桂林电子科技大学 2023-2024 学年 第 1 学期

# 计算机组成原理 B 实验报告

| | | | | | |
|---|---|---|---|---|---|
| 实验名称 | **存储系统设计** | | | | |
| 院　系 | **计算机与信息安全学院** | 专业 | **软件工程** | | |
| 学　号 | **2200350204** | 姓名 | **李禹佳** | | |
| 同 作 者 | | | | | |
| 实验日期 | **2023** 年 | **11** 月 | **25** 日 | | |

辅导教师意见：

成绩　　教师签名：

## 一、 实验目的和要求

1. 实验目的

（1） 掌握算术逻辑运算单元（ALU）的基本构成；

（2） 掌握 Logisim 中各种运算组件的使用方法，熟悉多路选择器的使用；

（3） 掌握 ALU 的设计和仿真方法。

2. 实验要求

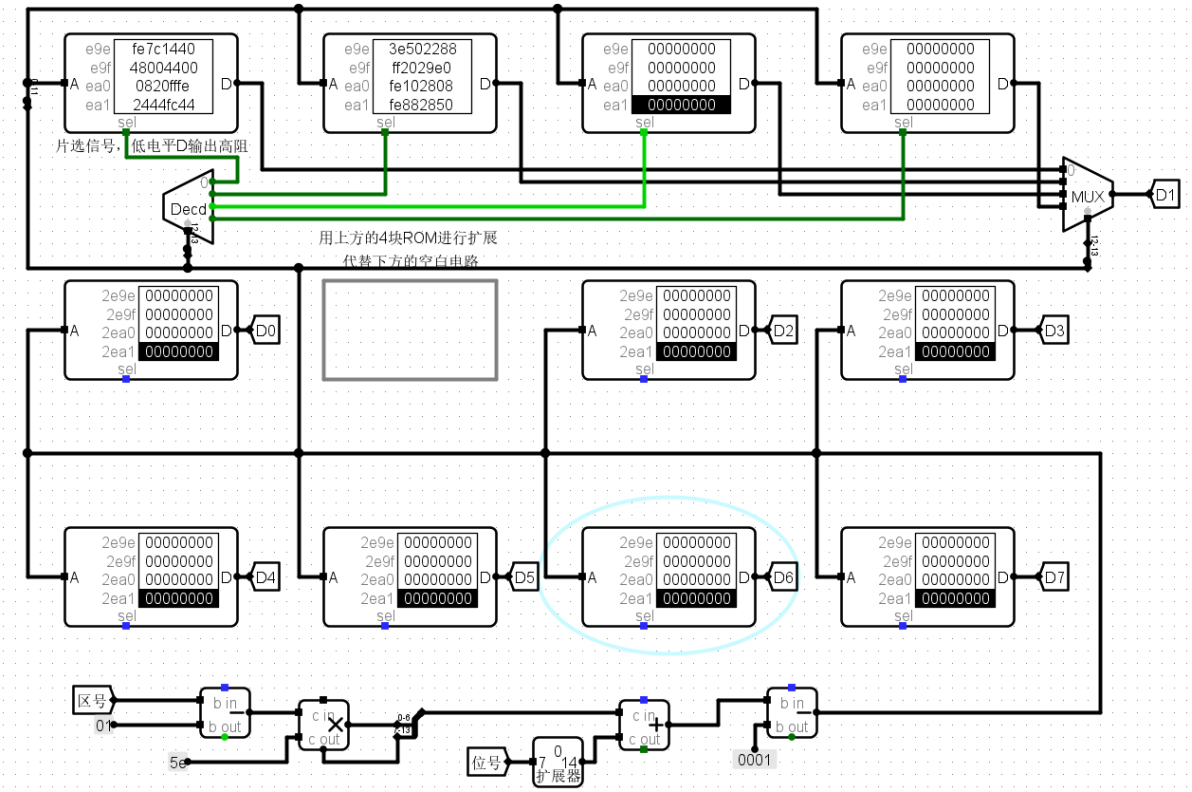
（1） 实验前，完成 Logisim 软件使用学习，并预习实验内容，准备好 ALU 设计方案；

（2） 独立完成 ALU 设计，在头歌平台完成指定闯关任务；

（3） 如实记录实验设计步骤，并对实验过程及结果进行分析总结，撰写实验报告。

## 二、 实验步骤

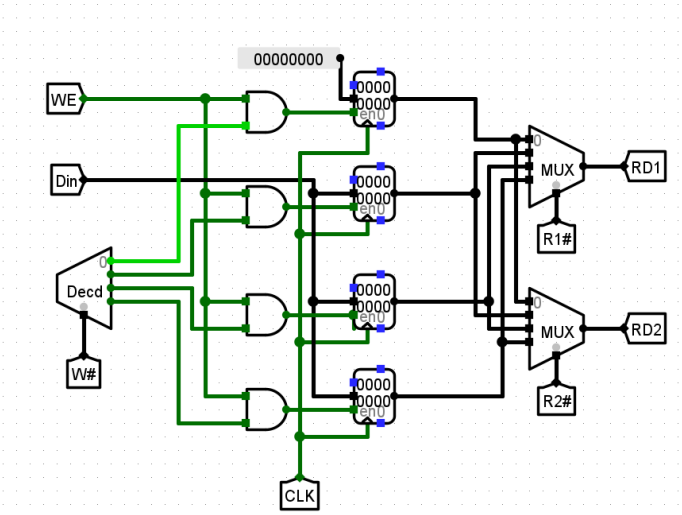
### 1. 第 1 关：汉字字库存储芯片扩展实验




根据题中描述，用四片小容量（4K*32 位）来代替原有（16K*32 位），将四片小容量 ROM 地址端并联，将并联后的地址段在分线器的 0~11 端，通过高两位链接译码器的输入端来分别与四个 ROM 的 sel 端连接，最后导入数据至所有 ROM，HZK16_1 .txt 中的 1～4096 个数据放到 0 号 4K 的 ROM 中，4097～ 8192 个数据放到 1 号 4K 的 ROM 中，8193～12288 个数据放到 2 号 4K 的 ROM 中，12289～16384 个数据放到 3 号 4K 的 ROM 中，地址线通过分线器来进行分离，其中还需要对其中的数据进行调整，分为高 2 位和低 14 位，高位为 1，低位为 0。对于片选信号进行连接，加入译码器，其中译码器位宽为 2，可选为 00，01，10，11 四个信号。检验
故障，位宽不匹配
解决。

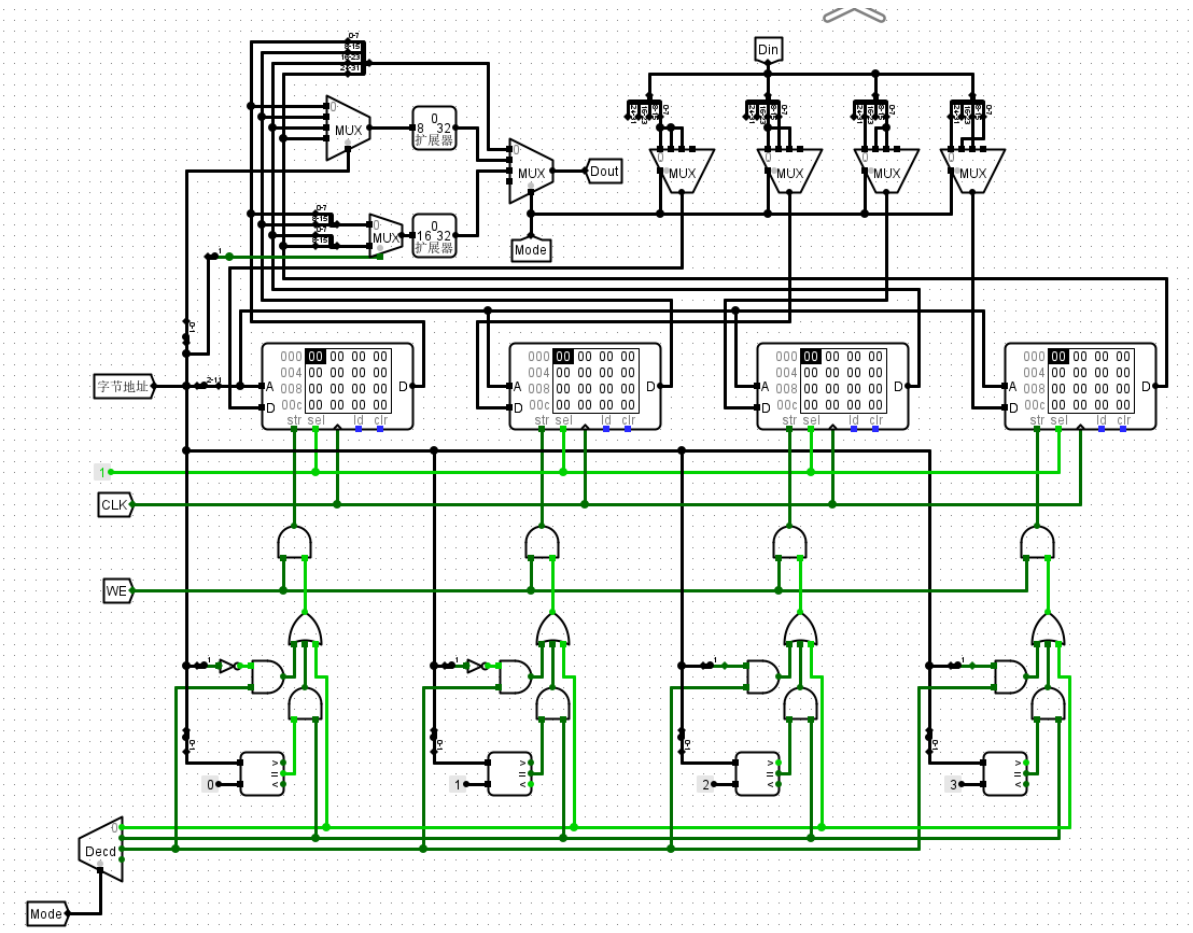
### 2. 第 2 关：MIPS 寄存器文件设计

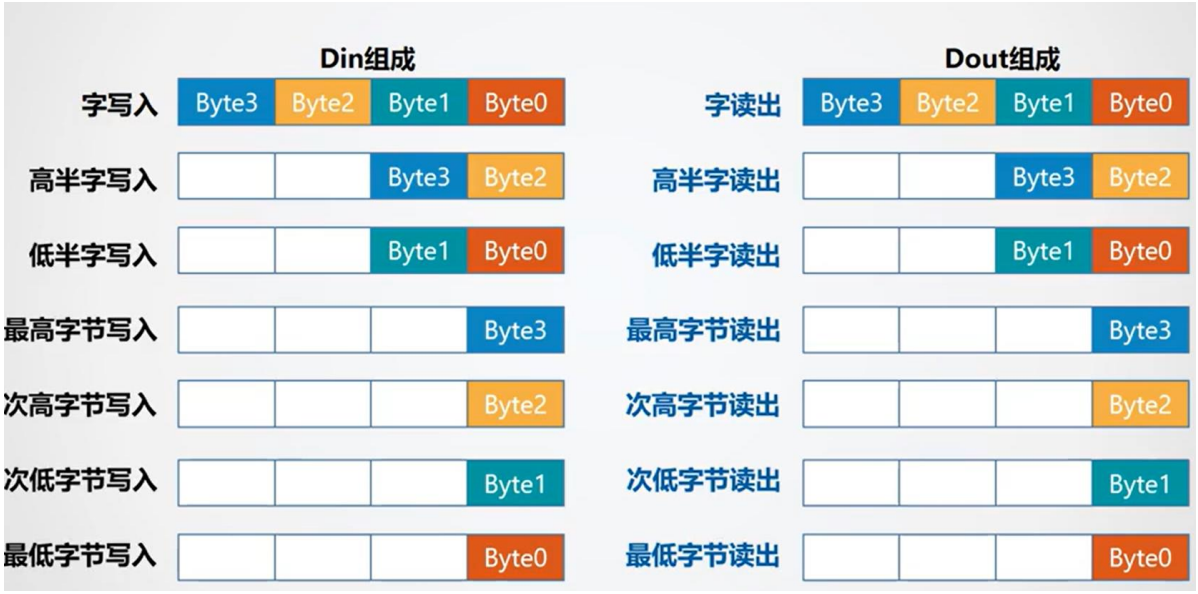
实现写入数据:



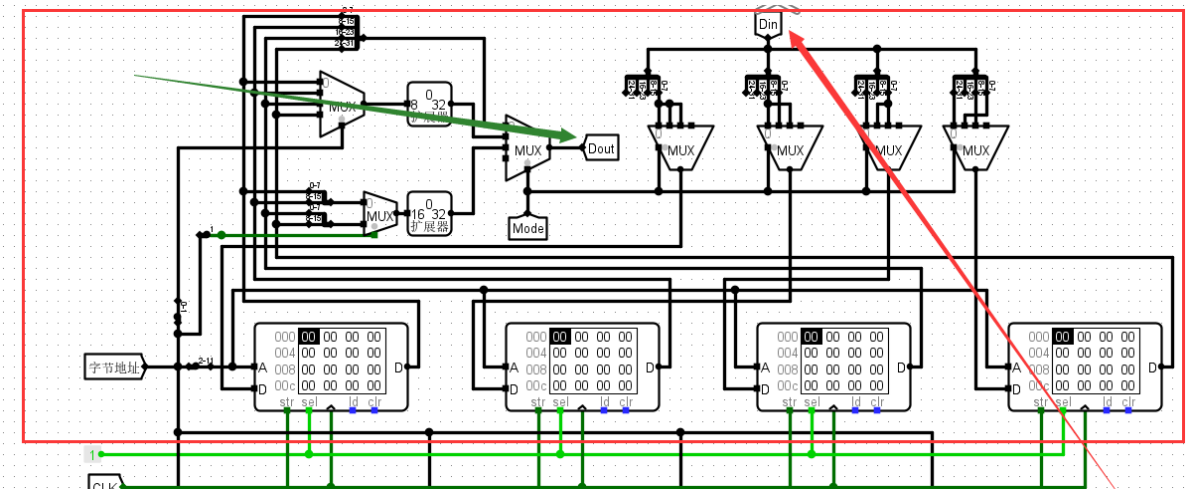

电路通过 R1#,R2#这两个信号来指定读取哪两个寄存器，值由 RD1,RD2 进行输出。在写寄存器时，要求 WE,W#信号有效，从 Din 输入引脚写入数据。同时 WE 有效，在时钟脉冲下，将 Din 端数据写到寄存器文件中。在指令执行时，通过分线器，将相应位提取出，在最终指令执行之后，数据被送到 Din 端，在 RegWrite 使能信号有效后再下一个时钟将指定寄存器的数据写入。写入时根据 W#收到的写入寄存器编号利用多路选择器选择寄存器

3. 第 3 关：MIPS RAM 设计





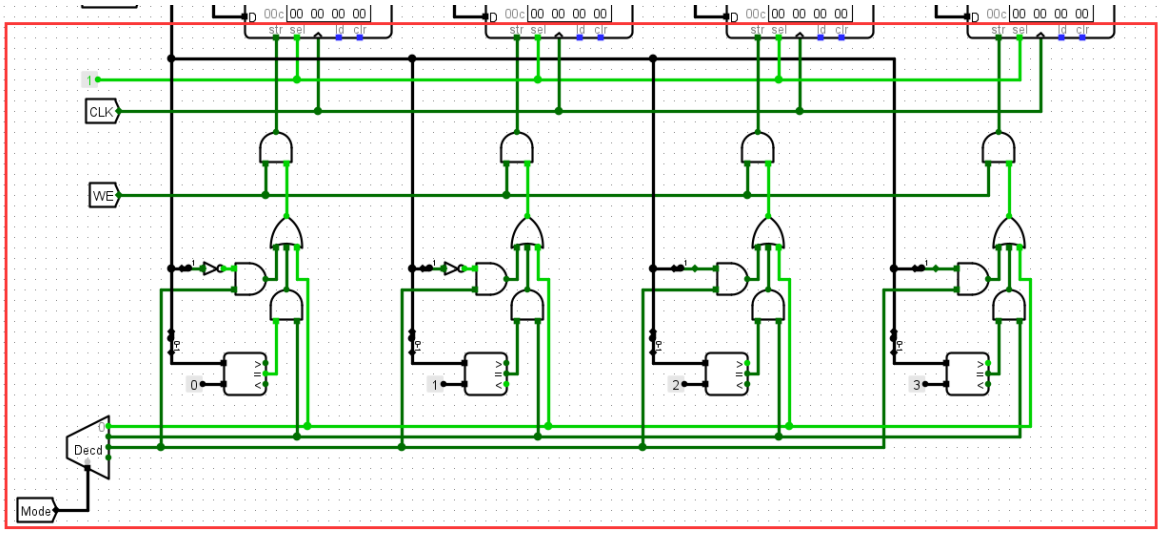

其中 Dout 的输出为：
字读出：4 个 ROM 分别给 Dout 提供一个字节数据
高半字读出：低十六位有效，3，2
底半字读出：低十六位有效，1，0
最高字节读出：最低八位
次高字节读出：最低八位
最低字节读出：最低八位
其中 Din 的输出为：最低八位
字写入：将 32 位子数据放在 32 位引脚，将四个字节写入四个 ROM 组件中
高半字写入：低十六位有效，3，2
底半字写入：低十六位有效，1，0
最高字节写入：最低八位
次高字节写入：最低八位
最低字节写入：最低八位




由数据选择器实现接线

| 引脚 | 输入/输出 | 位宽 | 功能描述 |
| --- | --- | --- | --- |
| Addr | 输入 | 12 | 字节地址 （字访问半字访问时应硬件强制对齐） |
| 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 |
| WE | 输入 | 1 | 写使能，高电平有效 |
| CLK | 输入 | 1 | 时钟信号，上跳沿有效 |
| Mode | 输入 | 2 | 访问模式 00：字访问，01：字节访问，10：2字节访问 |
| Dout | 输出 | 32 | 输出数据，不同访问模式有效数据均存放在最低位，高位忽略 |

由上可知，当为字输入输出时，即 Mode=00，此时直接输入输出 32 位。当为字节输入输出时，即 Mode=01，此时只需输入和输出 1Byte 的数据，并且该 Byte i 一定放置在 4 个字节的最低位在 Din 中 Byte i 由片选信号 wi 决定；在 Dout 中，由字节地址的最低两位来确定。当为半字节输入输出时，即 Mode=10，此时需要选择两个 Byte 的数据，分别是 01 或者 23，而他们的相对位置是确定的。在 Din 中，0 和 2 连接低 8 位，1 和 3 连接高 8 位，具体片选由 wi 决定；在 Dout 中，由第二低位来确定。*Mode* 和 *Wi* 引脚设置，先用解码器，将 *Mode* 值转化为对应的信号，再根据三种不同的输入输出方式，通过信号的改变，实现不同的片选型



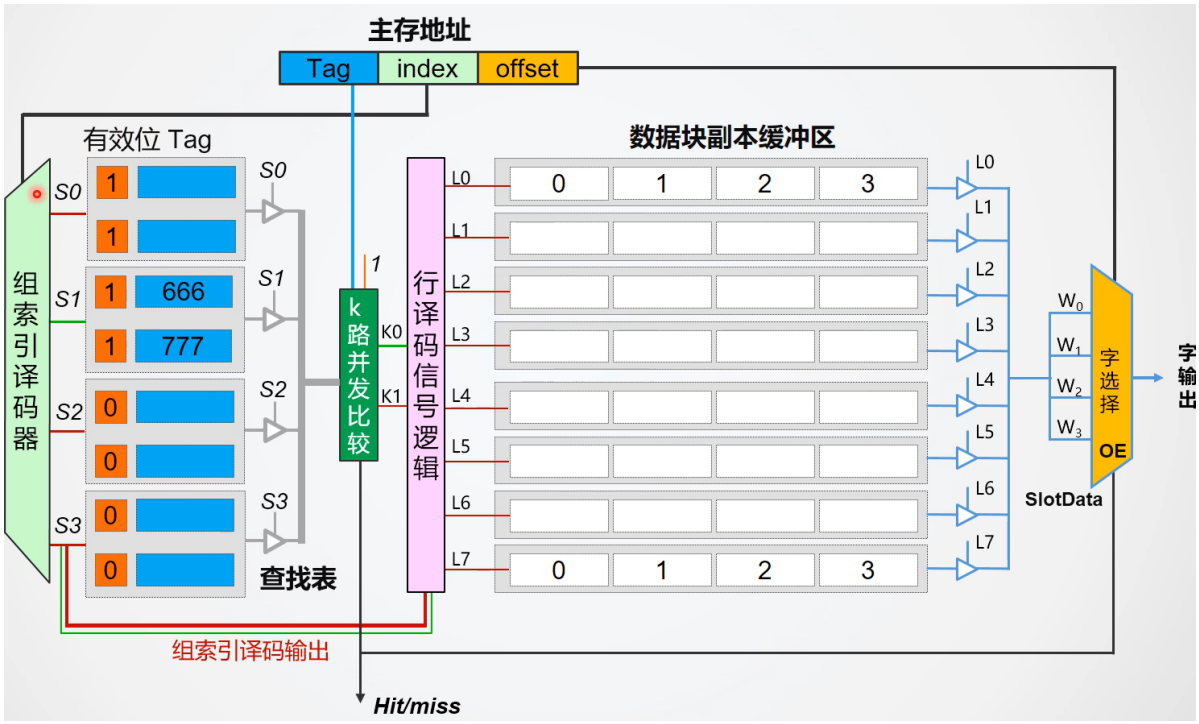

由于取字节地址，取 2~11 位，分别接入 4 片 ROM 中使能端由片选型号 *wi* 来决定是否有效，输入端位由 *di* 来输入数据

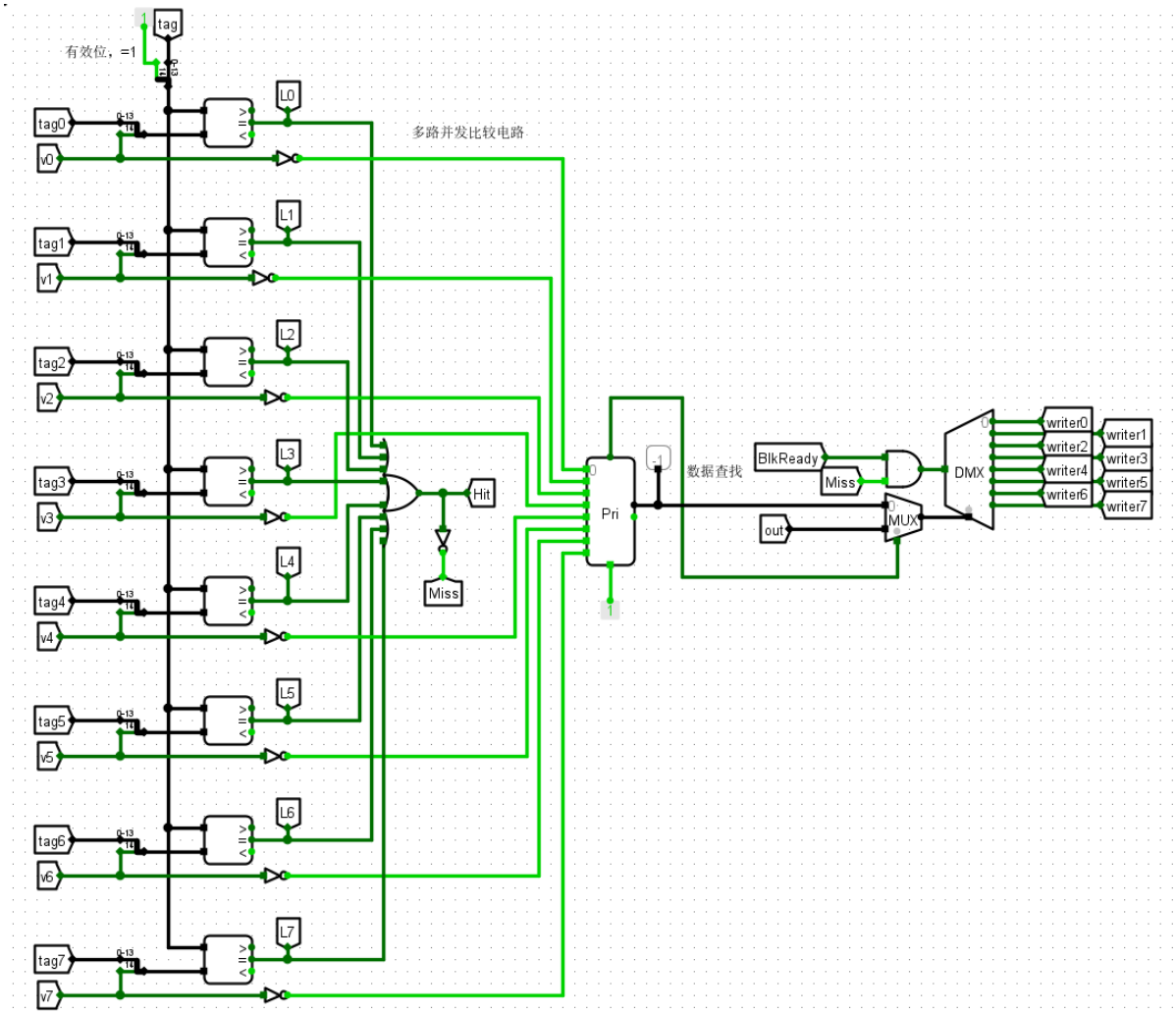
## 4. 第 4 关：全相联 cache 设计

实现全相联映射，可以直接存储而不用固定存储哪一行，命中率和存储率较高。主存中的任何一个数据块都可以放置在 cache 的任意一个数据块中。为了方便查找，主存数据块载入时，还需要记录若干的标记标志信息，主要包括有效位、主存块地址标记、脏数据标记位、淘汰计数等信息。

其中 cache 8 行，每个 cache 4 字节，offset 2 位，Tag 为 14 位。

数据查找和访问逻辑：

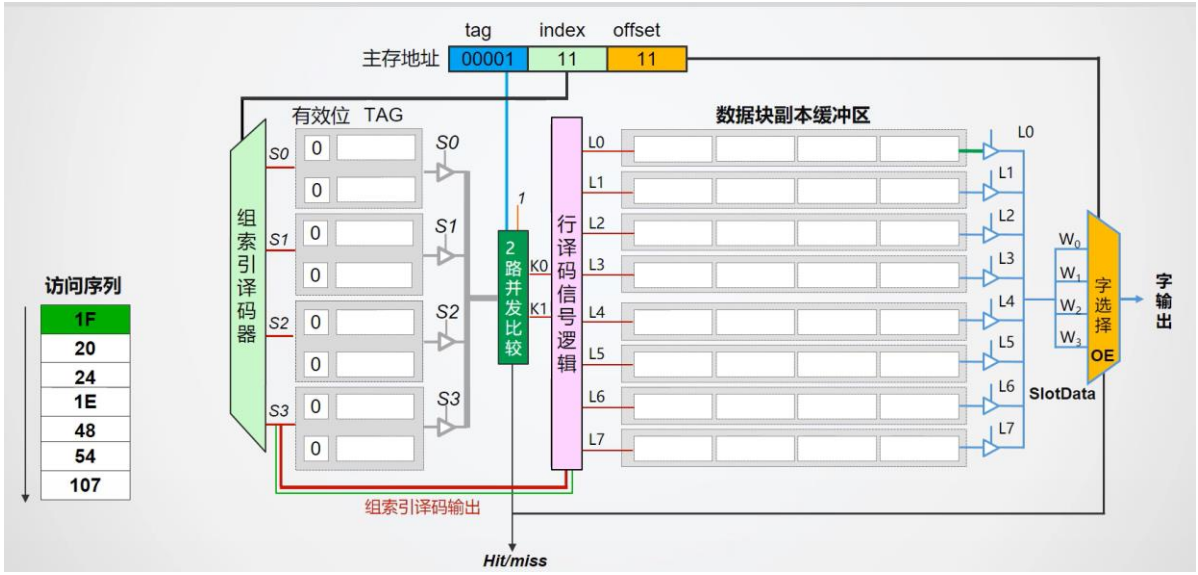


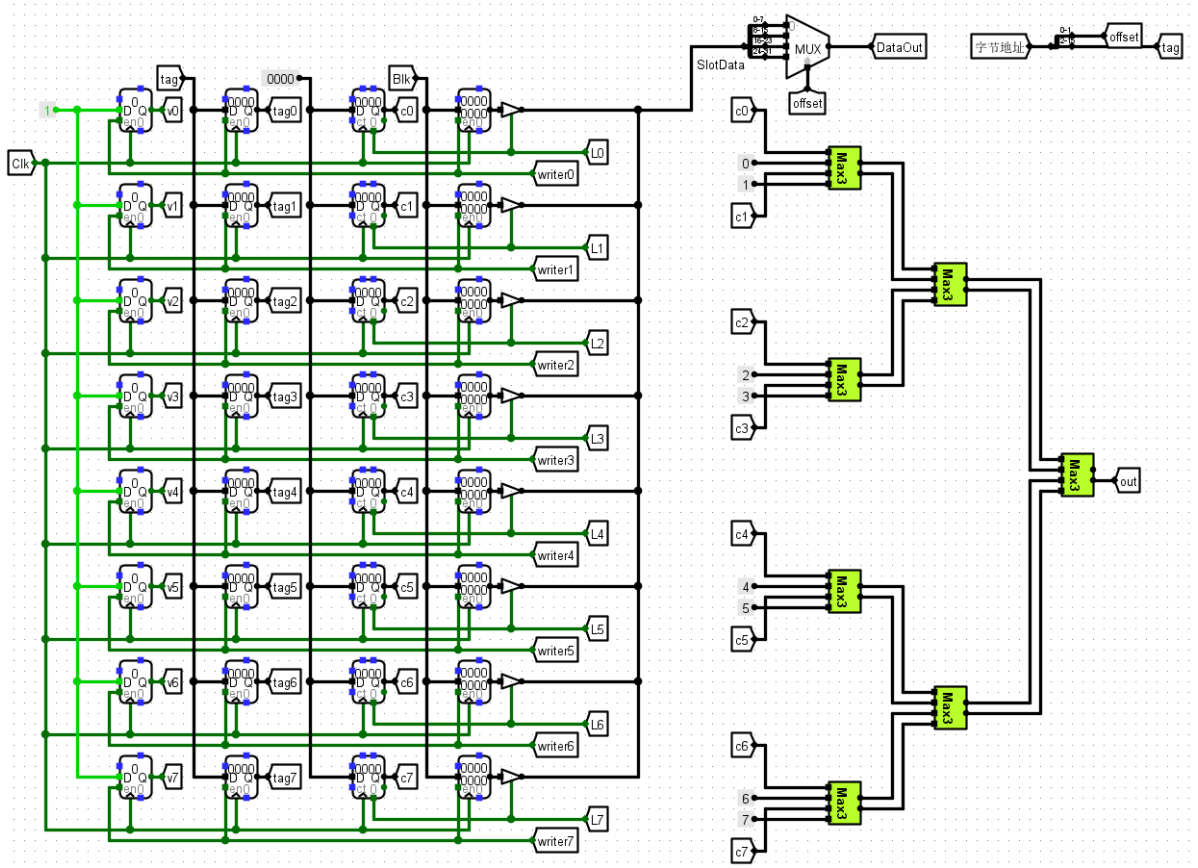




按图接入电路，使用或门将比较信号转换为打入信号（将 tag 和所有的 tag 寄存器里的值 x

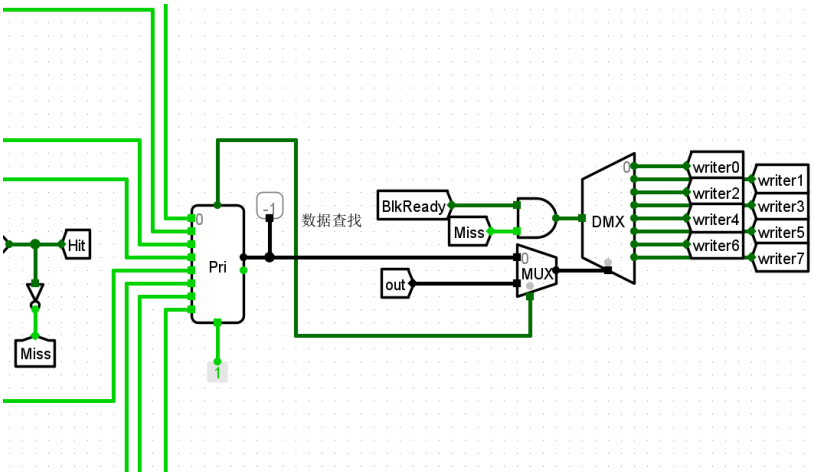
进行比较。某一行比较结果相同时，且有效位为 1 时，对应行的比较结果输出为 1），使用多路并发比较电路，输入行移码信号逻辑器写入该行。

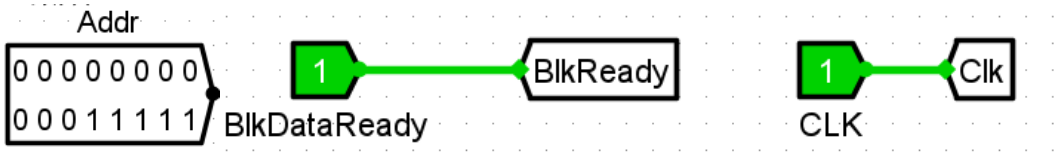
数据载入逻辑：



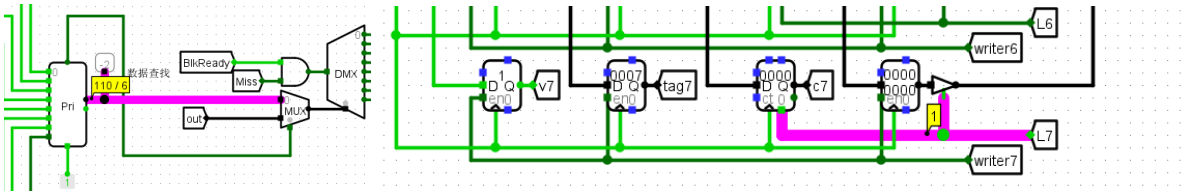




与直接相连一致，将所有的行淘汰数值归并得到最大值编号，如果需替换计数器最大行（载入 8 行数据）应改变行数。使用优先编码器，如果 8 行都载入数据了（也就是优先编码器输入为 0000 0000），那么就会使得 full 置为 1，否则就输出位数最小的空闲行。当 blk ready 并且 miss 的时候，向空闲行或者要淘汰的行写数据。



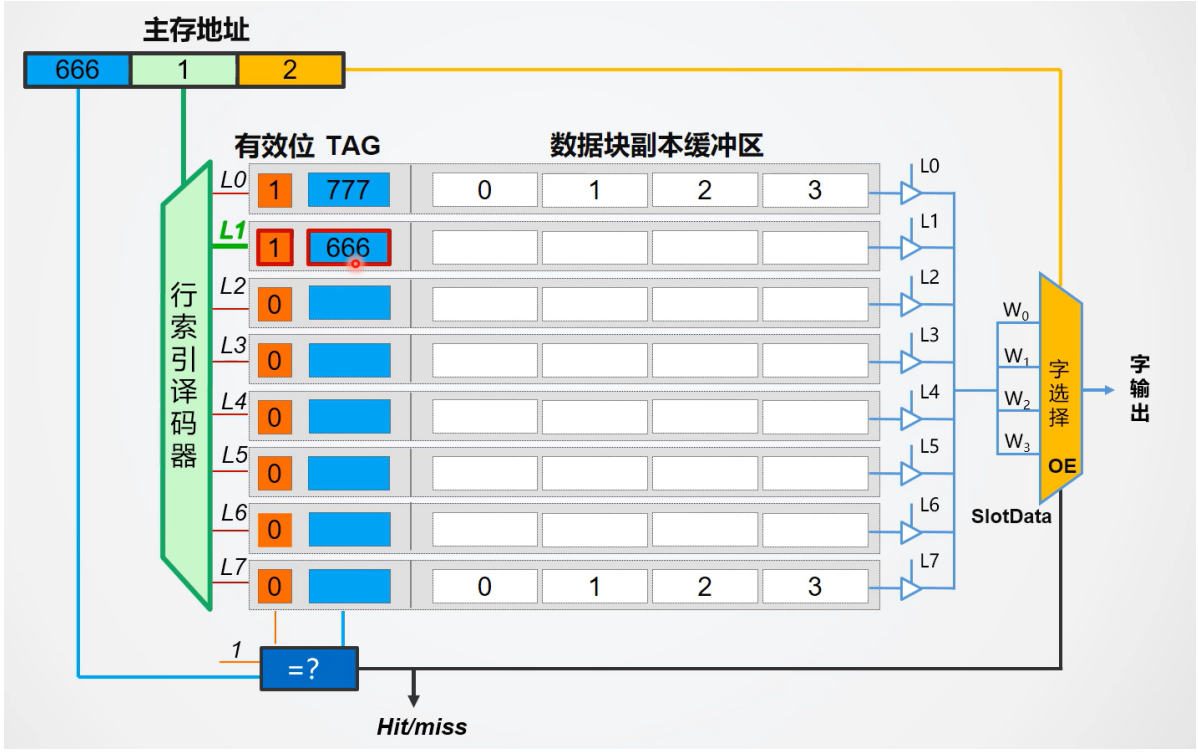



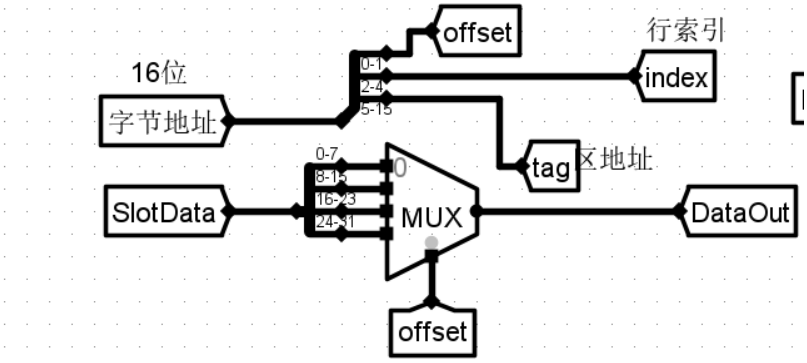

输入 1F，打入第七行，显示打入信息




5. 第 5 关：直接相联 cache 设计

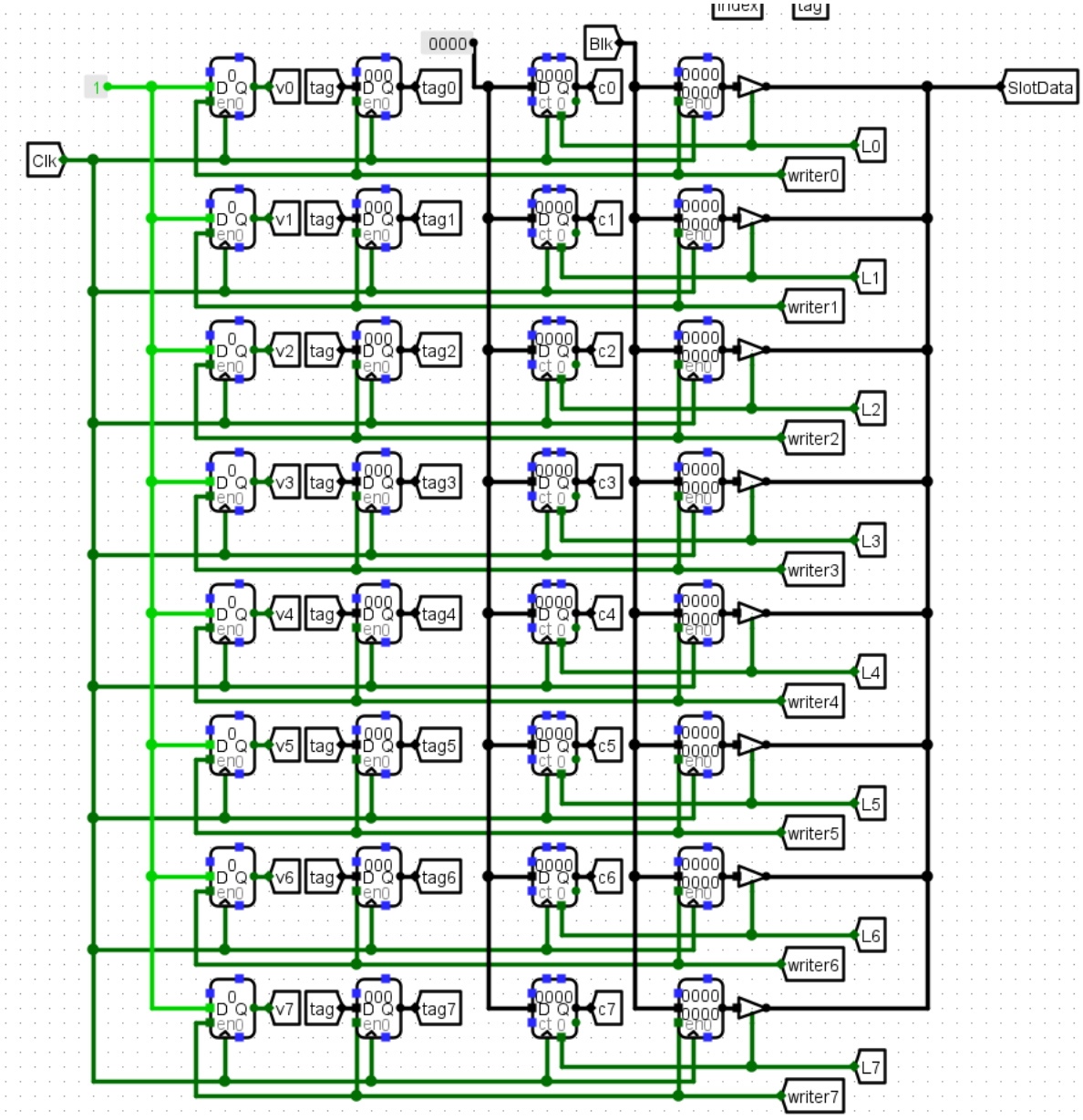
cache 有 8 行，4 字节，offset = 2，index = 3，tag = 11






Cache 行主要包括三种寄存器：valid，tag，数据块副本。当该行被选中时，将输出的值存到与之对应的标签。当 blkready=1 时，且该行被选中时，有效位置 1。

框部分为主存地址分配与 slotData 输出，旁边为字选择部分，下方为行地址译码器和比较器

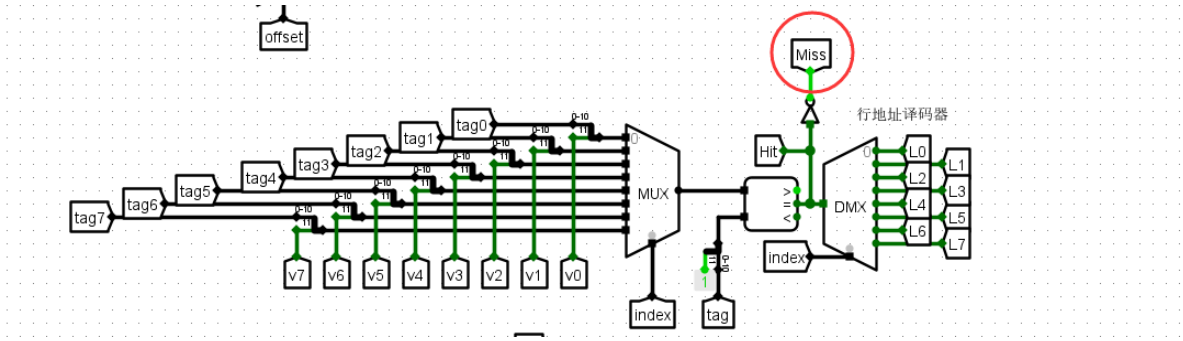



为缓冲区。行选中信号只用来控制数据块副本的输出。由于采用的是 LRU 算法，即近期最少使用

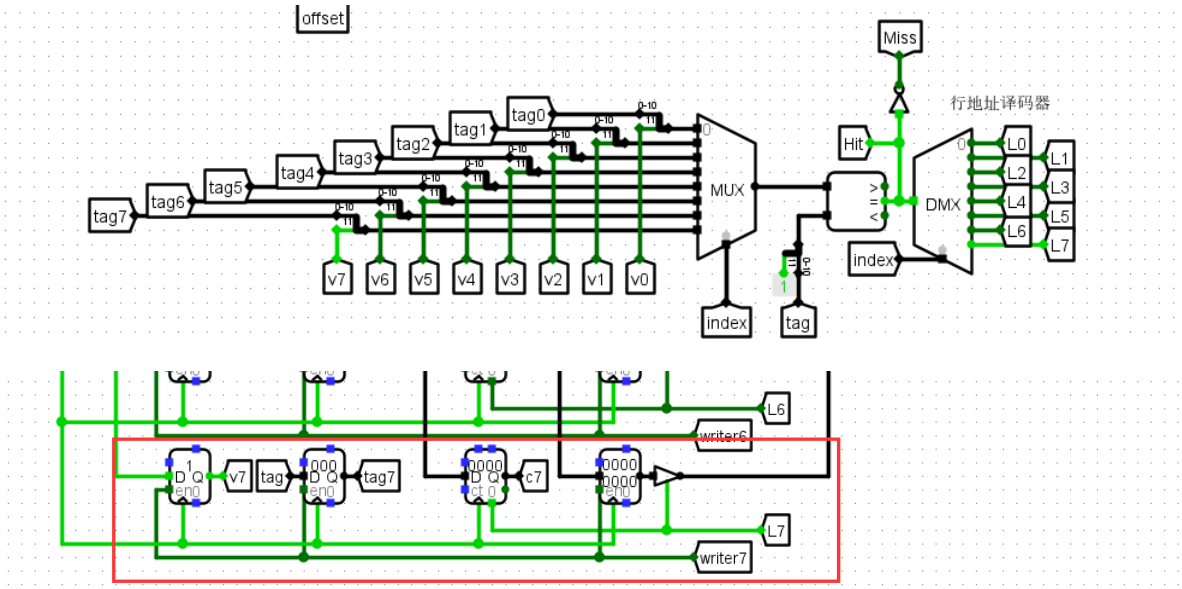
算法。当行被选中时，我们就将计数器清 0.随着时间的增大，如果某行 cache 块用的比较少，那么淘汰计数器的值就会比较大。

根据接线图可画出接线电路，tag 为区地址，为前 11 位，index 为行索引，在行地址译码器中找到对应行为区地址后面三位，offset 为字偏移地址，在后判断输出，为最后两位。

接好线后使用用例 1F 检测：

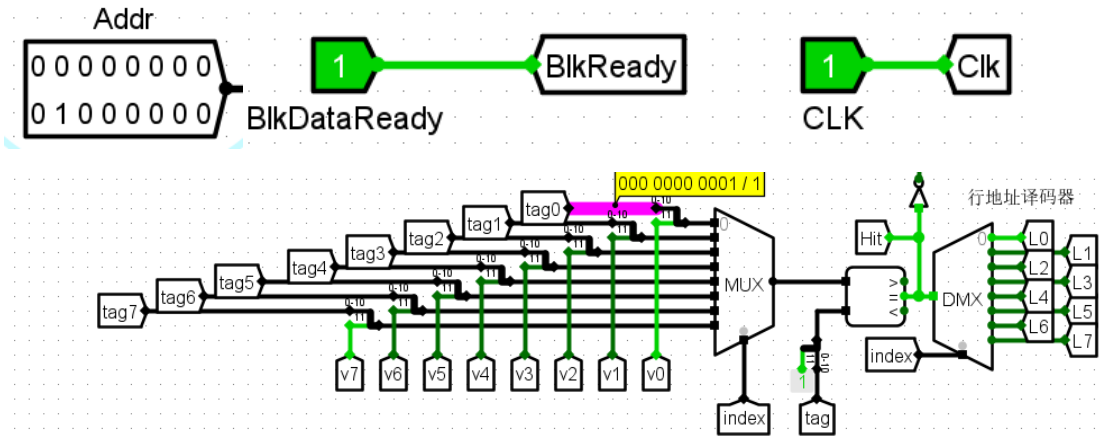



因为 cache 中为空，miss，在 BlkDataReady 为 1 的情况下 clk 置 1，后打入 hit 为 1，找到 L7

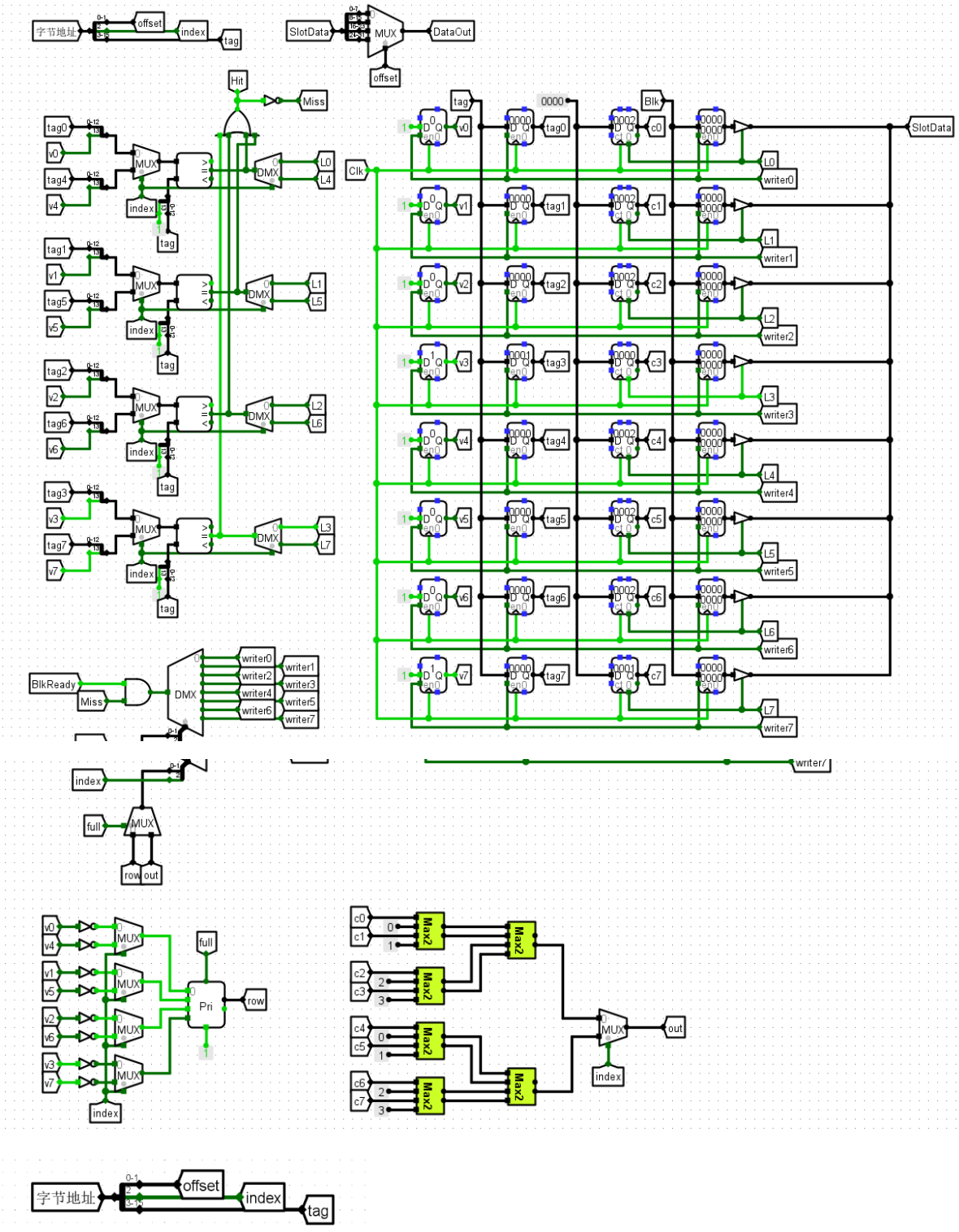



改变有效位为 1，tag 为 0000

下面是 20 举例

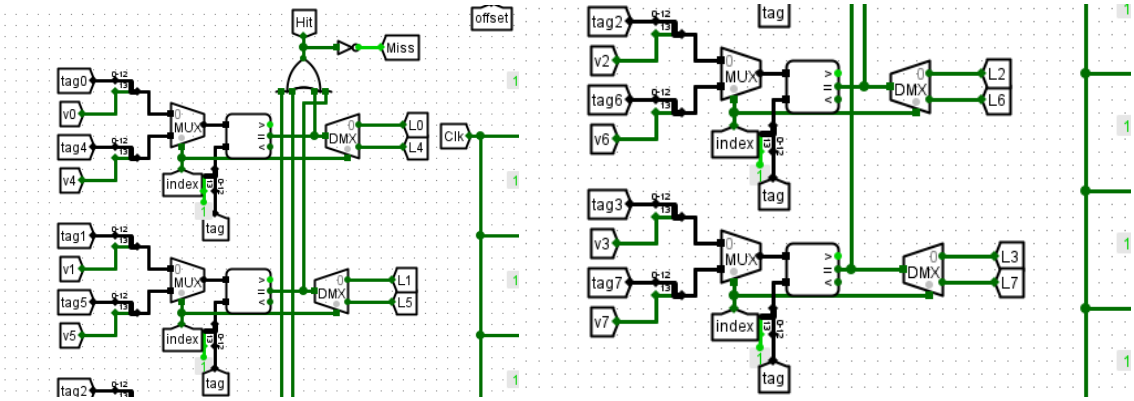


## 6. 第 6 关：4 路组相连 cache 设计




将主存地址按照一定的规则分成标记(tag)、组索引(index) 和块偏移(offset) 三个部分，将标记和组索弓|分别与缓存中的每个组的标记和索引进行比较，确定需要访问的组，4 路组相连缓存是将缓存地址空分成多个组，每个组包含 4 个缓存行每个组有个标记(tag)和一些数据(data) ，标记用于标识该组中的缓存行是否包含所需的数据

当 CPU 需要访某个地址时，缓存控制器会根据地址的标记来确定该地址是否在缓存中




找到访问的组，在该组中查找数据，如果命中 hit 则直接访问，不命中 miss 则缓存中为空，没有该信息，这时将这个数据的以及这个数据的下三位存入一个组中，组数随机

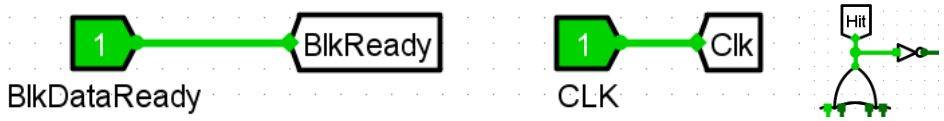
检验：

将 0100 打入并测试 0110，1000 是否在其中

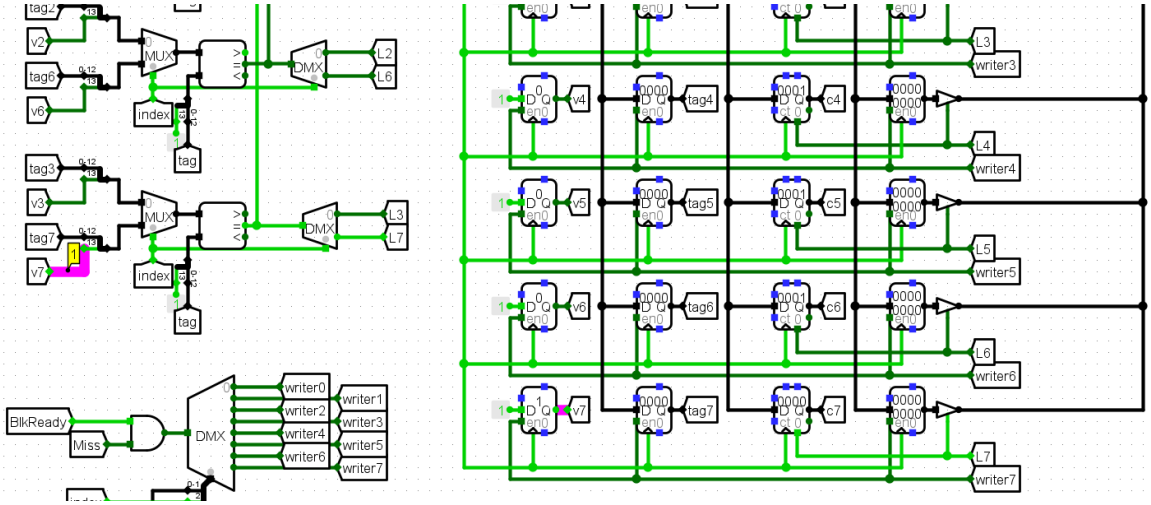
将 Addr 处置为 0100


显示 miss，所有组中没有 0100

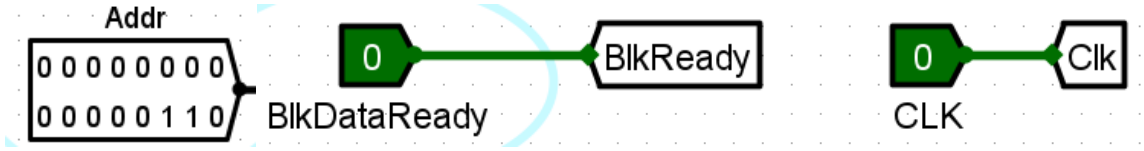
点击 BlkDataReady 开始运行后，将时钟置为 1



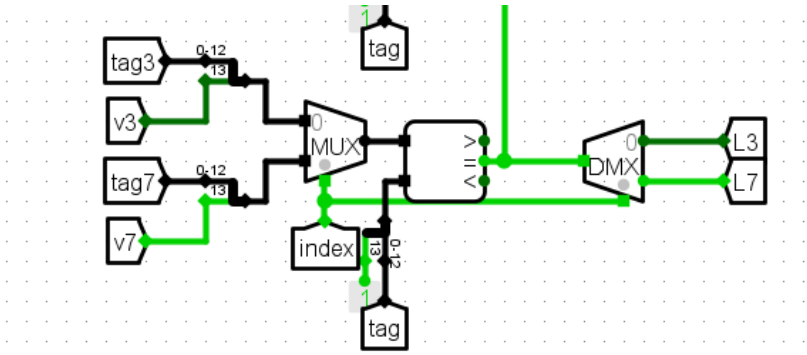

显示 hit 已打入，打入的组为 v7



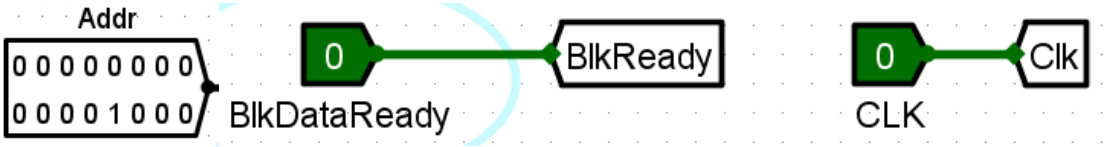

关闭 BlkDataReady，时钟，将 Addr 调至 0110

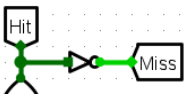



显示 hit，表明组中有这个数据，且这个数据在 v7 中

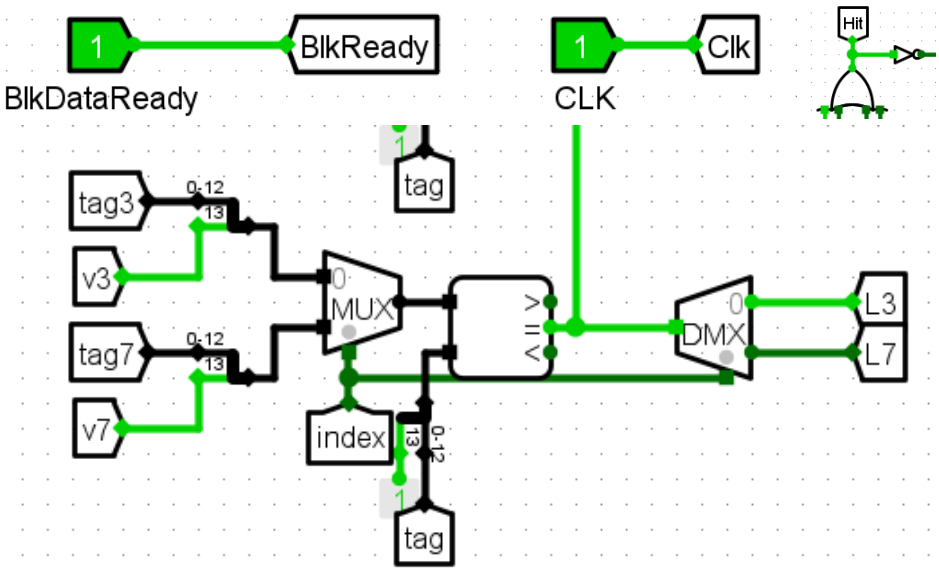



将 Addr 调至 1000



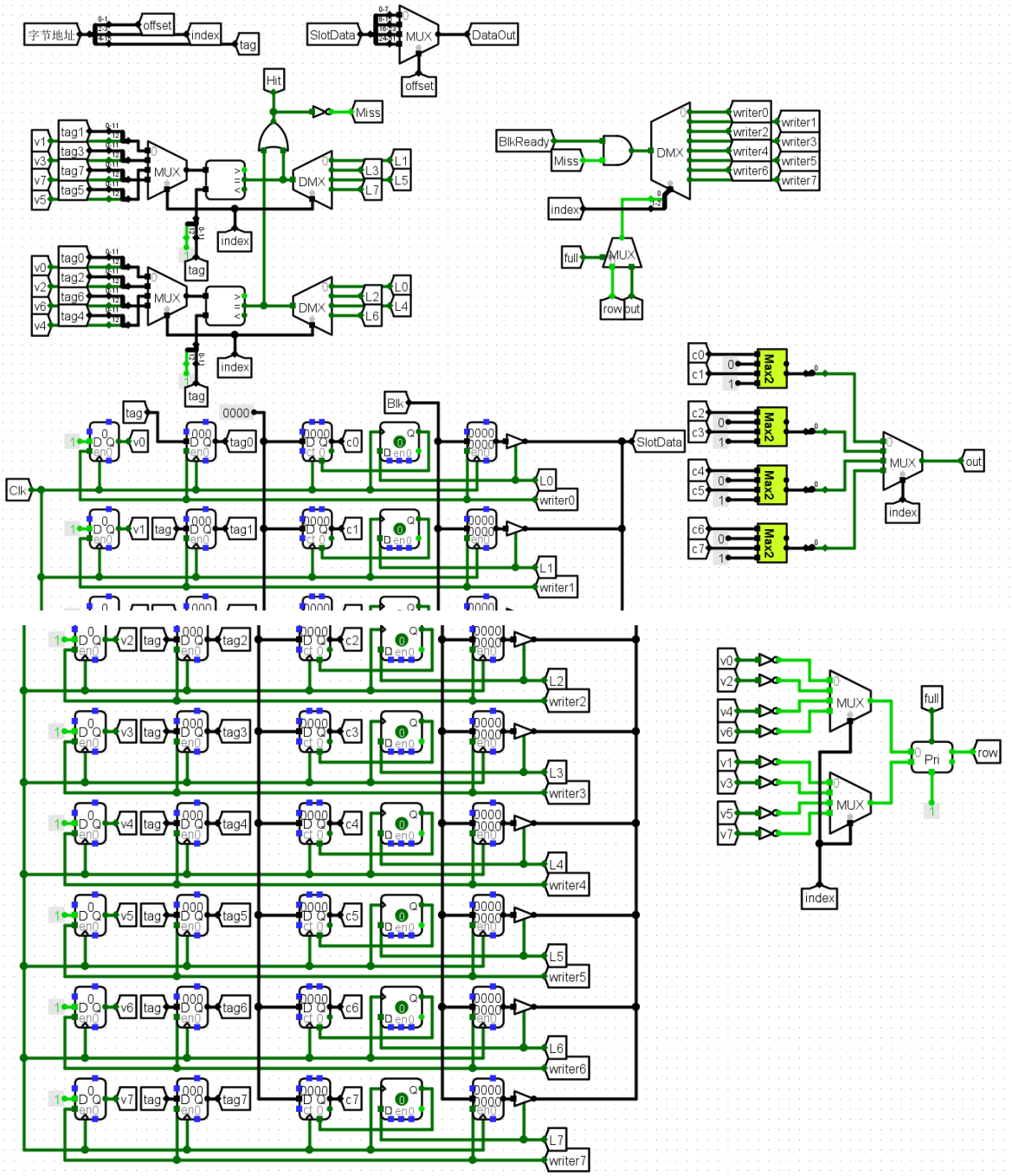

显示 miss，此时已超出存储的四个位置，故想要存储只能再次打入




打入后分配至 v3 组中

测试成功

7. 第 7 关：2 路组相联 cache 设计




将四路组相联改动组数为 2 组，可得到二路组相联

## 三、 实验小结

1. 在 Cache 全相联中未考虑满 Cache 时的替换，后加入 out 判断
2. 未考虑到有效位导致结果错误
3. 使用器件不习惯，某些器件，引脚需要搜索才能使用
4. 数据位宽不匹配现象，通过调试改变得出结果