|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | 实验一 Logisim逻辑功能基础实验 | | |
| **实验日期** |  | **实验地点** | J13-232 |
| **指导老师** | **刘连山** | **实验成绩** |  |

1. **实验目的**

学习使用Logisim，掌握基本功能。

了解基本组件库

掌握logisim自动生成电路方法

1. **实验内容**
2. 下载logisim，启动logisim
3. 学习使用工具栏，添加子电路，添加时钟，学习使用splitter，理解带宽含义。
4. 绘制一些基本电路
5. **实验过程**
6. **下载logisim安装包**

下载的安装包有可能为EXE或JAR格式，二者都需要JAVA运行环境JRE。如果运行文件后缀名为JAR，则需要通过命令行JAVA \*.JAR来运行LOGISIM。

运行成功如下：主画布位于右侧

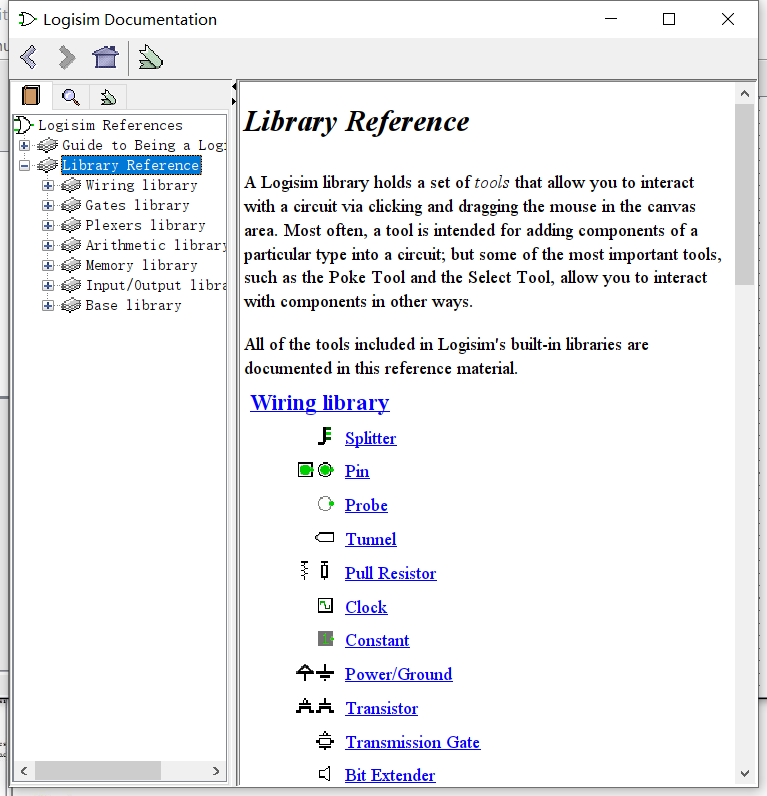


其中，File Edit…栏为菜单栏，其下方为工具栏

紧接着下方的8个按钮及整个左侧的文件夹部分被称为浏览面板。

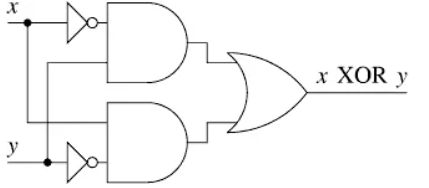
当点击某个原件的时候，浏览面板下方会出现属性详细信息

1. **基本操作**
2. 查看帮助



1. 制作简单电路

原理图如下：



具体操作：

添加两个AND门。单击工具栏中的AND工具 (列出倒数第二个工具）。然后在编辑区域中单击第一个AND门的位置。要左边的东西留出足够的空间。然后再次单击AND工具并将第二个AND门置于其下方。

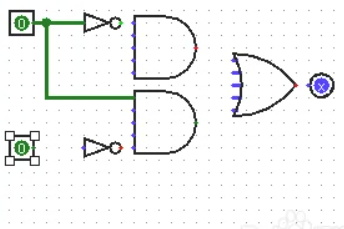
AND门左侧的五个点是可以连接电线的点。我们只将其中两个用于我们的XOR电路; 但对于其他电路，可能有两条以上的电线连接到AND门是有用的。

现在添加其他门。首先单击OR工具; 然后单击位置。使用NOT工具将两个NOT门放入画布。在NOT门和AND门之间留了一些空间; 也可以将它们相互对立。 现在将两个输入x和y添加到图中。选择输入工具，然后放下引脚。还可以使用输出工具在OR门的输出旁边放置一个输出引脚。（同样，可以在OR门和输出引脚之间留下一点空间）效果如下图所示：

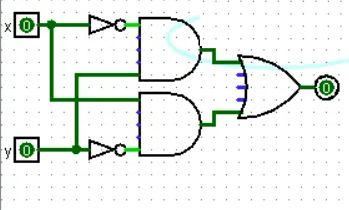


可以使用编辑工具将其选中并将其拖动到所需位置。或者，可以通过从“编辑”菜单中选择“删除”或按Delete键将其完全删除。 当您放置电路的每个组件时，只要放置组件，Logisim就会恢复到编辑工具，以便您可以移动最近放置的组件或连接组件通过创建电线。如果要添加最近放置的组件的副本，则快捷方式是按Ctrl-D复制选择。

在画布上阻止所有组件后，就可以开始添加电线了。选择编辑工具。当光标位于接收导线的点上方时，将在其周围绘制一个小的绿色圆圈。在那里按下鼠标按钮并尽可能地拖动电线。 添加电线时，每当电线在另一根电线上结束时，Logisim会自动连接。也可以使用编辑工具拖动其中一个端点来“扩展”或“缩短”导线。电线必须是水平或垂直的。为了将上输入连接到NOT门和AND门，现在添加了三条不同的线。



Logisim自动将电线连接到门。在T交叉点自动绘制圆，表示导线已连接。在绘制电线时，蓝色表示该点的值为“未知”，灰色表示该线未连接任何东西。当你完成电路时，电线都不应该是蓝色或灰色。将电线连接到正确的位置非常重要。Logisim在组件上绘制小点以指示电线应该连接的位置。然后应看到点从蓝色变为浅色或深绿色。 连接好所有电线后，插入的所有电线本身都会呈浅绿色或深绿色。最后，可以打开测试电路功能，如下图：



**实验总结/体会**

这次实验通过互联网资源，查找相关教程，使我对logisim的使用方法有了初步的了解。本次实验中，利用简单的操作实现了查看帮助，搭建电路，测试电路等过程。同时了解到了logisim中的自动连接功能和导线代表的含义。我们还在logisim中搭建了简单的电路图，并测试了其功能的正确性。希望可以在之后的学习中对logisim有更深的了解。

|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | 实验二 8位可控加减法电路设计 | | |
| **实验日期** |  | **实验地点** | J13-232 |
| **指导老师** | **刘连山** | **实验成绩** |  |

1. **实验目的**

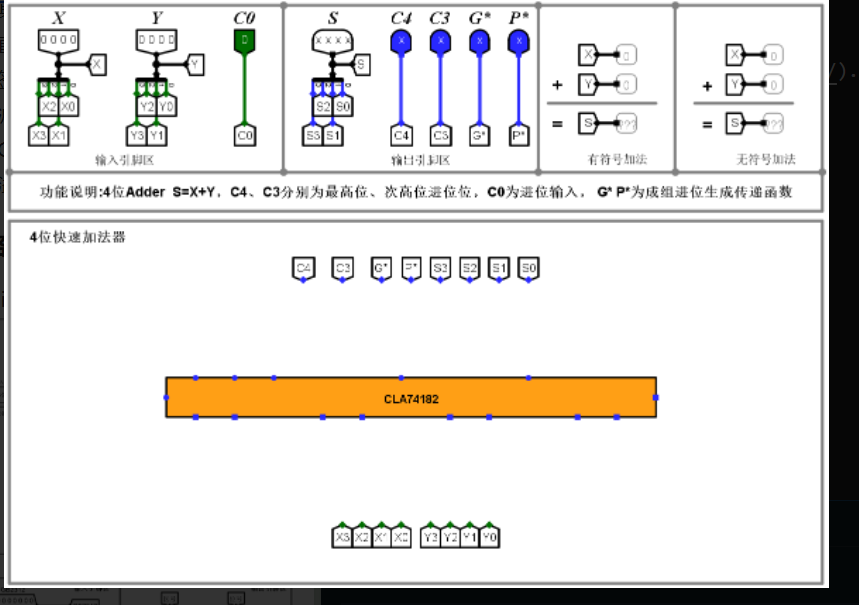
帮助学生掌握一位全加器的实现逻辑，掌握多位可控加减法电路的实现逻辑，熟悉 Logisim 平台基本功能，能在 logisim 中实现多位可控加减法电路。

1. **实验内容**

在alu.circ文件中原码一位乘法器子电路中增强控制电路和数据通路使得该电路能自动完成8位无符号的一位乘法运算，设置引脚初始值，然后驱动时钟自动仿真，电路可自动完成运算，运算结束结果传输到输出引脚，运算结束时电路应该自动停止。

**电路框架**

alu.circ



**电路引脚**

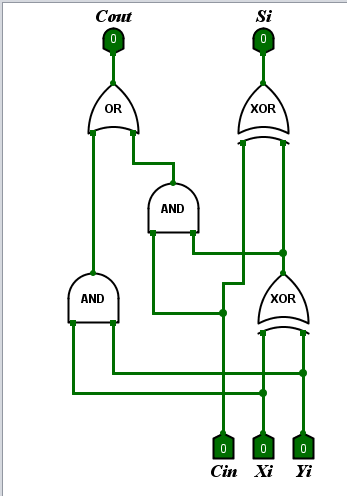
| **信号** | **输入/输出** | **位宽** | **说明** |
| --- | --- | --- | --- |
| X | 输入 | 8 位 | 加数/被减数 |
| Y | 输入 | 8 位 | 加数/减数 |
| Sub | 输入 | 1 位 | 运算控制位 1：减法 0：加法 |
| S | 输出 | 8 位 | 运算结果 和/差 |
| Cout | 输出 | 1 位 | 进位输出 |
| OF | 输出 | 1 位 | 有符号运算溢出检测位 |

**电路测试**

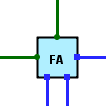
完成实验后，利用文本编辑工具打开 alu.circ 文件，将所有文字信息复制粘贴到 Educoder 平台的 alu.circ 文件中，再点击评测按钮即可进行本关测试。平台会对你设计的电路进行自动测试，为方便测试，请勿修改子电路封装，本关测试用例如下:

1. X Y Sub S Cout OF
2. 10 21 0 31 0 0
3. 7f 02 0 81 0 1
4. ff fe 0 fd 1 0
5. 81 82 0 03 1 1
6. 10 df 1 31 0 0
7. 7f fe 1 81 0 1
8. ff 02 1 fd 1 0
9. 81 7e 1 03 1 1
10. **模块电路**

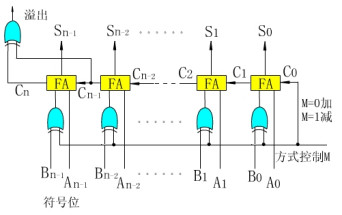
根据全加器真值表，首先设计1位全加器，结构图如下



将其封装成模块FA：

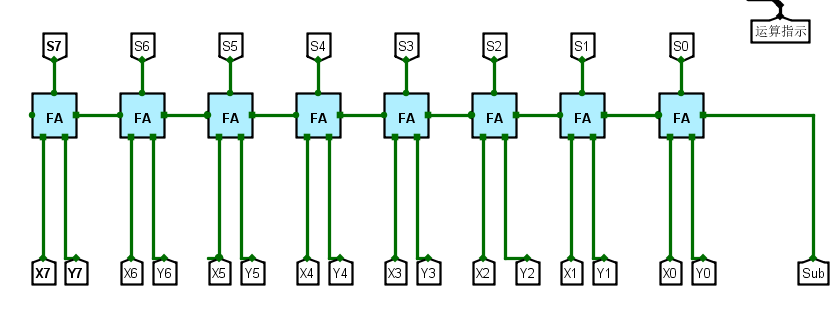


然后，由如下N位可控加法器原理图（如下）：

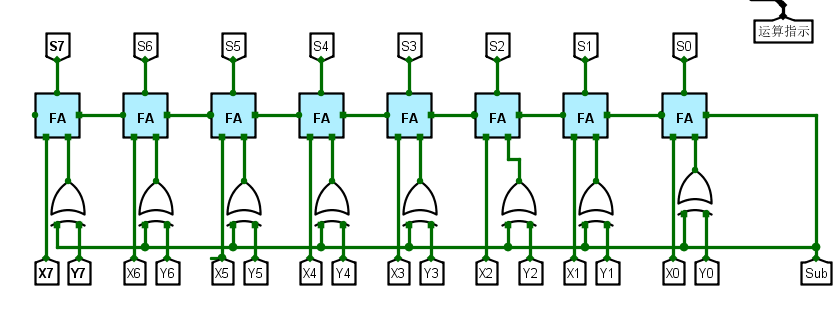


使用封装模块FA搭建如上电路：

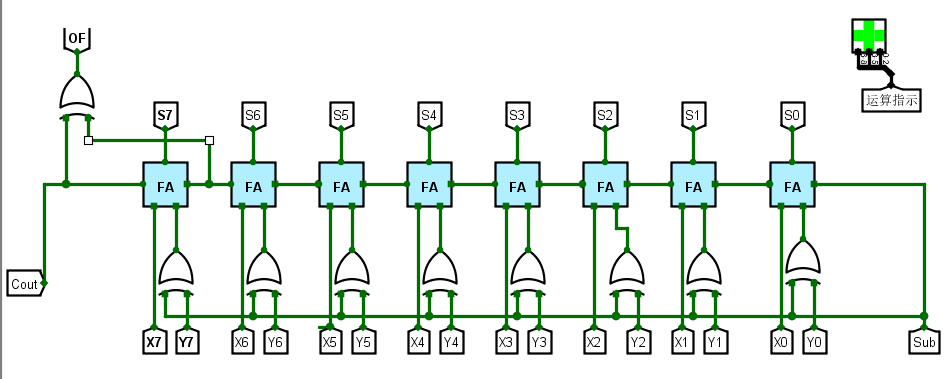
1.8位加法器：



2.8位可控加法器：

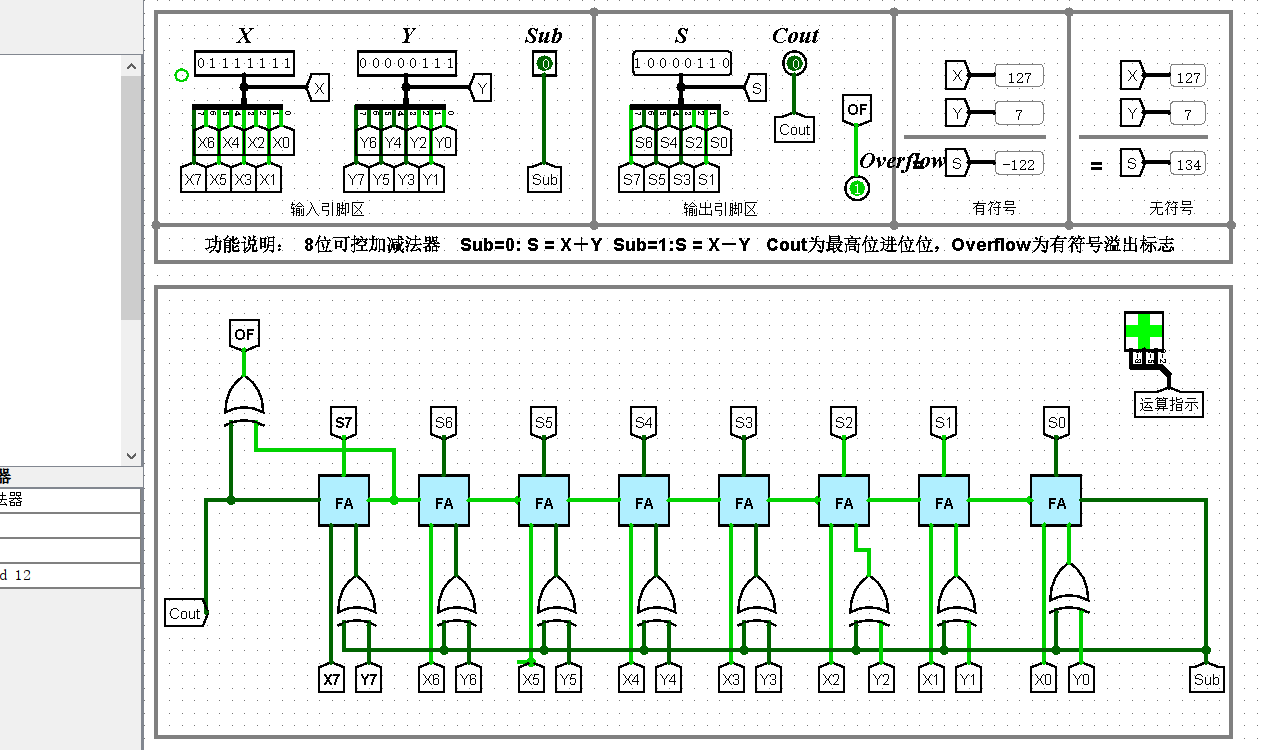


3.添加输入输出：

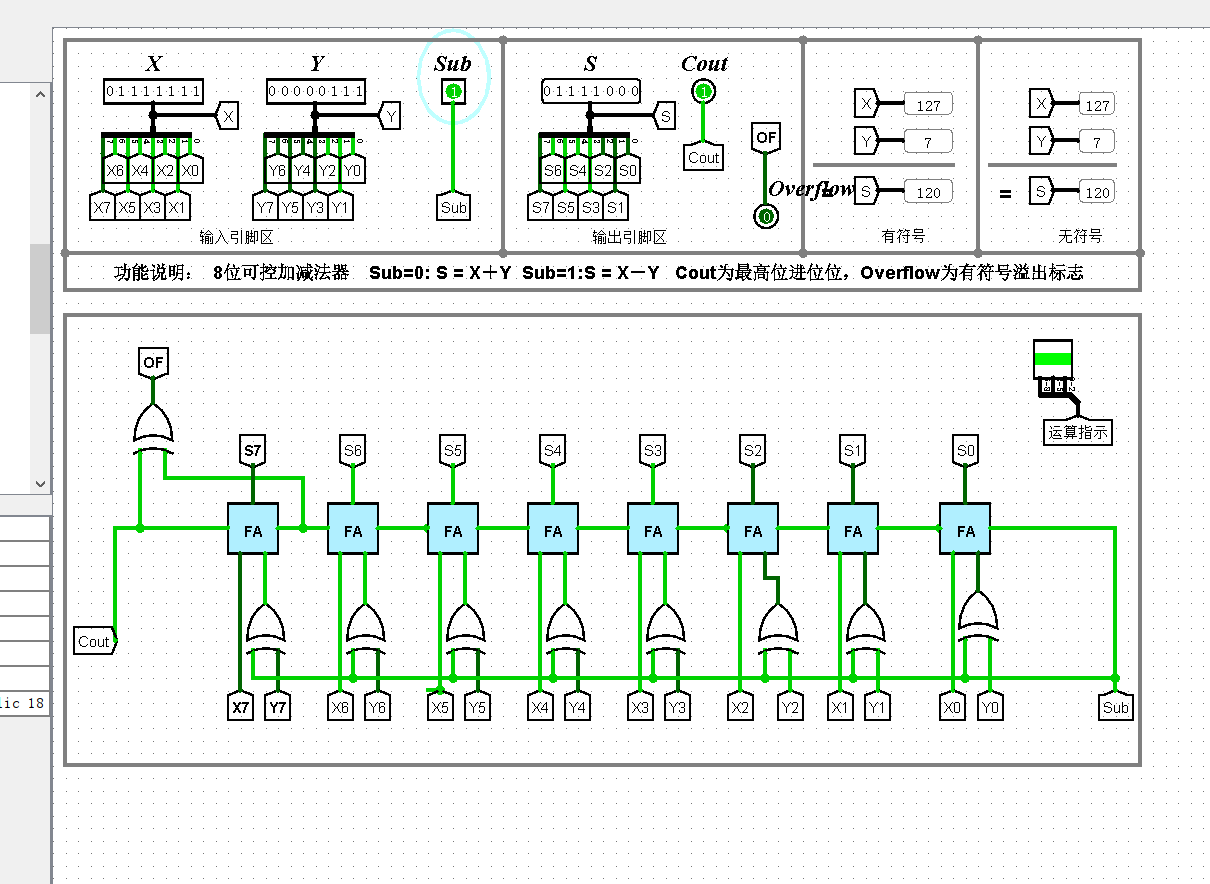


1. **实验结果**

**S=X+Y(SUB=0):COUT进位=0,OVERFLOW溢出=1**

****

**S=X-Y(SUB=1):COUT进位=1,OVERFLOW溢出=0**

****

****

1. **实验总结/体会**

完成本次实验后，我对logisim的基本使用方法有了一定的了解，同时，要得出正确的实验结果，需要了解全加器的原理，其可用公式表达为：X+Y=X+Y+0,X-Y=X+-Y+1,用硬件实现这一逻辑，需要使用可控反相异或门和异或门，在上述设计的电路中，通过将Sub置0置1,实现了Y到-Y的转化，同时，这也是Cin的信号输入。

|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | 实验三 MIPS寄存器文件设计 | | |
| **实验日期** |  | **实验地点** | J13-232 |
| **指导老师** | **刘连山** | **实验成绩** |  |

1. **实验目的**

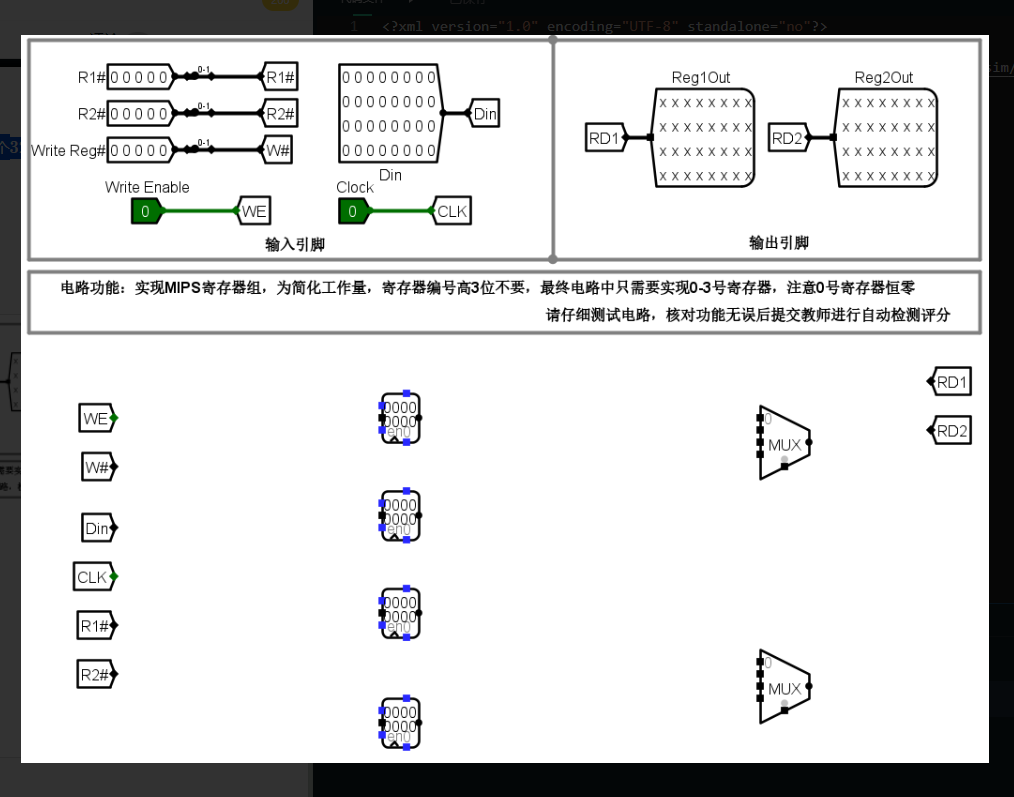
学生了解 MIPS 寄存器文件基本概念，进一步熟悉多路选择器、译码器、解复用器等 Logisim 组件的使用，并利用相关组件构建 MIPS 寄存器文件。

1. **实验内容**

利用 Logisim 平台构建一个简化的 MIPS 寄存器文件，内部包含4个32位寄存器，其具体引脚与功能描述如下表。

**电路框架**

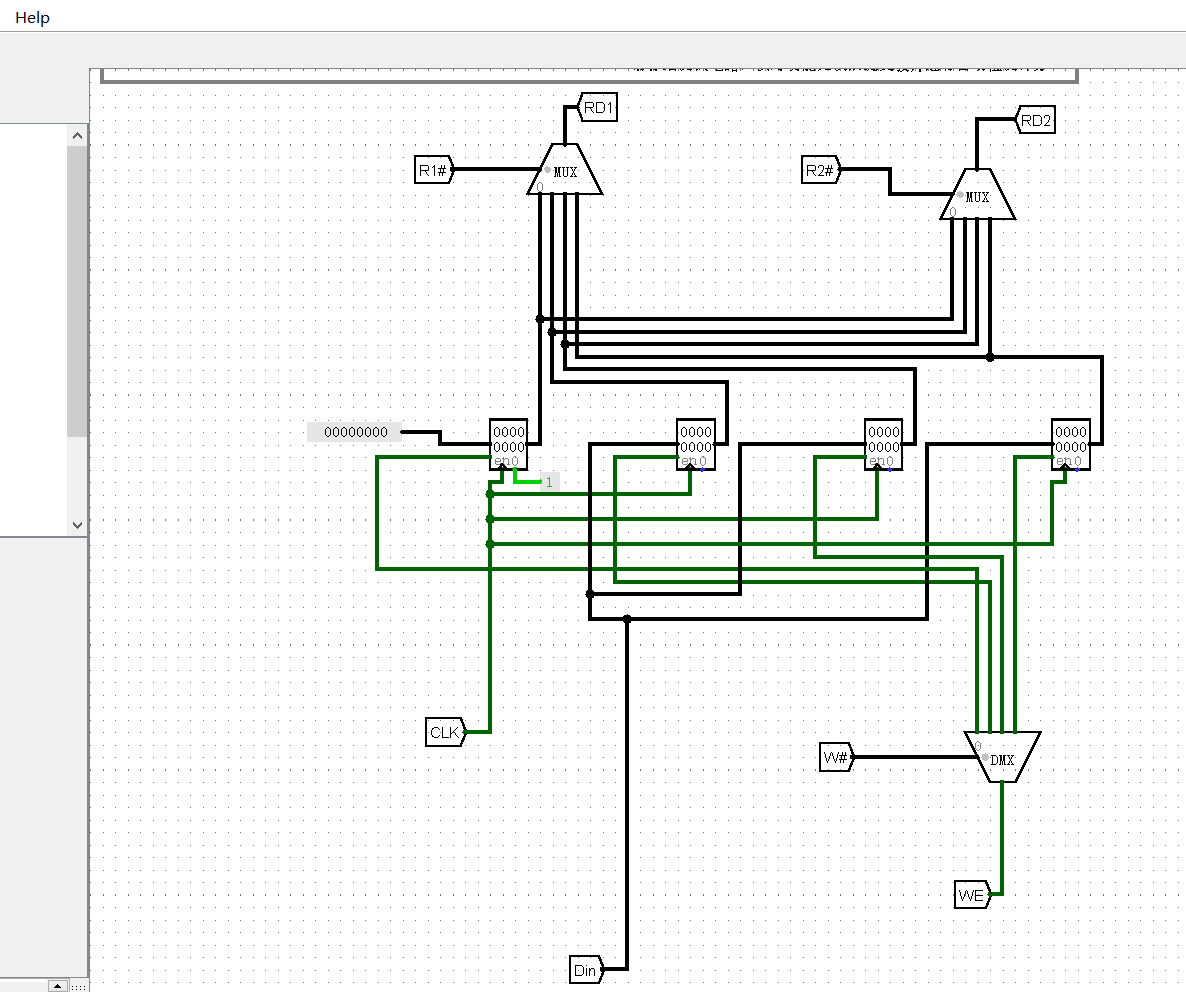
alu.circ



**电路引脚**

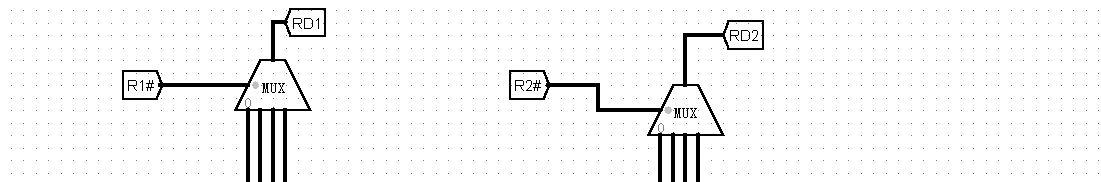
| **信号** | **输入/输出** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| R1# | 输入 | 5 | 第 1 个读寄存器的编号 |
| R2# | 输入 | 5 | 第 2 个读寄存器的编号 |
| W# | 输入 | 5 | 写入寄存器编号 |
| Din | 输入 | 32 | 写入数据 |
| WE | 输入 | 1 | 写使能信号，为 1 时在 CLK 上跳沿将 Din 数据写入W#寄存器 |
| CLK | 输入 | 1 | 时钟信号，上跳沿有效 |
| RD1 | 输出 | 32 | R1# 寄存器的值，MIPS 寄存器文件中 0 号寄存器的值恒零 |
| RD2 | 输出 | 32 | R2# 寄存器的值，MIPS 寄存器文件中 0 号寄存器的值恒零 |

1. **电路设计**

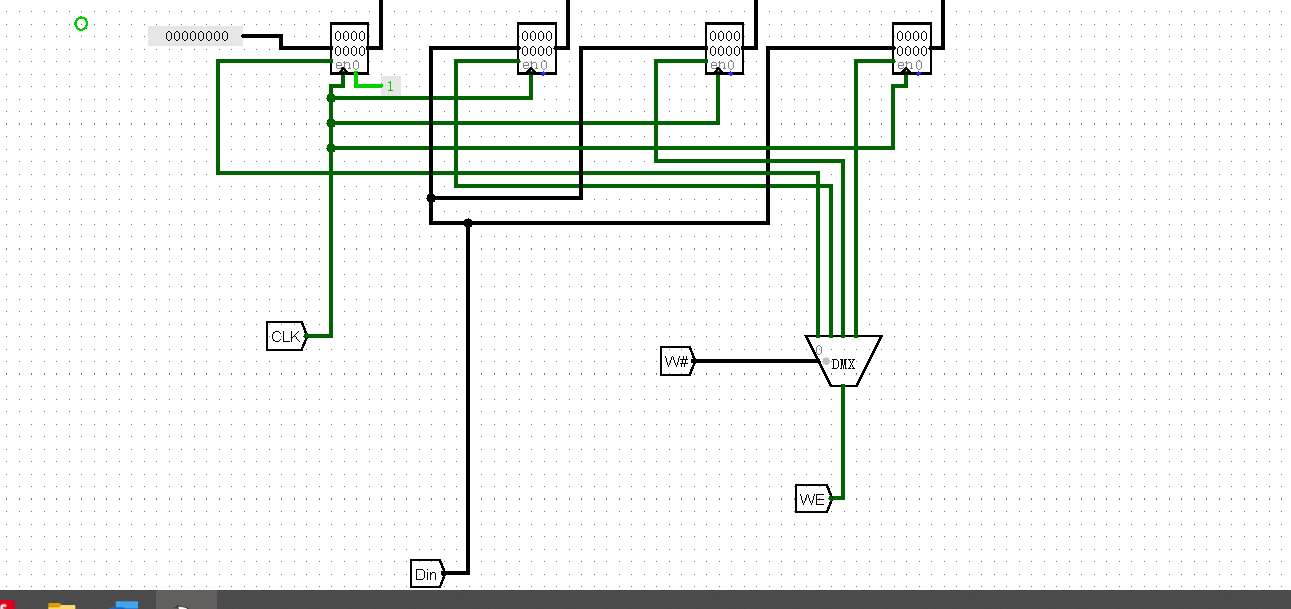


**电路分析：**

首先使用多路选择器，或者译码器+三态门来控制输出。当进行写入控制时，可以使用译码器或解复用器（即多路分配器）。0号寄存器的值恒为0，且给出异步清零信号（恒为1）。本次实验选用了复用器进行写入控制。



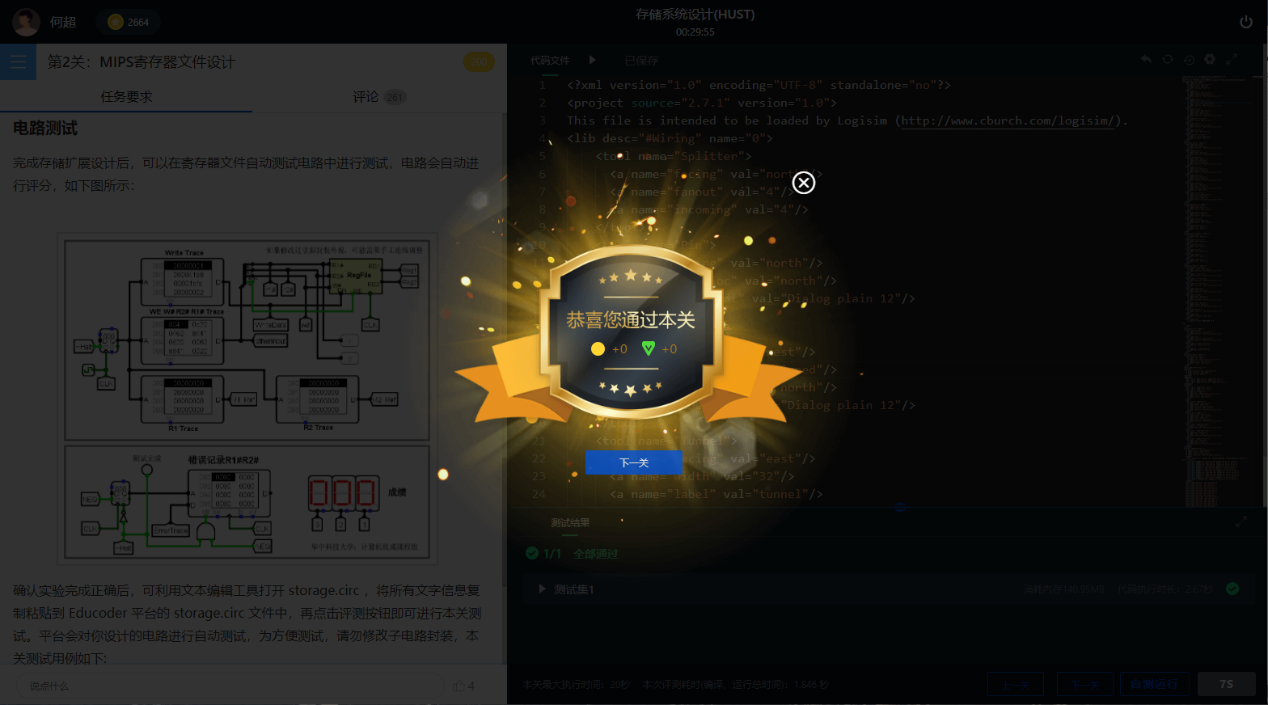
然后使解复用器的选择端位宽为2，数据位宽为1，使常量的数据位宽为32。当输入数据时，解复用器决定了写入信息的存储器位置。写入信号的输入由W#控制，W#选择相应存储器编码，并控制WE的信号输入对应存储器。0号存储器恒为零，通过直接设置数值控制0号寄存器的值恒零。



在输出数据时，多路选择器在R1#、R2#信号的控制下，会将对应编号存储器中的数据输出。随后，WE和DMX通过控制存储器使能端控制信息的读入。W#选择对应存储器，WE及CLK有信号1传入，控制对应存储器接收1信号，剩余存储器接收0信号。以此实现了只允许选中存储器读入Din信息。

1. **实验结果**

****

****

1. **实验总结/体会**

在实验中，可以通过电路中导线颜色，和logisim的提示语来判断出现问题的基本类型。此外我初步了解了解复用器：其有一个输入端口、选择端口、输出端口，不同于译码器的是，译码器只有选择和输出端口，本次实验，我还了解了MIPS寄存器基本结构与概念，熟悉了多路选择器、译码器、解复用器等基本原件的用途和使用方法。

|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | 实验四 原码一位乘法器设计 | | |
| **实验日期** |  | **实验地点** | J13-232 |
| **指导老师** | **刘连山** | **实验成绩** |  |

1. **实验目的**

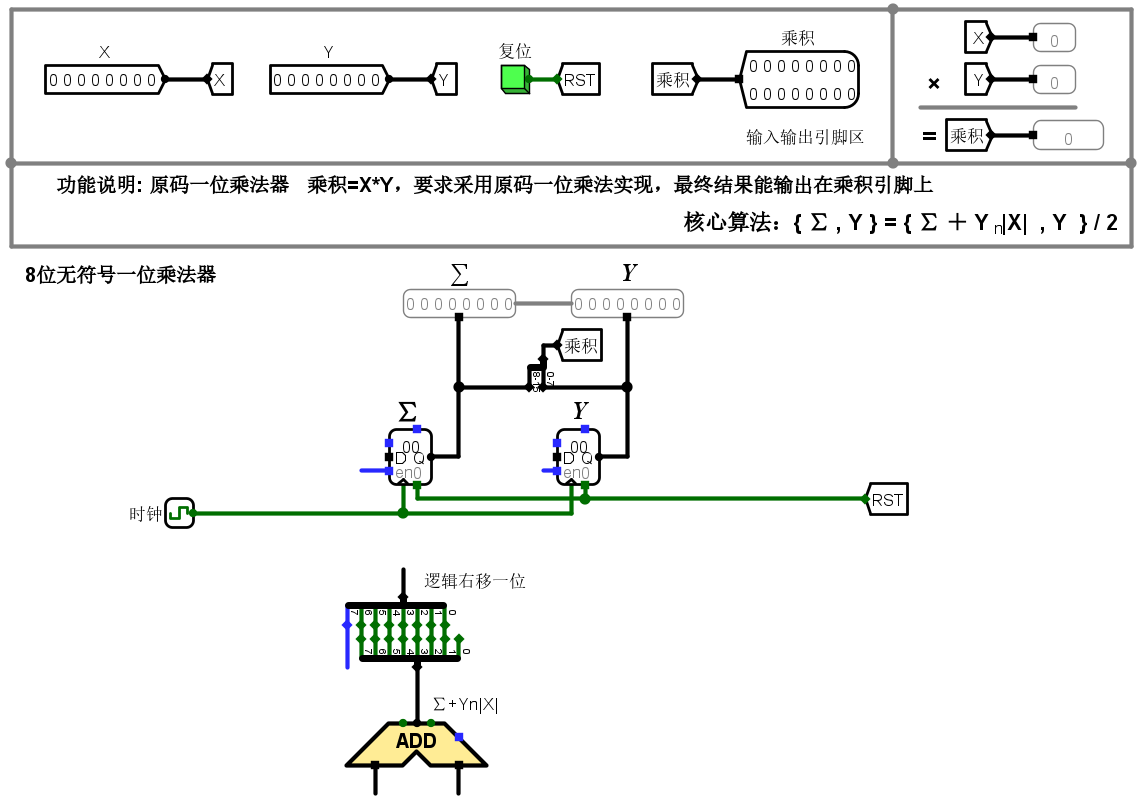
学生掌握原码一位乘法运算的基本原理，熟练掌握Logisim寄存器电路的使用，能在Logisim平台中设计实现一个8\*8的无符号数乘法器。

1. **实验内容**

在alu.circ文件中原码一位乘法器子电路中增强控制电路和数据通路使得该电路能自动完成8位无符号的一位乘法运算，设置引脚初始值，然后驱动时钟自动仿真，电路可自动完成运算，运算结束结果传输到输出引脚，运算结束时电路应该自动停止。

**电路框架**

alu.circ



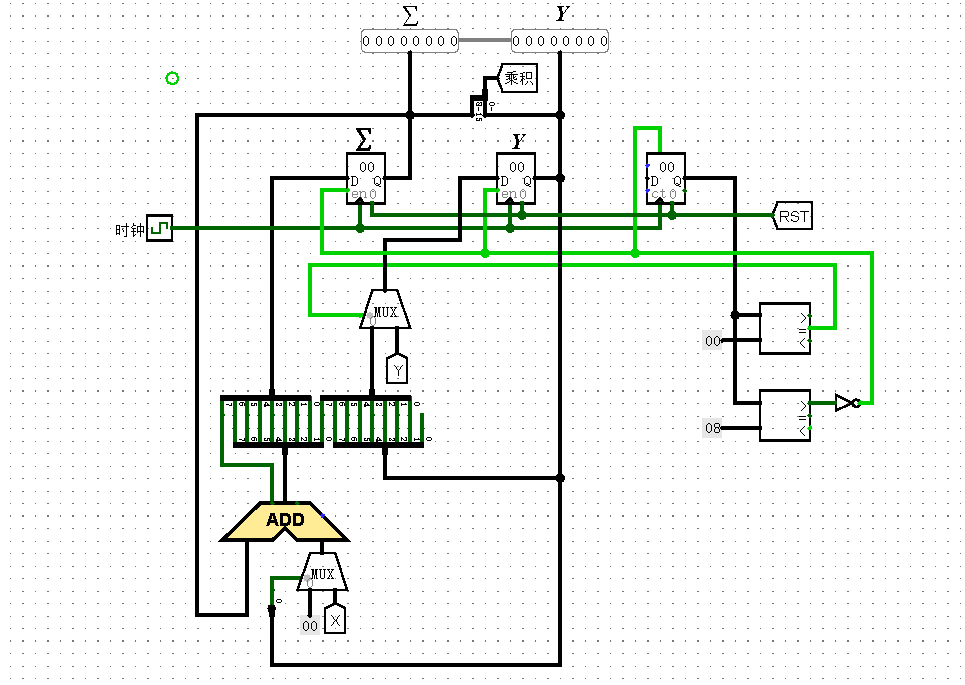
**电路引脚**

| **信号** | **输入/输出** | **位宽** | **说明** |
| --- | --- | --- | --- |
| X | 输入 | 8位 | 被乘数 |
| Y | 输入 | 8位 | 乘数 |
| 乘积 MulResult | 输出 | 16位 | 乘积 |
| ####电路测试 |  |  |  |
| 确实实验完成正确后可利用文本编辑工具打开alu.circ，将所有文字信息复制粘贴到educoder平台的alu.circ文件中，再点击评测按钮即可进行本关测试，平台会对你设计的电路进行自动测试，为方便测试，请勿修改子电路封装，本关测试用例如下: |  |  |  |

1. Cnt x y MulResult
2. 00 ff 03 0000
3. 01 ff 03 0003
4. 02 ff 03 7f81
5. 03 ff 03 bf40
6. 04 ff 03 5fa0
7. 05 ff 03 2fd0
8. 06 ff 03 17e8
9. 07 ff 03 0bf4
10. 08 ff 03 05fa
11. 09 ff 03 02fd
12. **模块电路**

原码一位乘法的算法是：乘数从最低位逐位乘以被乘数，再对结果进行累加。每次进行每位的乘法运算后，这个乘数就不再被用到了，可以右移一位，将最低位丢掉。累加时，要把部分积右移一位来对齐，因为之前做的每位的乘法是高了一位的。在这个过程中丢掉的一位数应该存储在乘数寄存器的最高位，让每次右移丢掉的一位都在最前列。运行8次后，第一次丢掉的数字就到达了最低位。当Yi=0时，输出为0，反之则结果为X。据此，可选择用数据选择器实现乘法来控制运算次数，首次运算将Y输入乘数寄存器。经过第8次运算后停止运算。如下图所示（无符号运算，不考虑符号位）

据此硬件逻辑设计了电路图如下：



1. **实验结果**

****

1. **实验总结/体会**

这次实验中对Logisim的使用了解的更为详细，熟悉了利用模拟电路平台完成实验的流程，同时，我认识到需要善于利用互联网资源查找知识点做到自主学习，这样对于知识的理解会更加的深刻一些，因为要了解实验过程背后的各种原理需要仔细琢磨，而不能大概的了解。

|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | 实验五、六 MIPS运算器设计 | | |
| **实验日期** |  | **实验地点** | J13-232 |
| **指导老师** | **刘连山** | **实验成绩** |  |

1. **实验目的**

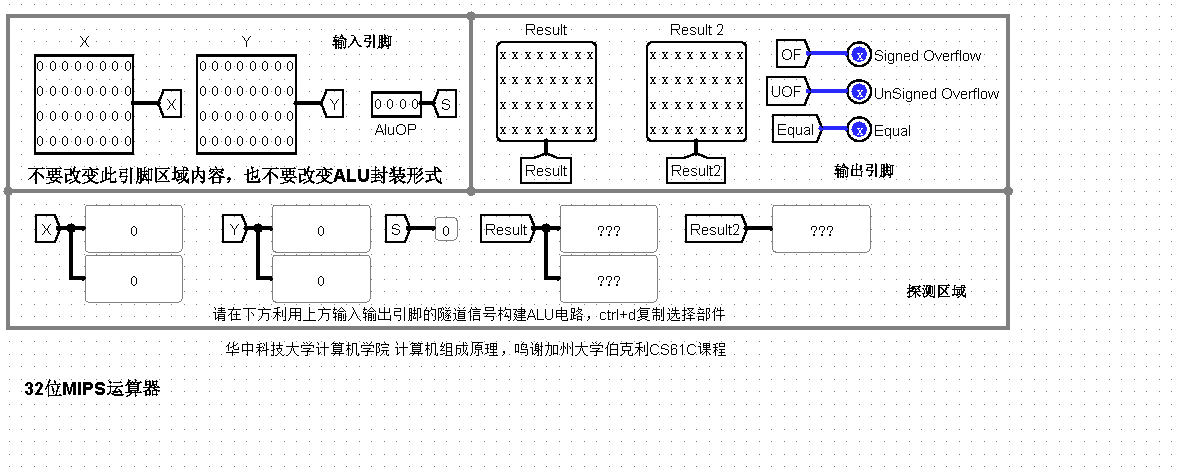
学生理解算术逻辑运算单元（ALU）的基本构成，掌握 Logisim 中各种运算组件的使用方法，熟悉多路选择器的使用，能利用前述实验完成的32位加法器、 Logisim 中的运算组件构造指定规格的 ALU 单元。

1. **实验内容**

利用前面实验封装好的32位加法器以及 Logisim 平台中现有运算部件，构建一个32位算术逻辑运算单元（禁用 Logisim 系统自带的加法器，减法器），可支持算术加、减、乘、除，逻辑与、或、非、异或运算、逻辑左移、逻辑右移、算术右移运算，支持常用程序状态标志（有符号溢出 OF 、无符号溢出 UOF ，结果相等 Equal ），ALU 功能以及输入输出引脚见后表，在主电路中详细测试自己封装的 ALU ，并分析该运算器的优缺点。

**电路框架**

alu.circ



**电路引脚**

| **信号** | **输入/输出** | **位宽** | **说明** |
| --- | --- | --- | --- |
| X | 输入 | 32 位 | 操作数 X |
| Y | 输入 | 32 位 | 操作数 X |
| ALU\_OP | 输入 | 4 位 | 运算器功能码，具体功能见下表 |
| Result | 输出 | 32 位 | ALU 运算结果 |
| Result2 | 输出 | 32 位 | ALU 结果第二部分，用于乘法指令结果高位或除法指令的余数位，其它运算时值为零 |
| OF | 输出 | 1 位 | 有符号加减运算溢出标记，其它运算时值为零 |
| UOF | 输出 | 1 位 | 无符号加减运算溢出标记，其它运算时值为零,溢出条件（加法和小于加数，减法差大于被减数） |
| Equal | 输出 | 1 位 | Equal=(x==y)?1:0, 对所有运算均有效 |

| **ALU\_OP** | **十进制** | **运算功能** |
| --- | --- | --- |
| 0000 | 0 | Result = X << Y 逻辑左移 （Y 取低五位） Result2=0 |
| 0001 | 1 | Result = X >>>Y 算术右移 （Y 取低五位） Result2=0 |
| 0010 | 2 | Result = X >> Y 逻辑右移 （Y 取低五位） Result2=0 |
| 0011 | 3 | Result = (X \* Y)[31:0]; Result2 = (X \* Y)[63:32] 无符号乘法 |
| 0100 | 4 | Result = X/Y; Result2 = X%Y 无符号除法 |
| 0101 | 5 | Result = X + Y (Set OF/UOF) |
| 0110 | 6 | Result = X - Y (Set OF/UOF) |
| 0111 | 7 | Result = X & Y 按位与 |
| 1000 | 8 | Result = X or Y 按位或 |
| 1001 | 9 | Result = X⊕Y 按位异或 |
| 1010 | 10 | Result = ~(X orY) 按位或非 |
| 1011 | 11 | Result = (X < Y) ? 1 : 0 符号比较 |
| 1100 | 12 | Result = (X < Y) ? 1 : 0 无符号比较 |

1. **模块电路**

首先先构建多路选择器，32位快速加法器，4位先行进位74182，4位快速加法器，16位快速加法器，32位快速加法器最后再利用之前的组件构建32位MIPS运算器，其中：总结4位74182原理公式：

Ci = Gi + Pi\*Ci-1，则有

C1 = G1 + P1\*C0

C2 = G2 + P2\*C1

C3 = G3 + P3\*C2

C4 = G4 + P4\*C3

逐步带入可得

C2 = G2 + P2\*（G1 + P1\*C0）

= G2 + P2G1 + P2P1C0

C3 = G3 + P3\*（G2 + P2G1 + P2P1C0）

= G3 + P3G2 + P3P2G1 + P3P2P1C0

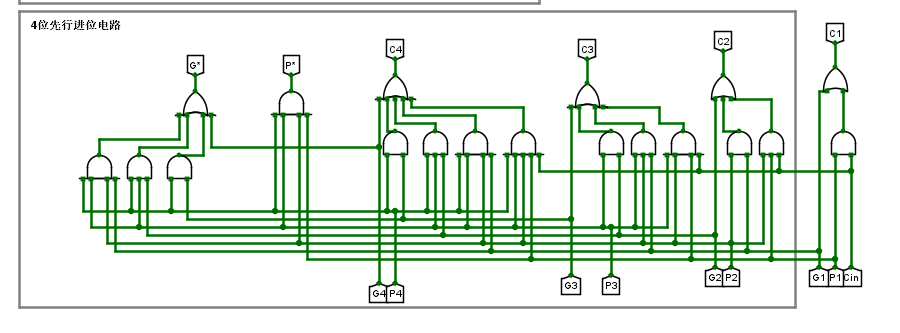
C4 = G4 + P4\*（G3 + P3G2 + P3P2G1 + P3P2P1C0）

= G4 + P4G3 + P4P3G2 + P4P3P2G1 + P4P3P2P1C0

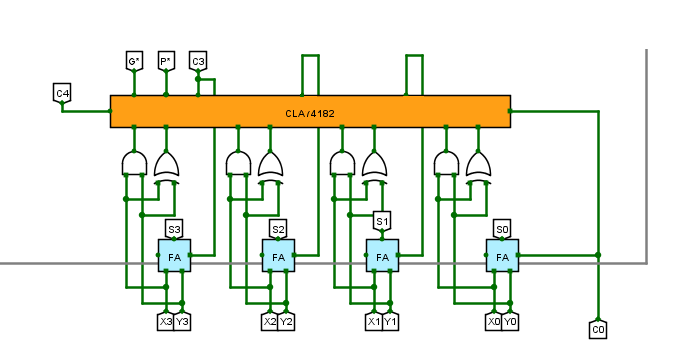
G\* = G4 + P4G3 + P4P3G2 + P4P3P2G1

P\* = P4P3P2P1

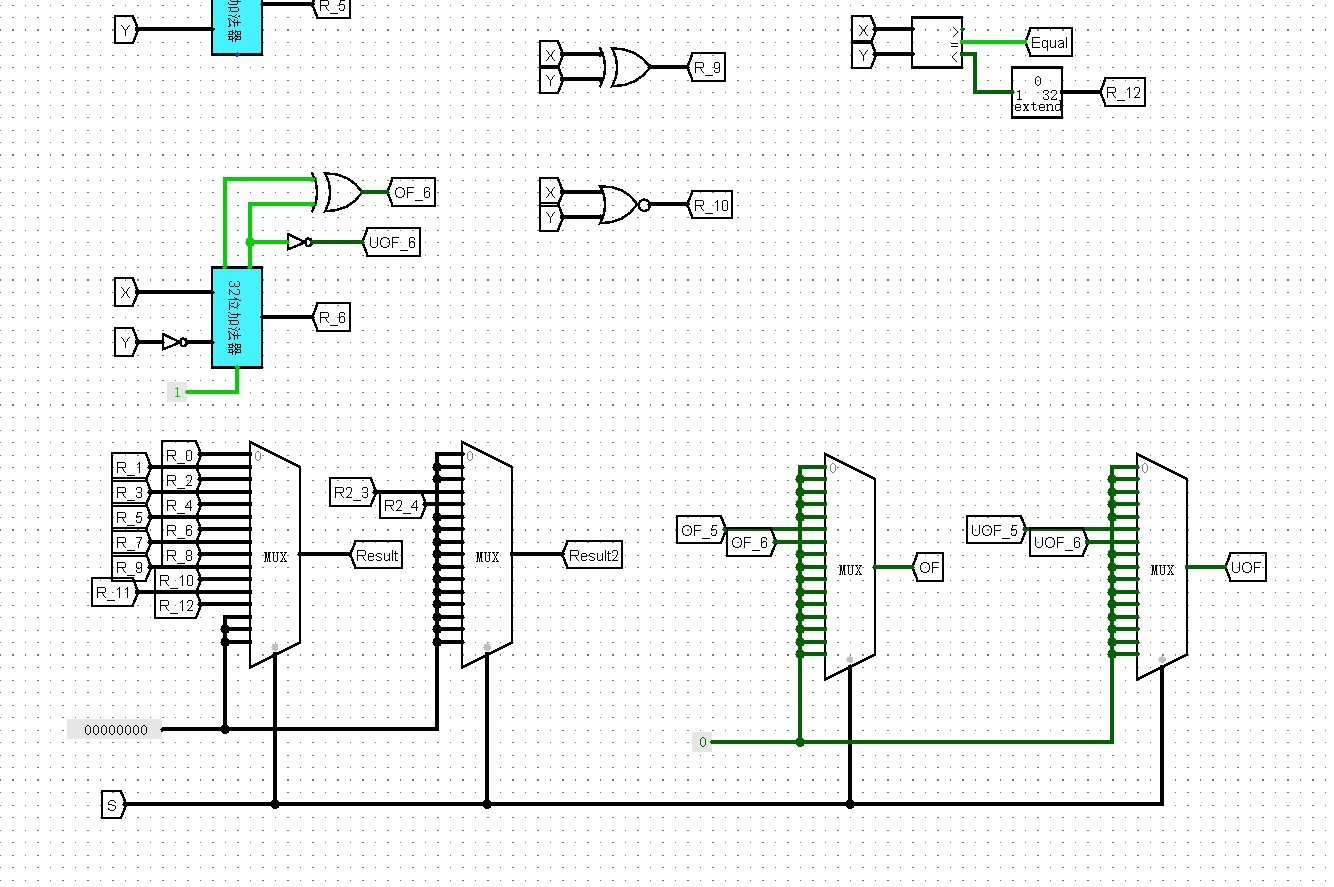
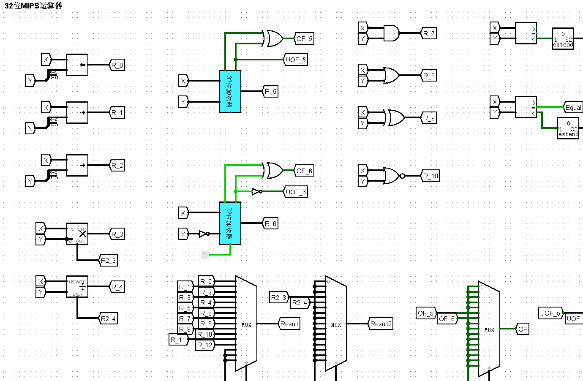
据此构建74182电路如下图 ：



另外，由前式得出：4位快速加法器Gi = X0Y0，Pi = X0 + Y0，对应构建电路如下：

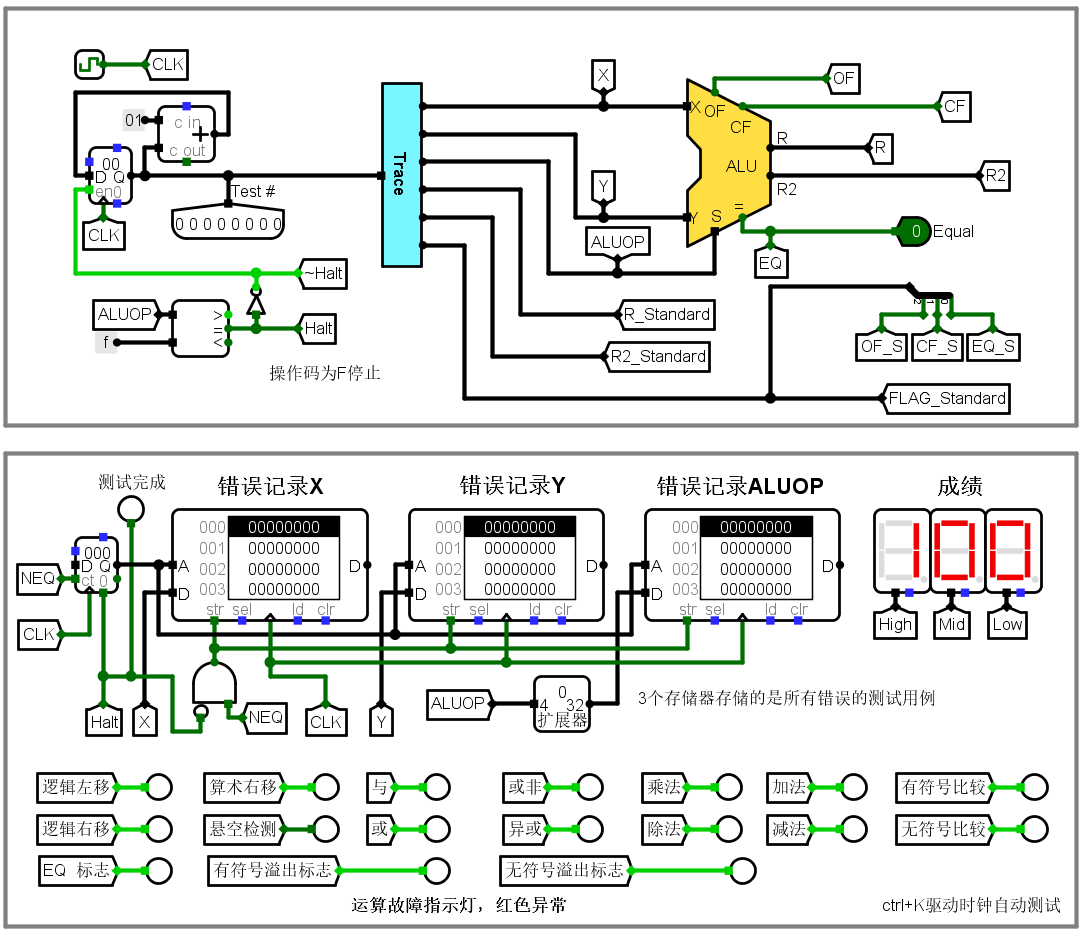


则据以上条件设计32位MIPS运算器：

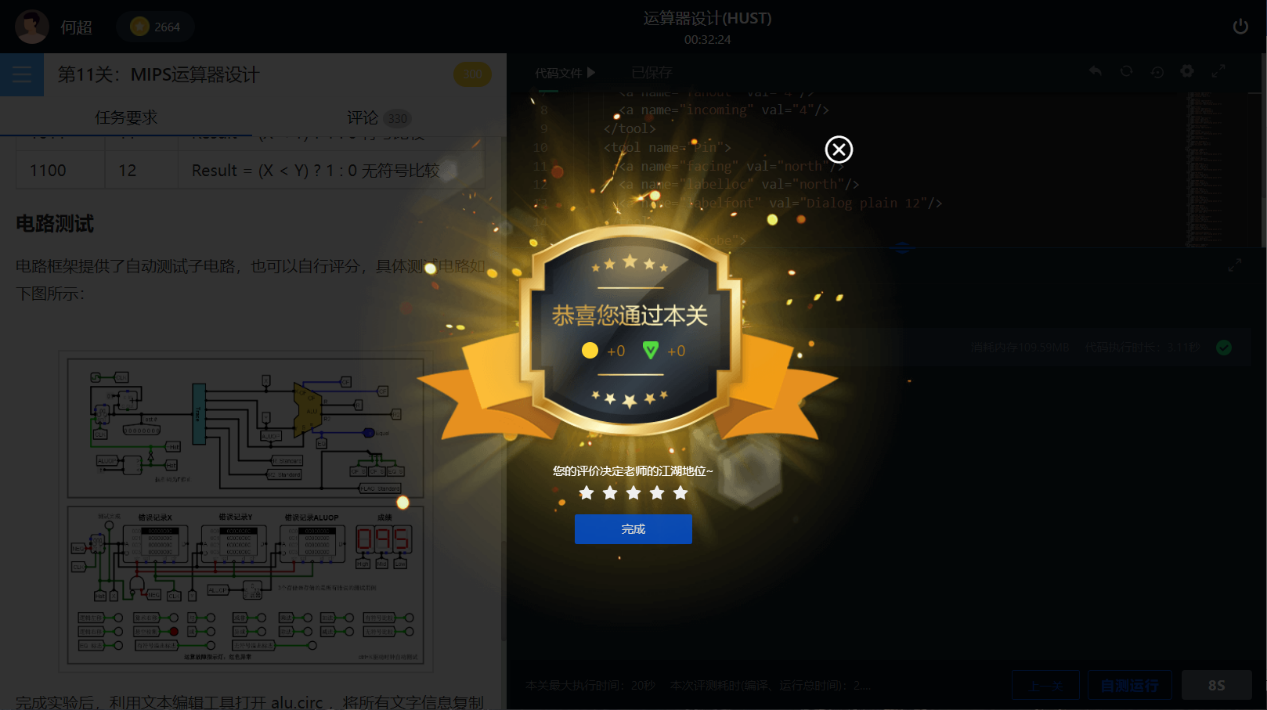


进行测试：

使用CTRL K进行自带测验电路模块测试：结果如下：



1. **实验结果**



1. **实验总结/体会**

本次实验中，我更加理解了关于多路选择器、乘法器、除法器等模块的使用。并再次使用logisim使自己对实验操作更为了解。了解了ALU运算过程。包括传统的四则运算，与或非，同或异或运算、逻辑左右移、以及算术右移，了解了各种状态标志： OF 、UOF 、Equal。希望可以在今后的实验中运用到本次积累下的经验

|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | 实验七 汉字字库存储芯片扩展实验 | | |
| **实验日期** |  | **实验地点** | J13-232 |
| **指导老师** | **刘连山** | **实验成绩** |  |

1. **实验目的**

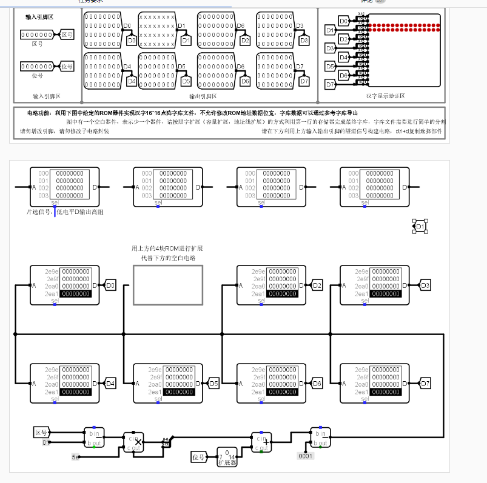
理解存储系统进行位扩展、字扩展的基本原理，能利用相关原理解决实验中汉字字库的存储扩展问题，并能够使用正确的字库数据填充。

1. **实验内容**

现有如下 ROM 组件，4片4K\*32位 ROM ，7片 16K\*32位 ROM，请在 Logisim 平台构建 GB2312 汉字编码的16K\*16点阵汉字字库，电路输入为汉字区号和位号，电路输出为8×32位（16K\*16=256 位点阵信息），待完成的字库电路输入输出引脚见后图，具体参见工程文件中的 storage.circ 文件，图中左侧是输入引脚，分别对应汉字区位码的区号和位号，中间区域为8个32位的输出引脚，可一次性提供一个汉字的256位点阵显示信息，右侧是实际显示区域，用于观测汉字显示是否正常。待完成字库子电路封装已经完成，请勿修改以免影响后续自动测试功能。

**电路框架**

alu.circ

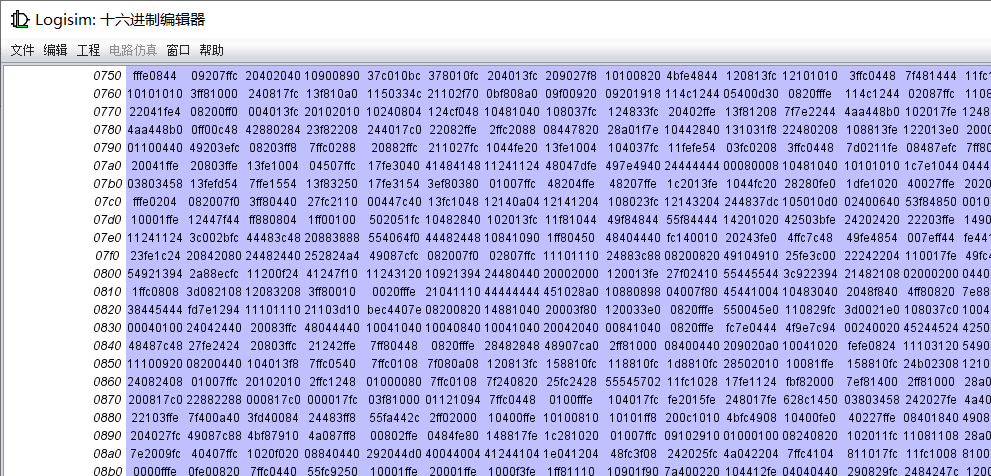


**电路引脚**

| **信号** | **输入/输出** | **位宽** | **说明** |
| --- | --- | --- | --- |
| 区号Qu | 输入 | 7 位 | 汉字区位码的区号 |
| 位号Wei | 输入 | 7 位 | 汉字区位码的位号 |
| Di | 输出 | 32 位 | 汉字点阵信息 |

1. **电路设计**

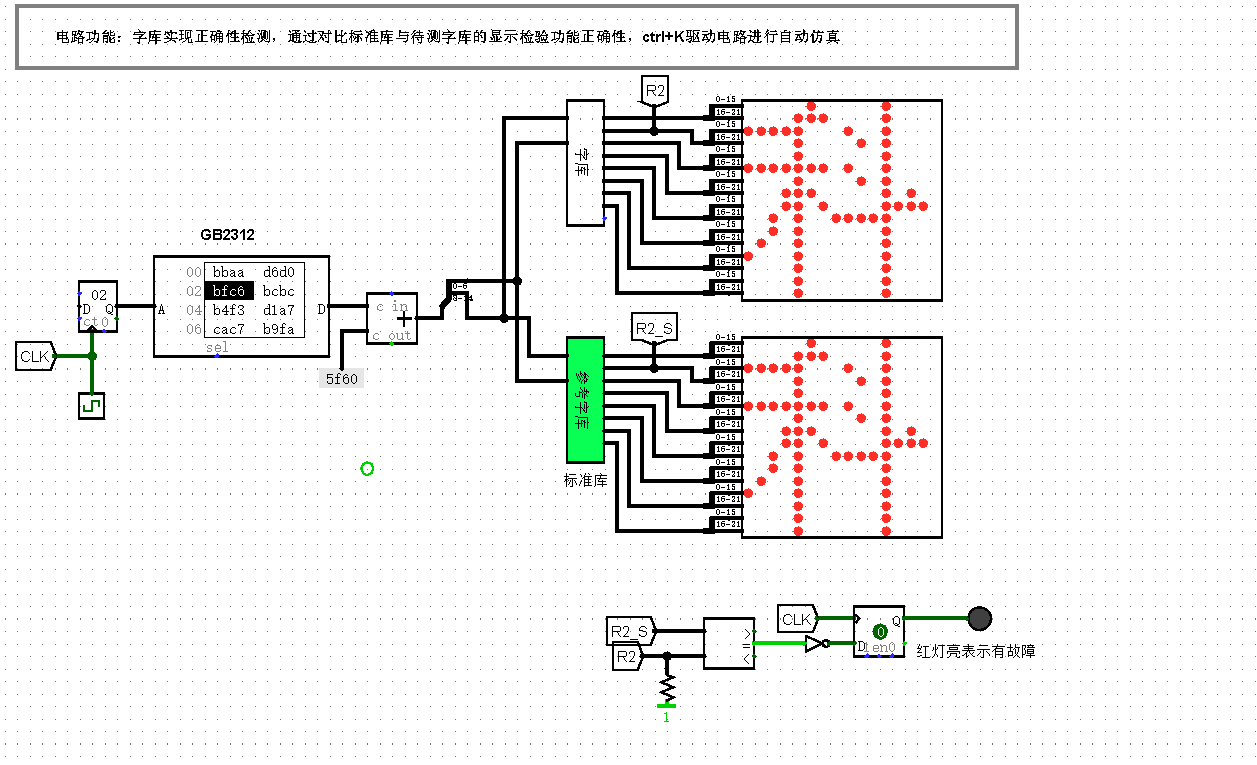
1.对4个ROM存储器的数据进行编辑，其内容为HZK16.1.txt，将此文本划分为4部分（0000-0ff0，1000-1ff0…）分别写入。



2.将4片ROM存储器进行级联，完成电路如下图所示：

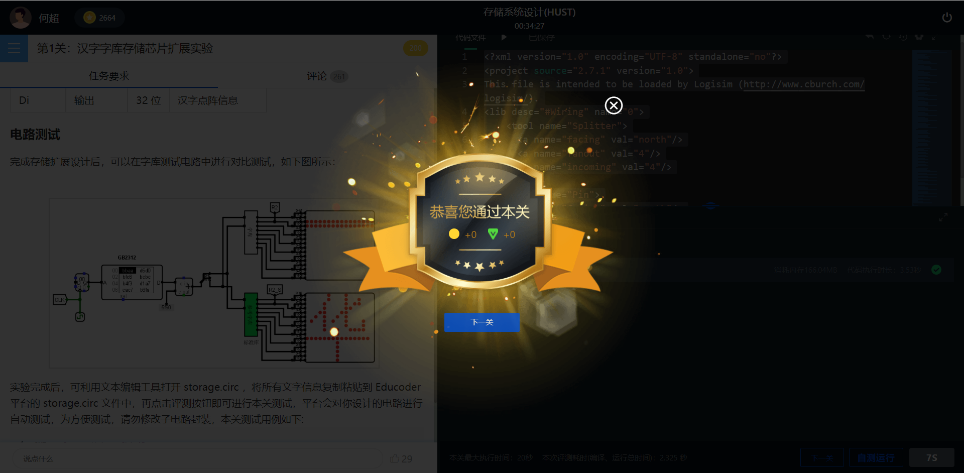
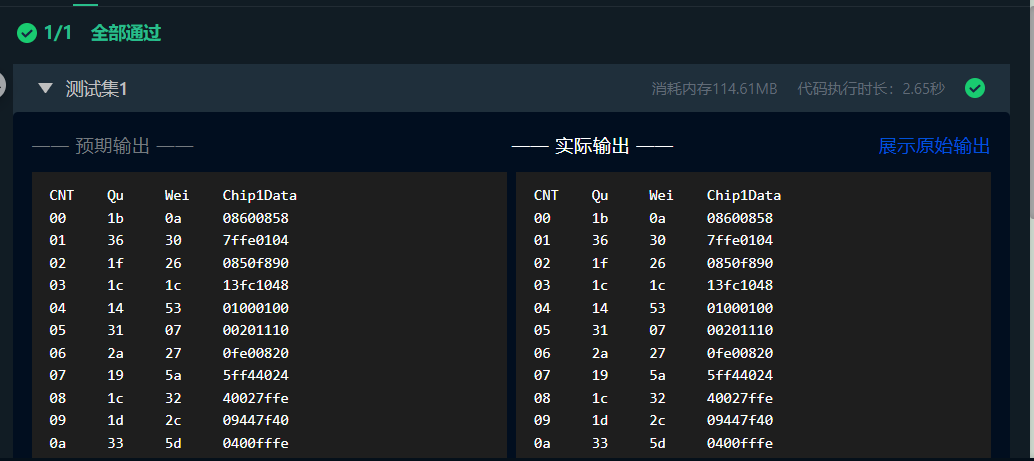


1. **实验结果**
2. 打开测试电路，进行测试，结果如下：

****

可以看到，字库电路设计正确

1. 将代码提交至平台评测：

****

1. **实验总结/体会**

通过本次实验，我了解到如何使用ROM芯片储存数据，比如将ROM作为汉字的字库，这其中涉及到了ROM拓展等知识，同时，使用测试电路验证拓展的成功与否也是让我比较惊奇的，值得研究探讨。本次实验中让我了解到ROM芯片的使用，操作，与级联的基本方法，并且进一步学习了一些逻辑电路基本元件的使用场景和具体使用方法。同时，我也更加了解了LOGISIM软件的使用。完成实验借助同学的帮助与互联网资料的查询将使知识点理解的更为具体透彻。为以后的实验课程打下了基础。

|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | 实验八、九 MIPS RAM设计 | | |
| **实验日期** |  | **实验地点** | J13-232 |
| **指导老师** | **刘连山** | **实验成绩** |  |

1. **实验目的**

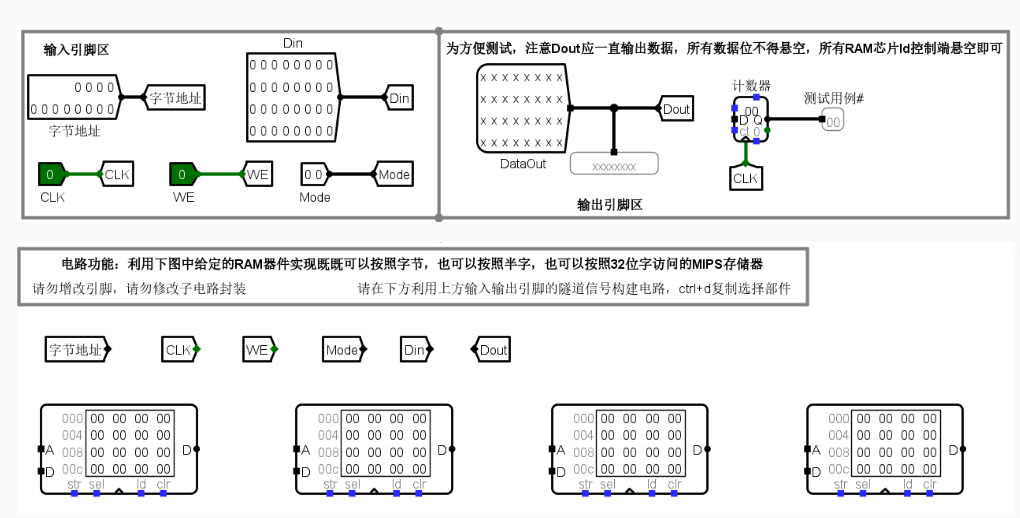
学生理解主存地址基本概念，理解存储位扩展基本思想，并能利用相关原理构建能同时支持字节、半字、字访问的存储子系统。

1. **实验内容**

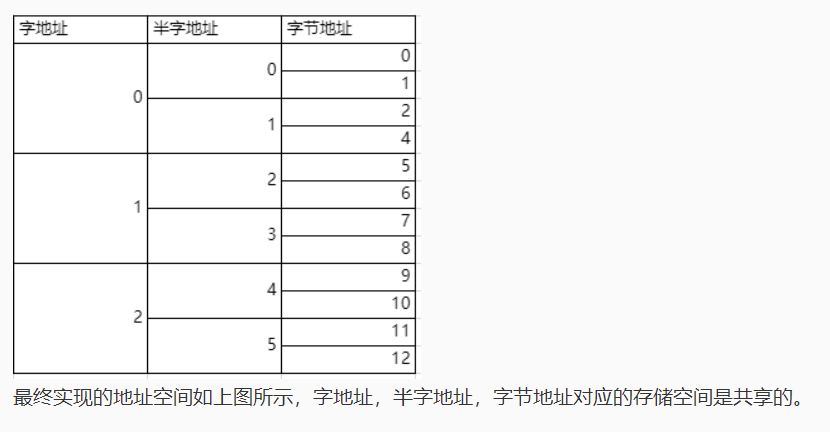
Logisim 中 RAM 组件只能提供固定的地址位宽，数据输出也只能提供固定的数据位宽，访问时无法同时支持字节/半字/字三种访问模式，实验要求利用4个8位的 RAM 组件进行扩展，设计完成既能按照8位、也能按16位、也能按照32位进行读写访问的32位存储器，最终存储器引脚定义如下图。

**电路框架**

alu.circ

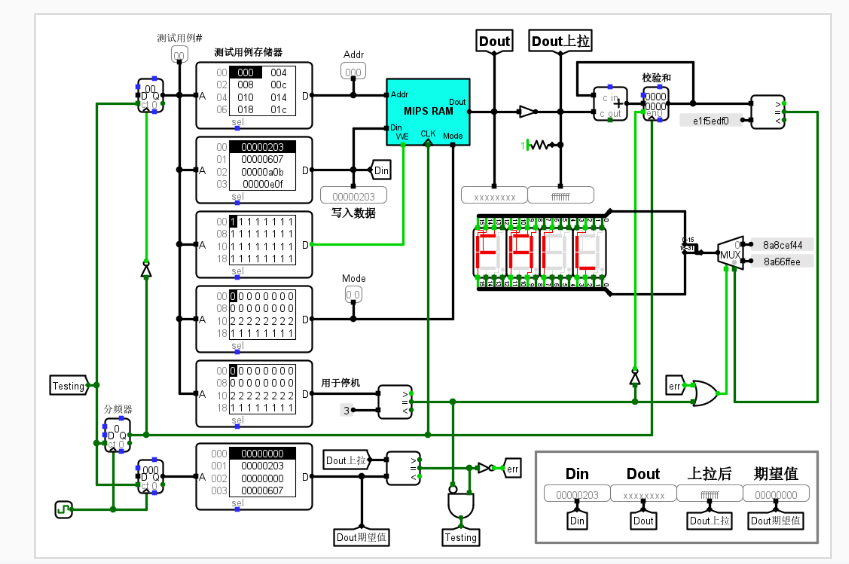
**电路引脚**

| **信号** | **输入/输出** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| Addr | 输入 | 12 | 字节地址输入（字访问时忽略最低两位，半字访问时忽略最低位，倒数第二位片选，字节访问时，低两位进行片选） |
| Din | 输入 | 32 | 写入数据 （不同访问模式有效数据均存放在最低位，高位忽略 |
| Mode | 输入 | 2 | 访问模式控制位（00 表示字访问，01 表示 1 字节访问，10 表示 2 字节访问） |
| WE | 输入 | 1 | 写使能，1 表示写入，0 表示读出 |
| Dout | 输出 | 32 | 读出数据 （不同访问模式有效数据均存放在最低位，高位补零）； |

****

**电路测试**

完成存储扩展设计后可以在自动测试电路中进行测试，电路通过会显示 Pass ，未能通过为停在出错位置，并显示 Fail ，如下图所示：

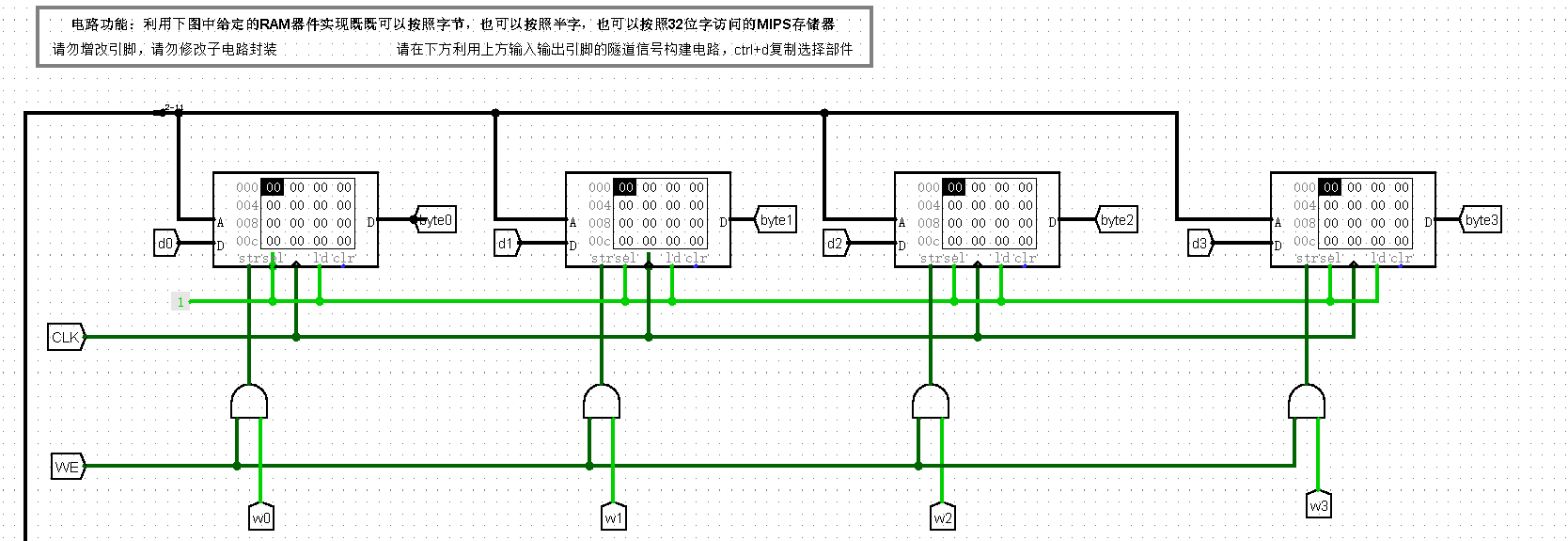


1. **电路设计**

**寄存器**

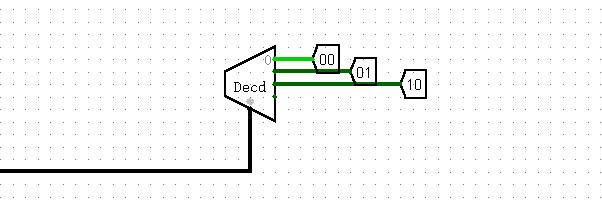
寄存器位宽为8，而最大单位字32，所以需要4片相同寄存器级联，分别为Byte1，2，3，4。将12位字节地址分出最低两位用来片选，接入寄存器。将d0，1，2，3作为数据输入。

构建此部分电路如下：（左上边为下方引出的字节地址线）



**设计电路将模式信号转换为位宽为1的信号**

采用译码器实现，如下：



**读取逻辑电路：**

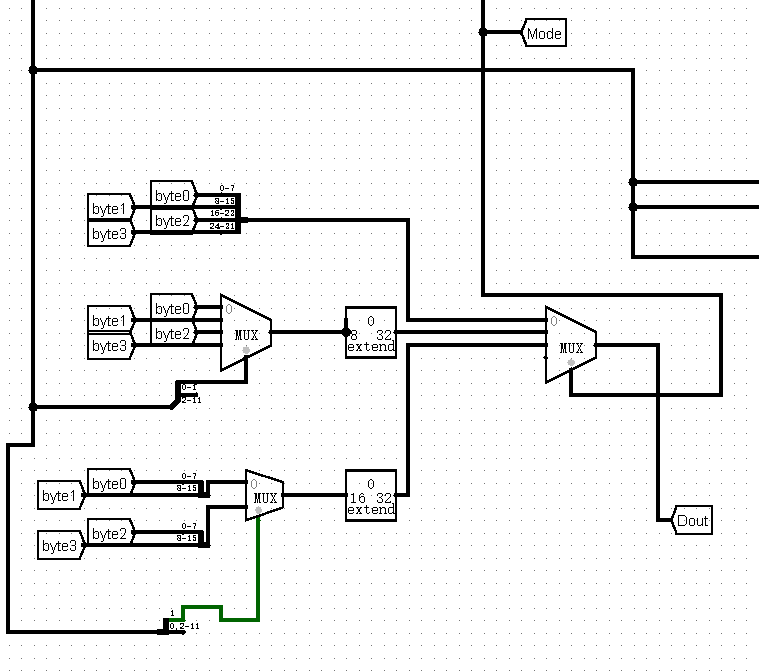
将ld信号设置为1，即恒为真，保持数据输出。

当需完成字读出时，需要输出全部数据。

当需完成半字读出时，需要字节地址的倒数第二位决定读出半字内容。

当需完成字节读出时，需要字节地址的倒数两位决定读出字节内容。

由此，搭建读取功能电路部分如下：



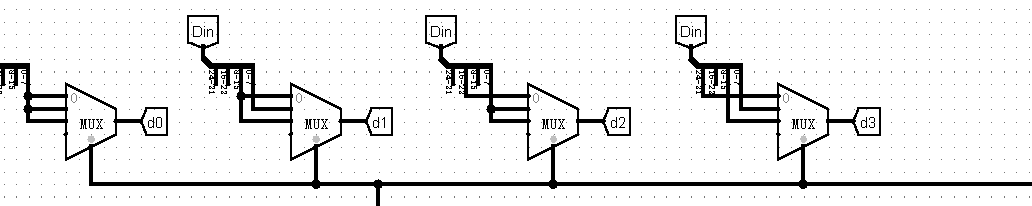
**写入逻辑电路：**

当mode==00时(字写入)，使0对应->7，1对应->15，2对应->23，3对应->31,从各个端口分别读取数据

当mode==01时(字节写入)，由写命令的片选信号挑选目的写入寄存器，将d0d1d2d3全部输入0-7位数据，

当mode==01时(半字写入)，会写入Din的0-15位数据，因为需要实现对齐，所以对于01，23号组合，较低编号都存放低位数据，较高编号存放高位数据，所以使d0，2输入0-7位数据，d1，3输入8-15位数据。

电路如下：



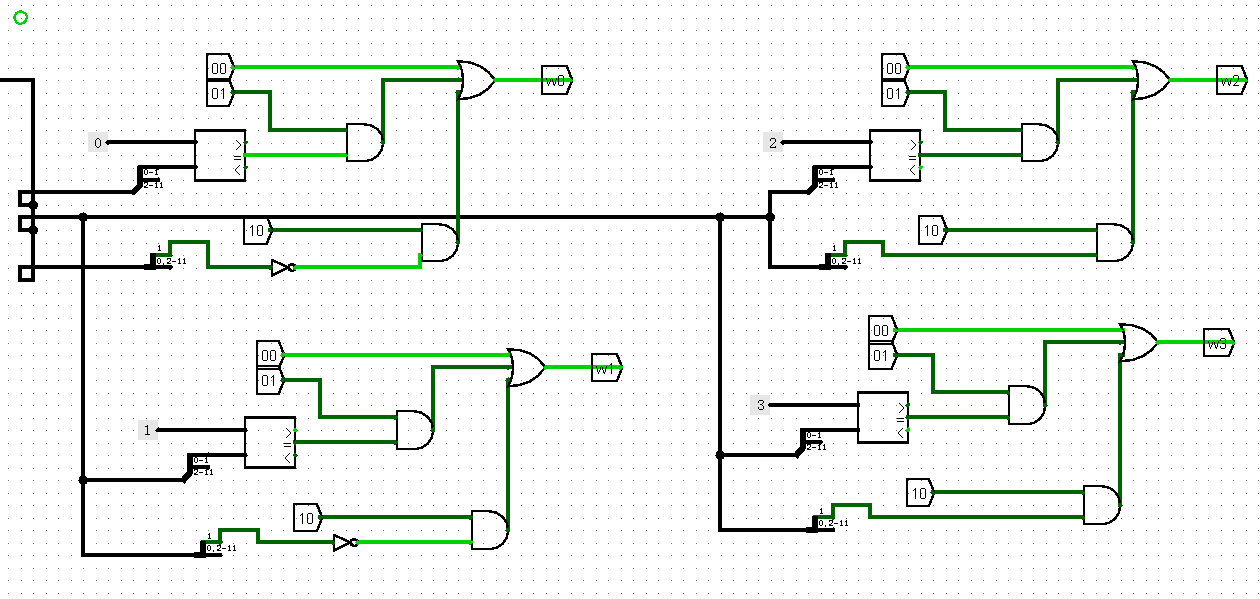
**写入片选逻辑电路：**

mode = 00时，所有片选信号都为1

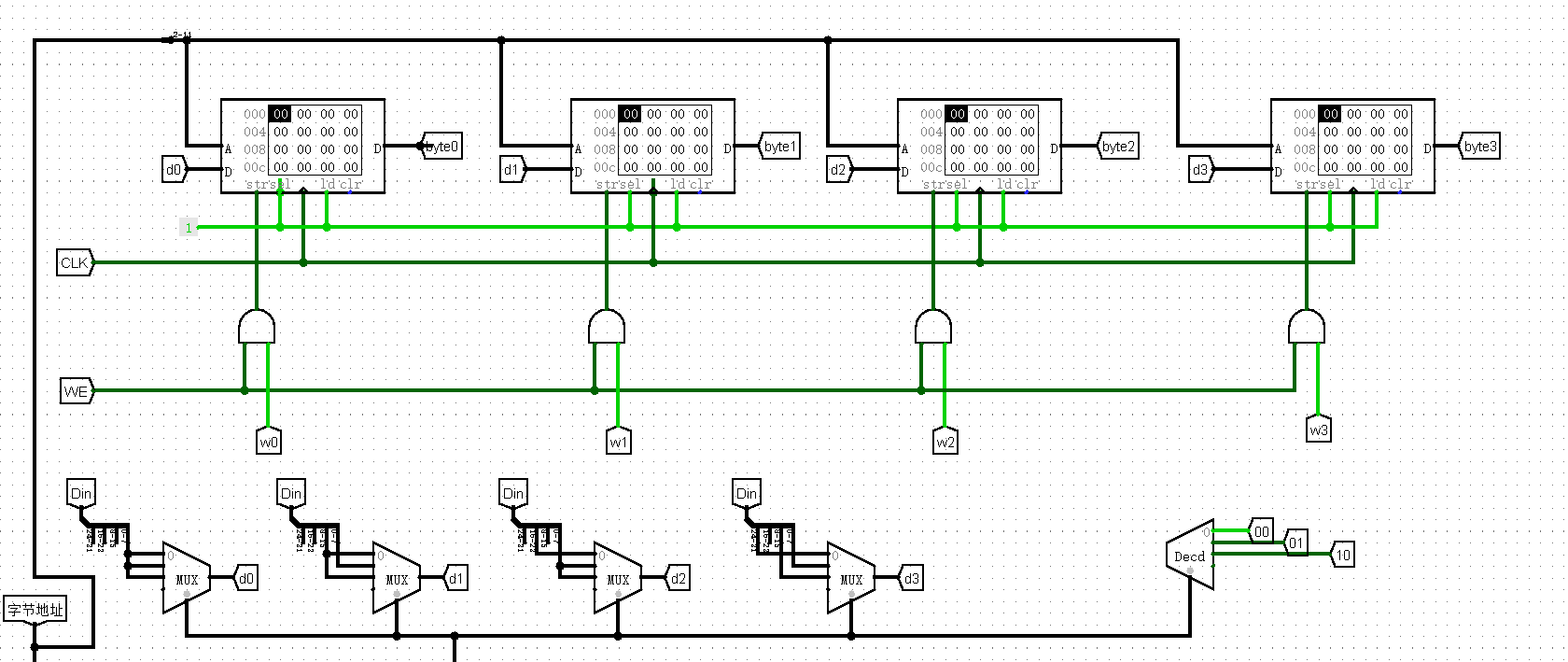
mode = 01时，字节地址最低两位和寄存器编号相对应的片选信号为1

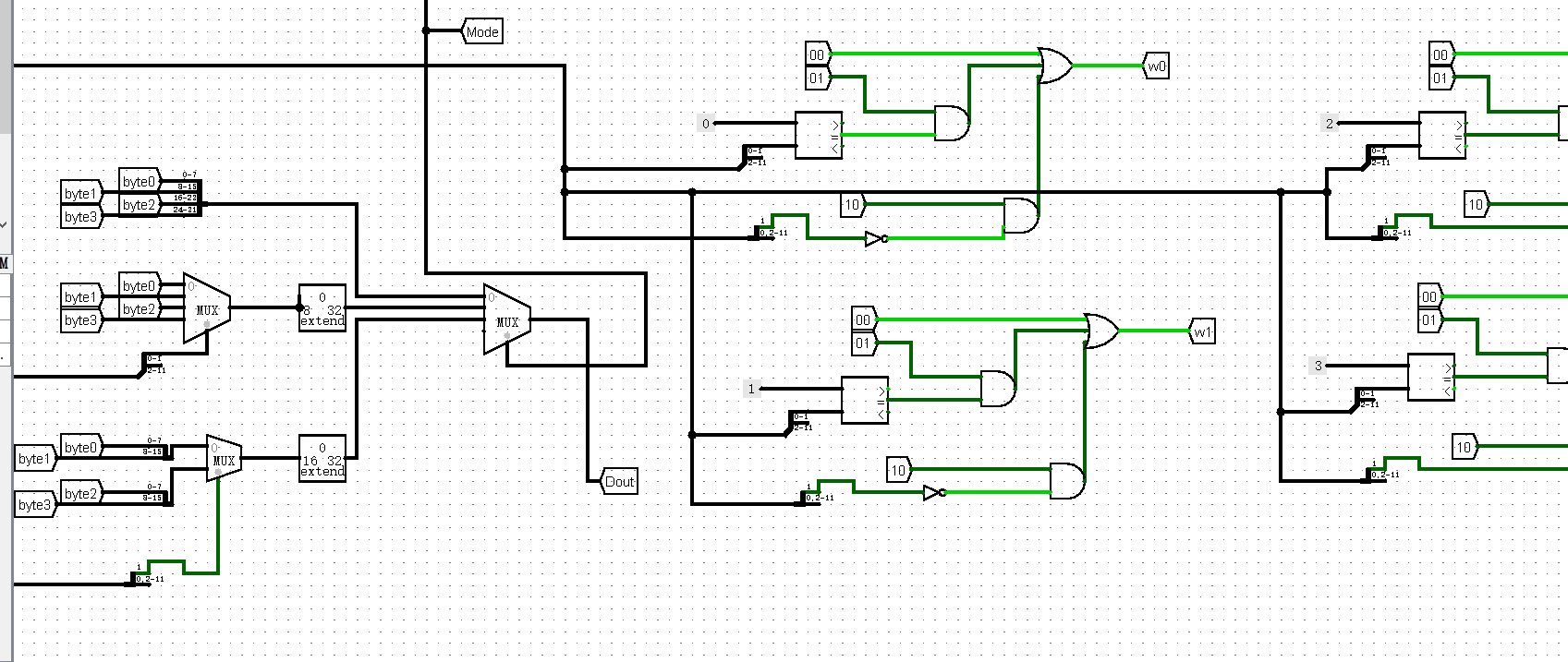
mode = 10时，字节地址的倒数第二低的一位来决定选择那两片。如果位0，选择01，选择23

设计电路如下：



**整体逻辑电路图：**

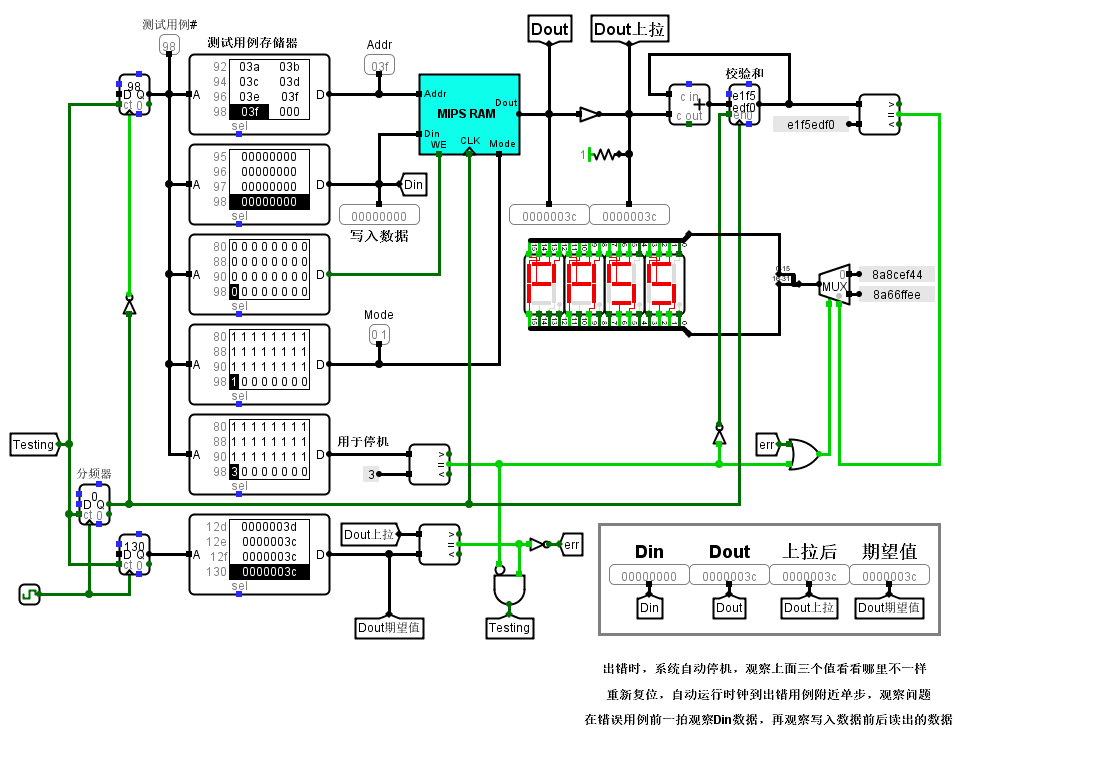




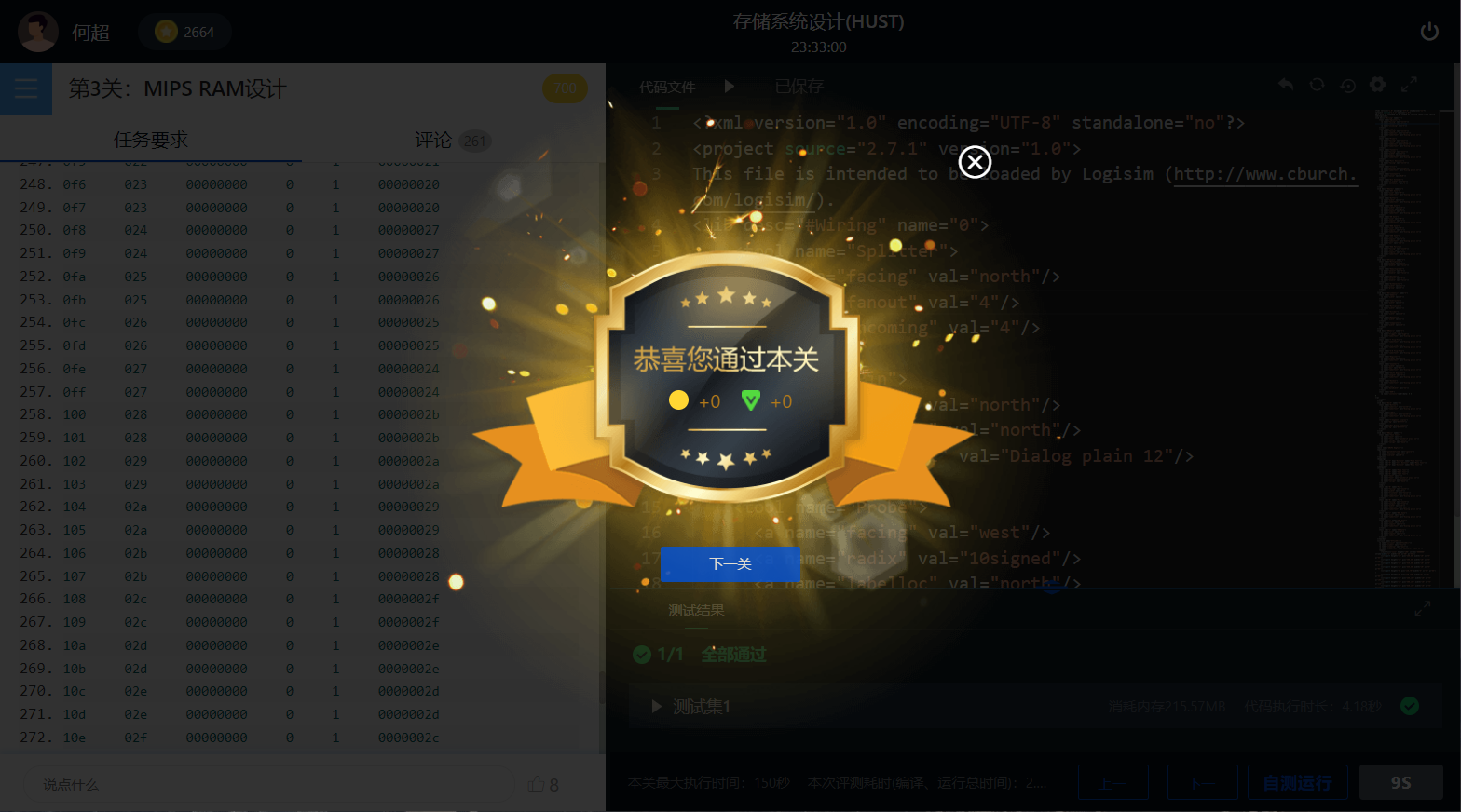
1. **实验结果**

**使用TEST电路按CTRL+K测试：**

**结果如下：**

****

**提交代码至平台：**

****

1. **实验总结/体会**

本次实验使我更加理解了存储位扩展基本原理，本次实验主要要求构建能同时支持字节、半字、字访问的存储子系统。本次实验电路规模较大，虽然电路通过了测试，但是没有做到模块化设计，直接观察自己设计的电路较难理解。但本次实验使用的部件较为简单基础，经过了本次实验，我收获了很多RAM设计相关基础知识，对Logisim有了更为深入的操作经验。在实验电路设计时，需要利用互联网查询陌生知识点，熟悉原理，以便于后续设计更为复杂的实验电路。

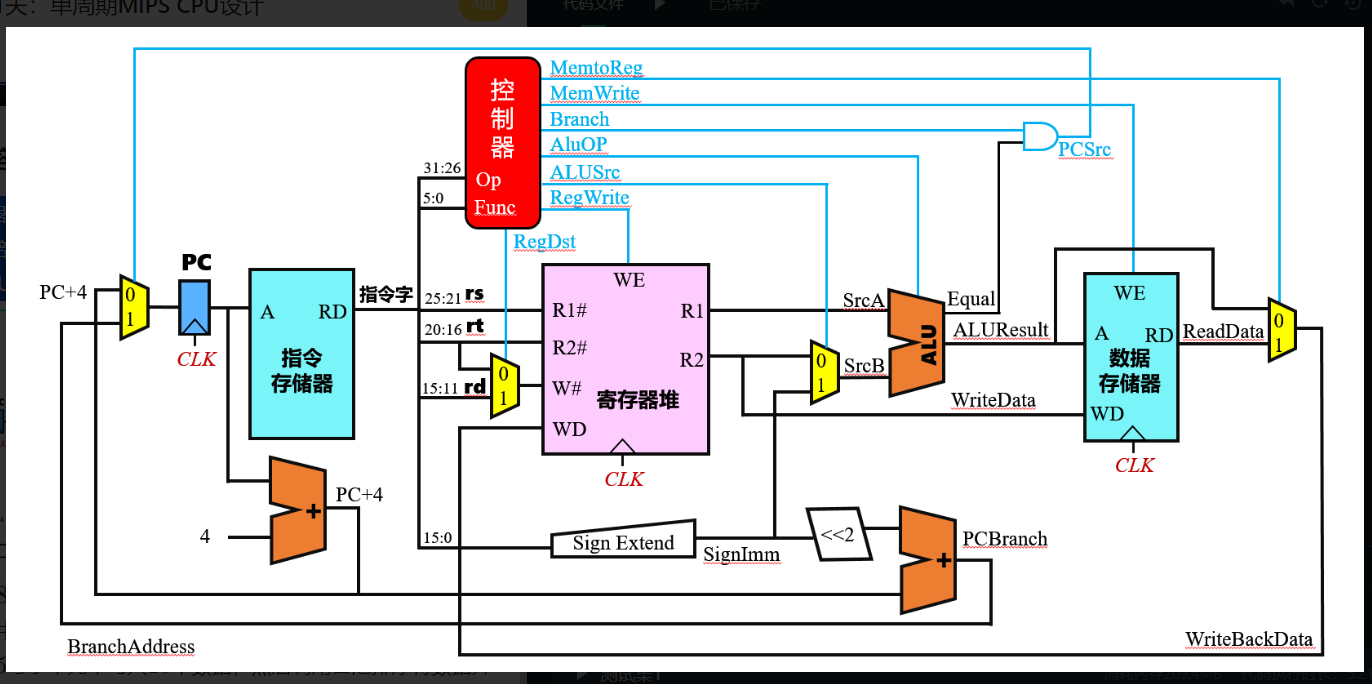
|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | 实验十、十一 单周期MIPS CPU设计 | | |
| **实验日期** |  | **实验地点** | J13-232 |
| **指导老师** | **刘连山** | **实验成绩** |  |

1. **实验目的**

学生掌握控制器设计的基本原理，能利用硬布线控制器的设计原理，在 Logisim 平台中设计实现 MIPS 单周期 CPU。

1. **实验内容**

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：

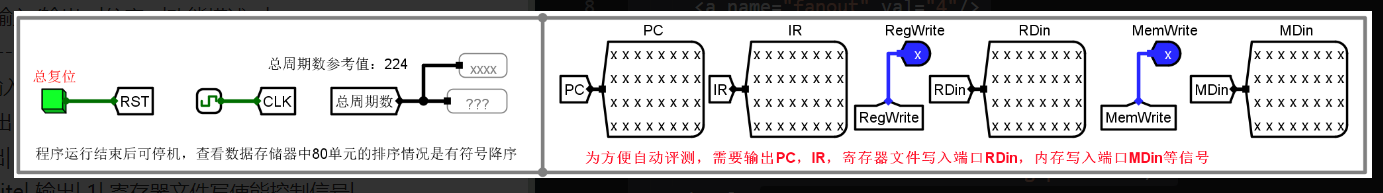


要求支持8条 MIPS 核心指令，最终设计实现的 MIPS 处理器能运行实验包中的冒泡排序测试程序 sort.asm，该程序自动在数据存储器0~15号字单元中写入16个数据，然后利用冒泡排序将数据升序排序，要求统计指令条数与 MARS 中的指令统计数目进行对比。



**电路框架**

cpu.circ



**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|程序寄存器的值|  
| 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|

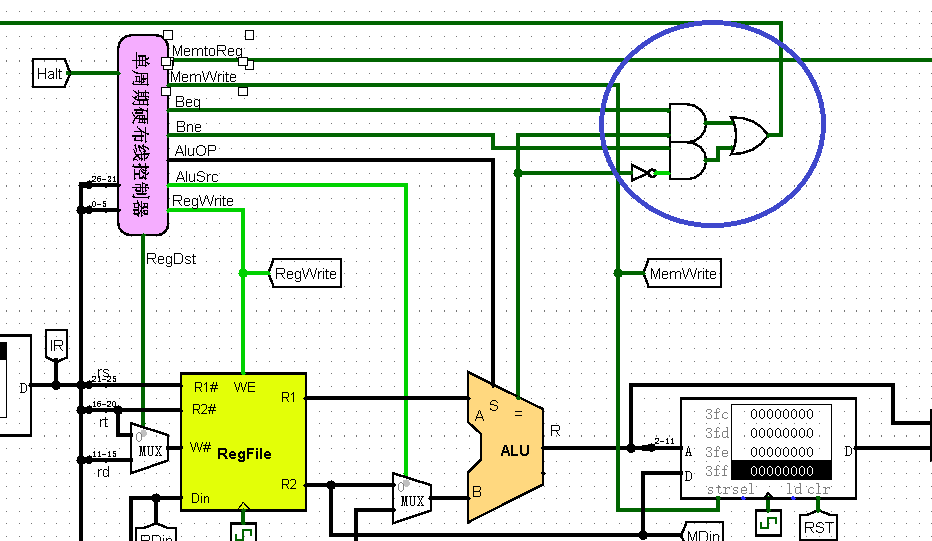
1. **设计电路**
2. **分离不同类型指令**

使用分线器，将不同类型的指令的不同段分离，如下：

****

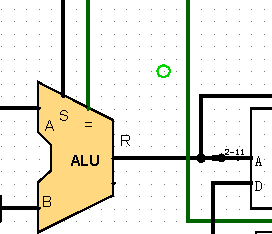
1. **设计PCsrc信号**

结合硬布线控制器，可知Beq和Bne对应数据通路中的Branch信号，Beq和Bne只要有一个有效时PCsrc为真，因此可设计出如下的逻辑电路：



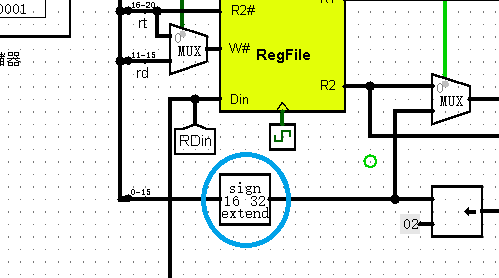
1. **设计输出储存器的地址码：**

32位的输入信号经过分线器的选择，均选取2-11位，即低2位不用，高20不用。MIPS指令的字长位定长的32位即4字节，所以指令计数器形式上的PC=PC+1实际上是PC=PC+4，而，所以低两位不会变化故不取。而储存器的容量为1024，所以只取2-11位。

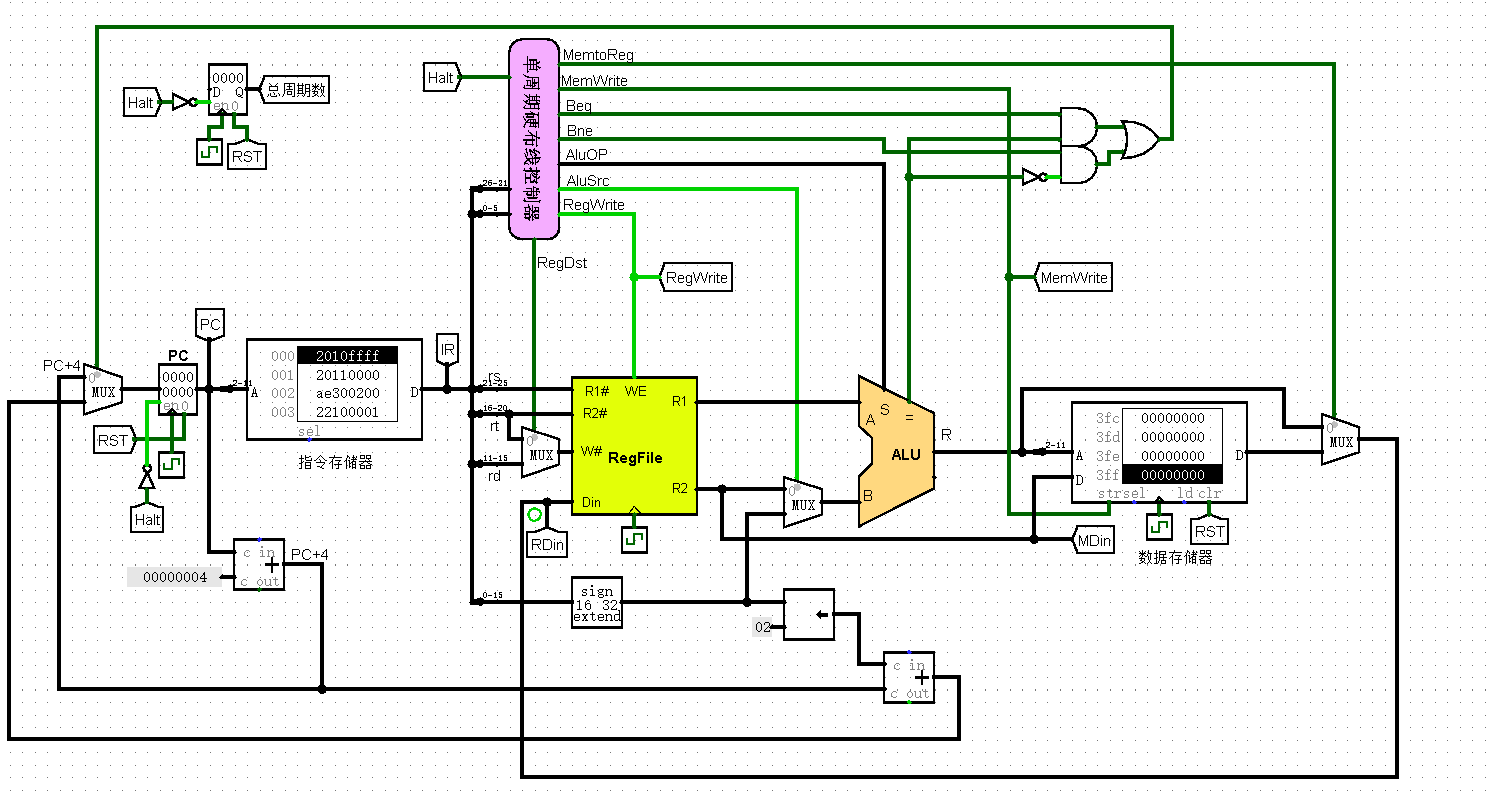
****

**4、设计ALU输入端**

根据数据通路以及所学知识可得，需要将I型指令的imm立即数带符号扩展到32位才能做运算。如下：



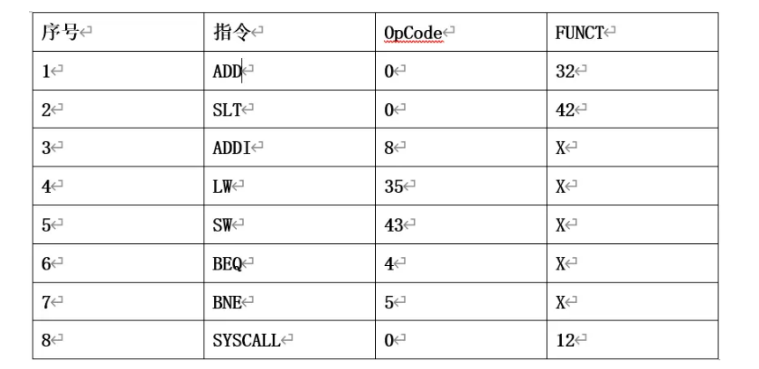
**则总的数据通路：**

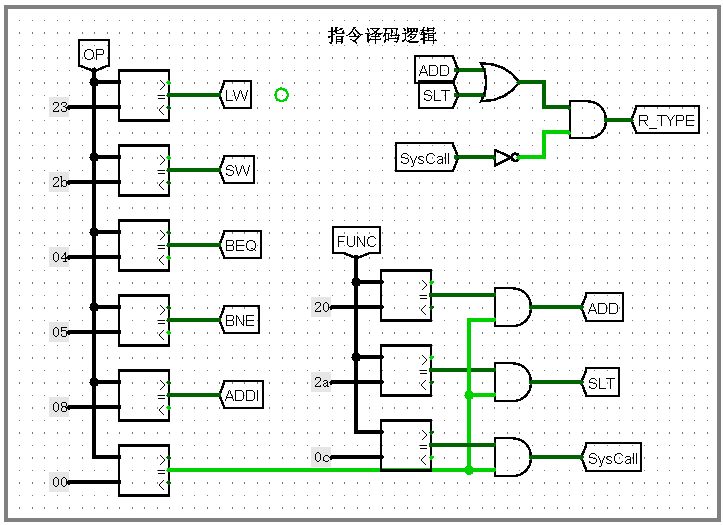
****

**5、设计单周期硬布线控制器**

**5.1 指令译码逻辑**

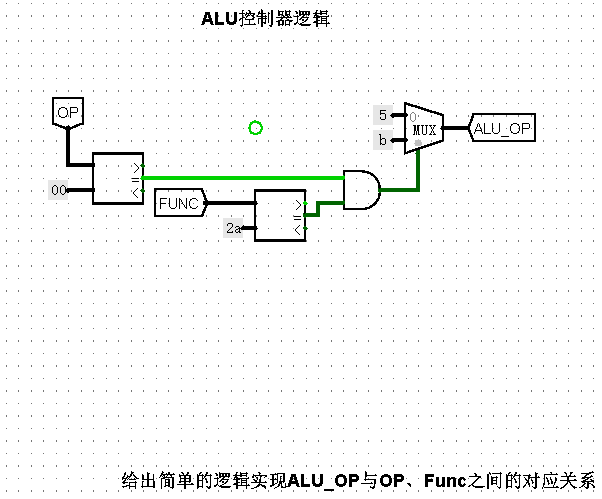
根据以上指令的OpCode字段和FUNCT字段结合logism自带的比较器，可以设计出指令译码逻辑部分，其中SysCall输入特殊的R型指令，所以R\_TYPE指令形成的信号中SysCall要取反再与其他R型指令取或的结果做与。

****

****

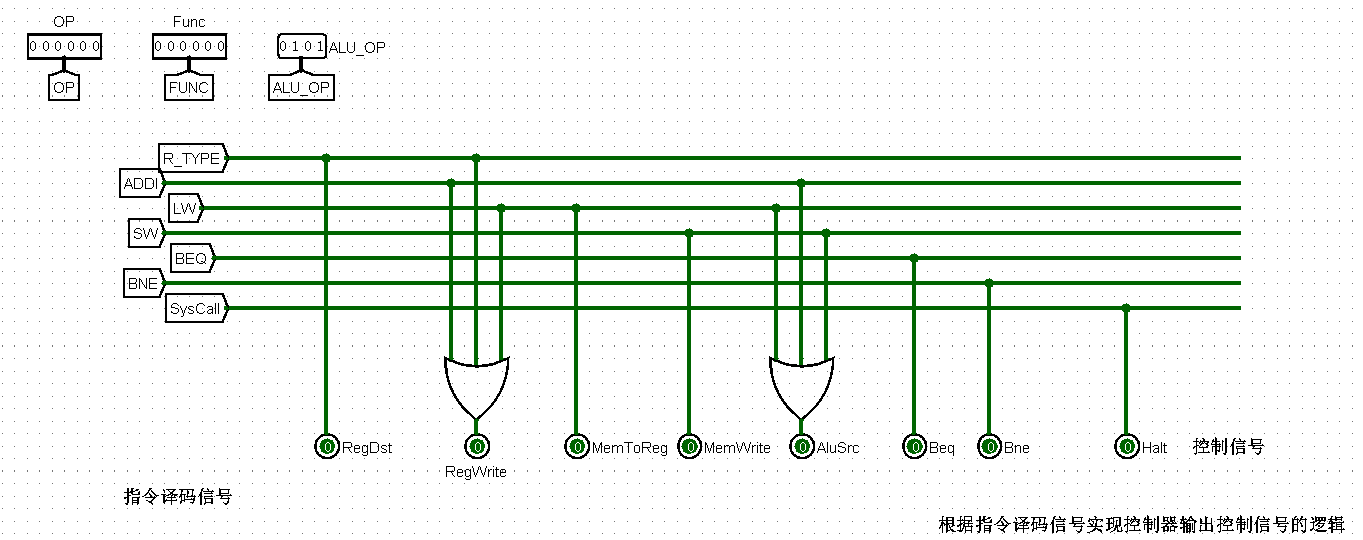
**5.2** **ALU控制逻辑**

由指令控制逻辑，可得只有SLT需要用到ALU的比较运算，其他均为加法运算，所以可以直接复制一个SLT的隧道加上数据选择器，直接设计ALU的控制逻辑，不需要用到给定的OP和FUNC字段的隧道。

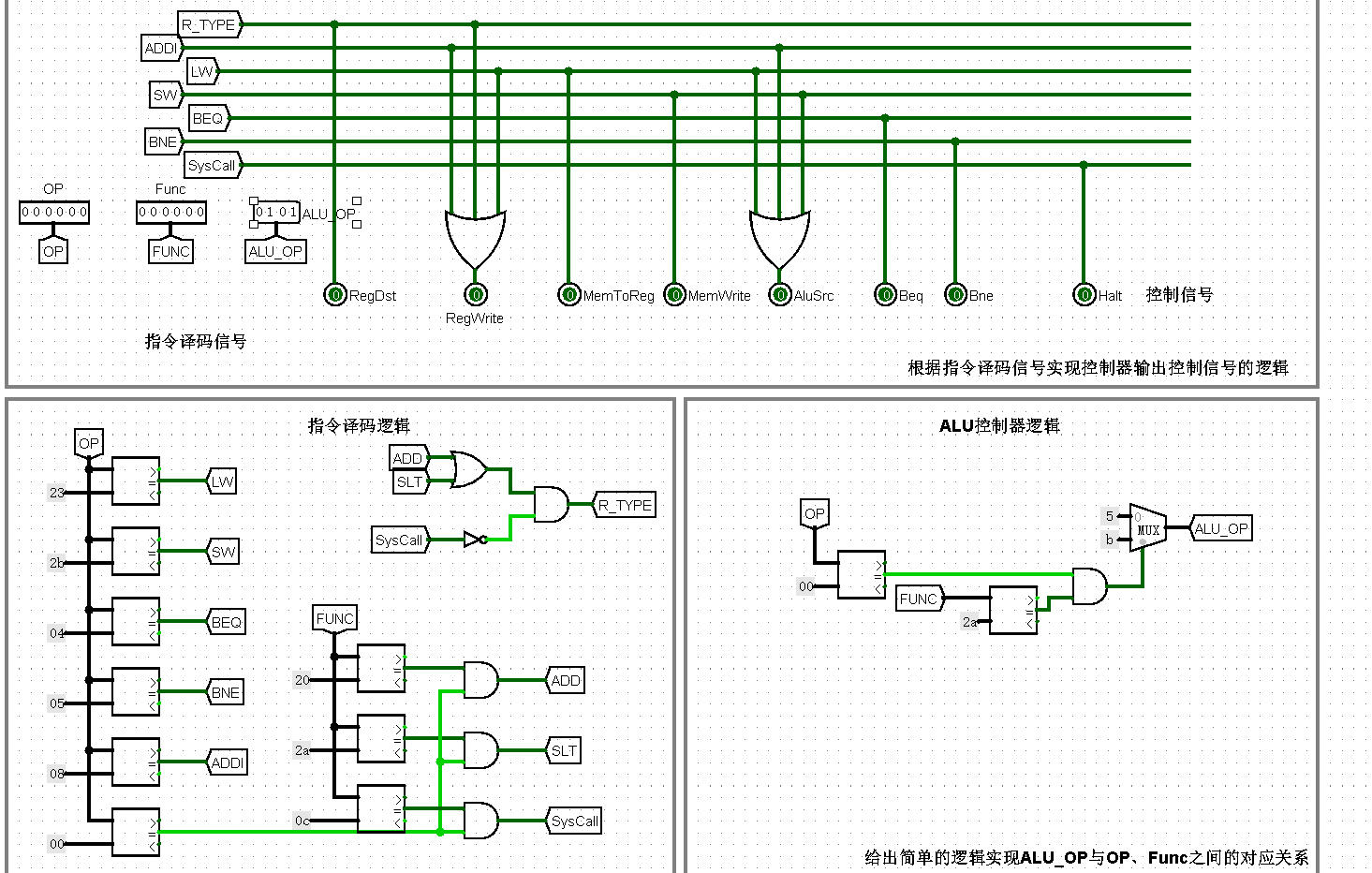


**5.3控制机器输出信号**

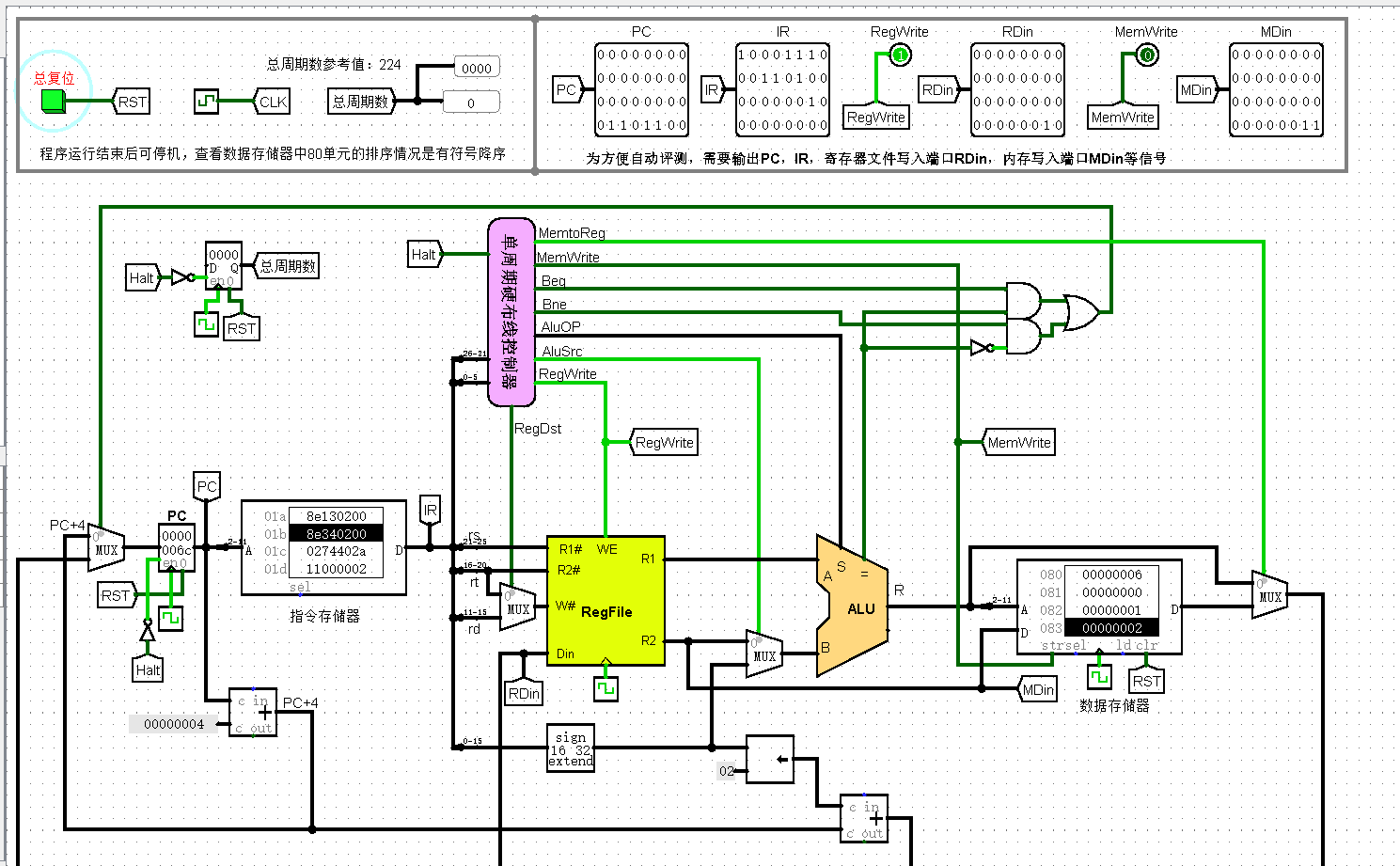
根据所给出控制信号的产生条件，结合数字逻辑知识，可以设计输出控制信号的逻辑

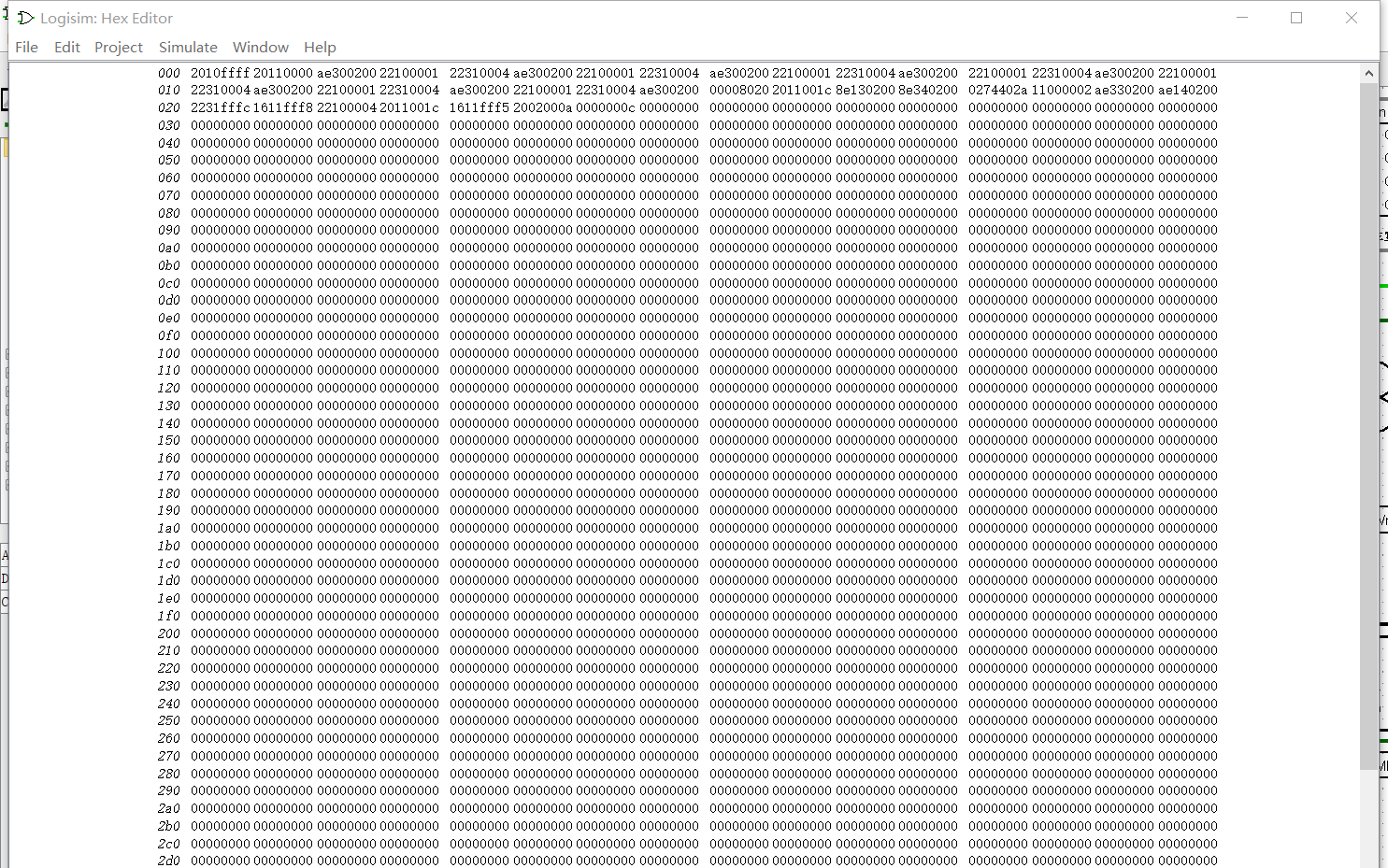
****

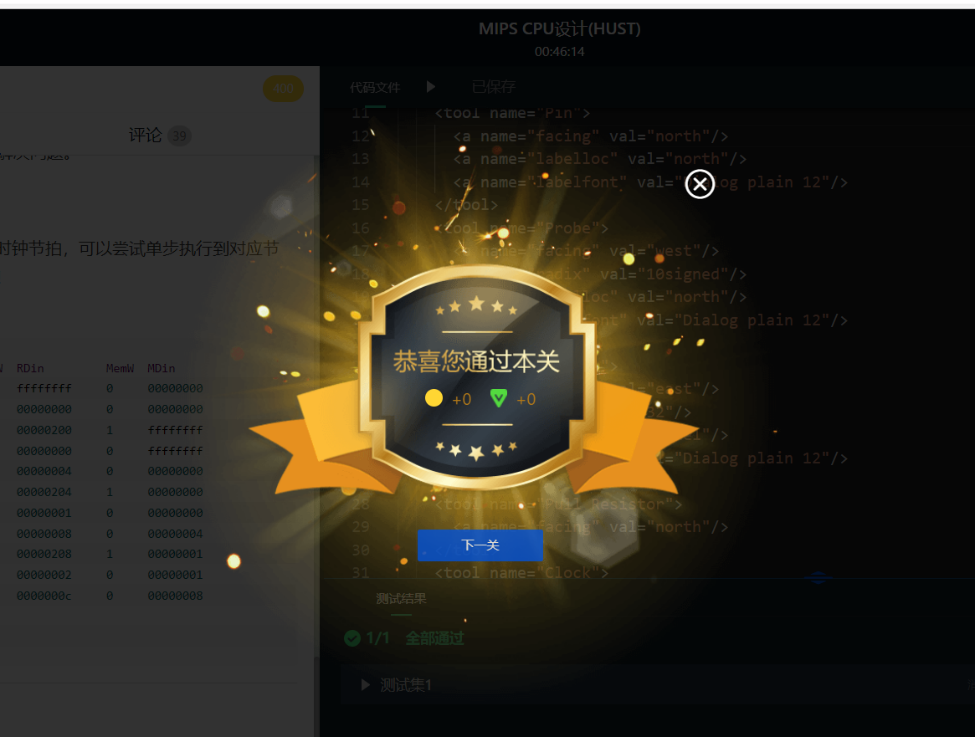
**单周期硬布线控制器总设计如图：**

****

1. **实验结果**

****

****



1. **实验总结/体会**

本次实验设计了单周期MIPS CPU，借助给定的数据通路，可以很快的连接好大致框架，其中PCsrc和输入储存器的地址段的选择是要解决的关键问题。PCsrc要根据数据通路中所给的信息，结合实验所给出的硬布线控制器基本框架，对已知数据通路进行适当的扩展，才能正确输出PCsrc信号。而输入储存器的地址则应结合RAM有关实验以及PC指令计数器在MIPS中是PC=PC+4，才能顺利的选出正确的十位地址码。在设计机器输出信号的时候应注意所给的提示，比如SysCall不在正常的R型指令中，所以设计R型指令信号的时候要对其进行取反处理。ALU控制器的设计时，只需要设计能选择两种运算方式的控制器即可，故不用所给的OP和FUNC隧道，直接选择SLT隧道加上数据选择器设计一个简洁的SLT控制器。