实验 4：算术逻辑部件

实验目的

1. 掌握使用 Logisim 软件设计、实现算术逻辑部件的方法
2. 学习 4 位先行进位加法器 CLA 和先行进位逻辑单元 CLU 的设计原理和实现方法
3. 学习 16 位先行进位加法器及相关标志位的设计原理和实现方法
4. 学习基本算术逻辑部件的设计原理和实现方法，实现 6 种操作的 ALU 器件

实验环境

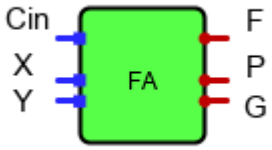
Logisim-ITA V2.16.1.2
https://sourceforge.net/projects/logisimit/

头歌线上评测平台
https://www.educoder.net/classrooms/10924/

实验内容

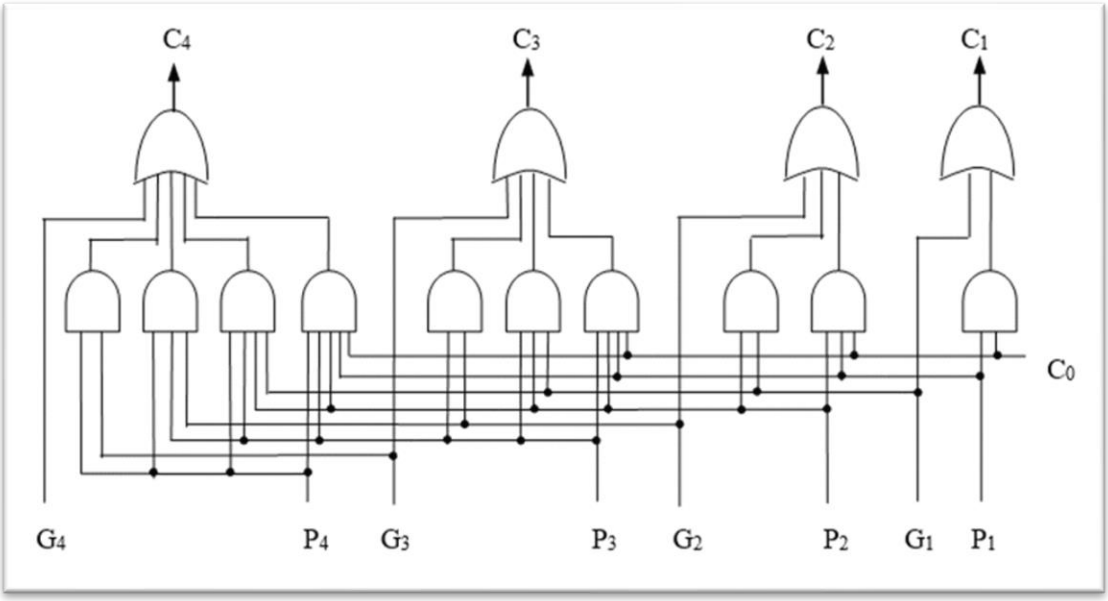
1. 4 位先行进位加法器

首先请同学们根据下列表达式在子电路中实现 1 位全加器，两个位输入 X、Y 和进位输入 Cin，输出加法计算结果 F、进位传递位 P 和进位生成位 G。注：如果使用 Logisim 内置的多位异或门，需要注意异或门输出为 1 的对应行为。

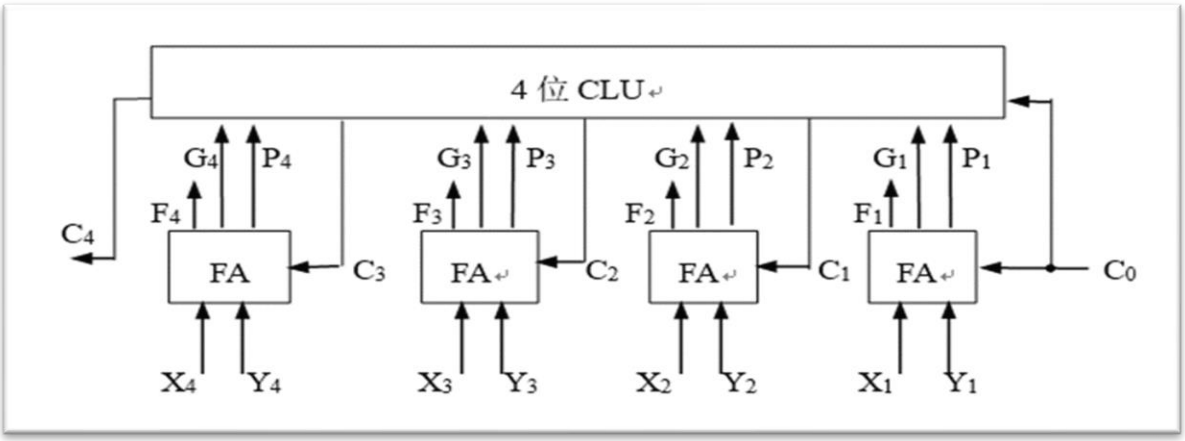


$$F = X \wedge Y \wedge Cin$$
$$P = X \mid Y$$
$$G = X \& Y$$

然后，根据下图在子电路中实现 4 位的组内先行进位部件（CLU），其输入为 4 位进位传递信号 P1、P2、P3、P4，4 位进位生成信号 G1、G2、G3、G4，和一位进位输入 Cin（同 C0）；其输出为四位进位信号 C1、C2、C3、C4。

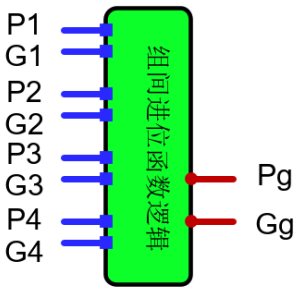


最后，请根据课件中的先行进行加法器原理图，思考并在子电路中完成 4 位 CLA 的实现，其输入为一位 Cin、四位操作数 X1~X4、四位操作数 Y1~Y4，输出为最高位进位 Cout、四位计算结果 S1~S4。



2. 16 位先行进位加法器

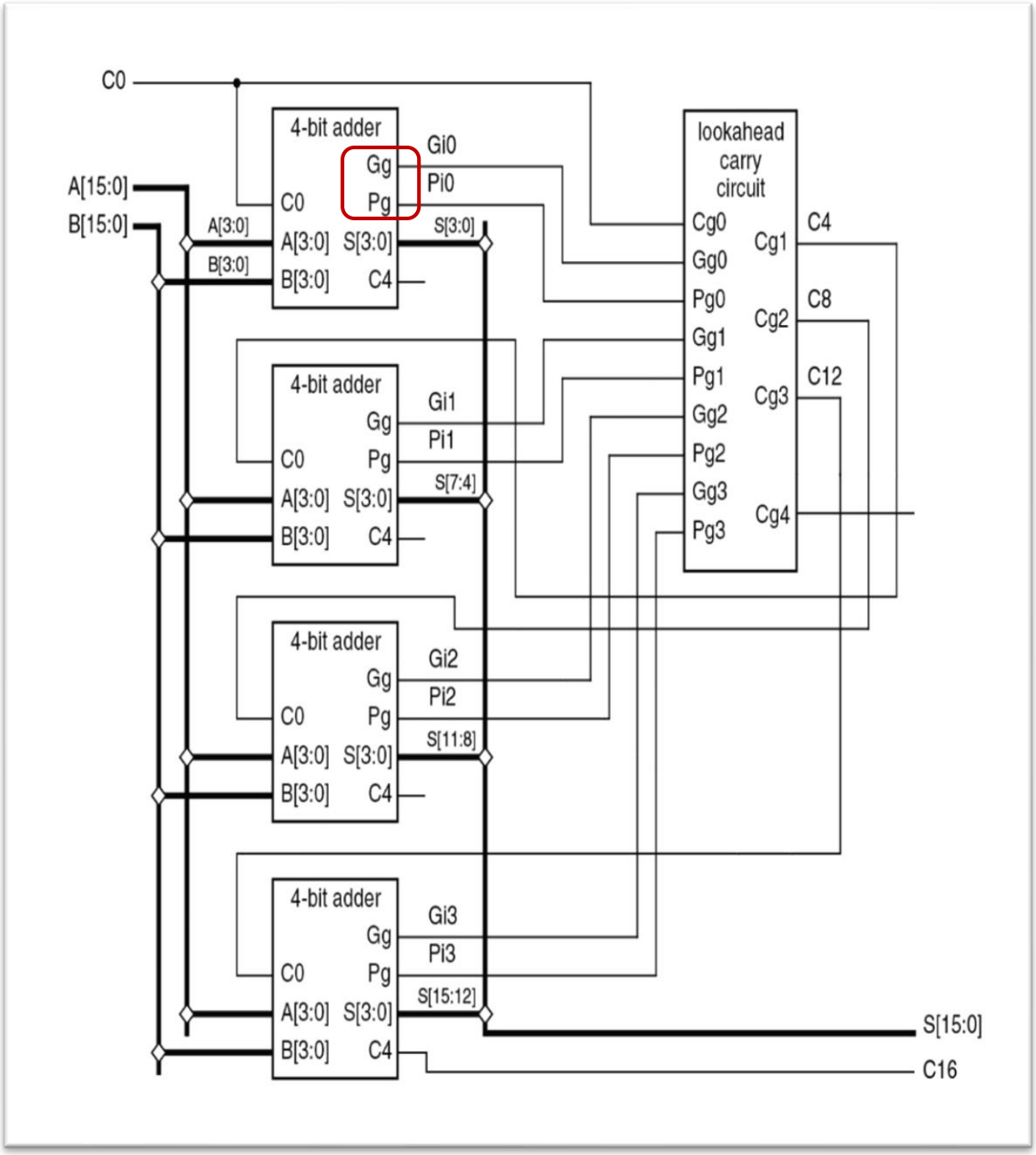
请在子电路中，首先根据下面的表达式在子电路中实现组间先行进位函数逻辑单元，其输入为 4 位进位传递信号 P1、P2、P3、P4 和 4 位进位生成信号 G1、G2、G3、G4，输出为组间进位传递信号 Pg 和组间进位生成信号 Gg。



$$Gg = G_4 + P_4 \cdot G_3 + P_4 \cdot P_3 \cdot G_2 + P_4 \cdot P_3 \cdot P_2 \cdot G_1$$
$$Pg = P_4 \cdot P_3 \cdot P_2 \cdot P_1$$

然后，下图是通过 4 位操作数一组的形式级联的 16 位先行进位加法器，请根据图中所示加法器工作原理，在“可级联的 4 位 CLA”子电路中对前一关卡中已经做过的 4 位 CLA 做出修改（可以将原始的“4 位 CLA”子

电路中的内容复制过来，然后再修改，避免重复劳动），使其能够支持组间级联，注意观察下图中的红色标记区域，思考应该如何利用刚刚实现的组间先行进位函数逻辑单元修改 CLA。

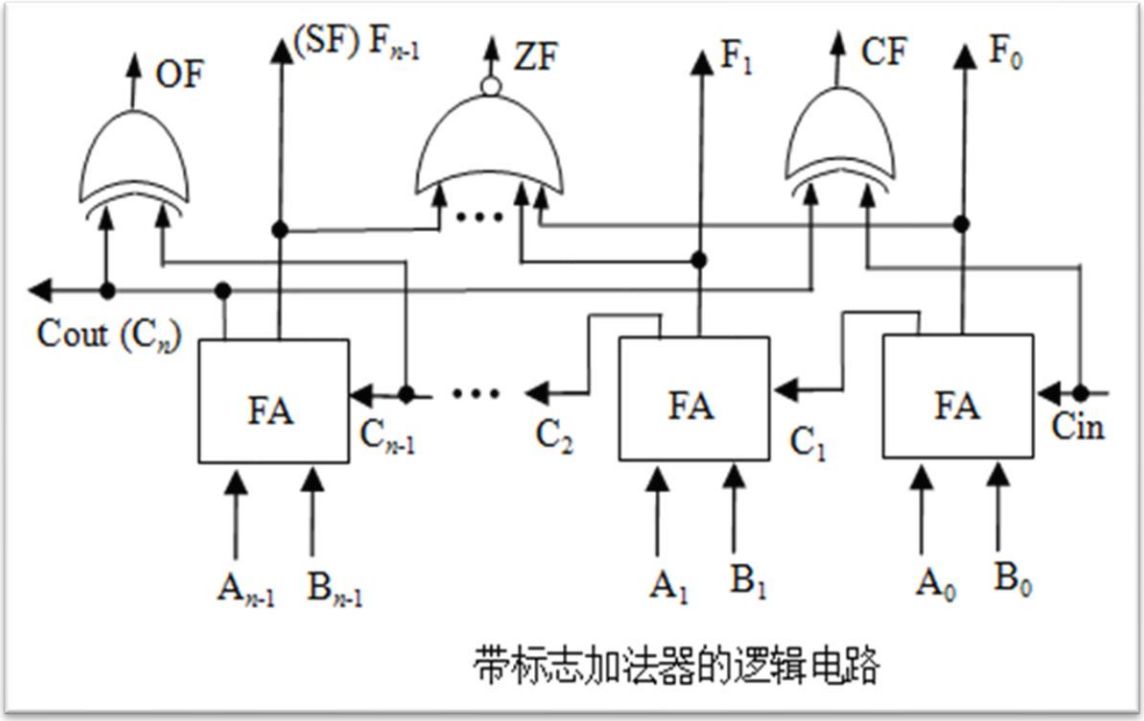


对于 16 位 CLA，输入为两个 16 位操作数 X、Y（同 A、B），一位进位 Cin（同 C0）；输出为 16 位计算结果 S 和最高位进位 Cout（同 C16）。在完成关卡任务后，请同学们思考，如果关卡要求实现的是 32 位先行进位加法器，那么该如何实现？如果在操作数以 4bit 为一个输入分组的条件下，是否需要对原有的 CLU 做出修改呢？

3. 算术逻辑部件（ALU）

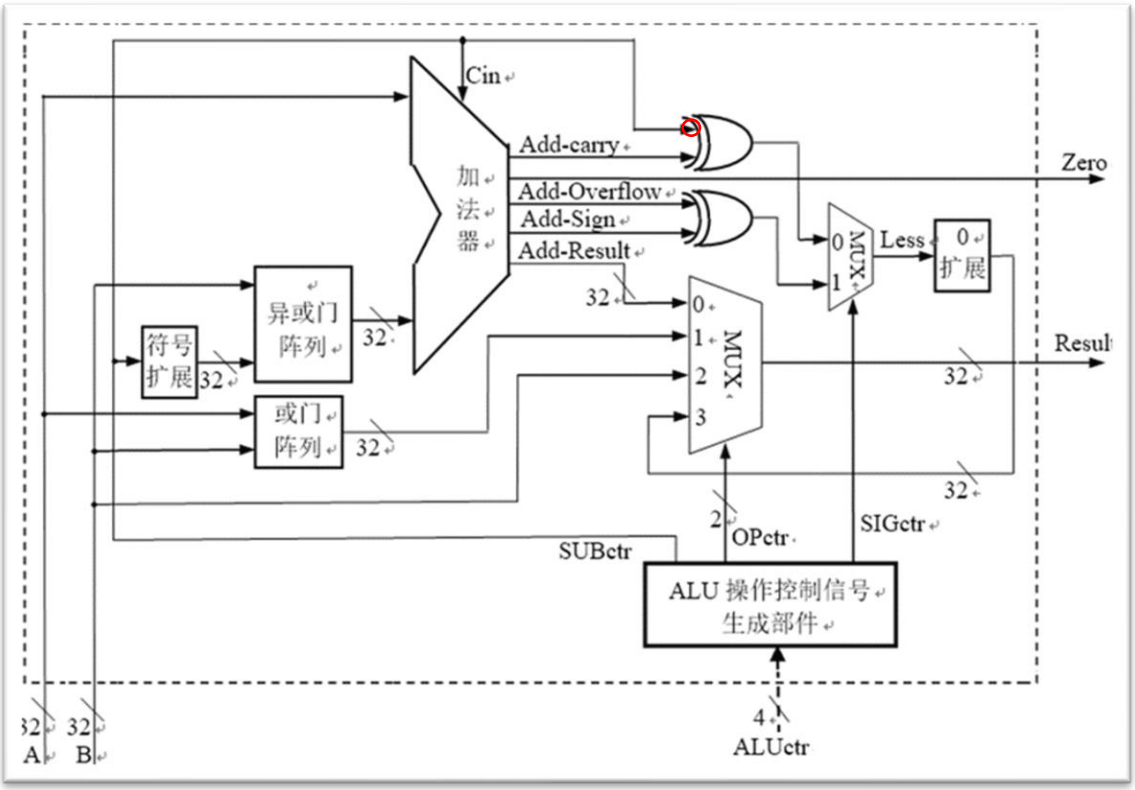
请根据下面给出的电路原理图，首先实现一个用于两个 4 位操作数的带标志位加法器件（可以使用 Logisim 内置库中的加法器作为 1 位全加器），其输入为两个 4 位操作数 X1、X2、X3、X4，Y1、Y2、Y3、Y4（由低位到高位，同图中 A0~A3、B0~B3），进位 Cin；输出为 4 位计算结果 S（同图中 F），最高位进位 Cout，溢出标志位 OF，符号标志位 SF，零标志

位 ZF，进/借位标志位 CF。



带标志加法器的逻辑电路

接下来，请仔细观察如下所示的 ALU 设计原理图，理解 ALU 操作控制信号 ALUctr 的生成原理，在子电路中实现一个支持 6 种操作（add、slt、sltu、sub、or、srcB）的 4 位 ALU。



如下表所示，是 ALUctr 的一种四位编码方案，本次实验中的作答区文件已经给出了 ALU 所需支持的 6 种操作对应的操作控制信号生成部件（可以直接从子电路中拖出来使用），其中 SUBctr、SIGctr、OPctr 是由 ALUctr 得到的关于 ALU 的三个操作信号，具体如下。

ALUctr<3:0>	操作类型	SUBctr	SIGctr	OPctr<1:0>	OPctr 的含义
0 0 0 0	add	0	×	0 0	选择加法器的结果输出
0 0 0 1	（未用）				
0 0 1 0	slt	1	1	1 1	使用减法做有符号整数的比较大小，如果 X 小于 Y 置位输出
0 0 1 1	sltu	1	0	1 1	使用减法做无符号整数的比较大小，如果 X 小于 Y 置位输出
0 1 0 0	（未用）				
0 1 0 1	（未用）				
0 1 1 0	or	×	×	0 1	选择按位或结果输出
0 1 1 1	（未用）				
1 0 0 0	sub	1	×	0 0	选择加法器的结果输出
其余	（未用）				
1 1 1 1	srcB	×	×	1 0	选择操作数 B 直接输出

本地测试用例，共 20 条 ALU 指令：4 条 add 操作测试、3 条 slt 操作测试、3 条 sltu 操作测试、2 条 srcB 操作测试、4 条 or 操作测试（二进制表示，左侧为最高位，右侧为最低位）

编号（Index）	ALUctr	操作数 X	操作数 Y	预期 Result
0000	0000	0001	0010	0011
0001	0000	1010	1010	0100
0010	0000	1111	0001	0000
0011	0010	0011	0111	0001
0100	0010	0011	1000	0000
0101	0010	0000	1010	0000
0110	0011	0011	0111	0001
0111	0011	0011	1000	0001
1000	0011	1010	1000	0000
1001	0110	1111	0000	1111
1010	0110	0000	0101	0101
1011	0110	1010	1001	1011
1100	1000	1111	0001	1110
1101	1000	0011	0011	0000
1110	1000	0001	0011	1110
1111	1111	0011	0101	0101