**第四次实验报告-221900180-田永铭**

**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

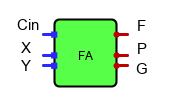
2. 学习4位先行进位加法器CLA和先行进位逻辑单元CLU的设计原理和实现方法

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

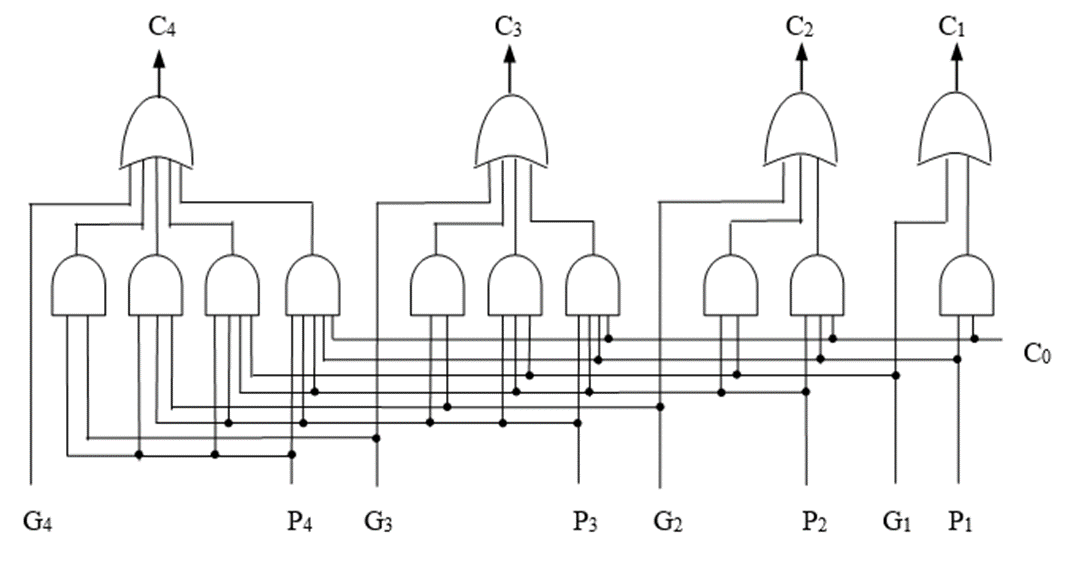
**实验1 4位先行进位加法器**

**（一）实验原理**

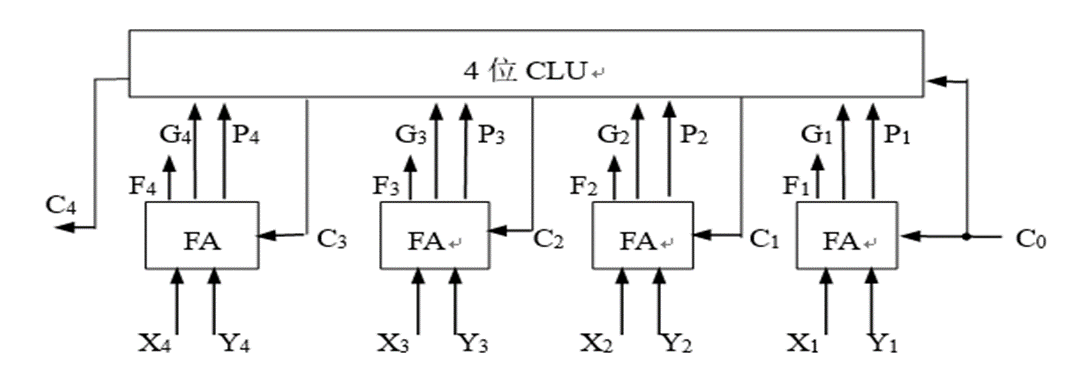
**4位先行进位加法器**

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。

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。



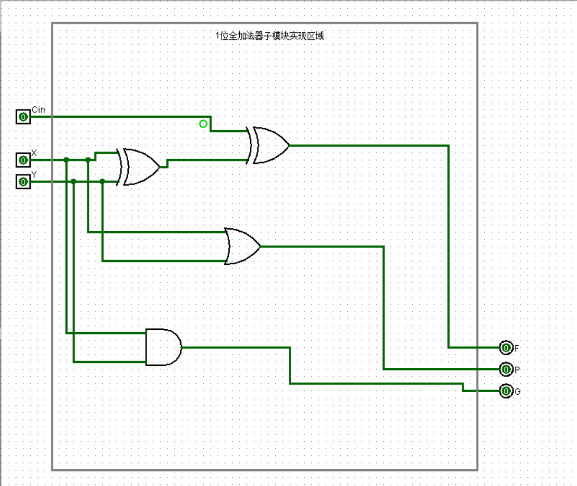
最后，请根据课件中的先行进行加法器原理图，思考并在子电路中完成4位CLA的实现，其输入为一位Cin、四位操作数X1~X4、四位操作数Y1~Y4，输出为最高位进位Cout、四位计算结果S1~S4。



**（二）实验步骤**

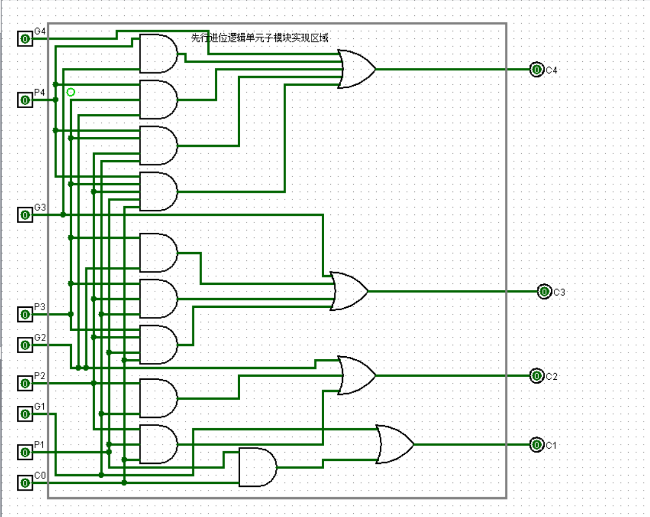
1.首先在子电路中实现1位全加器：

在电路中添加1个2输入的与门，1个2输入的或门，2个2输入的异或门，按照下图所示进行摆放与连接。



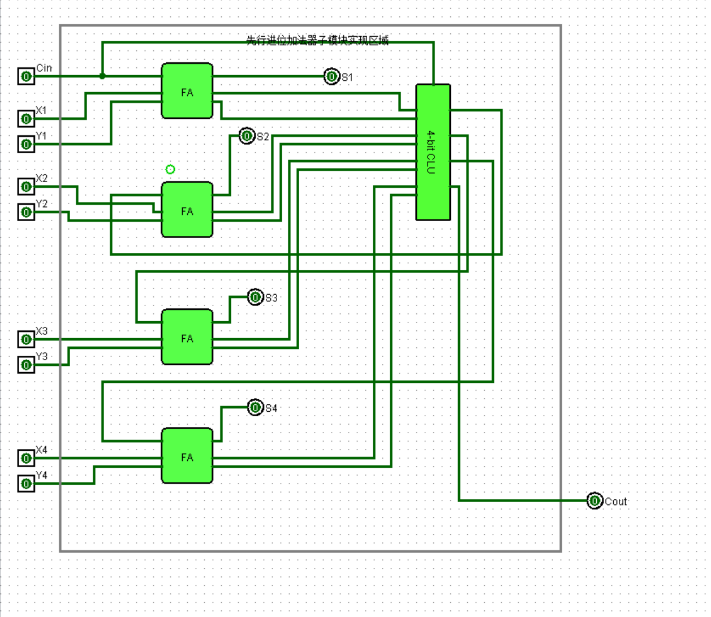
2.再实现4位的组内先行进位部件（CLU）

按照规律将与门分为四组，每组接一个或门，按照下图所示方法摆放并且连接。



3.实现CLA：

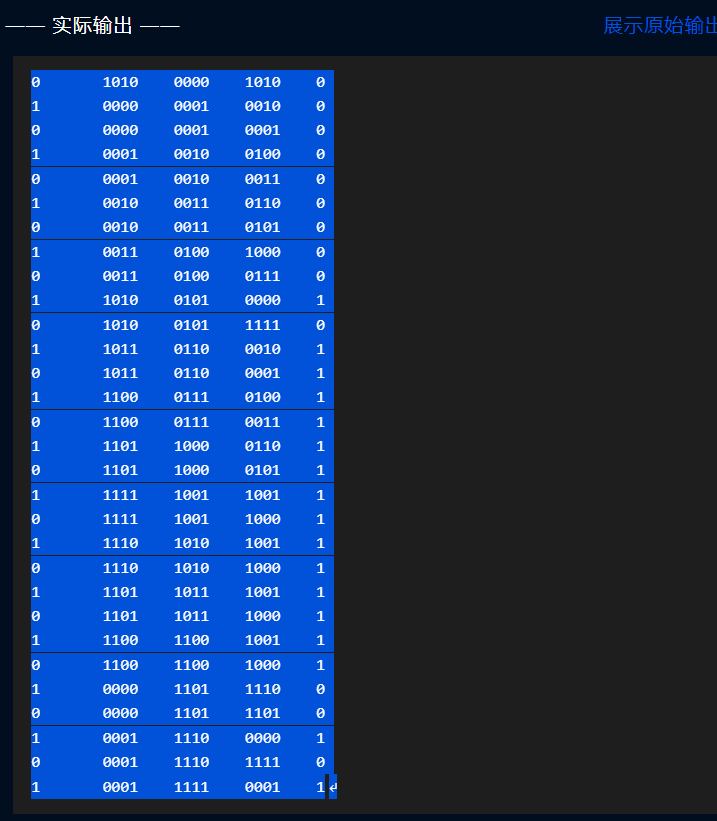
向电路中添加4个第一步实现的全加器，再添加1个第二步实现的CLU，按照下图方式摆放并且连接。

.

4.仿真验证电路，进入仿真状态，改变输入引脚赋值，记录输出引脚的数值，填写输入输出数据表，验证电路功能。

**（三）测试结果**

**Cin x1 y1 x2 y2 x3 y3 x4 y4 s1 s2 s3 s4 Cout**



**（四）总结与思考**

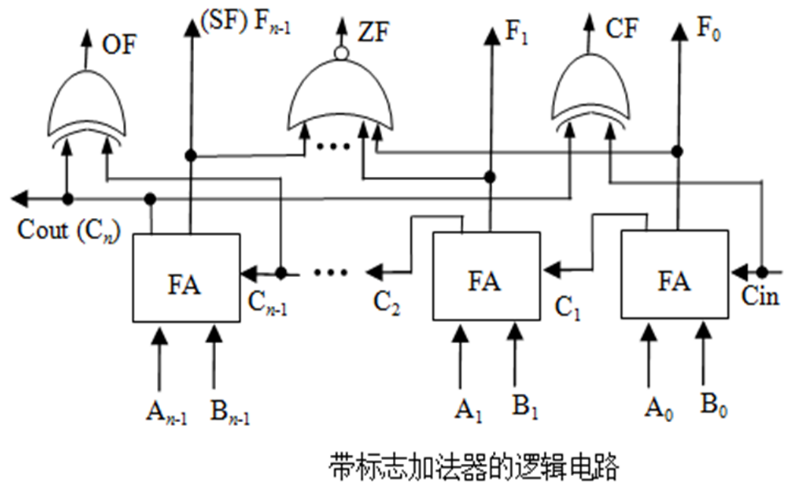
通过实验，我掌握了一些使用Logisim构建逻辑电路的方法，成功构建了一个学习4位先行进位加法器CLA。在搭建电路的时候，我出了一个小状况，某一步的连线，我一不小心连岔了。经过改正，我完成了电路。这让我反思：在连接电路的时候，应该非常细心，才能成功。

**实验2 4位算术逻辑部件**

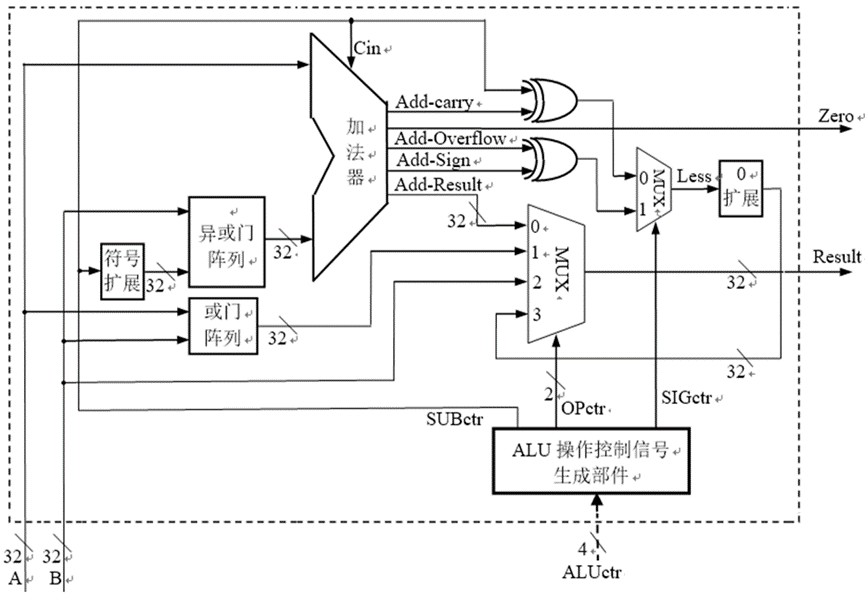
**（一）实验原理**

1. **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。



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。



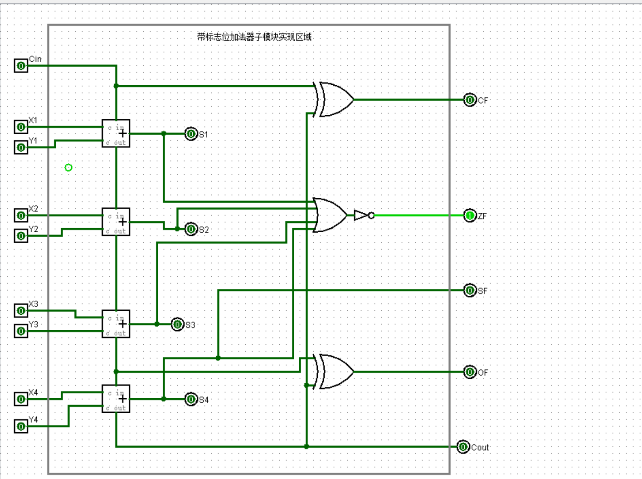
如下表所示，是ALUctr的一种四位编码方案，本次实验中的作答区文件已经给出了ALU所需支持的6种操作对应的操作控制信号生成部件（可以直接从子电路中拖出来使用），其中SUBctr、SIGctr、OPctr是由ALUctr得到的关于ALU的三个操作信号，具体如下。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 |

**（二）实验步骤**

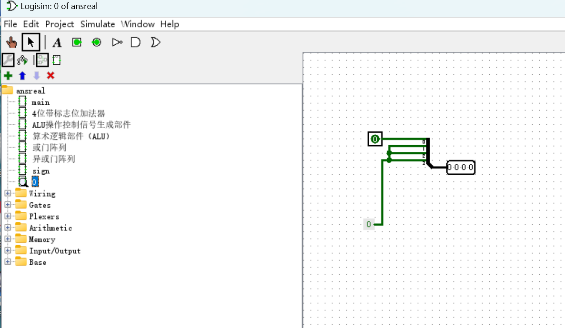
1.在子模块区域实现一个带标志位加法器：

利用系统提供的全加器以及若干异或门、非门、或门，按照下图方式实现。



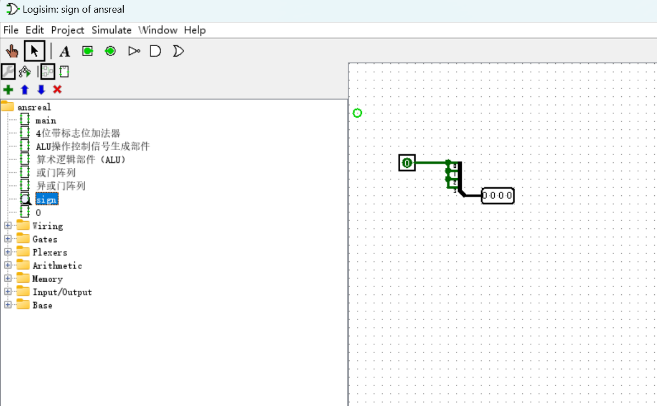
2.在子模块区域实现一个“0扩展器”：

利用一个变量和一个恒定输入0和一个分线器和一个4位输出按照下图方式实现。



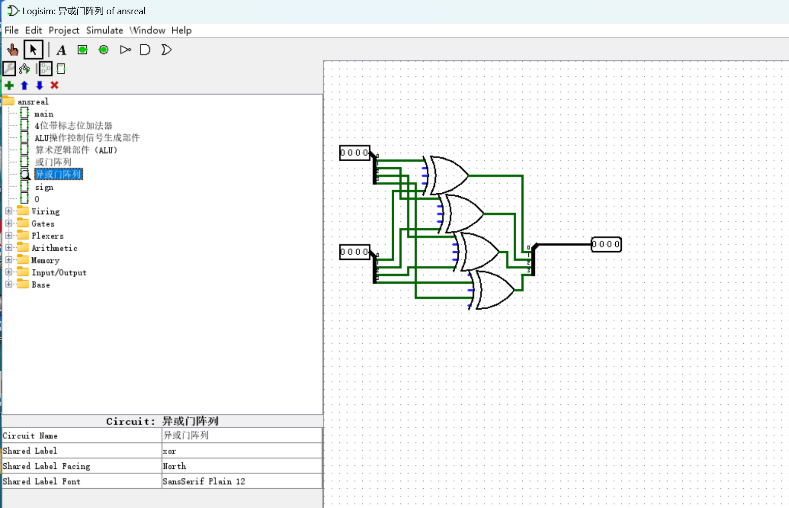
3. 在子模块区域实现一个“符号扩展器”：

按照下图方式实现。



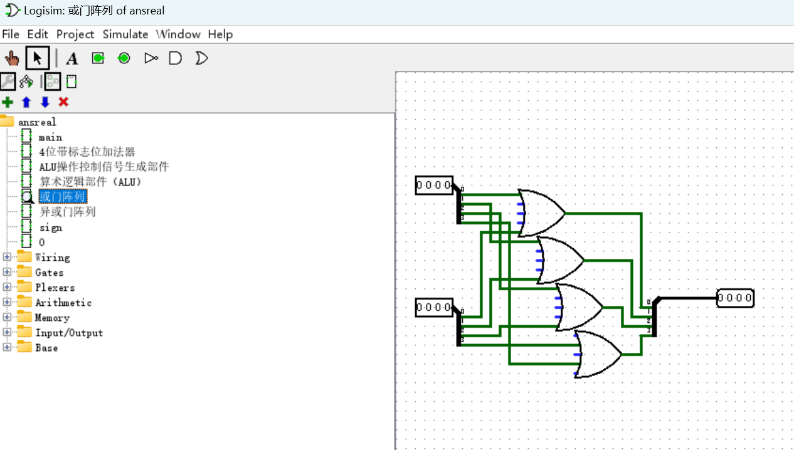
4. 在子模块区域实现一个“异或门阵列”：

按照下图方式实现。

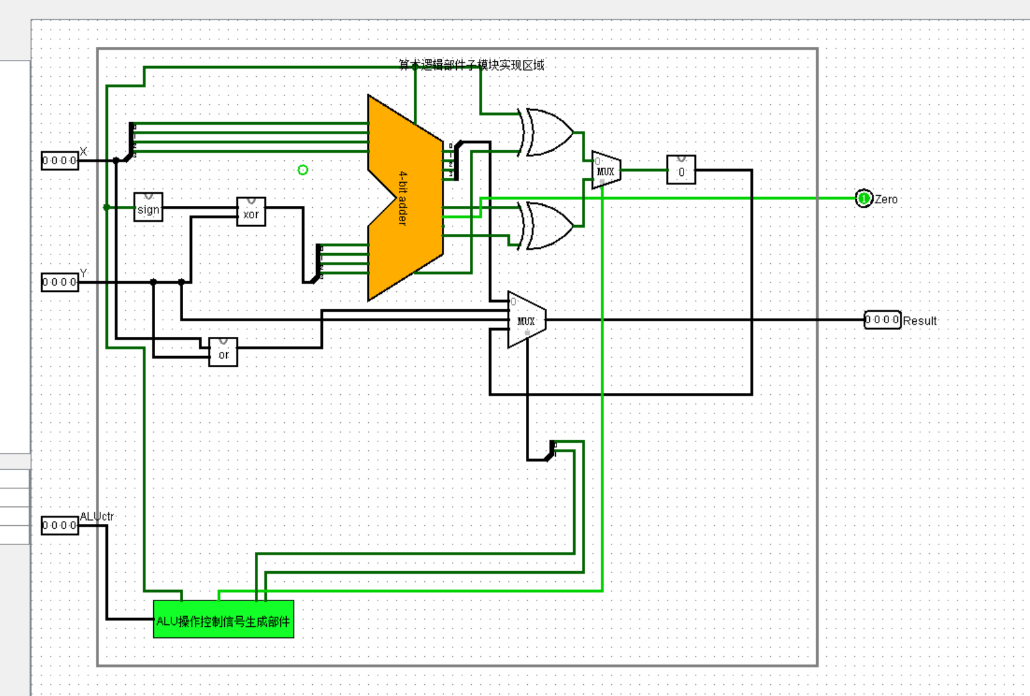


5. 在子模块区域实现一个“或门阵列”：

按照下图方式实现。



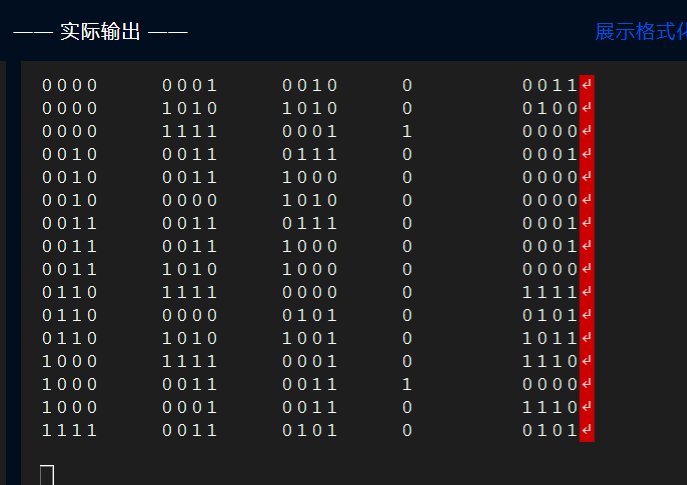
6.根据两个4位操作数的带标志位加法器件的工作原理，巧妙地利用上面几个步骤的实现结果，进行级联，按照下图方式实现。



7.仿真验证电路，进入仿真状态，改变输入引脚赋值，记录输出引脚的数值，填写输入输出数据表，验证电路功能。

**（三）测试结果**

**ALUctr X Y 0标志位 Result**

****

**（四）总结与思考**

通过这次简单的实验，我掌握了一些使用Logisim构建的方法，成功构建了一个带标志位加法器件。原本在课堂上，我对其理解并不深。但是通过这一次的电路搭建，我加深了对它的理解，更加清楚了其中的原理。

我总结到：对于复杂的电路图，其中的原理可能有些难理解，但是如果能像这题一样，拆成一个一个的子电路，一个一个模块化，就可以变得非常清晰并且易于理解。