在增加了一种结果，设 xor 对应的 OPctr 为 100，其中 SUBctr 和 SIGctr 无所谓，均为 X 即可。其他对应 OPctr 均为 0xx（原来的）

新的 ALU 操作控制部件电路：



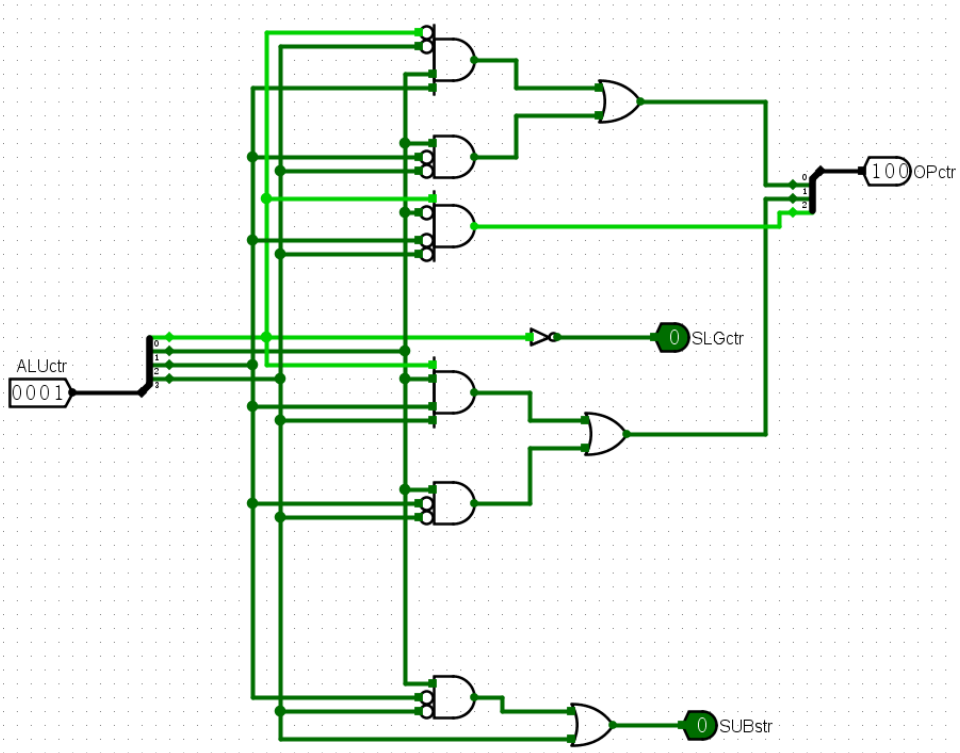
新的 ALU 电路



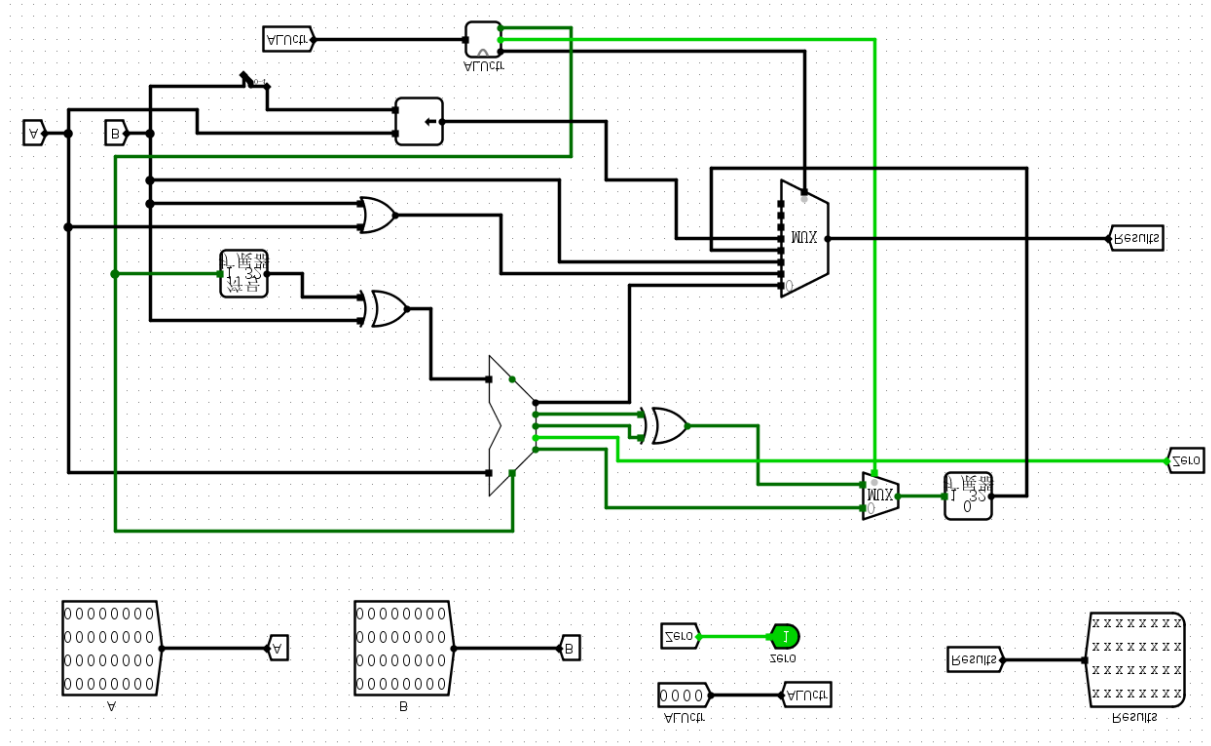
2. 若需要增加一条“sll rd, rs1, rs2”指令，则在所设计的 32 位 ALU 中要做哪些修改？

同理，首先要增加一条 ALUctr 的有效信息，不妨取第一个空信息 0001 为新的 ALUctr，再增加 OPctr 位数，设 sll 对应的 OPctr 为 100，其中 SUBctr 和 SIGctr 无所谓，均为 X 即可。其他对应 OPctr 均为 0xx（原来的）

新的 ALU 操作控制部件电路：



新的 ALU 电路



3. 如何验证运算器结果的正确性

设计数据并计算预期结果

运算器结果与预期结果比对
进行针对性数据检验，例如边界值等