華中科技大學

课 程 实 验 报 告

课程名称： 计算机组成原理课程实验

实验三

专业班级： 自实 1901

学　　号： U201915560

姓　　名： 肖力文

报告日期： 2021 年 12 月 08 日

人工智能与自动化学院

目录

实验三 存储系统实验

1 实验目的

熟悉 Logisim 软件平台;
熟悉 ROM、RAM 存储器的使用;
掌握存储器字扩展，位扩展的基本原理;
为 MIPS CPU 设计功能部件---寄存器文件;
进一步熟悉流水传输控制基本远离;

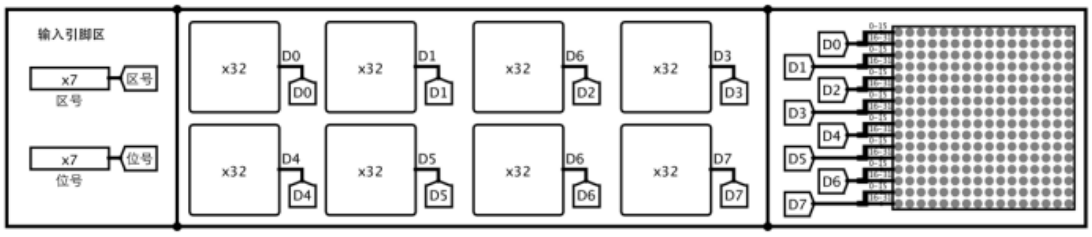
2 实验环境

Logisim 是一款数字电路模拟的教育软件，用户都可以通过它来学习如何创建逻辑电路，方便简单。 它是一款基于 Java 的应用程序，可运行在任何支持 JAVA 环境的平台，方便学 生来学习设计和模仿数字逻辑电路。Logisim 中的主要组成部分之一就在于设计并以图示来 显示 CPU。当然 Logisim 中还有其他多种组合分析模型来对你进行帮助，如转换电路，表达 式，布尔型和真值表等等。同时还可以重新利用小规模的电路来作为大型电路的一部分。

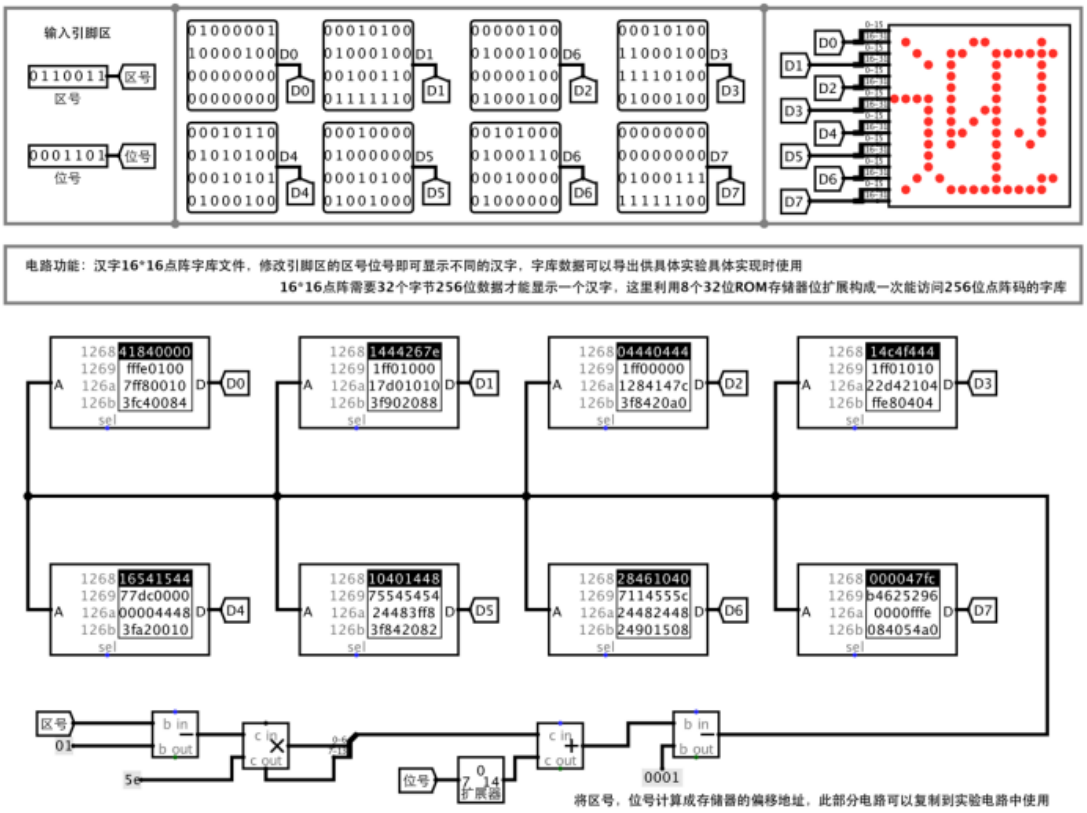
3 实验内容

3.1 汉字字库存储芯片扩展实验

实验目的：掌握存储扩展基本原理。
实验内容：设计字库文件，利用指定规格存储器进行存储器字扩展。
实验要求：现有如下 ROM 部件，4 个 4K*32 位 ROM，7 个 16K*32 位 ROM，请构建 GB2312 16*16 点阵字库存储器电路，电路输入为汉字区号和位号，由于 16*16 点阵的字模码需要 256 位点阵信息才能显示一个汉字，所以电路输出为 8*32 位（256 位点阵信息），实验电路 输入输出引脚如下图：



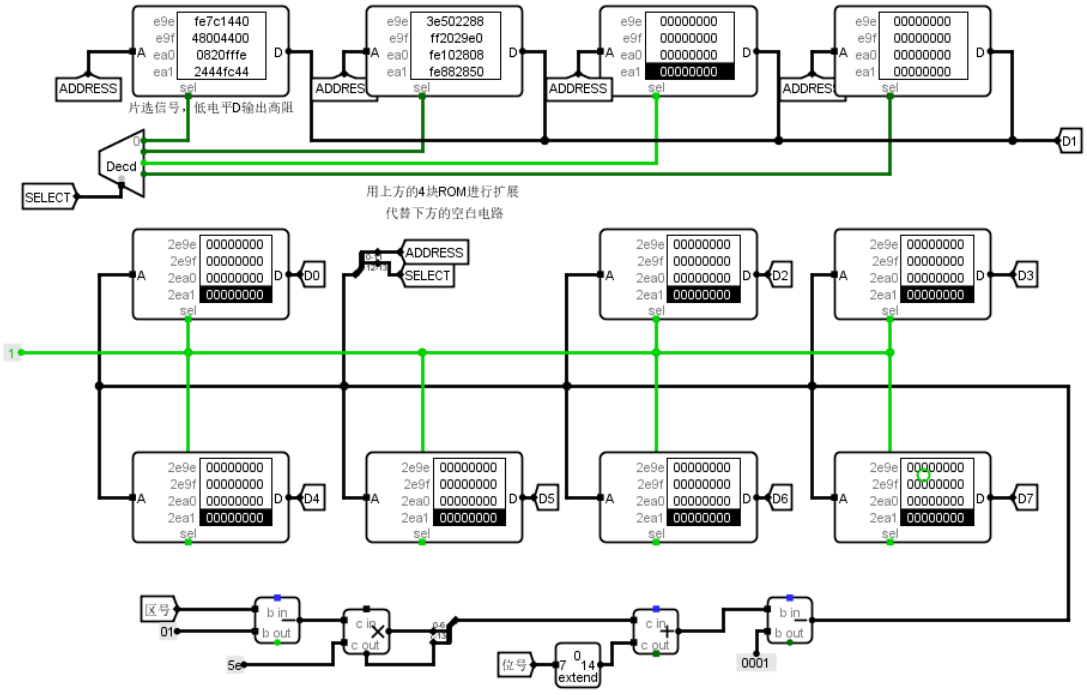
本实验的主要目的是进行存储器字扩展（容量扩展，地址总线扩展），故实验工程文件中 已经提供了一个参考实现，完成实验所需的点阵信息均可以通过该电路直接导出后载入，也可直接复制拷贝，区位码转存储器地址的电路也可一并参考使用。



3.1.1 原理

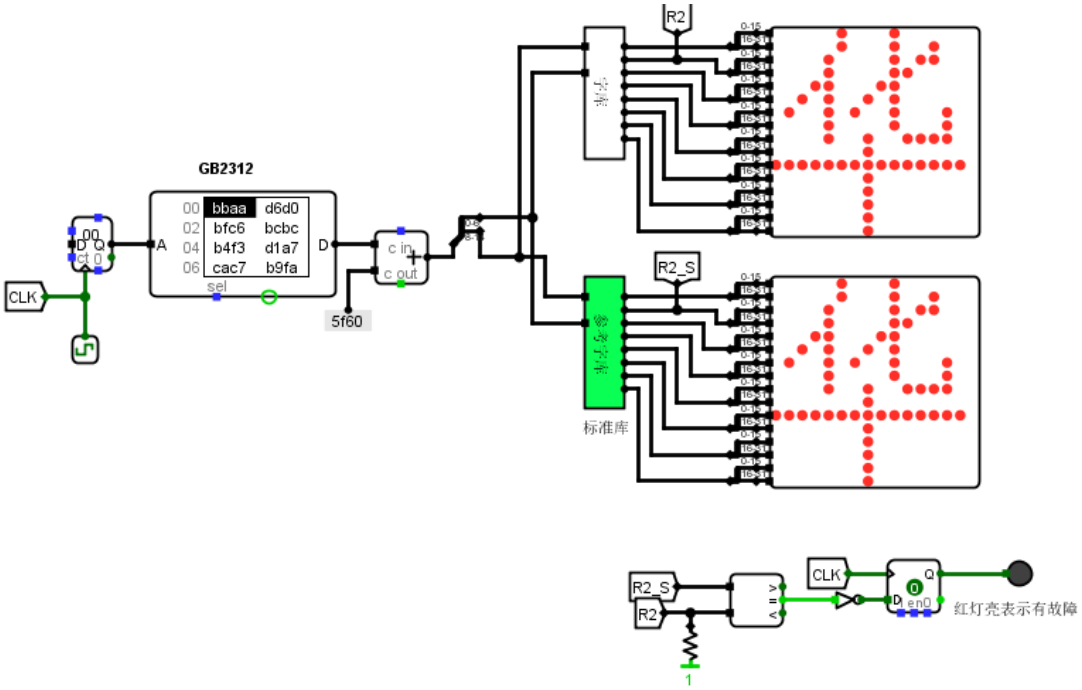
①这个实验的核心是位扩展
②地址高 2 位用来选择小容量 ROM：将地址高 2 位通入解码器，得到片选信号，将片选信号连接到小容量 ROM 的使能端，便实现了片选功能。
③剩下的地址位连接到小容量 ROM 的地址端，实现信息的读取

3.1.2 电路



3.1.3 测试

在“字库测试”模块测试电路的正确性，如图：



电路功能正常

3.2 MIPS RAM 设计

实验目的：熟练掌握存储扩展基本原理，进一步熟悉片选机制。

实验内容：计算机中主存储器通常即能按照字节访问也能按照半字访问，还能按照字进 行访问，如 MIPS 指令中的 LB/SB 指令（Load/Store byte）、LH/SH 指令（Load/Store Half），LW/SW 指令（Load/Stire Word）。X86 指令中 mov eax/ax/ah,[200]，而 logisim 中 RAM 存储器只能按 照一种模式访问，为此本实验要求设计完成既能按照 8 位，也能按 16 位，也能按 32 位进行 读写访问的 32 位存储器，最终存储器规格如下：

①字节地址 12 位（字访问时，忽略低两位，半字访问，忽略最低位，倒数第二位片 选，字节访问时，低两位进行片选）

②数据线宽度 32 位

③访问 Mode 位：2 位：00 表示字访问，01 表示 1 字节访问，10 表示 2 字节访问

④WE： 写使能，1 表示写入，0 表示读出

⑤Din： 32 位，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）

⑥Dout：32 位，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） 实验电路输入输出引脚如下图所示：

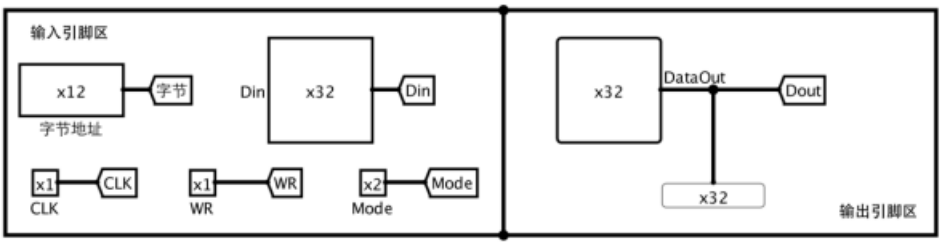
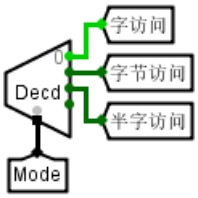


图 4 MIPS RAM 输入输出引脚

3.2.1 原理

（1）

通过 Mode 得到字访问，字节访问，半字访问的信号



（2）

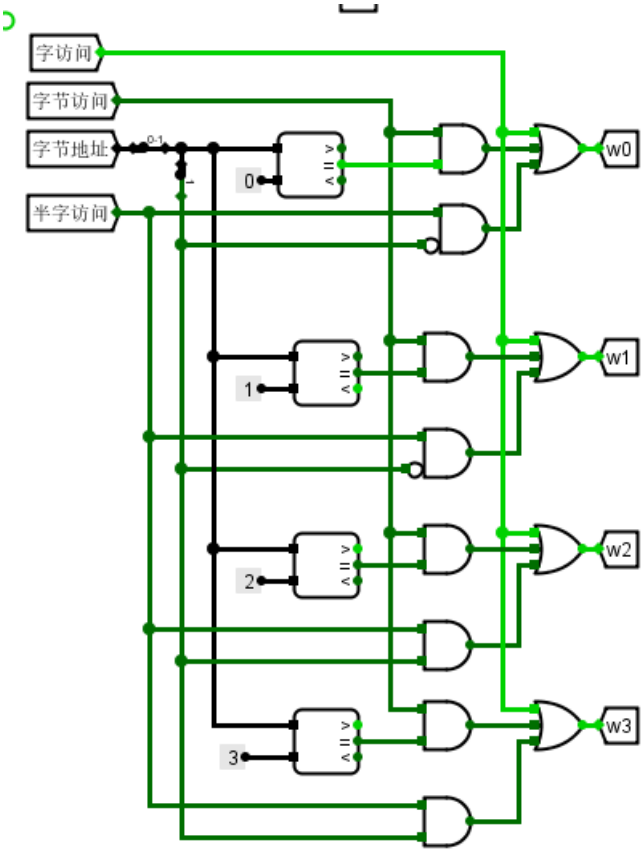
在不同的模式下，结合字节地址，得到四个寄存器的选中信号：

①在字访问模式下，四个寄存器均被选中

②在半字访问模式下，有两个寄存器被选中，结合字节地址，便可以使得被选中的寄存器对应的选中信号为 1

③在字节访问模式下，有 1 个寄存器被选中，结合字节地址，便可以使得被选中的寄存器对应的选中信号为 1

如图：



（3）

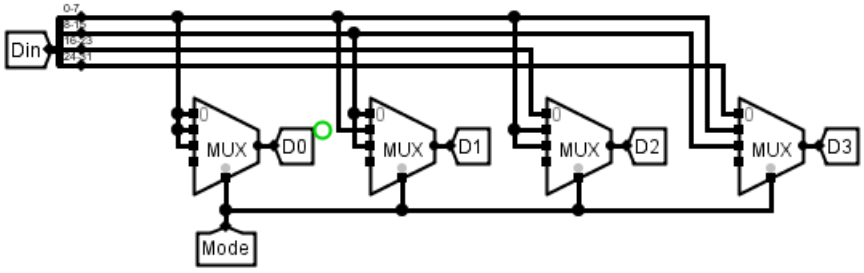
在写入时，在不同模式下，寄存器得到的输入数据的高低位不同：

①在字访问模式下，D0 得到输入数据的低八位，D1 得到输入数据的 8-15 位，D2 得到输入数据的 16-23 位，D3 得到输入数据的高八位

②在半字访问模式下，D0 得到输入数据的低八位，D1 得到输入数据的 8-15 位，D2 得到输入数据的低八位，D3 得到输入数据的 8-15 位

③在字节访问模式下，D0 得到输入数据的低八位，D1 得到输入数据的低八位，D2 得到输入数据的低八位，D3 得到输入数据的低八位

如图：

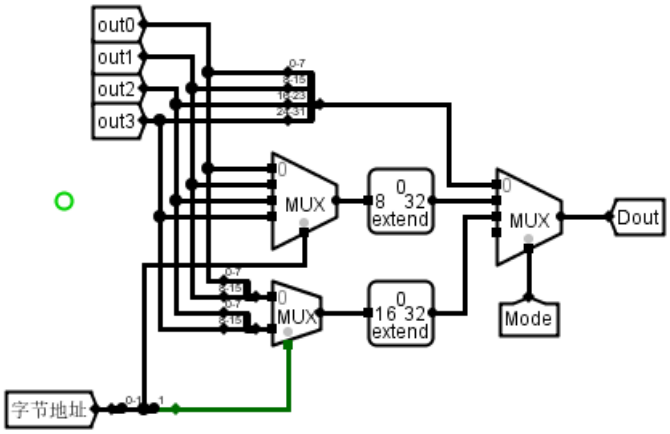


（4）

在读模式下，不同的模式下结合字节地址会得到不同的寄存器的数据：

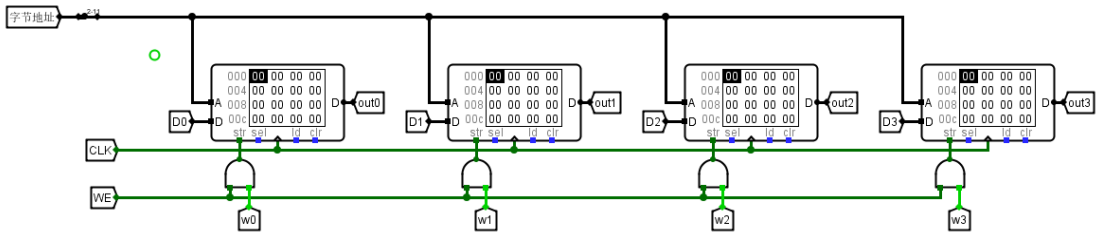
①在字访问模式下，输出数据为四个寄存器输出数据的组合
②在半字访问模式下，输出数据为 4 个寄存器中的 2 个寄存器数据的组合，结合字节地址决定是 01 组合还是 23 组合
③在字节访问模式下，输出数据为 4 个寄存器中的 1 个寄存器数据的组合，结合字节地址决定是哪一个寄存器中的值

如图：

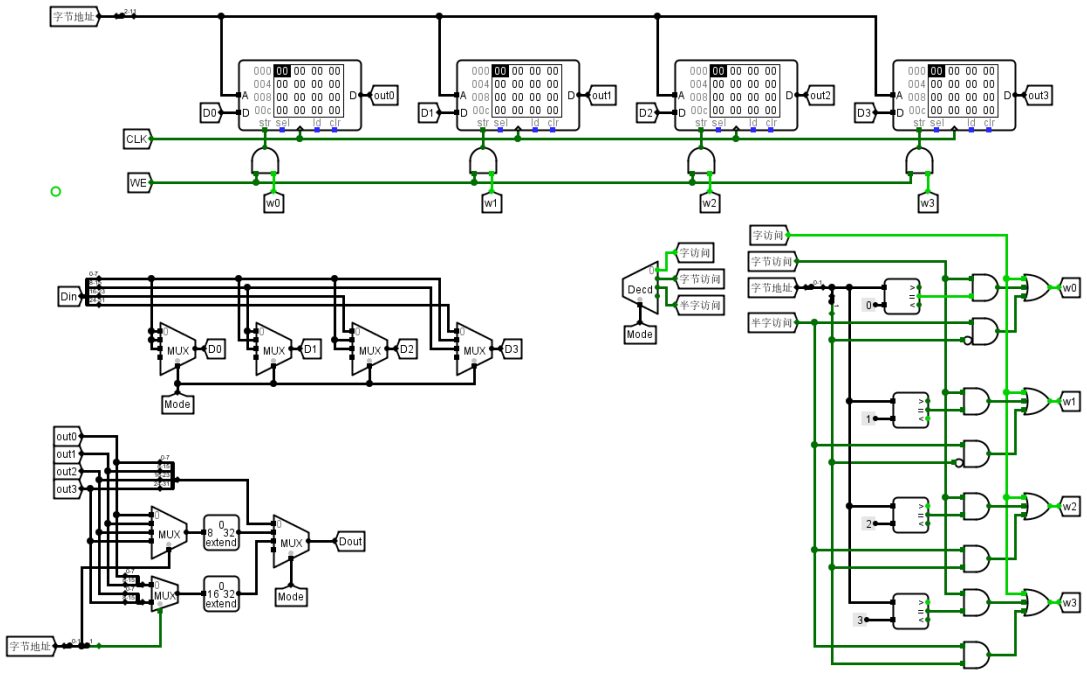


（5）
当写使能信号和写选中信号均为 1 时，选中的寄存器可以被写入数据

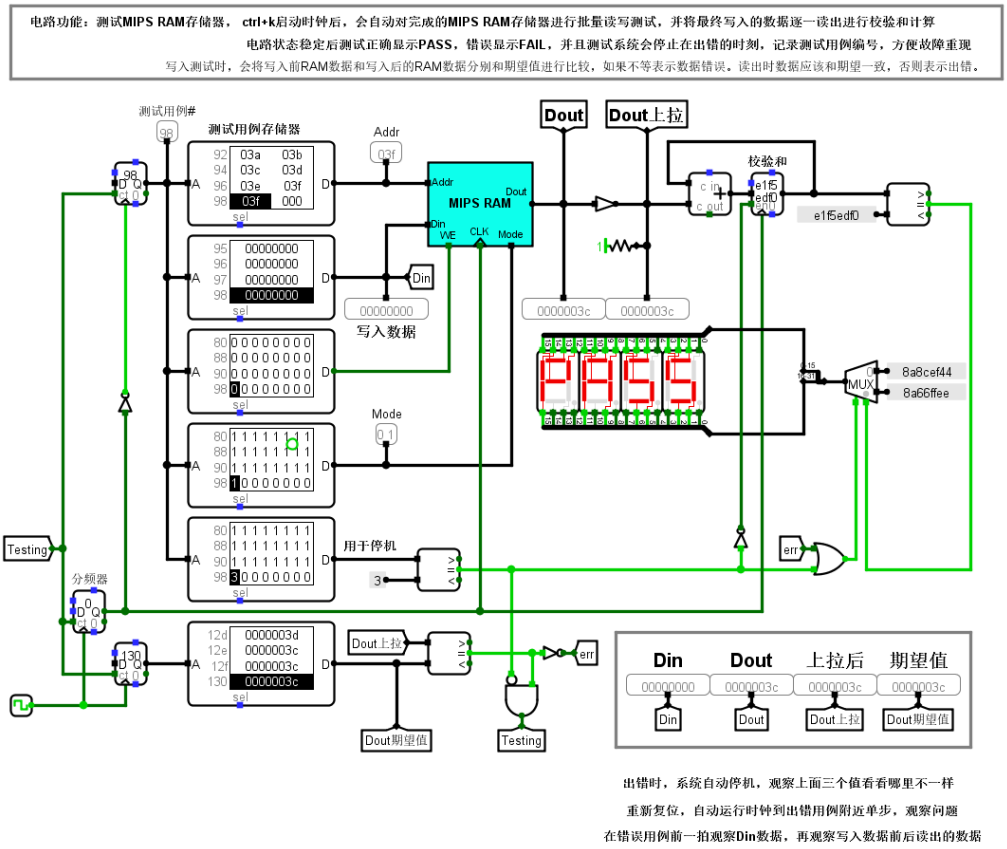
如图：



3.2.2 电路



3.2.3 测试

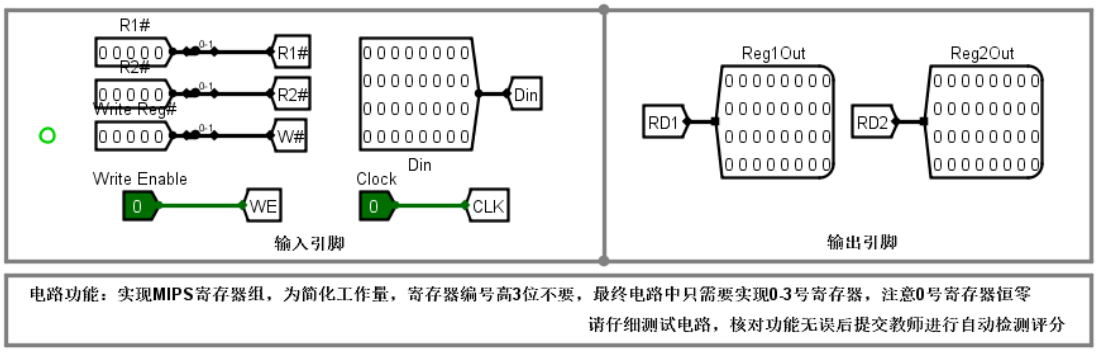


结果正确，测试成功

3.3 MIPS 寄存器文件设计

实验目的：为 MIPS CPU 构造核心功能部件，进一步熟悉多路选择器，译码器，解复 用器等 Logisim 部件的使用。

实验内容：设计完成满足如下规格要求的 MIPS 通用寄存器组。



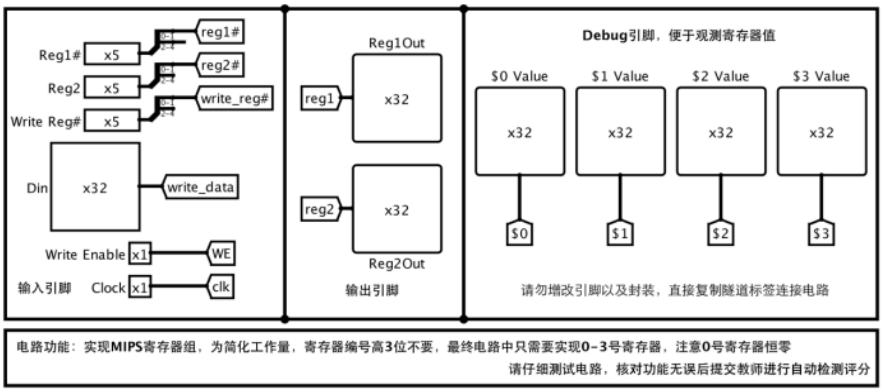
3.1.1 原理

1) 利用 logisim 平台构建一个 MIPS 寄存器组，内部包含 32 个 32 位寄存器，其具体 功能如下，具体封装文件为 regfile.circ.

表 1. 芯片引脚与功能描述

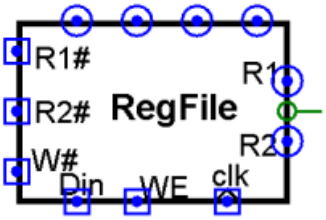
引脚	输入/输出	位宽	功能描述
R1#	输入	5	读寄存器 1 编号
R2#	输入	5	读寄存器 2 编号
W#	输入	5	写入寄存器编号
Din	输入	32	写入数据
WE	输入	1	写入使能信号，为 1 时，CLK 上跳沿将 Din 数据写入 W#寄存器
CLK	输入	1	时钟信号，上跳沿有效
R1	输出	32	R1#寄存器的值
R2	输出	32	R2#寄存器的值
$s0	输出	32	编号为 16 的寄存器的值
$s1	输出	32	编号为 17 的寄存器的值
$s2	输出	32	编号为 18 的寄存器的值
$ra	输出	32	编号为 31 的寄存器的值

实验电路输入输出引脚如下图：

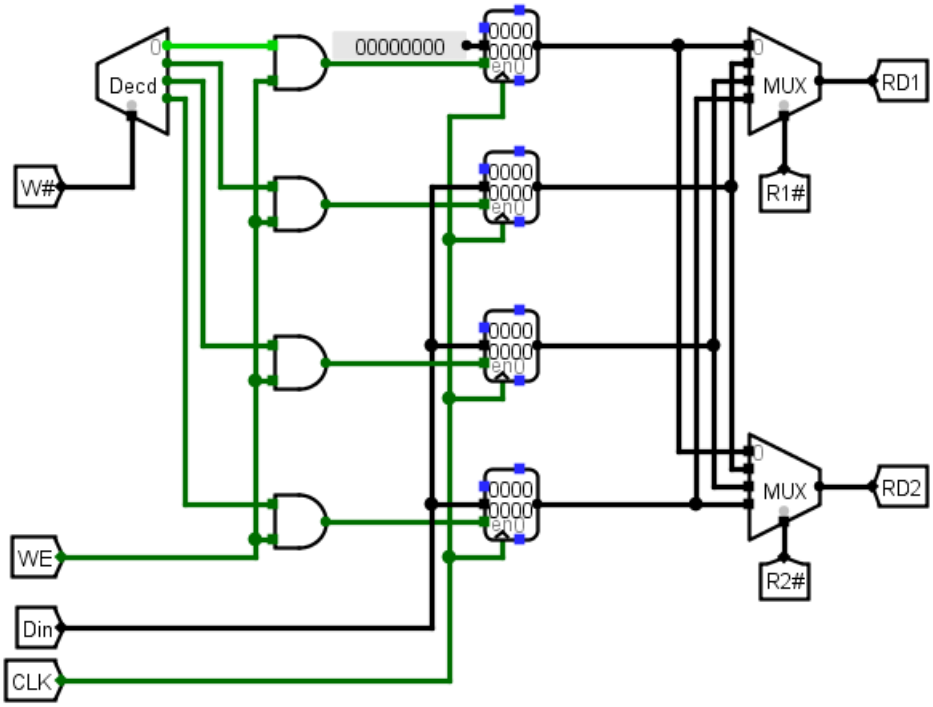


2) 为减少实验中画图工作量，实验工程文件中对 5 位寄存器地址进行了简化，具体 见引脚示意图，最终只需实现 4 个寄存器，0 号寄存器功能仍然是恒零。后续实验 中如需要使用 32 个寄存器的 MIPS 寄存器文件组，将提供标准组件。

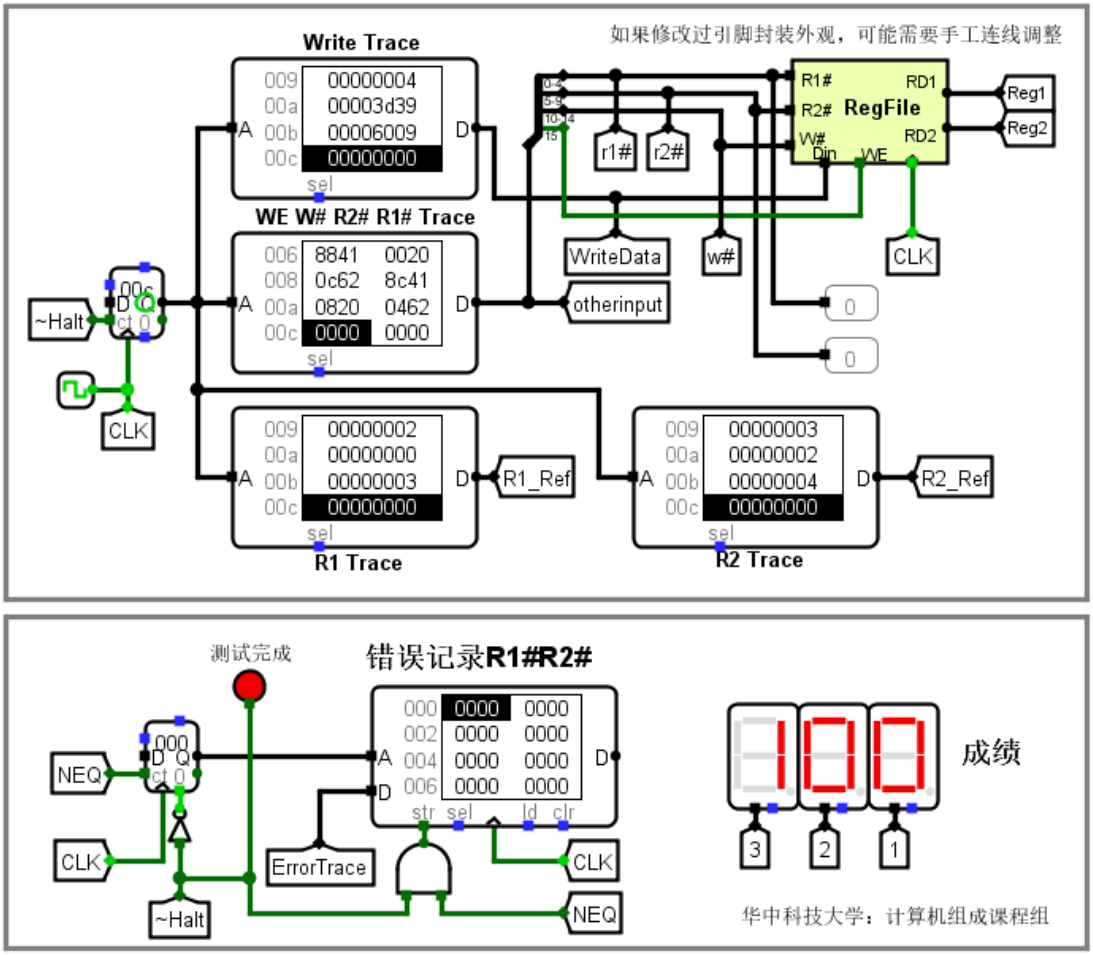
3) 注意时钟信号和电平信号不要混连，时钟仅仅触发状态改变。



3.3.2 电路

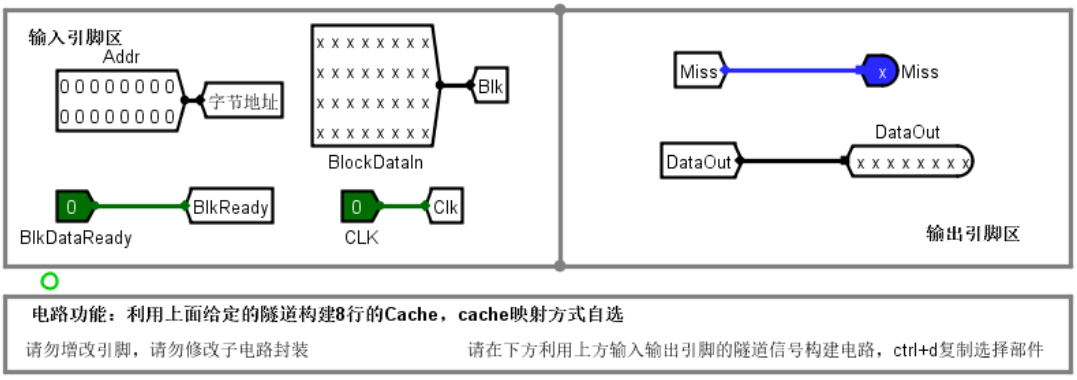


3.3.3 测试



结果正确，测试成功

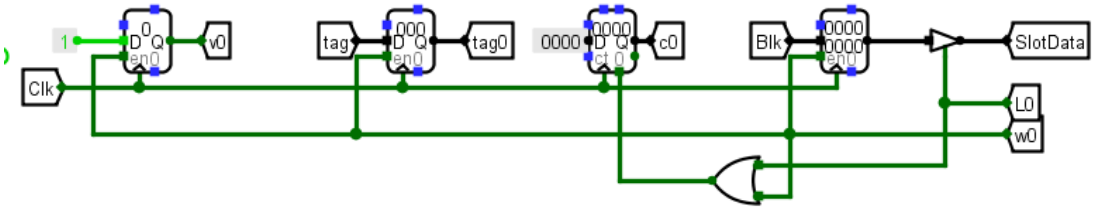
3.4 Cache 直接相连电路实现



3.4.1 原理

(1)

Cache 行封装如下：



（2）

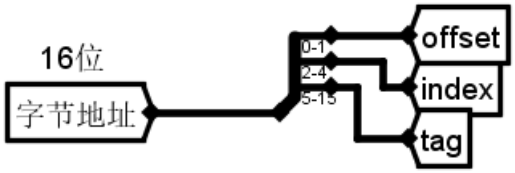
将字节地址分出偏移、行号、区地址：

①0-1 位为偏移

②2-4 位为行号

③5-15 位为区地址

如图 ：

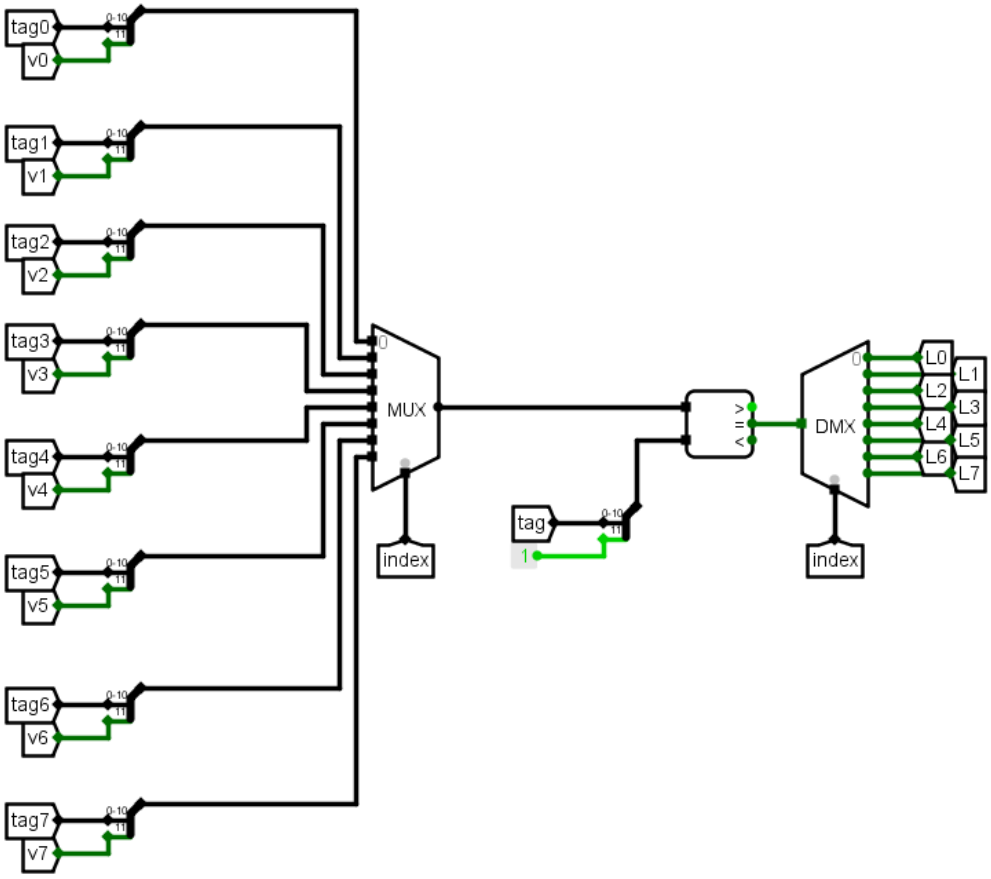


（3）

得到读信号：

当 Cache 中对应行 value 为 1 且其中存放数据的区地址与我们想要查找的数据块的区地址相同，则得到该行的读信号。否则所有行的读信号均为 0

如图：

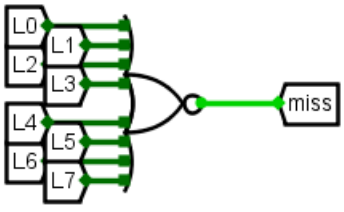


（4）

得到 miss 信号：

当没有选中信号为 1，即未命中时，miss 信号为 1，表明未命中

如图：

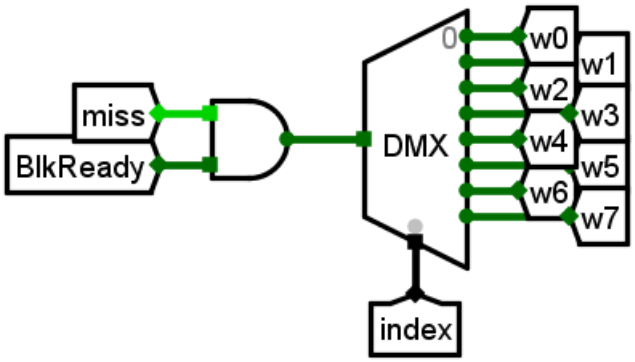


（5）

得到写入信号：

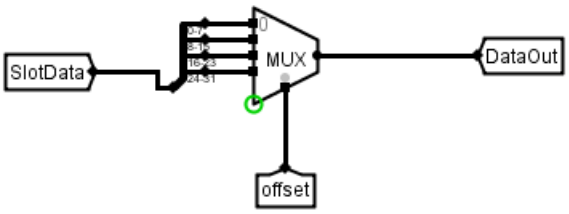
当 miss 信号为 1（Cache 中没有目标块），BlkReady 信号为 1（从显存中过来的数据块的数据已经准备好了）则根据行号输出该行写入信号。

如图：

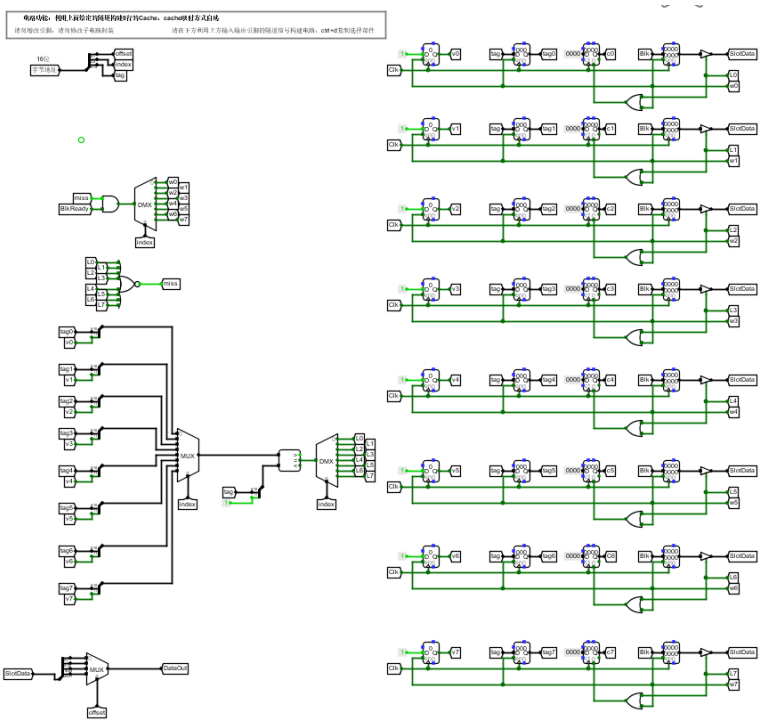


（6）
通过偏移量得到最终数据

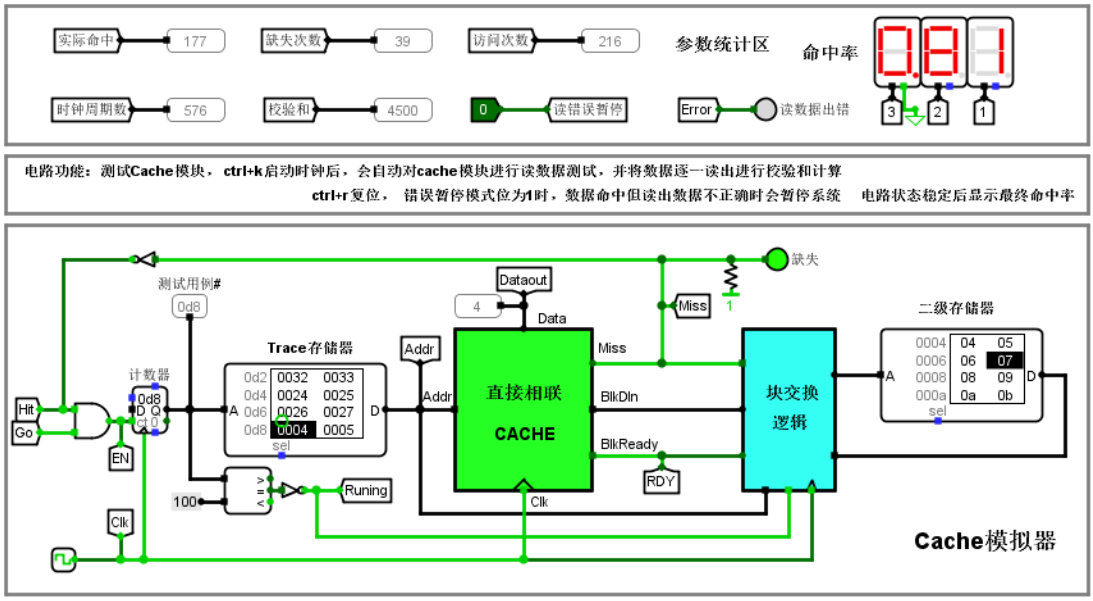
如图：



3.4.2 电路



3.4.3 测试



如果设计了不同的映射方式，可以将该电路中cache模块替换为对应Cache模块即可!

默认trace是顺序访问，所以测试系统中判断读错误的逻辑是地址和数据相等，如果更换了trace，这个逻辑不正确

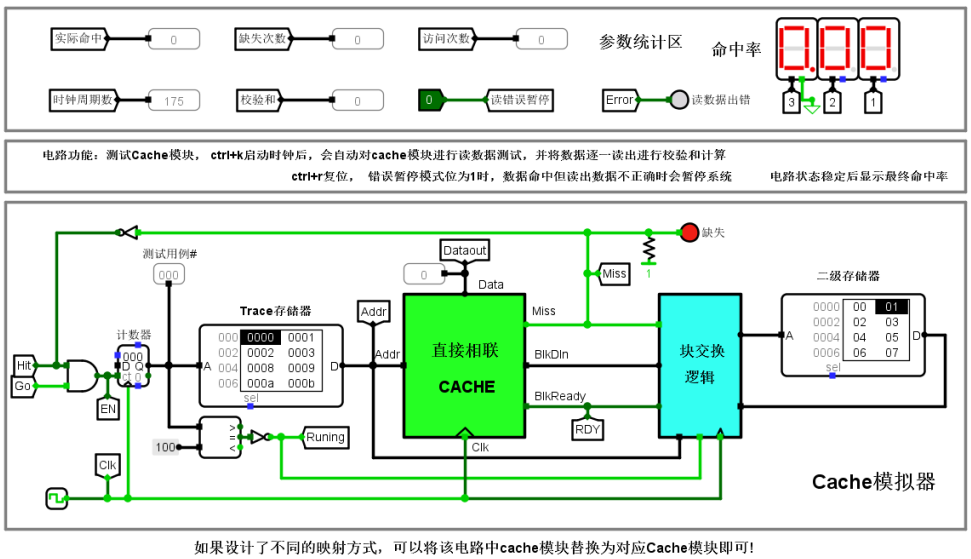
结构正确，测试成功

4 遇到的问题及解决办法

4.1 问题一

4.1.1 问题描述

①命中率一直为 0，说明测试错误



如果设计了不同的映射方式，可以将该电路中cache模块替换为对应Cache模块即可!

默认trace是顺序访问，所以测试系统中判断读错误的逻辑是地址和数据相等，如果更换了trace，这个逻辑不正确

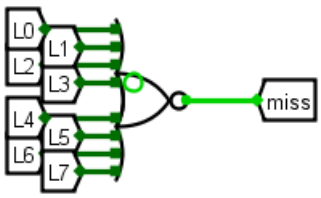
②二级存储器一直在读取前四个位置的数据

结合以上两个现象，我们可以大胆猜测：很有可能是 miss 出了问题

4.1.2 解决过程

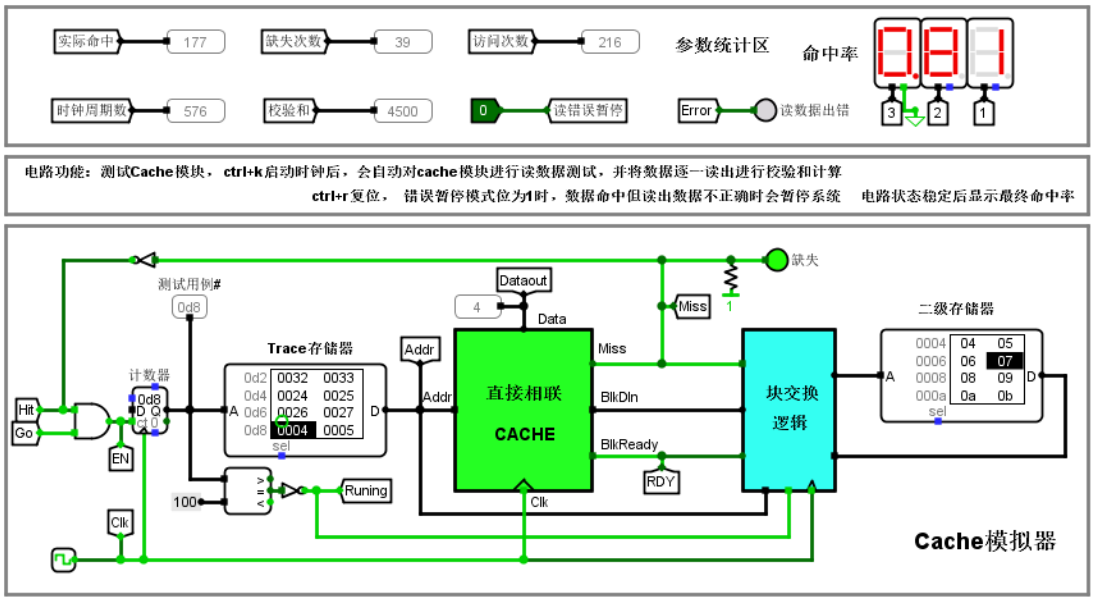
在回溯到电路中发现：Miss 没有输入值

于是去找 Miss 信号生成电路：



经过比对发现是大小写错了，将大小写修改一致之后再做测试

4.1.3 解决结果



修正之后结果正确，测试成功