

《 计算机组成原理 》

实验报告本

|  |  |
| --- | --- |
| 班 级： | **计203** |
| 学 号： | **20002462** |
| 姓 名： | **刘子言** |
| 指导教师： | **张欢欢** |

信息科学与工程学院

2022年 6 月

**实验一：数字逻辑——交通灯系统设计**

**子实验1：7段数码管驱动电路设计**

姓名： 刘子言 学号： 20002462 班级： 计203 指导教师(签章) 张欢欢

成绩:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**一、实验设计要求**

（1）理解利用真值表的方式设计电路的原理；

（2）利用Logisim真值表自动生成电路的功能，设计一个7段数码管显示驱动。

**二、实验方案设计**

7段数码管显示驱动的设计方案：

1. 输入：4位二进制
2. 输出：7段数码管7个输出控制信号
3. 电路引脚：



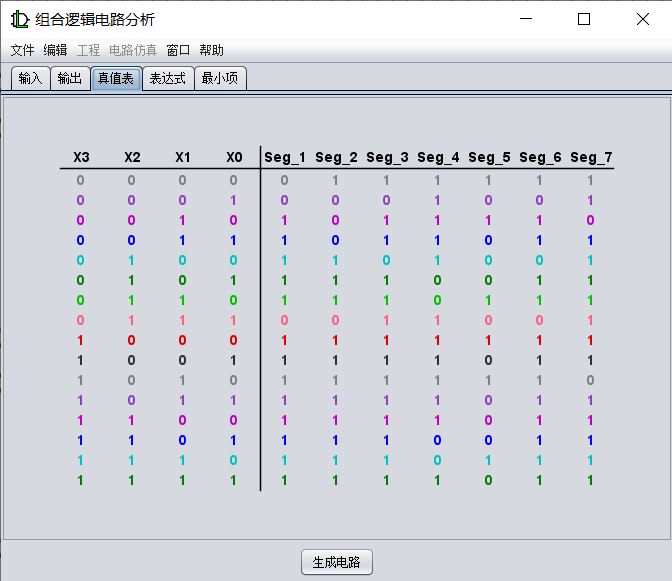
1. 实现功能：利用7段数码管显示4位二进制的16进制值
2. 设计方法：

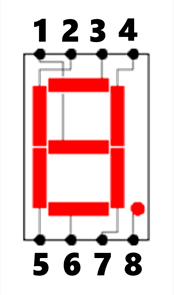
由于该实验若直接进行硬件设计会比较复杂，而7段数码管显示的真值表较容易掌握，所以我们选择由真值表自动生成电路的方法完成该实验。

先分析设计7段数码管显示驱动的真值表，再利用Logisim中的“分析组合逻辑电路”功能，将真值表填入，自动生成电路。

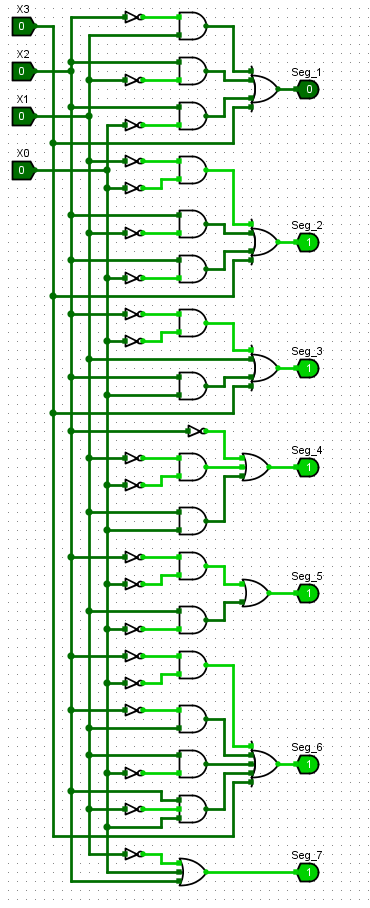
1. 真值表的设计：

由于是4输入7输出，真值表共有16行。7输出对应7个引脚，所以需要依次对照LED灯的引脚顺序进行设计，如下图所示（注意LED的引脚顺序）：

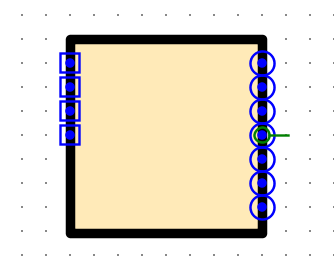




1. **实验步骤**
2. 在实验平台下载实验框架文件RGLED.circ；
3. 在Logisim中打开RGLED.circ文件，选择数码管驱动子电路；
4. 点击“工程”中的“分析组合逻辑电路”功能，先构建4输入和7输出，再在“真值表”中，将已设计好的真值表的所有数值仔细对照着填入表格中，确认无误后点击“生成电路”，自动生成的电路如下图所示：



1. 将子电路封装为如下形式：



（5）进行电路测试：

·自动测试

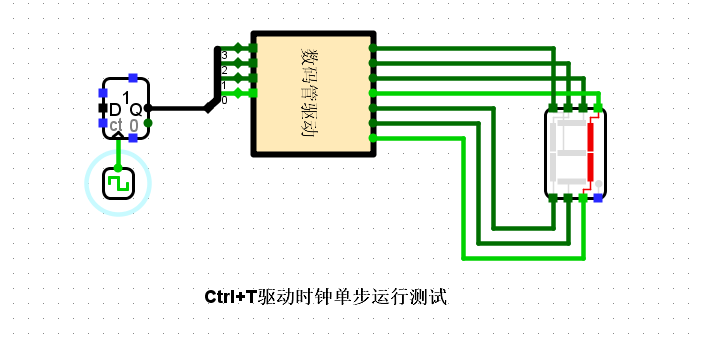
在数码管驱动测试子电路中进行测试；

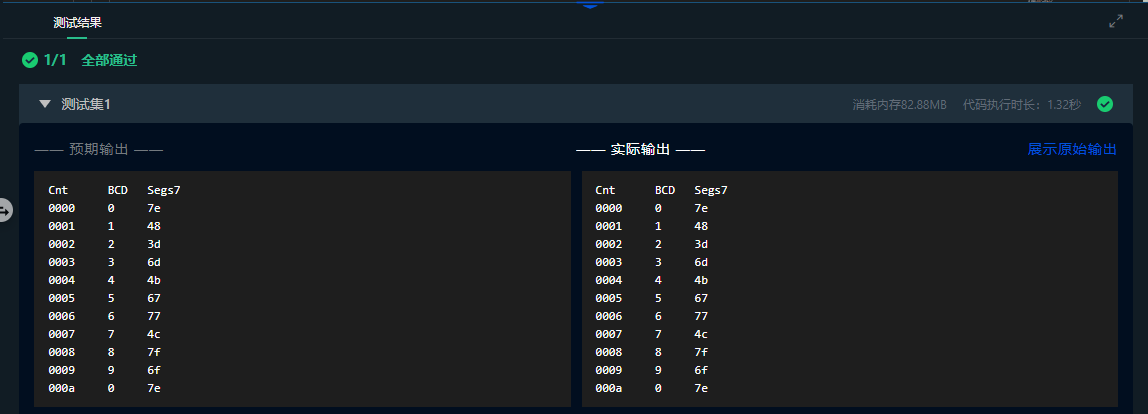
·平台评测

自动测试结果满足实验要求后，再利用记事本打开RGLED.circ文件，将所有文字信息复制粘贴到Educoder平台代码区域，点击评测按钮进行测试。

1. **实验结果测试与分析**

（1）自动测试的部分结果如下：

（2）平台测试结果如下：



综上，本实验测试结果为通过，无故障显示。

本实验的关键点在于：在设计时需要格外注重LED灯的引脚顺序，保证0-9数字显示的正确性，设计出正确的真值表。

**子实验2：4位无符号比较器设计**

姓名： 刘子言 学号： 20002462 班级： 计203 指导教师(签章) 张欢欢

成绩:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**一、实验设计要求**

（1）理解利用逻辑表达式设计电路的原理；

（2）利用Logisim中的逻辑表达式自动生成电路的功能，设计一个4位无符号比较器。

**二、实验方案设计**

4位无符号比较器的设计方案：

（1）输入：两个4位输入X、Y

（2）输出：大于，等于，小于，均为1位

（3）电路引脚：



（4）实现功能：两个4位输入值的无符号比较

（5）约束条件：只能使用线路库、逻辑门组件和输入输出库

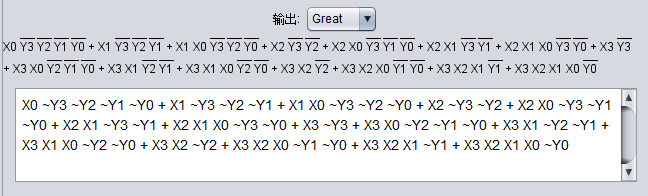
（6）设计方法：

该实验若直接进行硬件设计,电路会相当复杂；若利用真值表设计，该电路有8个输入，则真值表会有256项，过于繁琐且容易出错。所以对于这类电路的设计，我们可以采用构建逻辑表达式自动生成电路的方式来实现。

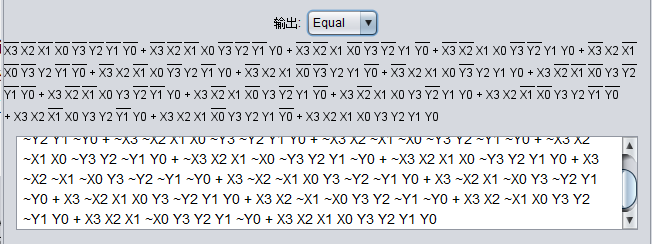
先分析设计4位无符号比较器的逻辑表达式，再利用Logisim中的“分析组合逻辑电路”功能，将表达式填入，自动生成电路。

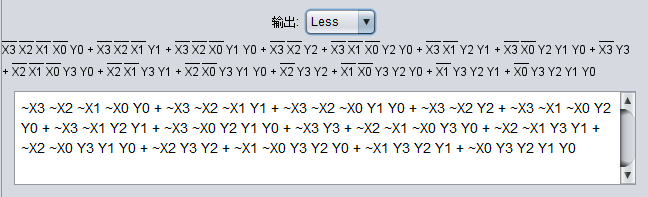
（7）逻辑表达式的设计：

该实验有4+4个输入，3个输出，需要列出大于、等于、小于这三种情况的表达式。比较时满足X>Y则Great输出1，每位完全相同时Equal输出1，满足X<Y则Less输出1。逻辑表达式构建如下：

·Great（X>Y，从X高位为1逐步向低位发展，寻找满足条件的情况）

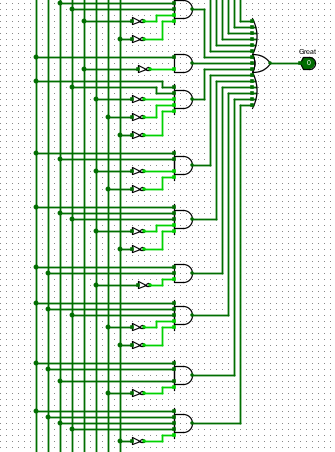
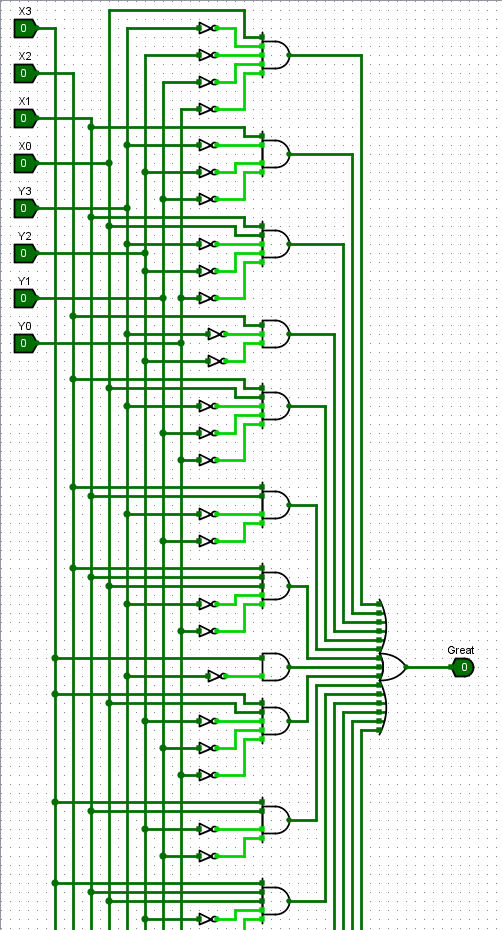
·Equal（X=Y，16种情况，所以共16个最小项）

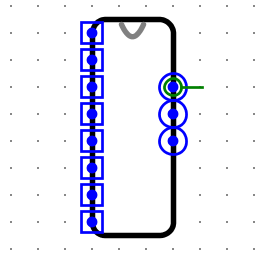
·Less（X<Y，从Y低位为1逐步向高位发展，寻找满足条件的情况）



**三、实验步骤**

1. 在Logisim中打开RGLED.circ文件，选择4位无符号比较器子电路；
2. 点击“工程”中的“分析组合逻辑电路”功能，先构建8输入和3输出，再在“表达式”中，将已设计好的三个逻辑表达式仔细对照着填入相应方框，确认无误后，可在“最小项”中选择对表达式进行化简；
3. 化简完成后点击“生成电路”。由于自动生成的电路十分复杂，此处只截取其中部分电路展示如下：



1. 将子电路封装为如下形式：
2. 进行电路测试：

利用记事本打开RGLED.circ文件，将所有文字信息复制粘贴到Educoder平台代码区域，点击评测按钮进行测试。

**四、故障与调试**

（1）故障现象

第一次调试结果报错，少数X、Y的大小判断时Great与Less输出均为1；

（2）原因分析

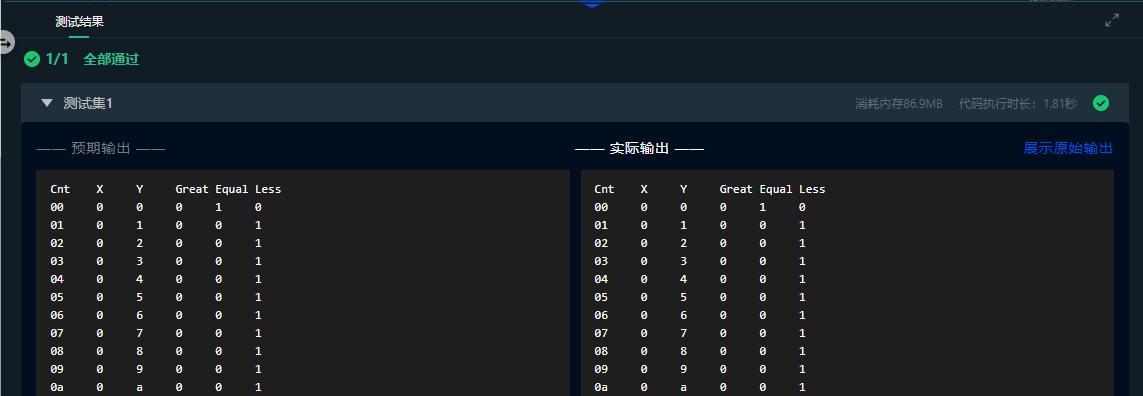
Great与Equal值输出均正确，初步判断是Less的逻辑表达式输入有误；

（3）解决方案

检查Less的逻辑表达式输入，发现有一项书写错误，修改后再进行测试，测试通过。

1. **实验结果测试与分析**

最终测试的部分结果截图如下：



综上，本实验测试结果为通过，无报错。

本实验的关键点在于：在设计时需要理清表达式的设计原理，依照一定的顺序设计表达式，保证设计时不会有疏漏；此外还要格外注意表达式的输入，需要仔细对照，确保表达式书写的正确性；

本实验设计了4位无符号比较器，利用4位无符号比较器的迭代还可以构建16位及以上的无符号比较器。

**实验二：运算器设计**

**子实验1：8位可控加减法电路设计**

姓名： 刘子言 学号： 20002462 班级： 计203 指导教师(签章) 张欢欢

成绩:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**一、实验设计要求**

（1）掌握一位全加器的实现逻辑，在此基础上实现8位可控加减法电路的逻辑；

（2）利用Logisim平台中已经封装好的全加器FA，以及其他的基本逻辑电路构建一个8位串行可控加减法电路。

**二、实验方案设计**

8位可控加减法电路的设计方案如下：

（1）输入：两输入数X，Y；加减控制信号Sub

（2）输出：运算结果输出S，进位输出Cout，有符号运算溢出位OF

（3）所需器件：全加器、异或门等

（4）电路引脚：



（5）可控加减法的逻辑设计

·加法部分：利用8个1位全加器串行实现；

·减法部分：根据“减去一个数等于加上这个数的相反数的补码”的原理，通过异或门对减数Y进行按位取反；

·具体过程：

Sub=0时，作加法操作：0和其他数异或得到该数本身，然后通过全加器FA执行加法运算。Sub=1时，作减法操作：1和其他数异或即对该数取反，然后通过全加器FA执行减法运算。注意，需要对加法器FA进行+1操作。

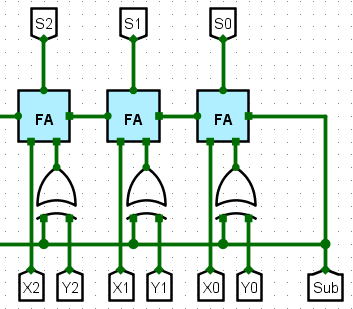
还要进行运算的有符号溢出检测（OF），可以通过最高位进位与符号位进位进行异或，相同为0，表示无溢出；不同为1，表示溢出。

**三、实验步骤**

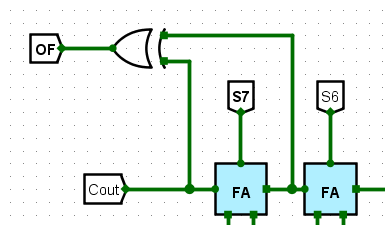
（1）在Logisim中打开alu.circ文件，选择8位可控加减法器子电路；

（2）准备8组封装好的全加器FA和异或门，对X、Y的各位数字进行分类排序，从左到右依次为7~0位；

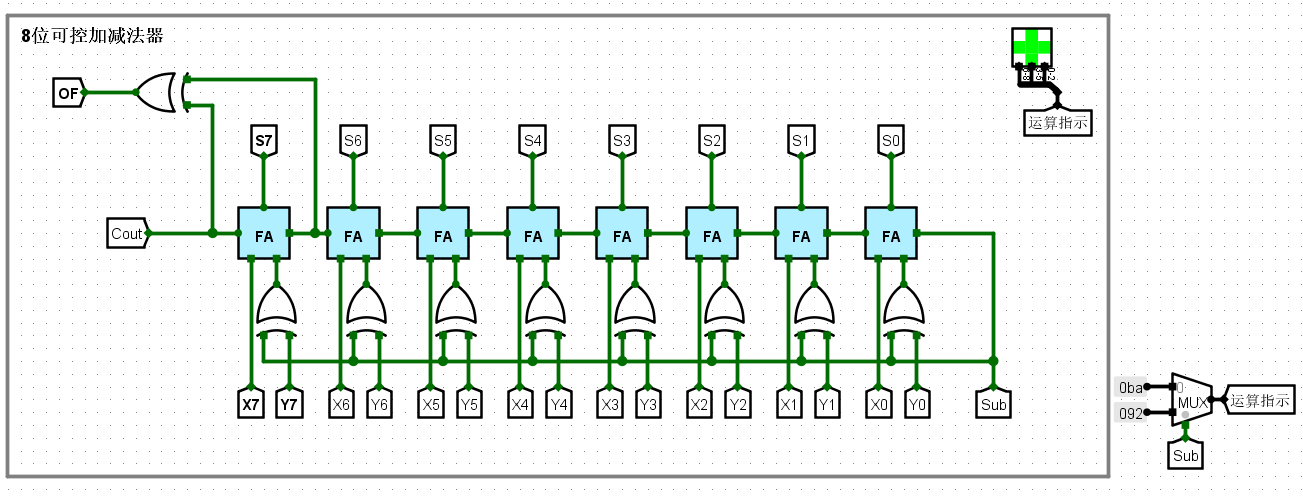
（3）将所有低位运算结果引脚连接全加器的输出端，X的位连接到FA的第一个输入端，Y的位与Sub加减控制信号进行异或以后接入FA的第二个输入端，同时也将Sub信号与全加器低位进位输入端相联，高位进位输出端与高一位的全加器低位进位输入端相联；



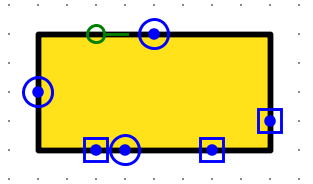
（4）将OF与最高位符号位进位和次高位进位异或得到的结果相连；



（5）最终得到的8位可控加减法器的电路图如下：



（6）将子电路封装为如下形式：



（7）进行电路测试：

·自动测试：

利用给出的测试电路，自定义X、Y、Sub值，进行8位可控加减法器的自动测试。

·平台测评：

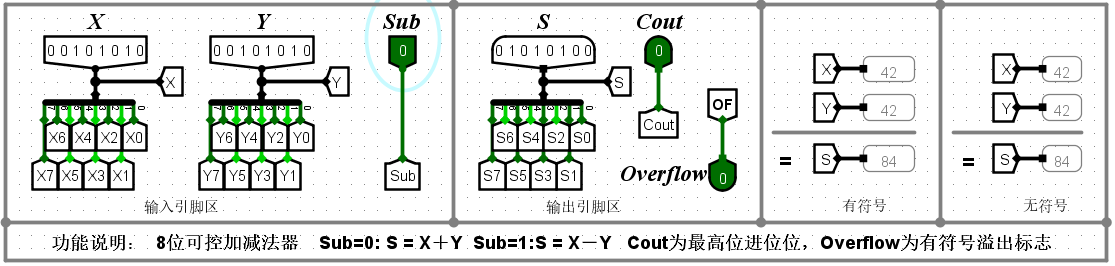
自动测试通过以后，利用记事本打开alu.circ文件，将所有文字信息复制粘贴到Educoder平台代码区域，点击评测按钮进行测试。

**四、实验结果测试与分析**

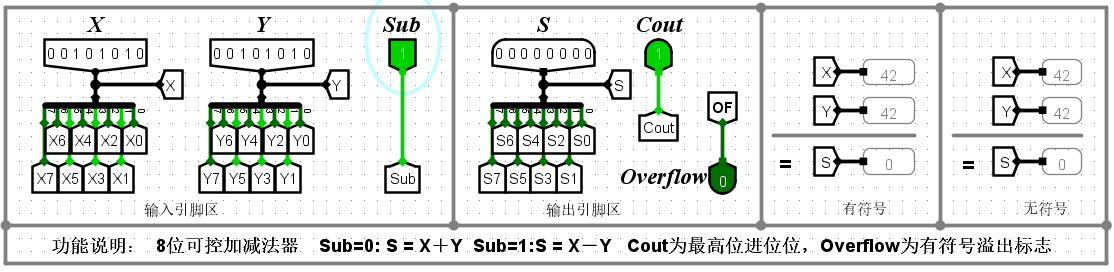
（1）自动测试结果如下：

自定义X = 00101010，Y = 00101010，即X=Y；

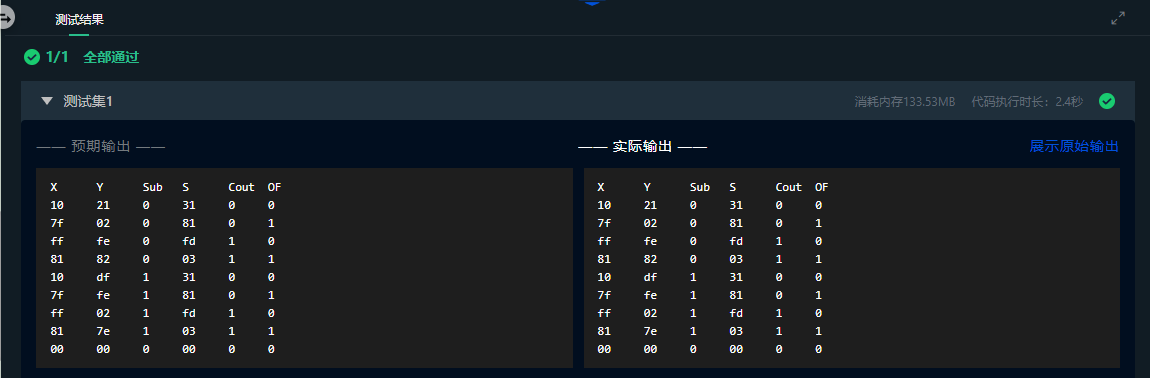
Sub=0时，加法：



Sub=1时，减法：



（2）平台测试结果如下：



综上，本实验测试结果为通过，无故障与报错。

本实验的关键点在于：正确设计可控加减法器的加减法逻辑，利用全加器作为基本部件，结合异或门的使用实现可控的加减法功能，注意进位信息。

**子实验2：CLA182四位先行进位电路设计**

姓名： 刘子言 学号： 20002462 班级： 计203 指导教师(签章) 张欢欢

成绩:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**一、实验设计要求**

（1）掌握快速加法器中先行进位的设计原理，并能利用相关知识设计4位先行进位电路。

（2）本实验的传递函数P采用异或逻辑关系实现。

**二、实验方案设计**

CLA182四位先行进位电路的设计方案如下：

（1）输入：进位输入Cin，进位生成函数Gi，传递函数Pi

（2）输出：进位输出Ci，成组进位生成函数G，成组进位传递函数P

（3）所需器件：基本逻辑门电路等

（4）电路引脚：

（5）先行进位加法器的设计逻辑



4位先行进位加法器表示进位输出，并不依赖于低位的进位，而是与最低位的进位输入有关；

根据先行进位加法器的原理，可以设计出以下逻辑表达式：

C4 = P4 P3 P2 P1 Cin + P4 P3 P2 G1 + P4 P3 G2 + P4 G3 + G4

C3 = P3 P2 P1 Cin + P3 P2 G1 + P3 G2 + G3

C2 = P2 P1 Cin + P2 G1 + G2

C1 = P1 Cin + G1

G = P4 P3 P2 G1 + P4 P3 G2 + P4 G3 + G4

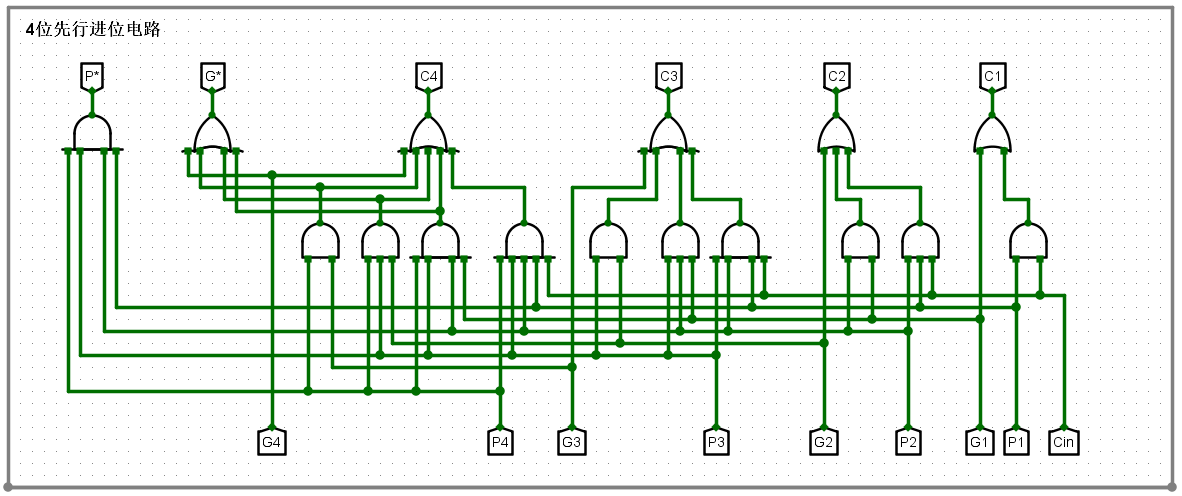
P = P4 P3 P2 P1

**三、实验步骤**

（1）在Logisim中打开alu.circ文件，选择4位先行进位74182子电路；

（2）根据C1、C2、C3、C4进位项的表达式（只与P\*、G\*、Cin相关），将与门、或门及其他各项引脚连接；

（3）最终设计出的4位先行进位的电路图如下；

（4）进行电路测试

·自动测试：

利用给出的测试电路，自定义输入值，进行4位先行进位的自动测试。

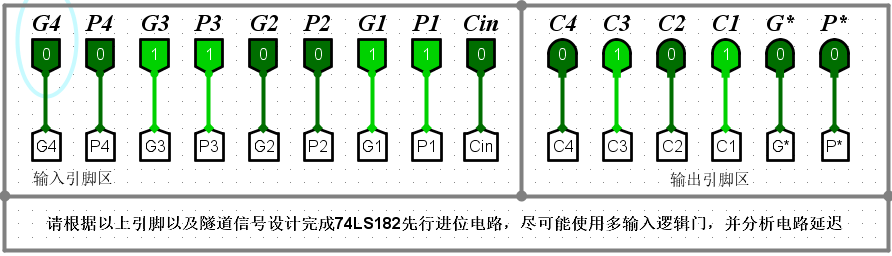
·平台测评：

自动测试通过以后，利用记事本打开alu.circ文件，将所有文字信息复制粘贴到Educoder平台代码区域，点击评测按钮进行测试。

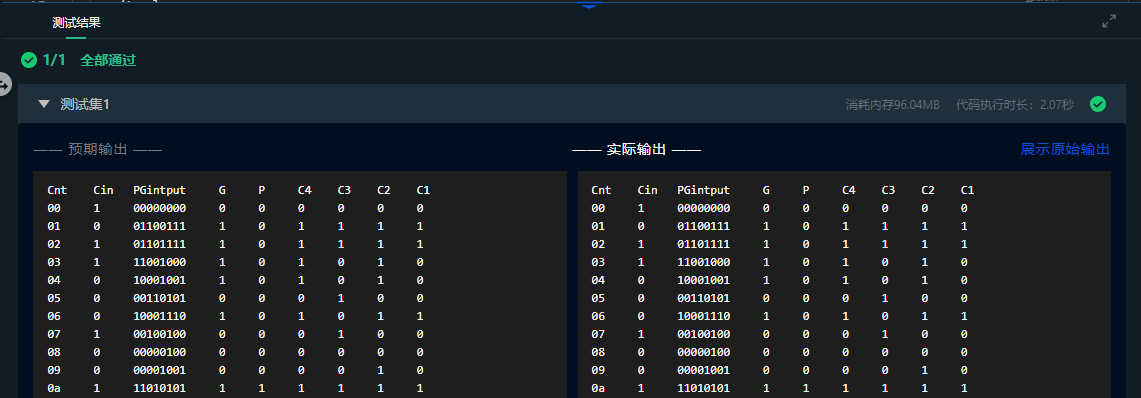
**四、实验结果测试与分析**

（1）自动测试结果如下：

自定义Pi、Gi的值进行测试，观察进位信号Ci的输出；



（2）平台测试的部分结果如下：



综上，本实验测试结果为通过，无故障与报错。

本实验的关键点在于：正确设计4位先行进位的各个输出的逻辑表达式，并依照逻辑表达式完成电路的设计。

**子实验3：4位快速加法器设计**

姓名： 刘子言 学号： 20002462 班级： 计203 指导教师(签章) 张欢欢

成绩:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**一、实验设计要求**

（1）掌握快速加法器中先行进位的原理，4位快速加法器运行的原理；

（2）利用上一个子实验已经设计好的4位先行进位电路构造4位快速加法器，并分析对应电路的时间延迟。

**二、实验方案设计**

在前一个子实验已经设计好的4位先行进位电路的基础之上，构造4位快速加法器。4位快速加法器的设计方案如下：

（1）输入：四位相加数X、Y，进位输入C0

（2）输出：和数输出S，进位输出C，4位成组进位生成函数G，4位成组进位传递函数P

（3）所需器件：CLA74182先行进位器，基本逻辑门电路等

（4）电路引脚：



（5）快速加法器的设计逻辑

根据快速加法器的原理，可以得到输出信号的逻辑表达式，经过相应的化简整理可以得到如下的逻辑表达式：

S4 = X4⊕Y4⊕C3 = P4⊕C3

S3 = X3⊕Y3⊕C2 = P3⊕C2

S2 = X2⊕Y2⊕C1 = P2⊕C1

S1 = X1⊕Y1⊕C0 = P1⊕C0

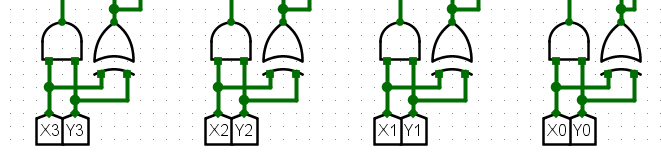
Gi = Xi Yi

Pi = Xi⊕Yi

**三、实验步骤**

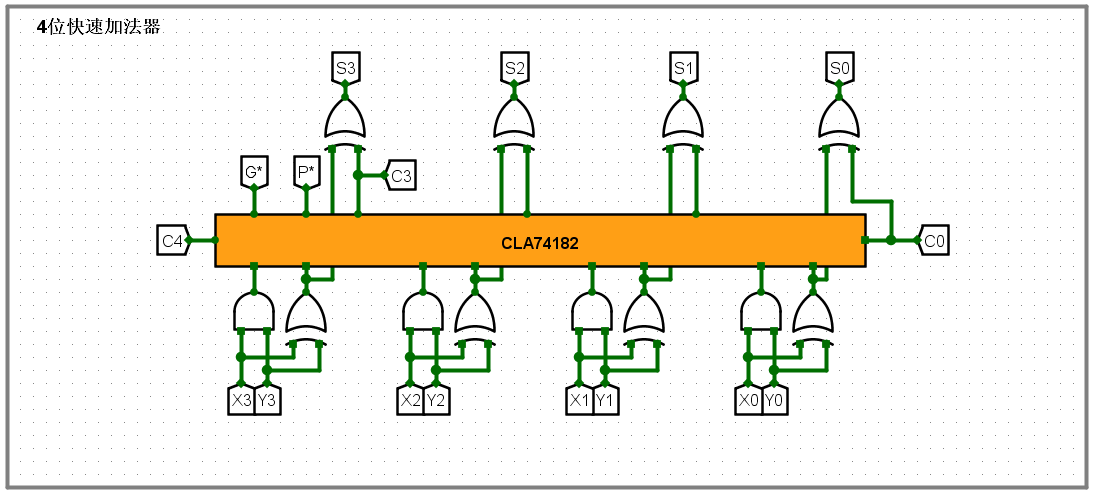
（1）在Logisim中打开alu.circ文件，选择4位快速加法器子电路；

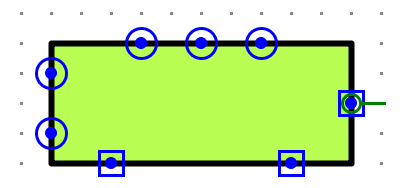
（2）根据成组生成函数和成组传递函数的表达式，利用与门和异或门设计Gi、Pi电路；



（3）根据各个和数输出S的逻辑表达式，基于上个实验已经封装好的CLA74182先行进位器，将各个引脚及逻辑门连接完整；

（4）最终设计出的4位先行进位的电路图如下；

（5）将子电路封装为如下形式：



（6）进行电路测试

·自动测试：

利用给出的测试电路，自定义输入，进行4位快速加法器的自动测试。

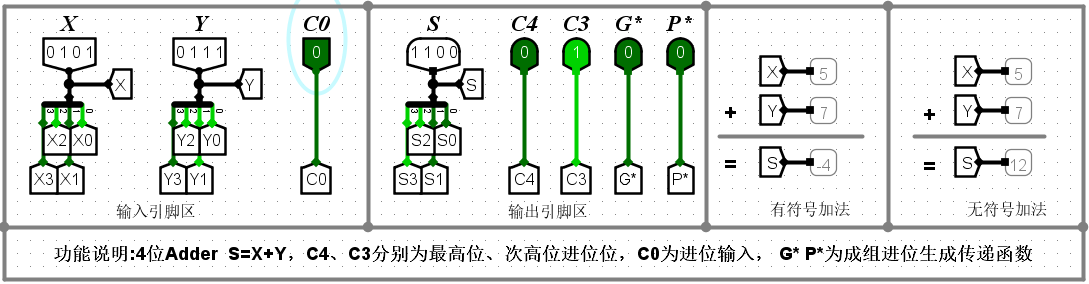
·平台测评：

自动测试通过以后，利用记事本打开alu.circ文件，将所有文字信息复制粘贴到Educoder平台代码区域，点击评测按钮进行测试。

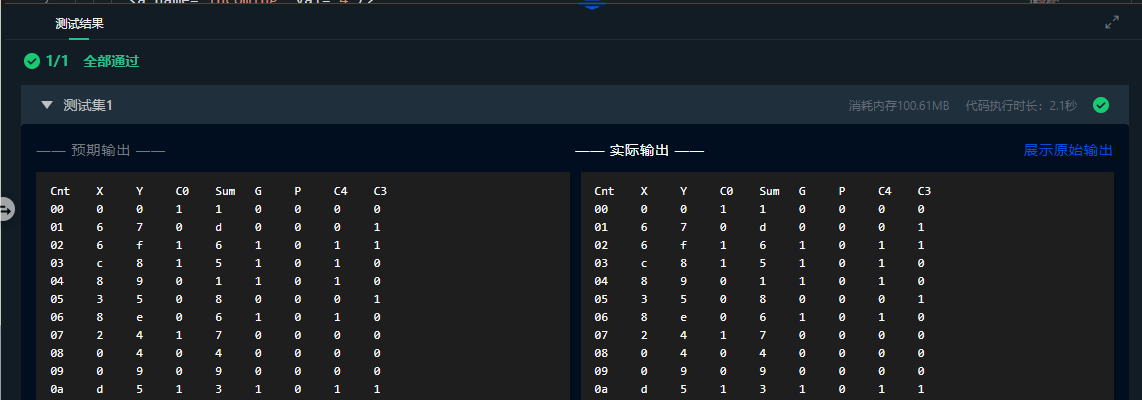
**四、实验结果测试与分析**

（1）自动测试结果如下：

自定义X、Y、C0的值进行测试，观察S、C4、C3、G\*、P\*的输出；



（2）平台测试的部分结果如下：



综上，本实验测试结果为通过，无故障与报错。

·实验关键点在于：

正确设计4位快速加法器的各个输出的逻辑表达式，基于上一个子实验设计的4位先行进位电路，完成快速加法器电路的设计。

·4位快速加法器**时间延迟**分析：

4位快速加法器也称并行加法器，可由4位先行进位电路加上生成G、P的与门异或门电路，再加上4个异或门构成。

运行过程中生成的所有Gi、Pi输入需要一级异或门电路时间延迟3T，先行进位电路的时间延迟为2级门电路的时间延迟2T；当所有进位信号产生后，再增加一级异或门时间延迟3T，即可得到所有位的和数，从而完成多位加法运算。

因此4位快速加法器的时间延迟为8T，相比4位串行加法器的时间延迟(2n+4)T=12T，其性能提升了1.5倍。

**子实验4：16位快速加法器设计**

姓名： 刘子言 学号： 20002462 班级： 计203 指导教师(签章) 张欢欢

成绩:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**一、实验设计要求**

（1）理解成组进位产生函数、成组进位传递函数的概念；

（2）理解电路并行的概念，利用Logisim平台中前述子实验封装好的4位先行进位电路以及4位快速加法器电路，构建16位快速加法器，并能利用相关知识分析对应电路的时间延迟。

**二、实验方案设计**

在前述子实验已封装好的4位先行进位电路、4位快速加法器电路的基础之上，构造16位快速加法器。16位快速加法器的设计方案如下：

（1）输入：16位相加数X、Y，进位输入C0

（2）输出：和数输出S，进位输出C，16位成组进位生成函数G，16位成组进位传递函数P

（3）所需器件：CLA74182先行进位器，4位快速加法器，基本逻辑门电路等

（4）电路引脚：



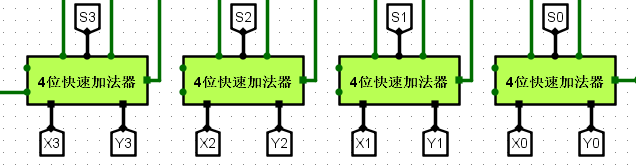
（5）16位快速加法器的设计逻辑

16位快速加法器需要将加数的16个数据位分为4组，每组包含4位数据；利用四个4位快速加法器和一个CLA74128先行进位电路，得到每一组的进位输入，即C0、C4、C8和C12；最后，利用4位快速加法器依次计算得到运算结果S3、S2、S1和S0。

**三、实验步骤**

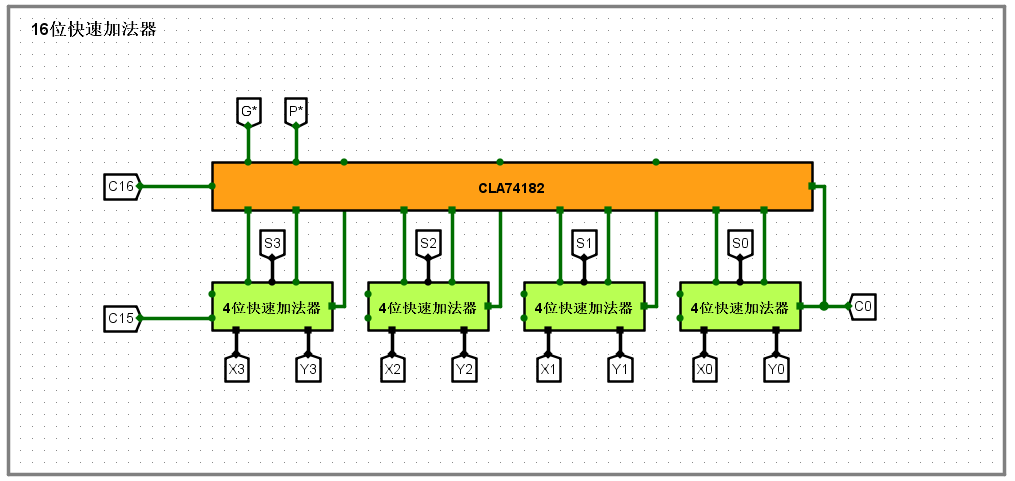
（1）在Logisim中打开alu.circ文件，选择16位快速加法器子电路；

（2）先将两个加数分成四组，分别与四个4位快速加法器相连；

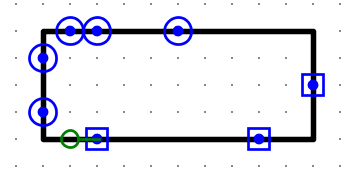


（3）再将上述四组器件依次与CLA74128先行进位电路相连；注意C0需要同时连接CLA74128和第一个4位快速加法器的输入端；

（4）最终设计得到的16位快速加法器的电路图如下：



（5）将子电路封装为如下形式：



（6）进行电路测试

·自动测试：

利用给出的测试电路，自定义输入，进行16位快速加法器的自动测试。

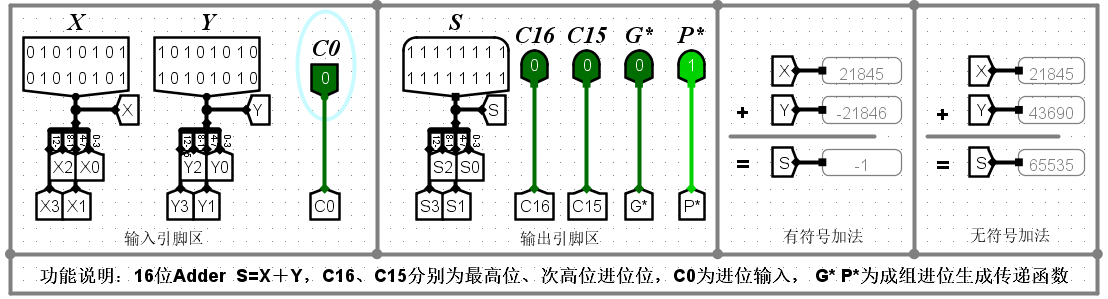
·平台测评：

自动测试通过以后，利用记事本打开alu.circ文件，将所有文字信息复制粘贴到Educoder平台代码区域，点击评测按钮进行测试。

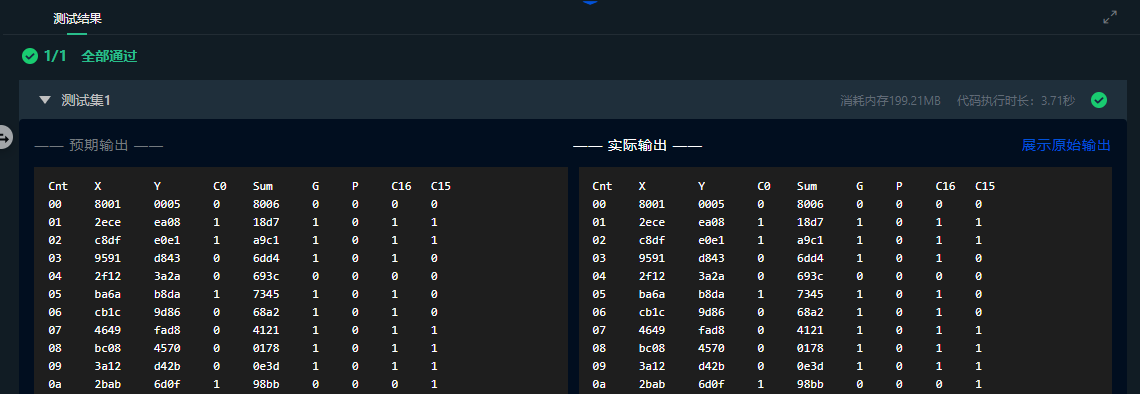
**四、实验结果测试与分析**

（1）自动测试结果如下：

自定义X、Y、C0的值进行测试，观察S、C16、C15、G\*、P\*的输出，右侧显示有符号和无符号加法结果；



1. 平台测试的部分结果如下：



综上，本实验测试结果为通过，无故障与报错。

·实验关键点在于：

将16位数据划分为4组，正确利用前述子实验设计的4位快速加法器和CLA74128先行进位电路，完成16位快速加法器电路的设计。

·16位快速加法器**时间延迟**分析：

本实验设计的是16位组内并行、组间并行加法器，由四个4位快速加法器和一个可级联CLA74128先行进位电路实现。

运行过程中所有4位快速加法器产生成组生成的进位函数G\*、P\*的时间延迟为5T，而先行进位电路的时间延迟为2T，所以生成C4、C8、C12、C16的时间延迟为7T。

当这些信号就绪以后，此时4位快速加法器内部的与门、异或门电路已经运算完毕，需要2T生成进位输出信号，需要5T输出和数。

因此，整个电路的关键时间延迟为和数信号的时间延迟12T，相比16位组内并行、组间串行加法器的时间延迟14T，其关键时间延迟减少了2T。

**子实验5：32位快速加法器设计**

姓名： 刘子言 学号： 20002462 班级： 计203 指导教师(签章) 张欢欢

成绩:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**一、实验设计要求**

（1）理解电路并行的概念；

（2）利用Logisim中前述子实验已经封装好的4位先行进位电路和16位快速加法器电路，构建32位快速加法器，并能利用相关知识分析对应电路的时间延迟。

**二、实验方案设计**

在前述子实验已封装好的4位先行进位电路、16位快速加法器电路的基础之上，构造32位快速加法器。32位快速加法器的设计方案如下：

（1）输入：32位相加数X、Y，进位输入C0

（2）输出：和数输出S，进位输出C，有符号加法运算溢出信号Overflow

（3）所需器件：CLA74182先行进位器，4位快速加法器，基本逻辑门电路等

（4）电路引脚：



（5）32位快速加法器的设计逻辑

32位快速加法器设计的可能方案有以下三种：

·方案1：2个16位加法器直接串联，C16信号采用下层的进位输出；

·方案2：2个16位加法器直接串联，C16进位输入采用上层的进位输出；

·方案3：在16位快速加法器的基础上再增加一级组间先行进位电路，类似64位快速加法器的方法；

本实验选择方案3：在16位快速加法器的基础上，再增加一级组间先行进位电路，构建32位快速加法器；以第一个CLA74128先行进位电路的最高位进位输出作为第二个CLA74128的进位输入；最后再利用八个4位快速加法器依次计算得到运算结果S7~S0。

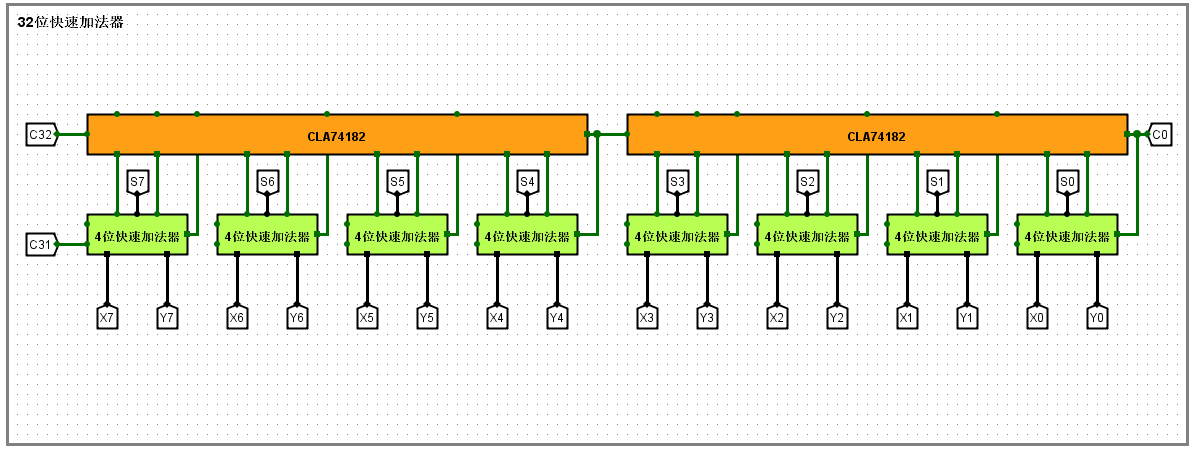
**三、实验步骤**

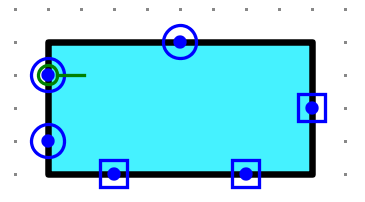
（1）在Logisim中打开alu.circ文件，选择32位快速加法器子电路；

（2）先利用上个子实验的分组分块方法，将32位数据分为8组，再将每4个为一大组，构建两个16位快速加法器；

（3）再将第一个CLA74128先行进位电路最高位进位输出，作为第二个CLA74128的进位输入；将C0连接到第一个CLA74128和4位快速加法器的进位输入；

（4）最终得到的32位快速加法器的电路图如下：

（5）将子电路封装为如下形式：



（6）进行电路测试

·自动测试：

利用给出的测试电路，自定义输入，进行32位快速加法器的自动测试。

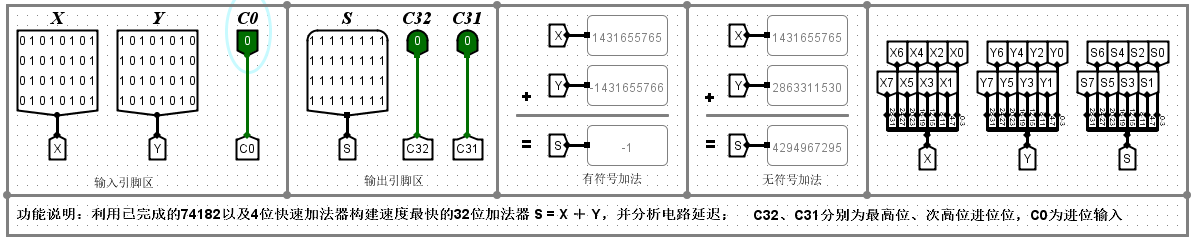
·平台测评：

自动测试通过以后，利用记事本打开alu.circ文件，将所有文字信息复制粘贴到Educoder平台代码区域，点击评测按钮进行测试。

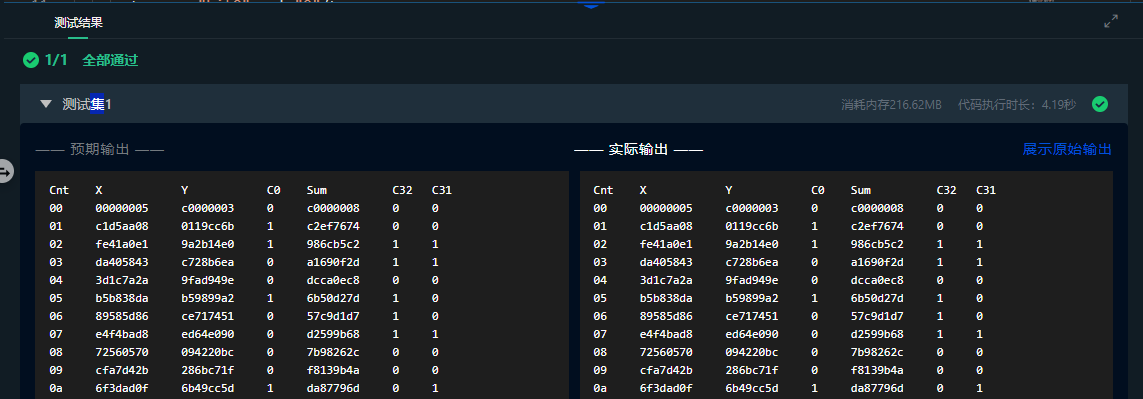
**四、实验结果测试与分析**

（1）自动测试结果如下：

自定义X、Y、C0的值进行测试，观察S、C32、C31的输出，中间显示了有符号和无符号加法的结果；



（2）平台测试的部分结果如下：



综上，本实验测试结果为通过，无故障与报错。

·实验关键点在于：

将32位数据划分为8组，4个一大组构建两个16位快速加法器；正确利用前述子实验中设计得到的16位快速加法器和CLA74128先行进位电路，完成32位快速加法器电路的设计。

·32位快速加法器**时间延迟**分析：

本实验选择将两个16位快速加法器采用组内并联、组间串联的方式构建32位快速加法器。

由上一个子实验分析可知，一个16位快速加法器产生最高位进位信息C16的时间为7T，所以与之串联的第二个16位快速加法器在7T时收到该进位信号后，其内部4位快速加法器的先行进位电路开始工作，时间延迟为2T，再经过2T延迟产生进位信息C20、C24、C28、C32；然后再经过2T（四位先行电路）+3T（异或门延迟）得到每一个4位快速加法器的和数。

所以，第二个16位快速加法器得到和数结果的总时间延迟，即32位快速加法器的总时间延迟为：7T+2T+2T+5T=16T。

如果同样采用这种方式，进一步级联构成64位快速加法器，则总时间延迟还是16T。目前，计算机内部常用的加法器普遍是采用类似的方法构建的。

**子实验6：MIPS运算器设计**

姓名： 刘子言 学号： 20002462 班级： 计203 指导教师(签章) 张欢欢

成绩:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**一、实验设计要求**

（1）理解算术逻辑运算单元ALU的基本构成；

（2）掌握Logisim中各种运算组件的使用方法，能利用前述子实验完成的32位加法器、Logisim中的运算组件，来构造MIPS运算器（为32位算术逻辑运算单元），构造时不使用Logisim系统自带的加法、减法器。

（3）实验完成后，对封装好的ALU进行测试，并分析该运算器的优缺点。

**二、实验方案设计**

在前述子实验已封装好的32位快速加法器电路的基础之上，结合Logisim平台中现有运算部件，构造MIPS运算器。MIPS运算器的设计方案如下：

（1）输入：32位相加数X、Y，进位输入C0

（2）输出：和数输出S，进位输出C，有符号加法运算溢出信号Overflow

（3）实现功能：实现算术加、减、乘、除运算，逻辑与、或、非、异或运算，逻辑左移、逻辑右移、算术右移运算；实现常用程序状态标志的功能（有符号溢出OF、无符号溢出UOF，结果相等Equal）

（4）所需器件：32位快速加法器，Logisim平台中其他现有运算部件

（5）电路引脚：



（6）MIPS运算器设计方法

·运算实现

将运算器整体分为多个功能模块来逐一设计，以此简化硬件线路的实现。每次运算时，先通过ALU\_OP信号选择对应的运算方式进行运算，然后通过多路选择器，将所选运算方式对应的结果给Result，将乘法运算时高位结果或除法运算时余数给Result2，其余情况下Result2的输出为0。

·溢出判断

判断加法溢出时，UOF加连C32引脚，OF加连C31引脚；判断减法溢出时，可以利用比较器，通过对X和Y的比较得到UOF减，通过加法器引脚得到OF减。

（7）运算器功能码的功能对照表如下：

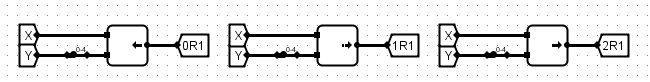


**三、实验步骤**

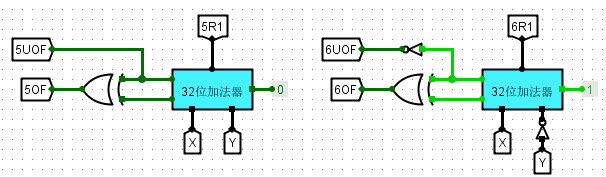
（1）在Logisim中打开alu.circ文件，选择算术逻辑运算单元子电路；

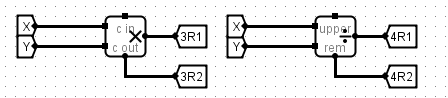
（2）利用前述子实验中已经设计封装好的32位加法器，以及Logisim平台中的现有运算部件，实现基本运算的逻辑电路；

·移位运算（逻辑左移、运算右移、逻辑右移）

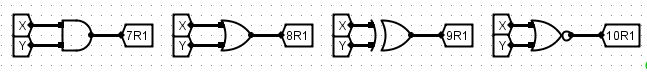


·加减运算（减运算在加运算基础上控制常量设为1，Y加上非门）

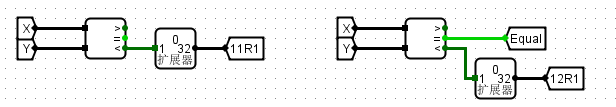
·乘除运算（乘法、除法器实现）



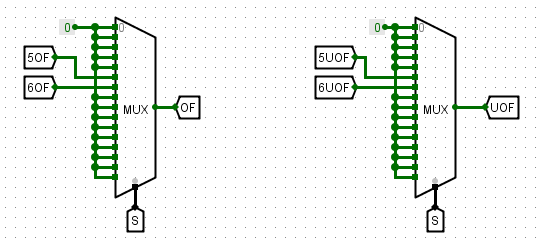
·逻辑与、或、非、异或运算



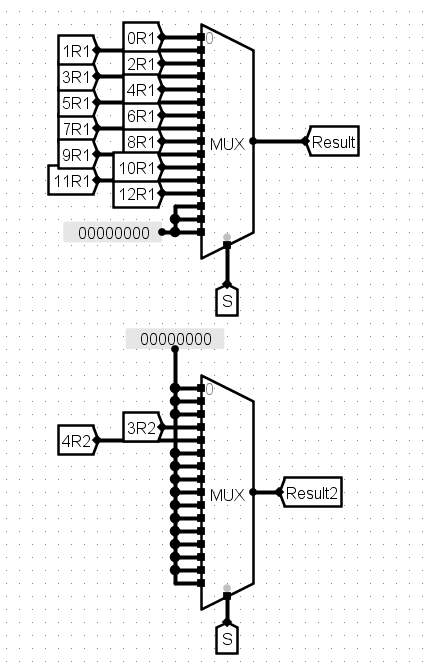
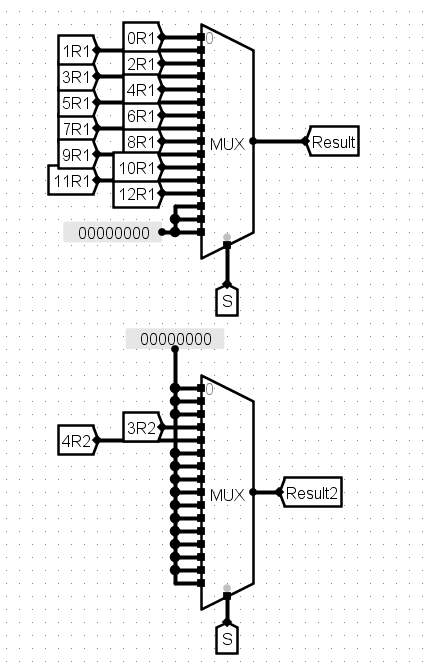
·比较结果运算（需要进行位扩展，以保证数据位的统一）



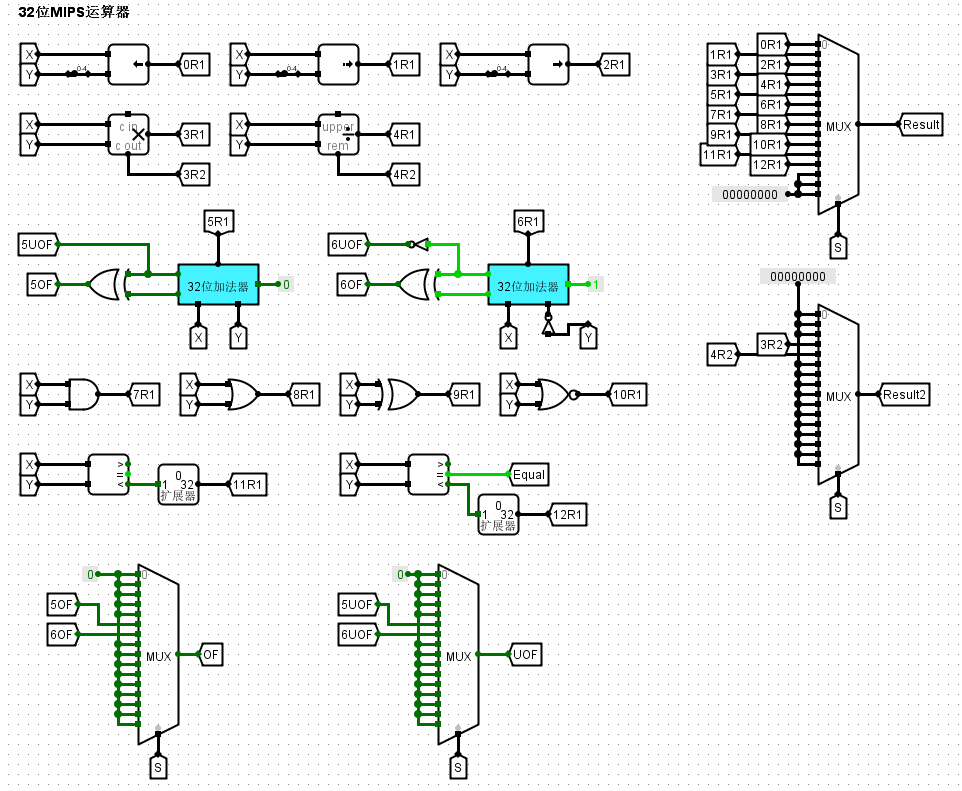
（3）再利用多路选择器，设计加减溢出判断标记的电路图；



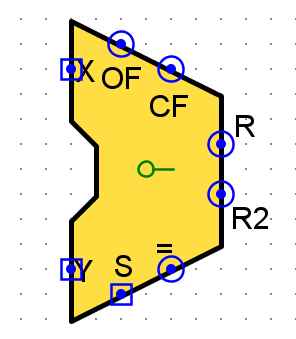
（4）最后再利用多路选择器，将上述各种运算结果合并，以ALU\_OP（S）为选择控制信号，实现最终的结果输出；



（5）将上述部件组合，最终得到的MIPS运算器的电路图如下：



（6）将子电路封装为如下形式：



（6）进行电路测试

·自动测试：

利用给出的测试电路进行MIPS运算器的自动测试，记录评分。

·平台测评：

自动测试通过以后，利用记事本打开alu.circ文件，将所有文字信息复制粘贴到Educoder平台代码区域，点击评测按钮进行测试。

**四、故障与调试**

（1）故障一

·故障现象

ALU\_OP = 6时，运算结果报错；

·原因分析

ALU\_OP = 6为减法运算，可能是减法逻辑电路的设计出现问题；

·解决方案

未将32位加法器的右端的常量输入改为1，将0改为1以后，运算结果正确。

（2）故障二

·故障现象

减法运算时，溢出标记出现高阻态；

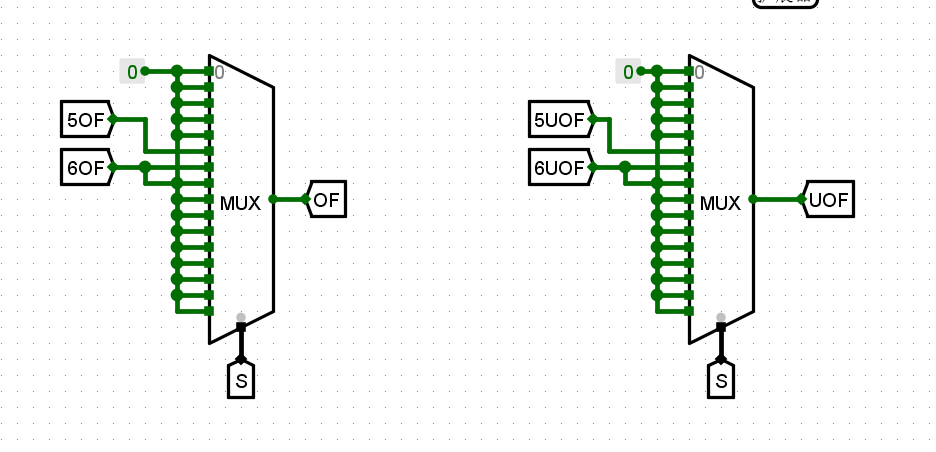


·原因分析

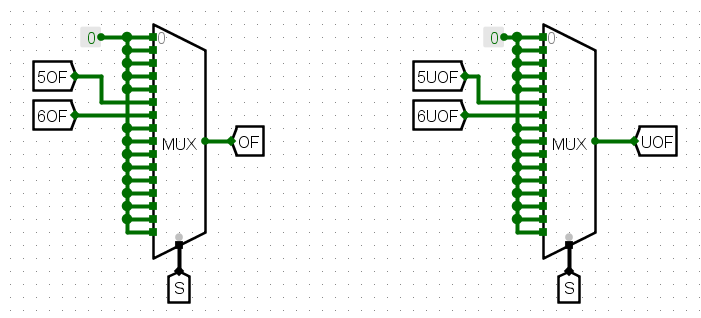
加减溢出判断标记的电路图设计出现问题；初步判断是6OP以及6UOP处连线出现错误，才会导致减法运算时的溢出标记出现高阻态；

·解决方案

检查6OP、6UOP处溢出判断标记电路图的连线，发现6OP、6UOP隧道后的连线多出了一条分支，去掉多余连线以后，测试正常。

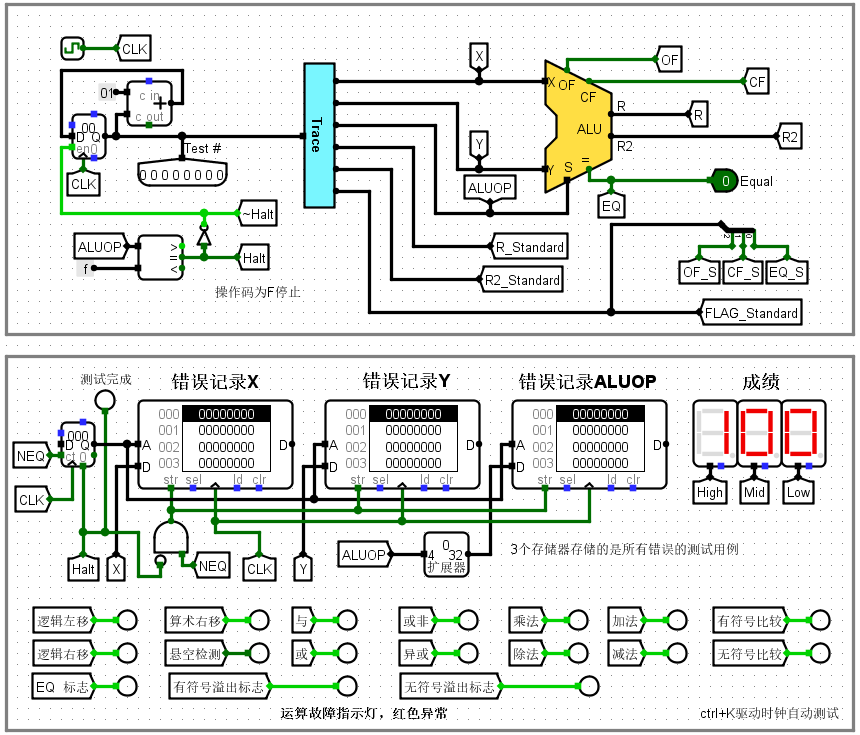


去掉多余的连线，改为以下电路：

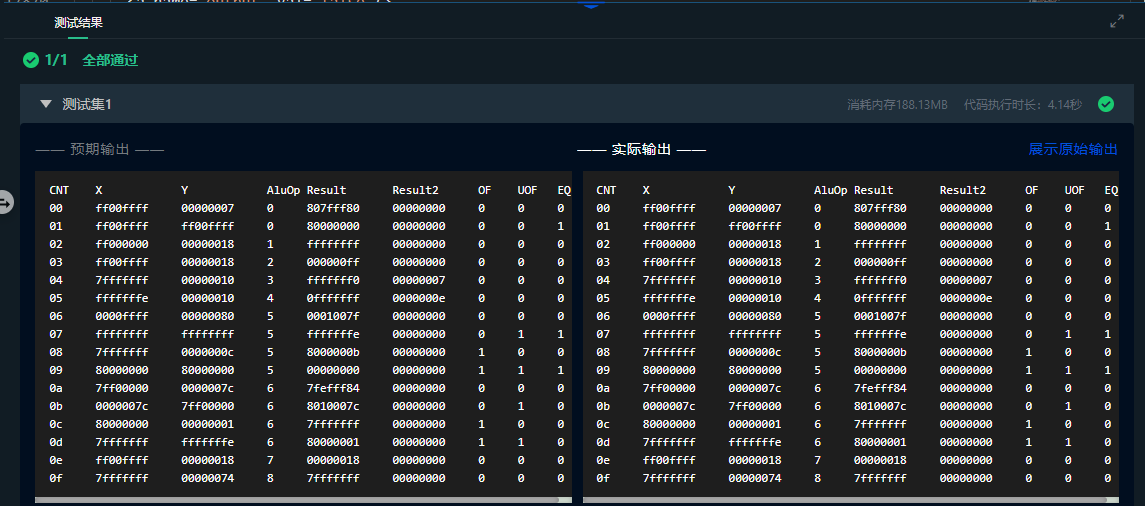


**五、实验结果测试与分析**

（1）自动测试的结果如下，测试评分为100分：



（2）平台测试结果如下：



综上，本实验测试的最终结果为通过。

本实验的关键点在于：先将MIPS运算器的电路设计“化整为零”，针对不同功能的运算逐个设计，并利用多路选择器正确设计加减法溢出标记的电路；最后再利用多路选择器对不同运算功能的运算结果进行“汇总”，以ALU\_OP为选择信号，得到最终的运算结果。

**实验三：存储系统设计**

**子实验1：MIPS寄存器文件设计**

姓名： 刘子言 学号： 20002462 班级： 计203 指导教师(签章) 张欢欢

成绩:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**一、实验设计要求**

（1）了解MIPS寄存器文件基本概念；

（2）进一步熟悉多路选择器、译码器、解复用器等Logisim组件的使用，并利用相关组件构建MIPS寄存器文件。

**二、实验方案设计**

MIPS寄存器文件又称为MIPS寄存器组或MIPS寄存器堆。利用Logisim构建一个简化的MIPS寄存器文件，设计方案如下：

1. 寄存器个数：包含4个32位寄存器
2. 所需器件：多路选择器、译码器、解复用器、寄存器等
3. 电路引脚：
4. 实现功能：对所有寄存器的写入、读取
5. 实验简化：

为了减少实验的绘图工作量，该实验不要求设计32个寄存器，而是选择利用一组分线器，将5位的读寄存器地址编号进行分线。由于仅仅使用了5位地址中的低两位，所以最终只需实现0~3号4个寄存器即可。

由于0号寄存器的值恒为0，所以在设计时不需要一个真正的寄存器，可以运用常量进行设计。

**三、实验步骤**

（1）在Logisim中打开已下载的storage.circ文件，选择MIPS Regfile子电路；

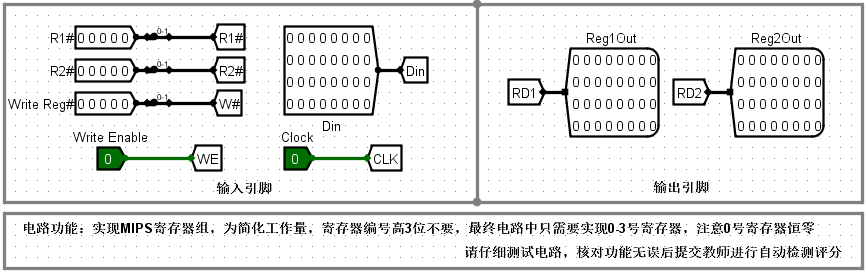
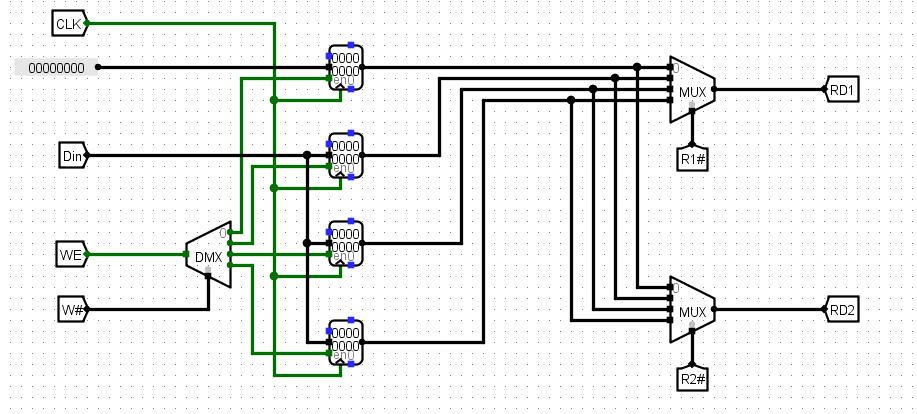
（2）利用分线器，将5位读寄存器地址编号的低两位，作为所选的寄存器编号0~3，并赋给R1#、R2#、W#；

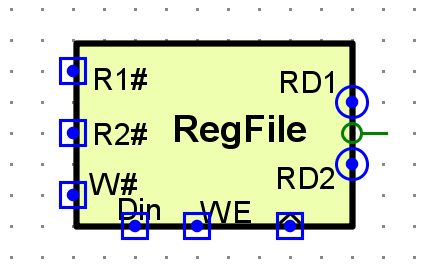
（3）利用解复用器控制写使能信号WE以及写入寄存器编号W#；

（4）结合时钟信号CLK，以及解复用器输出端的写入信息，将写入数据Din存入4个寄存器中；注意0号寄存器值恒为0，所以利用常数0作为写入数据；

（5）再利用两个多路选择器，四个寄存器数据作为备选数据，由R1#、R2#提供所选寄存器的编号，然后读出对应数据，将读出的两个寄存器的值分别给RD1、RD2进行输出；

（6）综上设计得到MIPS寄存器文件的电路图如下所示：

（7）将子电路封装为如下形式：

（8）进行电路测试：

·自动测试

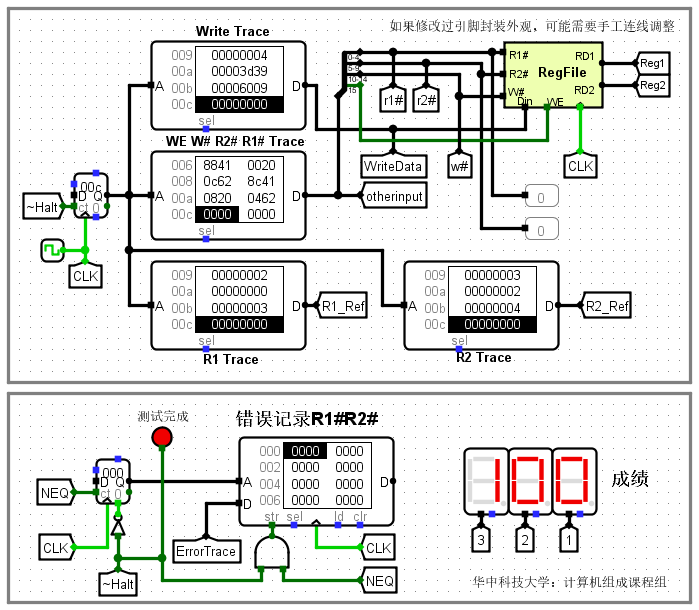
先在MIPS Regfile测试子电路中进行测试。测试思路：先对某一寄存器进行写入，再将所有寄存器中的值读出，与正确答案进行对比；再写入下一个寄存器，读出所有并比较；以此类推，完成所有寄存器的写入和读出比较。在测试时需要将测试频率调整到最快（4KHz），并用Ctrl+k快捷键控制时钟进行自动测试。

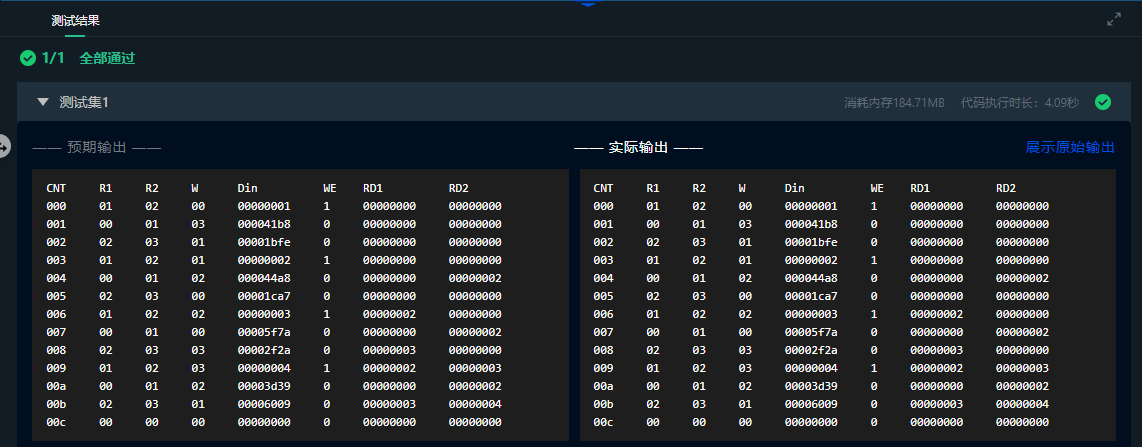
·平台测评：

自动测试结果满足实验要求后，再利用记事本打开storage.circ文件，将所有文字信息复制粘贴到Educoder平台代码区域，点击评测按钮进行测试。

**四、实验结果测试与分析**

（1）自动测试的结果如下，测评分数为100分：

（2）平台测试结果如下：



综上，本实验测试结果为通过，无故障和报错。

本实验的关键点在于：正确使用多路选择器、解复用器等器件，处理寄存器文件的读写信号，以及根据寄存器编号选择正确的寄存器读出和写入值；特别注意0号寄存器的设计。

本实验设计的MIPS寄存器文件经过优化以后，还可用于实验四中的MIPS CPU的设计。

**子实验2：直接相联cache设计**

姓名： 刘子言 学号： 20002462 班级： 计203 指导教师(签章) 张欢欢

成绩:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**一、实验设计要求**

（1）掌握cache实现的三个关键技术：数据查找，地址映射，替换算法；

（2）熟悉译码器、多路选择器、以及寄存器的使用，能根据直接映射的映射策略在Logisim平台中用数字逻辑电路实现cache机制。

**二、实验方案设计**

直接相联的cache的设计方案如下：

（1）cache槽：所设计的cache具有8个cache槽

（2）设计规格：主存地址16位，一个cache模块共包括8个cache行，每个数据块包含4个字节共32位数据。

（3）所用器件：译码器、多路选择器、编码器、寄存器、比较器等

（4）电路引脚：



（5）直接相联映射的逻辑设计：

在明确主存地址划分（标记字段tag为11位，行索引字段index为3位，块内偏移字段offset为2位）以后，cache需要实现一下逻辑：

**·数据查找逻辑**

对于直接相联映射，由于一个数据块只能对应到cache中的特定行，查找算法较为简单，只需查看tag和有效位，即可判断数据是否命中。

从硬件实现的角度来设计，需要一个行索引译码器和一个比较器：只需要将index字段输送到一个行索引译码器，就可以得到8个行索引译码输出，其中输出高电平的那一行就是当前主存地址对应的cache行；利用8个行译码输出信号，控制8个三态门，将所有行的有效位以及标记字段输出到一个比较器；根据译码器的定义，只有当前主存地址对应的那一行的数据才会输出，因此我们将当前行的有效位和1进行比较，同时将tag位和主存地址中的区地址进行比较，若两个都相等表示命中，否则表示缺失。

**·数据访问逻辑**

用上述的8路译码输出信号控制cache块的数据输出。将所有cache行的数据块都通过三态门输出到数据总线（槽数据）上，三态门的控制端就是这8个译码输出信号。

命中时，命中行的数据会输出到数据总线上。接下来具体选择数据块中的哪一个字，由主存地址中的offset字段决定，硬件上可以使用多路选择器实现。注意，命中信号要直接连接到多路选择器的使能端，缺失时多路选择器的输出为高阻态。

**·数据载入逻辑**

将主存地址访问序列先进行地址划分，将区地址字段直接送入比较器，行索引字段送入行地址译码器，选中输出为高电平的那一行，控制该行的有效位和标记位通过三态门输出到比较器，与区地址和有效位1进行比较，判断比较结果是否相等。

若结果不相等，则数据发生缺失，需要对新的数据进行载入：需要从二级存储器（内存）中载入该主存地址所在的数据块。由于内存的DRAM比cache的SRAM要慢很多，所以这个数据块的载入过程比较漫长，在这段时间内，CPU会死锁等待相应数据块的载入，此时CPU不能进行任何其他操作。

数据载入过程一般需要若干个时钟周期，对应的数据块会载入到之前选中的行中；同时，会对选中的行的标记位进行更新，并将有效位设置为1。

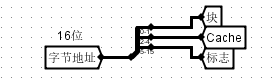
这些字段更新完毕以后，比较器结果会变为相等，即数据最终会变为命中。然后将该行数据块中的数据输出到数据总线，再由offset字段进一步选择具体的字，进行最后的输出。

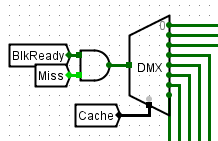
若起初比较时有效位为1，但标记位与区地址字段并不相同，则接下来应该进行替换操作：将主存地址访问序列对应内存中的数据块载入选中的这一行替换原来的数据，同时更新对应的标记字段。

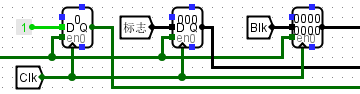
**三、实验步骤**

（1）在Logisim中打开已下载的storage.circ文件，选择Cache（直接相联）子电路；

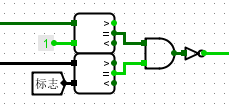
（2）先利用分线器对主存地址进行地址划分；

（3）再利用解复用器构造行索引译码器，对index字段进行译码，选定某一行；

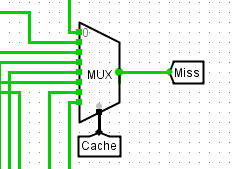
（4）每行中对应地址三个寄存器存储，第一个寄存器存储有效位（数据位宽为1），第二个寄存器存储标记位（数据位宽为11），第三个寄存器存储块内数据（4个字节，数据位宽为32）；8行共使用24个寄存器；



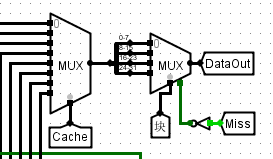
（5）利用比较器和基本逻辑门构造综合比较器，判断有效位是否为1，且标记位与区地址字段是否相等；



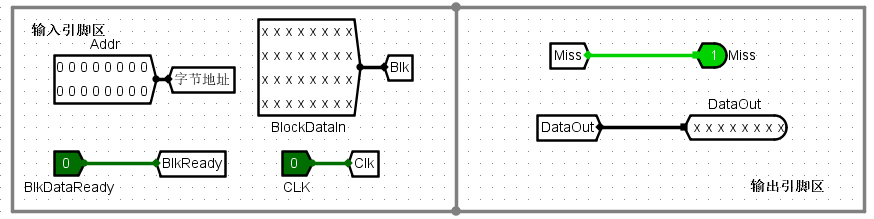
（6）再结合多路选择器对判断结果进行汇总，用于判断结果是命中、缺失还是替换；

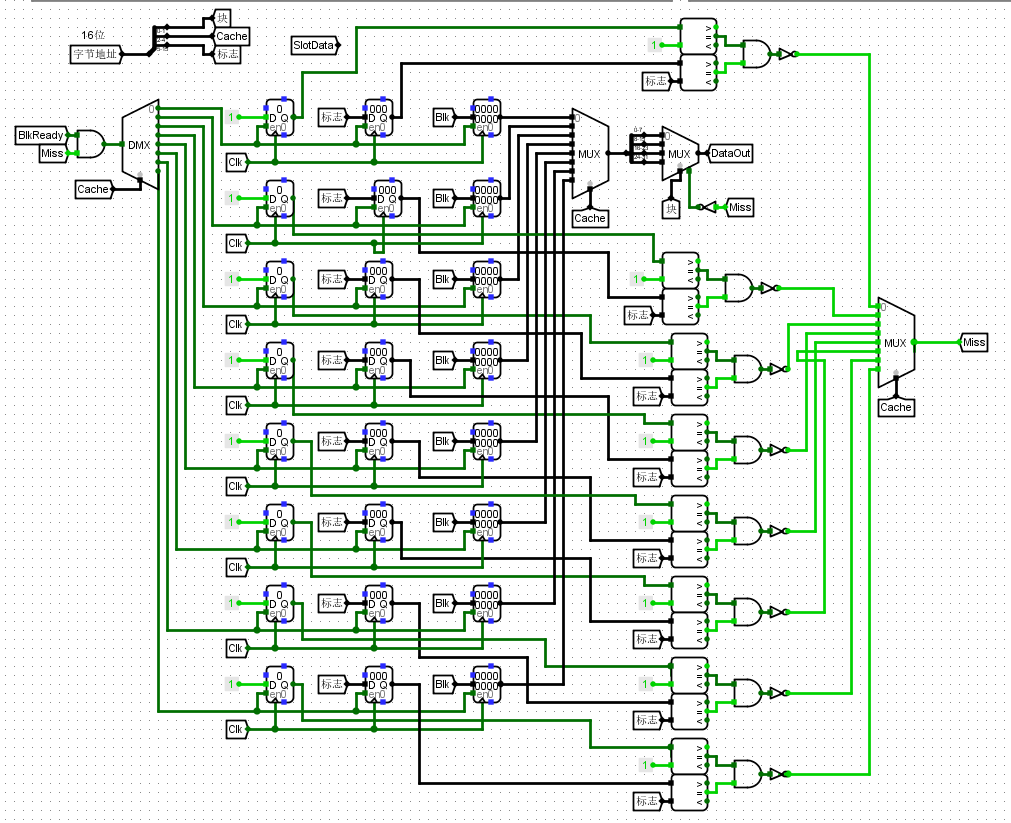


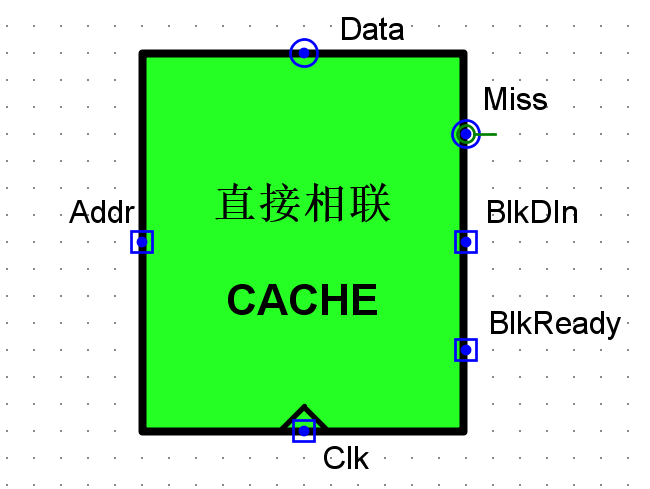
（7）最后利用多路选择器，结合offset字段，实现对数据块中具体字的选择和输出；



（8）综上设计得到MIPS寄存器文件的电路图如下所示：



（9）将子电路封装为如下形式：



（10）进行电路测试：

·自动测试

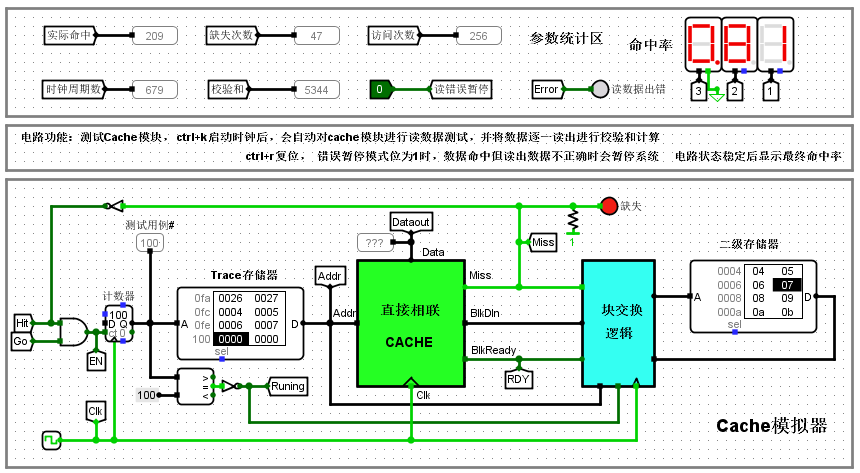
先在Cache自动测试子电路中进行测试。测试思路：电路左侧计数器与存储器部分会在时钟驱动下逐一生成地址访问序列给cache模块；计数器模块的使能端受命中信号驱动，缺失时使能端无效，计数器不计数，等待系统将待请求数据所在块从二级存储器中调度到cache后才能继续计数。cache与二级存储器之间通过块交换逻辑实现数据块交换，由于二级存储器相比cache慢很多，所以一次块交换需要多个时钟周期才能完成；cache 模块判断数据块准备好的逻辑是blkready信号有效，该信号有效且时钟到来时，cache将块数据从 BlkDin端口一次性载入到对应cache行缓冲区中，此时cache数据命中，直接输出请求数据，解锁计数器使能端，继续访问下一个地址。自动测试电路会逐一取出trace存储器中的主存地址去访问存储系统，并逐一将数据从cache模块取出送校验和计算电路计算校验和，计数器值为256时会停止电路运行，此时所有存储访问的cache命中率将会在右上角LED数码管显示。

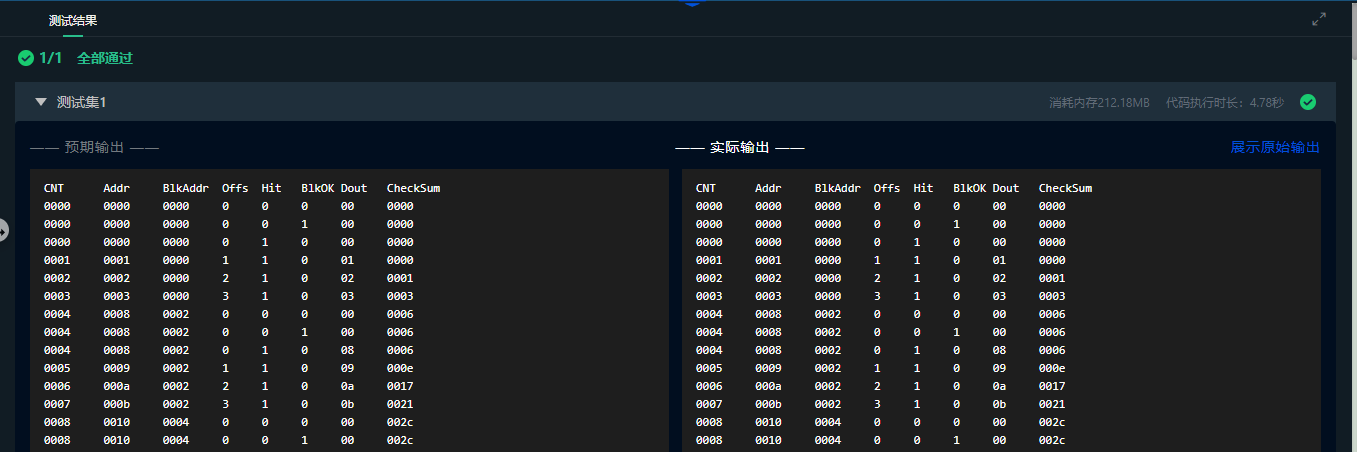
·平台测评

自动测试结果满足实验要求后，再利用记事本打开storage.circ文件，将所有文字信息复制粘贴到Educoder平台代码区域，点击评测按钮进行测试。

**四、实验结果测试与分析**

（1）自动测试的结果如下，访问次数256次，命中率为0.81：

（2）平台测试的部分结果如下：



综上，本实验测试结果为通过，无故障和报错。

本实验的关键点在于：正确使用译码器、多路选择器、解复用器等器件，能够判断命中、缺失、替换三种情况，实现直接相联cache的数据查找、访问、载入三大逻辑。

cache读命中的时间一般最短，现在的CPU中取指令、取操作数都是从cache得到，通常在一个时钟周期内完成，因此cache的性能直接决定了CPU的性能。

此外，cache还可能在没有填满的状态下就发生替换导致数据的淘汰，所以直接相联映射的cache利用率并不是很高，命中率也较差，一般在cache容量较大的情况下使用。

**实验四：MIPS CPU设计**

**单周期MIPS CPU设计**

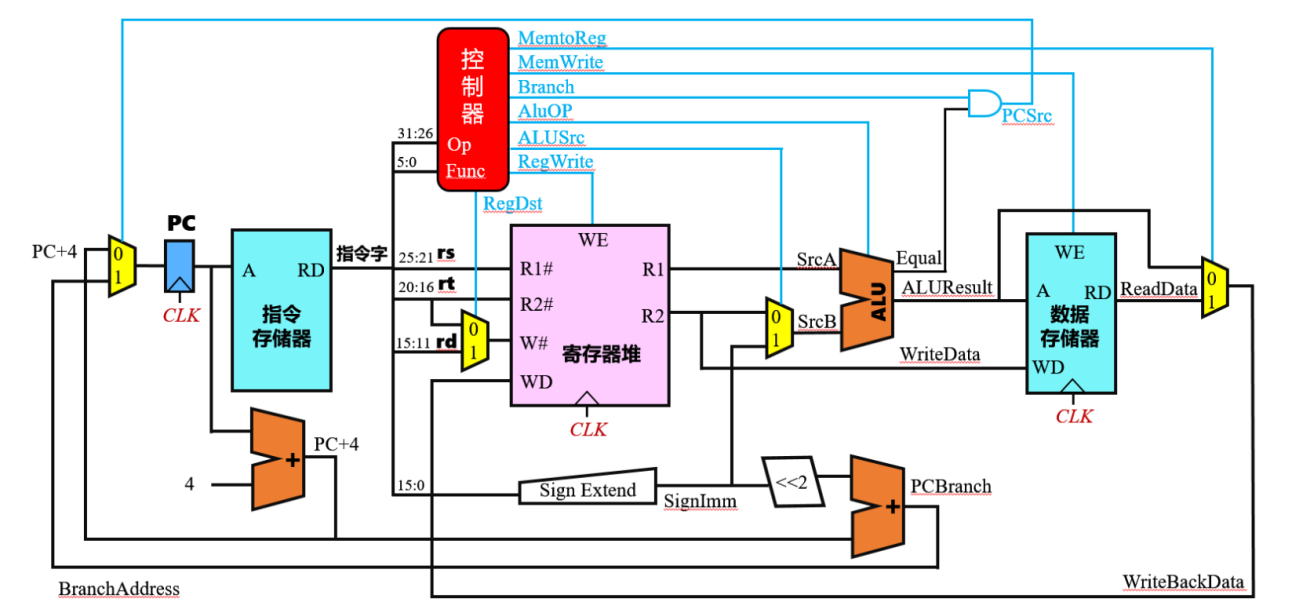
姓名： 刘子言 学号： 20002462 班级： 计203 指导教师(签章) 张欢欢

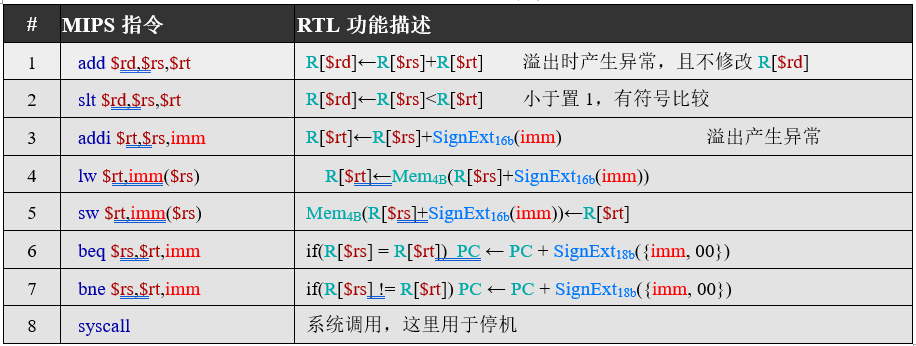
成绩:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**一、实验设计要求**

（1）掌握控制器设计的基本原理。

（2）结合硬布线控制器的设计原理，利用运算器实验、存储系统实验中构建的运算器、寄存器文件、存储系统等部件以及Logisim中的其它功能部件，构建一个32位MIPS CPU单周期处理器。数据通路如下图所示：

（3）要求设计的MIPS CPU单周期处理器支持8条MIPS核心指令，最终设计实现的MIPS处理器能运行实验包中的冒泡排序测试程序sort.hex。该程序会自动在数据存储器0~15号字单元中写入16个数据，然后利用冒泡排序将数据升序排序，要求统计指令条数，并与MARS中的指令统计数目进行对比。8条MIPS核心指令如下：



**二、实验方案设计**

单周期MIPS CPU的设计方案如下：

（1）主要任务：绘制MIPS CPU数据通路，实现单周期硬布线控制器，并完成测试联调；

（2）所用器件：程序计数器PC、寄存器文件Regfile、指令存储器IMEM、数据存储器DMEM、运算器ALU、单周期硬布线控制器Controller；

（3）电路引脚：

|  |  |  |  |
| --- | --- | --- | --- |
| **信号** | **输入/输出** | **位宽** | **功能描述** |
| CLK | 输入 | 1 | 时钟信号 |
| PC | 输出 | 32 | 程序计数器的值 |
| IR | 输出 | 32 | 当前指令字 |
| RegWrite | 输出 | 1 | 寄存器文件写使能控制信号 |
| RDin | 输出 | 32 | 寄存器文件写入端口的数据 |
| MemWrite | 输出 | 1 | 存储器写使能控制信号 |
| MDin | 输出 | 32 | 存储器写入端口的数据 |

（4）单周期硬布线控制器的设计：

1）输入信号：指令字OP、Func字段（12位）

2）输出信号：多路选择器选择信号、内存访问控制信号、寄存器写使能信号、运算器控制信号、指令译码信号，具体表示如下表所示：

3）实现逻辑：

·指令译码

对于非R型指令，可以根据给出的6位指令操作码OP产生对应的指令译码信号，LW、SW、BEQ、BNE、ADDI等。硬件实现中为了简化设计，选择利用比较器进行相应比较，得出译码信号；

将OP与常量0进行比较，若相等，则为R型指令。对于R型指令要根据Func字段来确定具体是ADD、SLT还是SysCall指令。注意：R\_TYPE表示R型运算类指令（不包括SysCall这样的特殊指令）。

**·**ALU控制器

利用OP、Func字段生成对应的运算器AluOP信号，即4位的运算器功能选择信号。

·其余控制信号

通过增加简单的组合逻辑，根据给出的指令译码信号，实现剩下的8个控制信号逻辑。

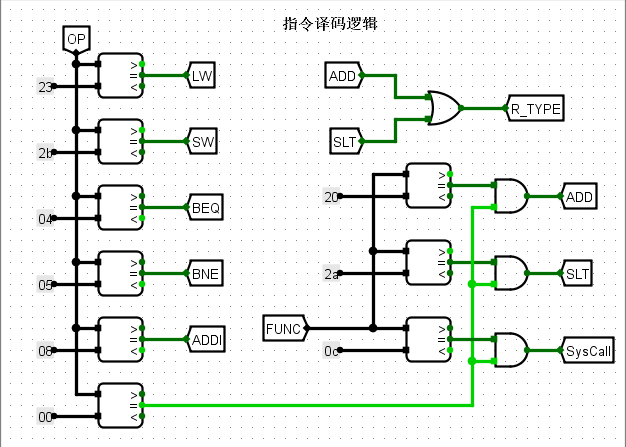
（5）单周期MIPS CPU通路设计：参照“实验设计要求”中数据通路图进行设计。

**三、实验步骤**

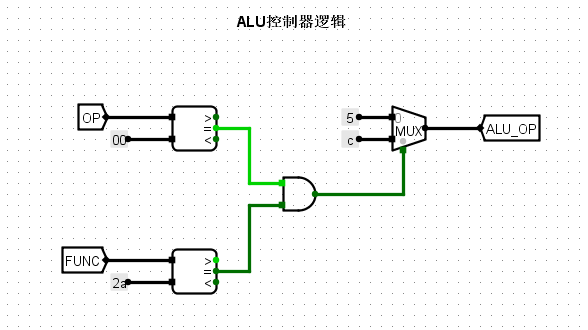
（1）在Logisim中打开已下载的cpu.circ文件，同时加载cs3410.jar文件（包含设计所需器件）；

（2）先选择单周期硬布线控制器子电路，在已有框架的基础之上，完成单周期硬布线控制器的设计：

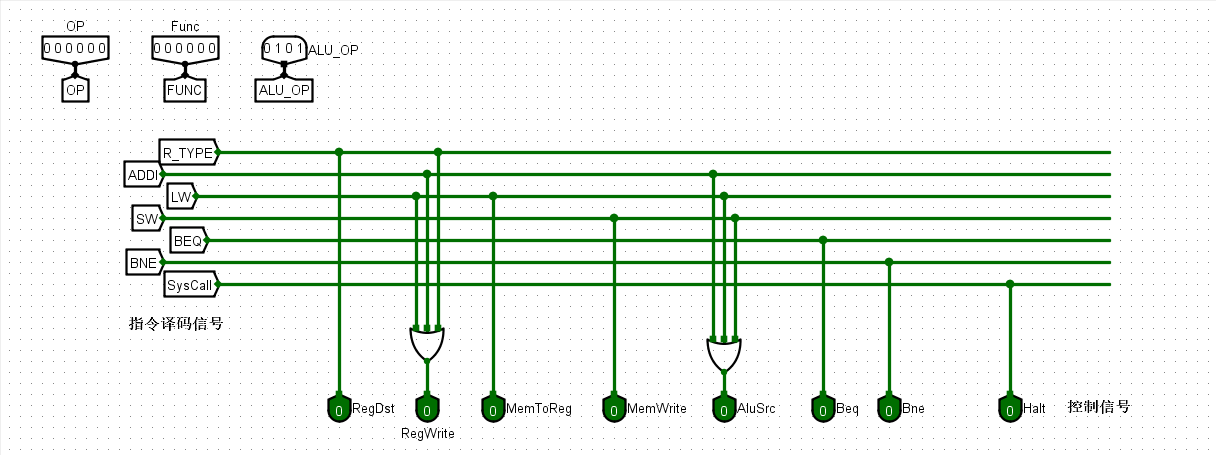
·指令译码逻辑（利用比较器和基本逻辑门，将OP与对应常量进行比较）



·ALU控制器逻辑（利用比较器、基本逻辑门及多路选择器实现AluOP的表示）



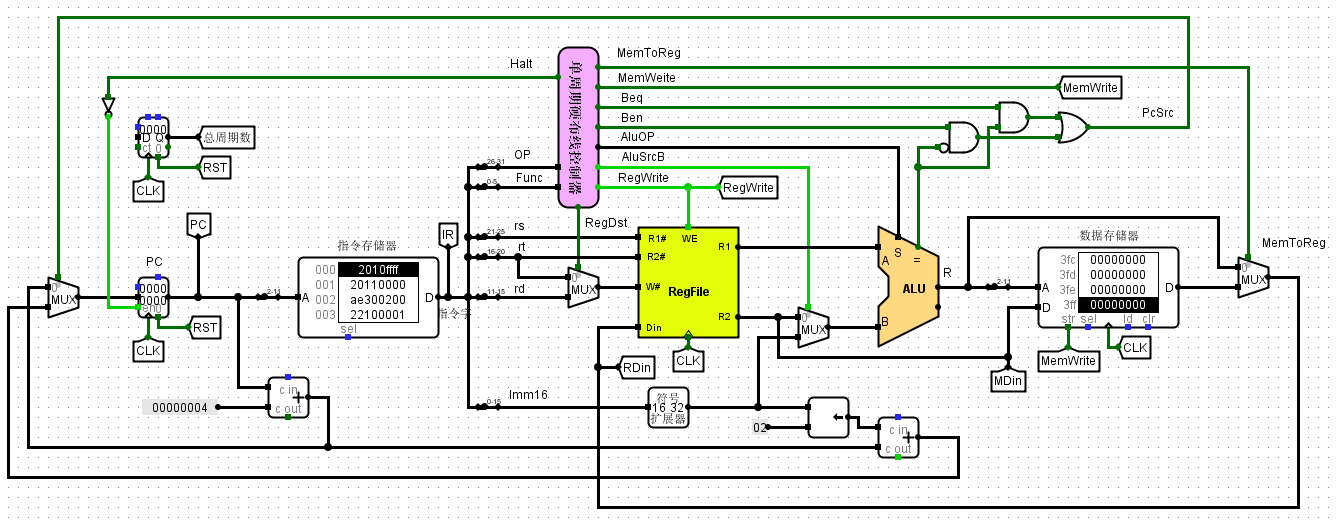
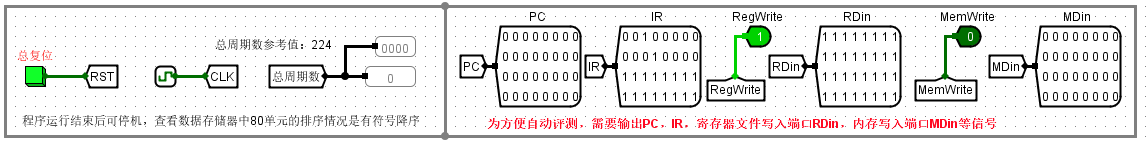
·其他控制信号逻辑（在给出的指令译码信号的基础上增加简单的组合逻辑）



（3）停机处理的设计：

设计停机逻辑处理时，不能对时钟进行直接的与操作，而是应该控制相应的寄存器的写使能信号来达到停机的目的，即应当控制电平信号而不是控制时钟信号；

（4）选择单周期MIPS（硬布线）电路，开始构建MIPS主机通路，依照数据通路图完成通路的连接：

其中，左上方的计数器用来记录CPU运行的总周期数（参考值为224），受时钟周期和复位信号的控制；

（5）进行电路测试：

·自动测试

在指令存储器中载入排序程序sort.hex，利用Logisim的时钟自动仿真（快捷键Ctrl+k）运行程序；

当程序运行完成以后，整个CPU处于停机状态（因为最后一个指令设置为SysCall）。程序停机以后，查看数字存储器中的排序情况，在080位置应该得到一个有符号降序排列。

·平台测评

自动测试结果满足实验要求后，再利用记事本打开cpu.circ文件，将所有文字信息复制粘贴到Educoder平台代码区域，点击评测按钮进行测试。

**四、故障与调试**

（1）故障一

·故障现象

实际输出报错：“Cannot find Jar Library：cs3410.jar”；

·原因分析

查阅平台相关提示资料，发现出现以上故障可能是由于本地cs3410.jar和测试平台中的cs3410.jar路径不一样导致的。

·解决方案

先将本地的cs3410.jar文件移动到cpu.circ文件的同目录下，再重新打开 cpu.circ文件；Logisim会提示找不到cs3410.jar文件，在窗口中重新选择指向同目录下的cs3410.jar文件，保存再退出，重新上传电路文本内容即可。

（2）故障二

·故障现象

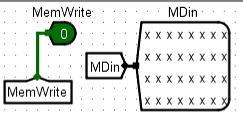
实际输出中出现以下故障：第二个时钟到来时，CPU的MDin端口和标准值不一致，是悬空态。

·原因分析与解决方案

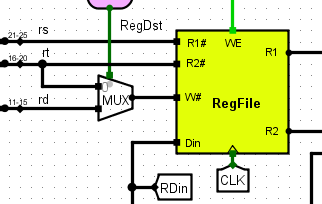
起初查阅头歌平台的常见故障提示资料，以为是Logisim新旧版本兼容性的问题，在使用新版本Logisim打开以后，依旧会出现该故障。

所以最终还是选择通过单步运行测试的方法来逐步排查故障所在：

单步运行到第二个时钟周期时观察，发现Mdin的值显示为悬浮态，运行到第三个周期，RMin、Mdin的值均为悬浮态；

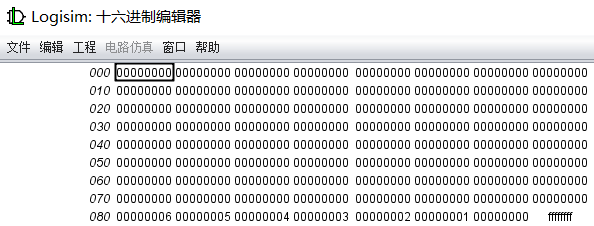


分析可知，与两者有直接联系的器件即为寄存器文件Regfile，仔细检查Regfile的输入输出后发现，多路选择器到写入寄存器编号W#之间缺少一根连线，线路连接以后再进行测试，测试通过。

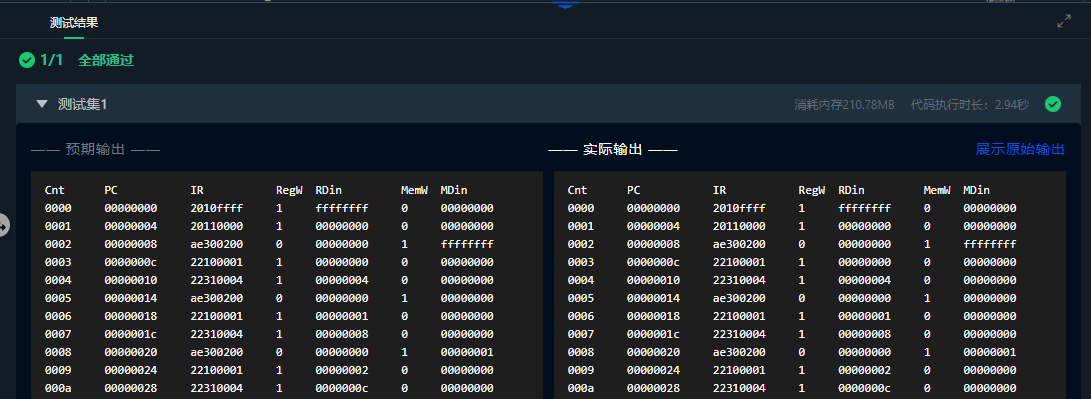


**五、实验结果测试与分析**

（1）自动测试结果如下，程序停机以后，查看数字存储器中的排序情况，在080位置得到了一个有符号降序排列，其中ffffffff表示-1：



（2）平台测试的部分结果如下：



综上，本实验最终的测试结果为通过。

本实验的关键点在于：正确设计单周期硬布线控制器中的三大逻辑（指令译码逻辑、ALU控制器逻辑、控制信号逻辑）；正确利用程序计数器、寄存器文件、指令存储器、数据存储器、运算器、以及单周期硬布线控制器来设计单周期MIPS主机通路；正确设计停机逻辑。

单周期MIPS CPU通路较为复杂，设计时需要仔细连接线路，注意计数器、寄存器、多路选择器、加法器、移位计数器、符号扩展器等器件的使用，并灵活使用各种隧道以达到简化电路的目的。