

计算机组成原理

课程设计报告

专业班级：计算机科学与技术（大数据方向）

19G231

学生姓名：张淋迩

指导老师：樊媛媛

**中国地质大学计算机学院**

**2023年 6 月**

目录

[实验一、全相联cache设计 1](#_Toc12443)

[1、 实验目的 1](#_Toc31731)

[2、 主要任务 1](#_Toc6482)

[3、 实验方案的总体设计 1](#_Toc3950)

[4、 实验的实现过程 1](#_Toc7462)

[5、 电路图及相关说明 2](#_Toc21452)

[6、 实验数据测试 5](#_Toc16543)

[实验二、4路组相连cache设计 7](#_Toc30533)

[1、实验目的 7](#_Toc3489)

[2、主要任务 7](#_Toc18066)

[3、实验方案的总体设计 7](#_Toc26333)

[4、实验的实现过程 7](#_Toc10004)

[5、电路图及相关说明 8](#_Toc15325)

[6、实验数据测试 12](#_Toc4908)

[实验三、2路组相联cache设计 14](#_Toc19057)

[1、实验目的 14](#_Toc12921)

[2、主要任务 14](#_Toc16520)

[3、实验方案的总体设计 14](#_Toc13524)

[4、实验的实现过程 14](#_Toc27061)

[5、电路图及相关说明 15](#_Toc3640)

[6、实验数据测试 19](#_Toc11756)

[实验四、单周期MIPS CPU设计 21](#_Toc25349)

[1、实验目的 21](#_Toc4938)

[2、主要任务 21](#_Toc12091)

[3、 实验方案的总体设计 21](#_Toc17114)

[4、 实验的实现过程 21](#_Toc18587)

[5、 电路图及相关说明 22](#_Toc20656)

[6、 实验数据测试 27](#_Toc6871)

[实验五、微程序地址转移逻辑设计 34](#_Toc18912)

[1、实验目的 34](#_Toc13878)

[2、主要任务 34](#_Toc24339)

[3、实验方案的总体设计 34](#_Toc19537)

[4、实验的实现过程 34](#_Toc30914)

[5、电路图及相关说明 35](#_Toc21471)

[6、实验数据测试 37](#_Toc10856)

[实验六、MIPS微程序CPU设计 38](#_Toc5025)

[1、实验目的 38](#_Toc25640)

[2、主要任务 38](#_Toc5614)

[3、实验方案的总体设计 38](#_Toc32441)

[4、实验的实现过程 38](#_Toc20853)

[5、电路图及相关说明 38](#_Toc23445)

[6、实验数据测试 39](#_Toc31018)

[实验心得 40](#_Toc25533)

# 实验一、全相联cache设计

## 实验目的

（1）理解全相联缓存的工作原理：通过实验，加深对全相联缓存如何根据地址信息直接定位到缓存行的理解。

（2）掌握缓存映射算法：通过实践，学会如何实现全相联映射算法，并理解其与直接映射和组相联映射的区别。

（3）分析缓存性能：通过实验，学会如何评估和比较不同缓存策略对系统性能的影响，包括命中率和访问时间。

（4）培养问题解决能力：通过解决实验过程中遇到的问题，提高分析问题和解决问题的能力。

（5）增强理论联系实际的能力：将课堂上学到的理论知识应用到logisim的实际电路设计中，加深对计算机组成原理的掌握。

## 主要任务

1. 构建一个存储系统，输入字节地址（16bits）、块数据Blk（32bits），数据准备信号BlkReady（1bit）、时钟周期信号Clk（1bit），输出是否命中信号Miss（1bit）、目标字节数据DataOut（8bits）。
2. 构建一个核心模块cache子电路，该 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，无写入机制，只读（类似指令 cache）。
3. 涉及cache 实现的三个关键技术：数据查找，地址映射，替换算法。

## 实验方案的总体设计

1. 需求分析：地址映射，cache槽中要有与主存地址块号位数相同的标记位，通过比对标记位来判断是否为所寻找块信息；数据查找，子电路要能够通过主存地址中的块地址偏移（2bits）找到cache块数据（4B）中的对应字节；替换算法，当cache槽全部未命中时，要优先将主存信息载入空槽，没有空槽则载入使用频率最低槽。
2. 实验步骤：设计cache槽；查找命中逻辑；空行载入逻辑；替换载入逻辑；数据读取逻辑。

## 实验的实现过程

（1）设计cache槽：有效位VALID（1bit）来表示此槽是否为空，VALID=0为空；标记位TAG（14bits）与主存地址相对应，通过TAG的比对来判断是否为所寻地址信息；具有淘汰计数器CNT（16bits）来统计cache槽使用频率，方便找到最少使用的cache槽进行替换；块数据（32bits）四字节。

（2）查找命中逻辑：每个时钟周期，读取每个cache槽的当前有效位valid（1bit）与当前标记位tag（14bits），通过将每个cache槽与所寻主存地址块号进行比较，相等且有效，即当前槽valid=1时为命中，该槽为命中行；若八行全未命中，则需要将当前主存信息重新载入cache。

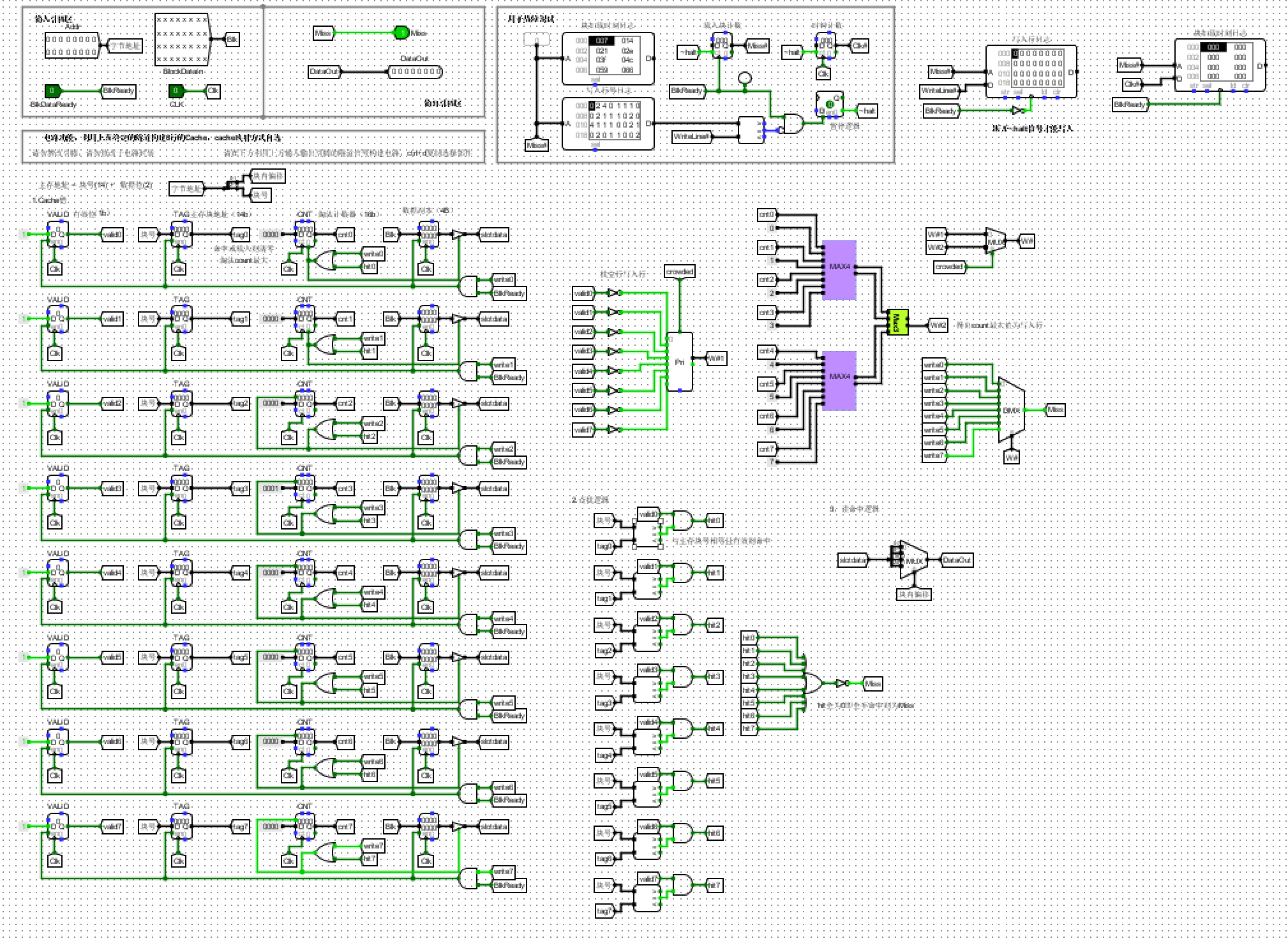
（3）空行载入逻辑：当未命中时，首选空行进行主存信息直接载入，当valid=0时为空行，首次出现的空行为载入行，若valid全为1，则无空行。

（4）替换载入逻辑：无空行情况，需要找到使用频率最低行，用计数器统计调用频率，每个时钟周期计数器加一，每次调用，包括命中和载入，计数器清零，计数器值越大，使用频率越低，选择计数器最大值槽为载入行，进行替换。

（5）数据读取逻辑：当命中时，输出当前数据块，当未命中时，先载入主存信息，再输出当前新数据块，再根据主存地址的块内偏移量选中数据块内对应字节为输出数据。

## 电路图及相关说明

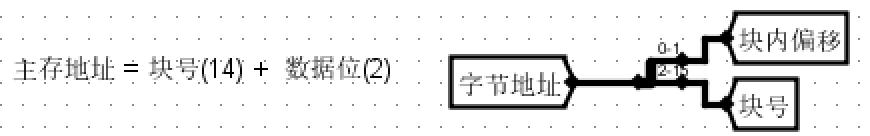
（1）总电路图展示



**图1.1全相联cache图 总电路图**

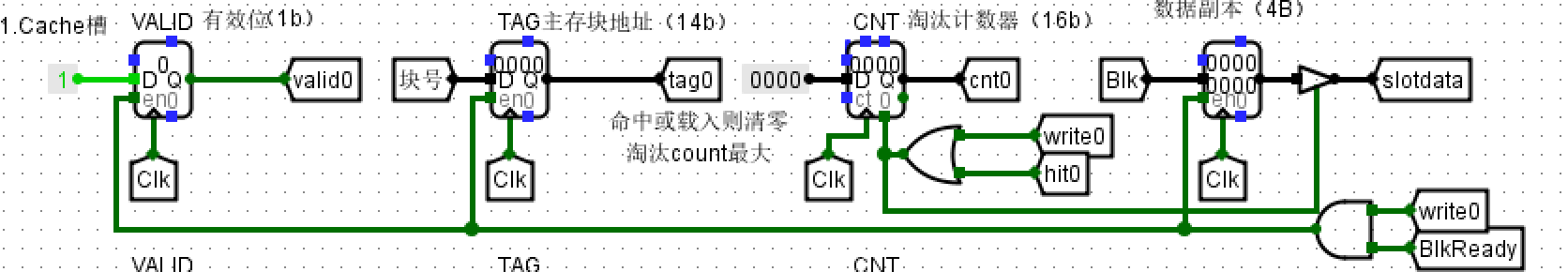
（2）细节图说明

通过分线器对主存字节地址（16bits）进行分线，前两位为块内偏移地址，代表块内四个字节中的其中一个字节对应序号，剩下的位数为块号，将主存按块（32bits，4B）分布，块号代表第几块。分线后更方便读取和查找。具体电路如图1.2所示。



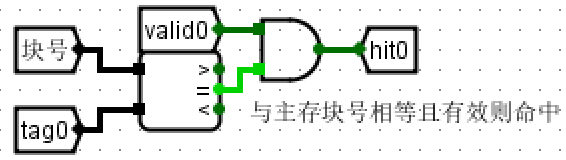
**图1.2全相联cache图 字节地址分线图**

一位寄存器存放有效位VALID，每个时钟周期，该寄存器将存放数传给valid；十四位寄存器存放标记位TAG，每个时钟周期，该寄存器将存放内容传给tag；十六位计数器存放淘汰计数位，每个时钟周期，计数器加一；三十二位寄存器存放数据副本；当write信号与数据准备信号同时为1，即被选为载入行且数据准备好时，将1传入有效位寄存器，将主存地址中块号传入标记位寄存器，将Blk数据传入数据寄存器；当write或hit为1，即命中或被选为载入行时，被调用一次，淘汰计数器清零，三态门联通，数据副本寄存器将存放内容传给slotdata。具体电路如图1.3所示：



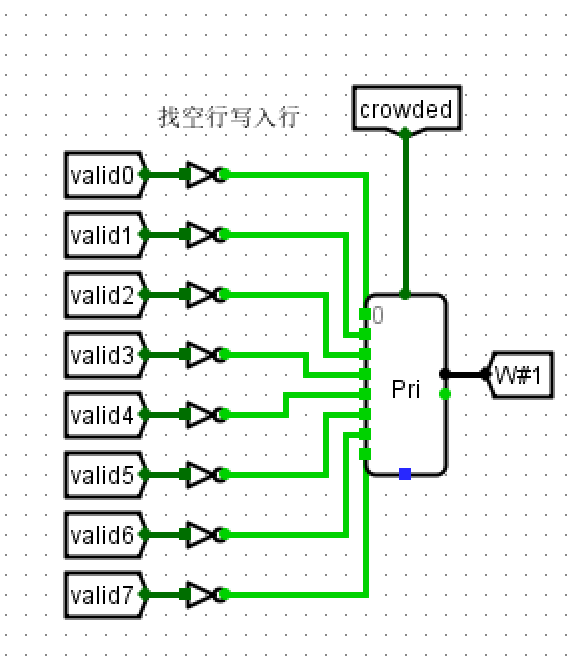
**图1.3全相联cache图 单cache槽图**

通过十四位比较器，比较主存地址块号与每行传出的tag是否相同，通过与门，确定每行传出的有效位valid=1及tag匹配这两个条件同时成立时为命中此行。具体电路如图1.4所示。



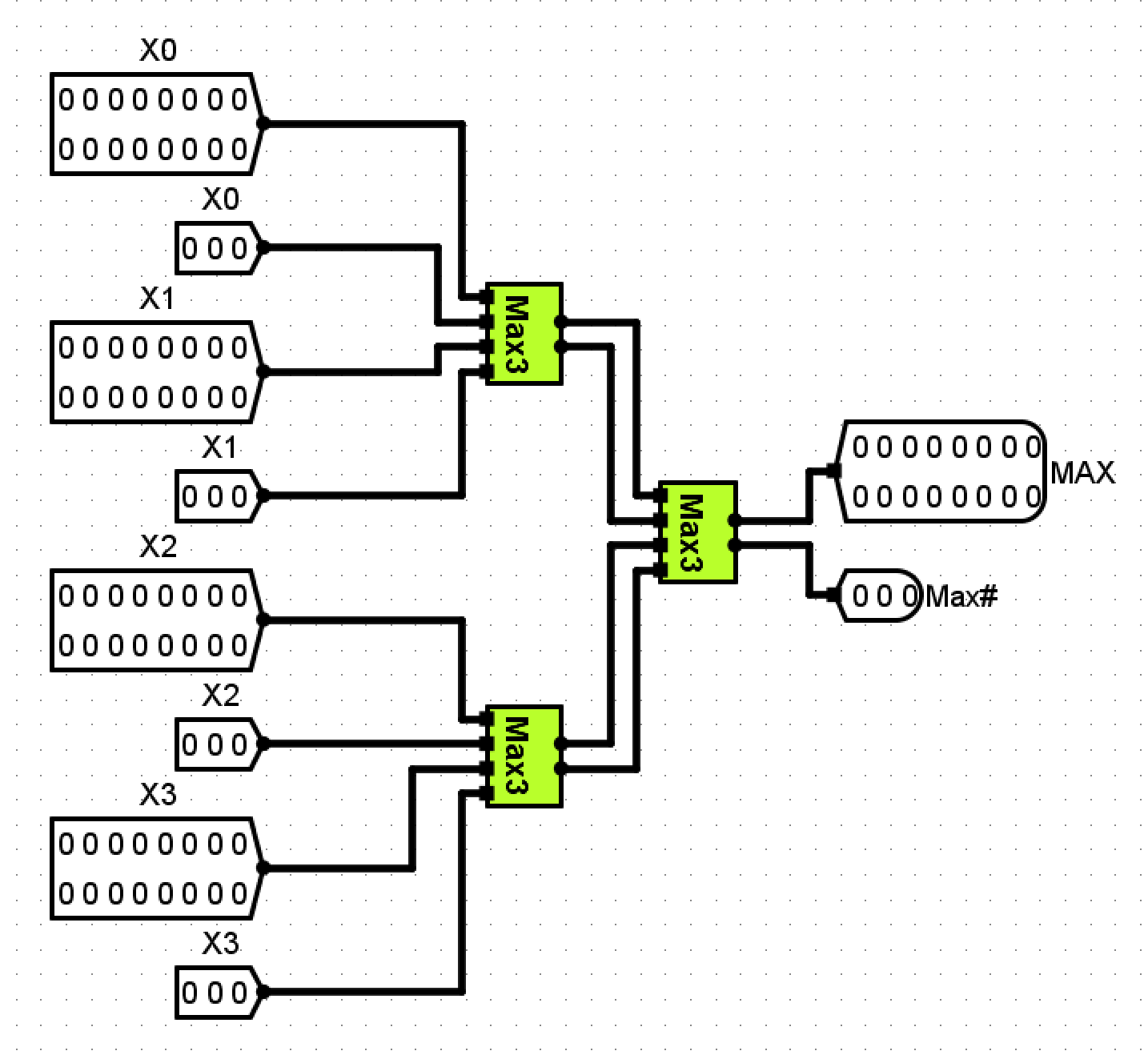
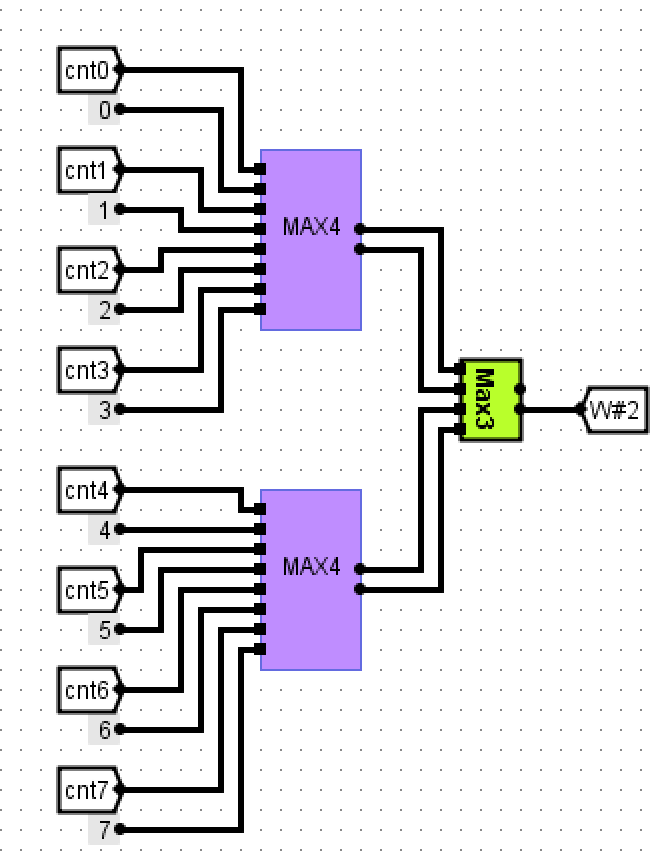
**图1.4全相联cache图 单命中查找图**

通过三位优先编码器，输入通过非门后的八行valid值的非值，输出从下至上首次出现1（valid=0），即为空的值的三位序号，若输入无1，即无空行，输出1至crowded。具体电路如图1.5所示。



**图1.5全相联cache图 空行查找图**

根据题目给出的Max3模块（输入两个十六位数及其副本三位数，比较输出较大值的副本），设计可以输入四个十六位数及其三位数副本并比较输出最大值副本的Max4（1.6右），通过两个Max4及一个Max3层层比较出八行输出的淘汰计数器的最大值所在行号。具体电路如图1.6所示。



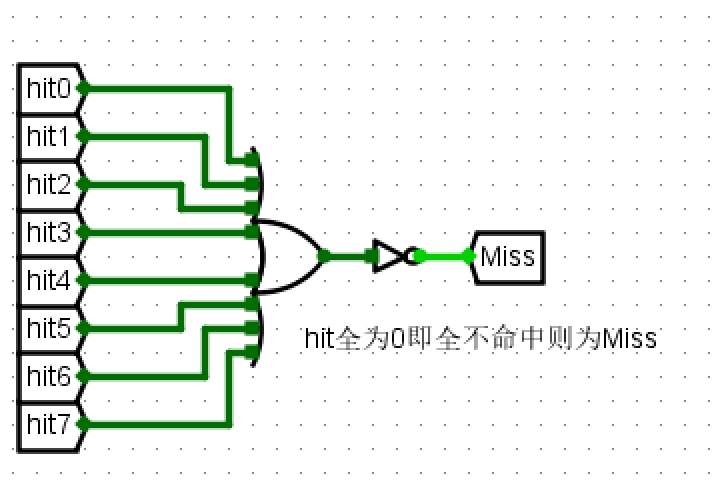
**图1.6全相联cache图 cnt最大值查找图**

通过多路选择器，以crowded为选择条件，当crowded为0时，即不为满时，选择第0行的空行W#1为最终载入行，当crowded为1时，即为满时，选择第1行的替换行W#2为最终载入行。具体电路如图1.7所示。



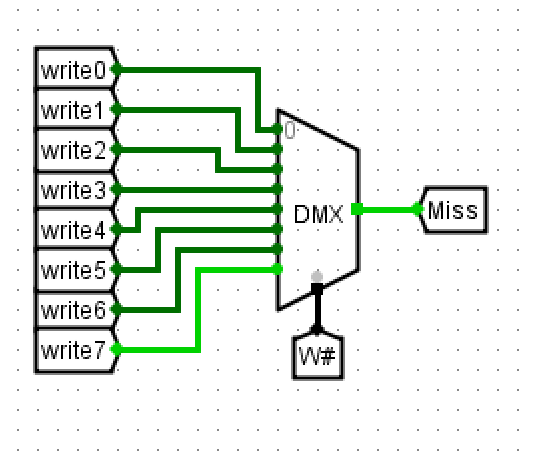
**图1.7全相联cache图 最终载入行选择图**

当全部不命中，即全部hit为0时才为特殊情况，所以选择用或门，只有在全部为0时输出0，其他全为1，但特殊情况下需要Miss为1，所以还需要经过一个非门后传给Miss。具体电路如图1.8所示。



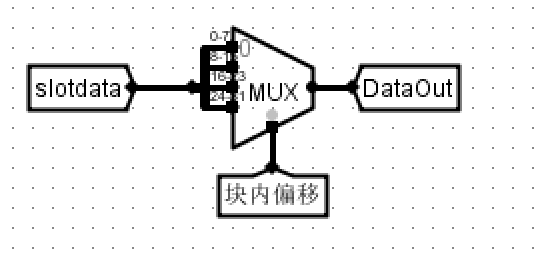
**图1.8全相联cache图 Miss值确定图**

当确定最终载入行后，需要给对应行的write信号赋值1，这里通过解复用器，当Miss为1时，将1传给W#所指向序号的write。具体电路如图1.9所示。



**图1.9全相联cache图 赋值write信号图**

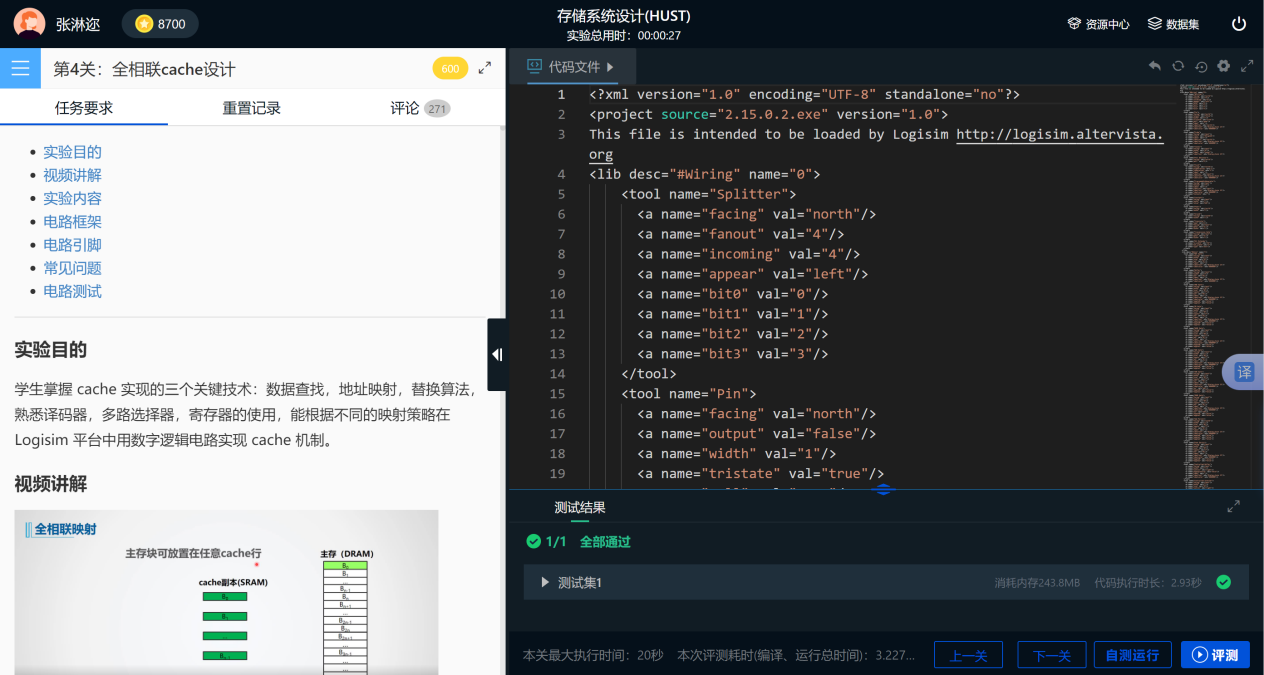
通过分线器，将块数据（32位）按字节分为四段，再通过多路选择器根据主存地址块内偏移选择出最终输出字节。具体电路如图1.10所示。



**图1.10全相联cache图 数据输出图**

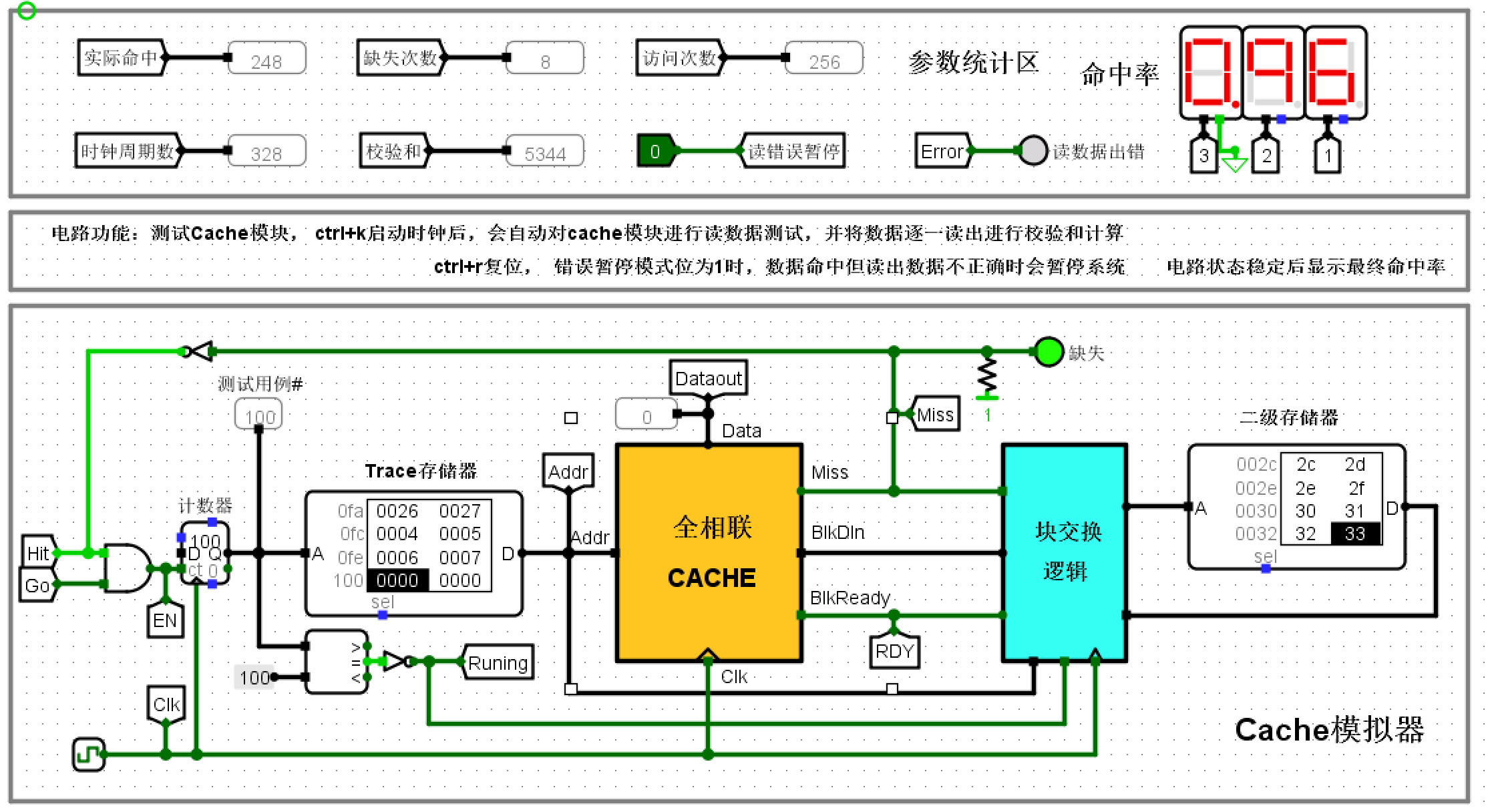
## 实验数据测试

（1）头歌平台实验通过截图。



**图1.11全相联cache图 头歌平台实验通过截图**

（2）程序自动测试截图。



**图1.12全相联cache图 程序自动测试截图**

# 实验二、4路组相连cache设计

## 1、实验目的

（1）理解4路组相联缓存的工作原理：通过实验，加深对4路组相联缓存如何根据地址信息直接定位到缓存行的理解。

（2）掌握缓存映射算法：通过实践，学会如何实现4路组相联映射算法，并理解其与直接映射和全相联映射的区别。

（3）分析缓存性能：通过实验，学会如何评估和比较不同缓存策略对系统性能的影响，包括命中率和访问时间。

（4）培养问题解决能力：通过解决实验过程中遇到的问题，提高分析问题和解决问题的能力。

（5）增强理论联系实际的能力：将课堂上学到的理论知识应用到logisim的实际电路设计中，加深对计算机组成原理的掌握。

## 2、主要任务

（1）构建一个存储系统，输入字节地址（16bits）、块数据Blk（32bits），数据准备信号BlkReady（1bit）、时钟周期信号Clk（1bit），输出是否命中信号Miss（1bit）、目标字节数据DataOut（8bits）。

（2）构建一个核心模块cache子电路，该 cache 模块共包括8个 cache 行，每四行为一个组，每个数据块包含4个字节共32位数据，无写入机制，只读（类似指令 cache）。

（3）涉及cache 实现的三个关键技术：数据查找，地址映射，替换算法。

## 3、实验方案的总体设计

1. 需求分析：地址映射，每四块为一个组，一个cache中有两个组，主存按组号被分为两个群，根据主存的群偏移量找到组号后，cache槽要有对应的标记位与主存地址的群号进行比对；数据查找，子电路要能够通过主存地址中的块地址偏移（2bits）找到cache块数据（4B）中的对应字节；替换算法，当cache槽全部未命中时，要优先将主存信息载入空槽，没有空槽则载入使用频率最低槽。
2. 实验步骤：设计cache槽；查找命中逻辑；空行载入逻辑；替换载入逻辑；数据读取逻辑。

## 4、实验的实现过程

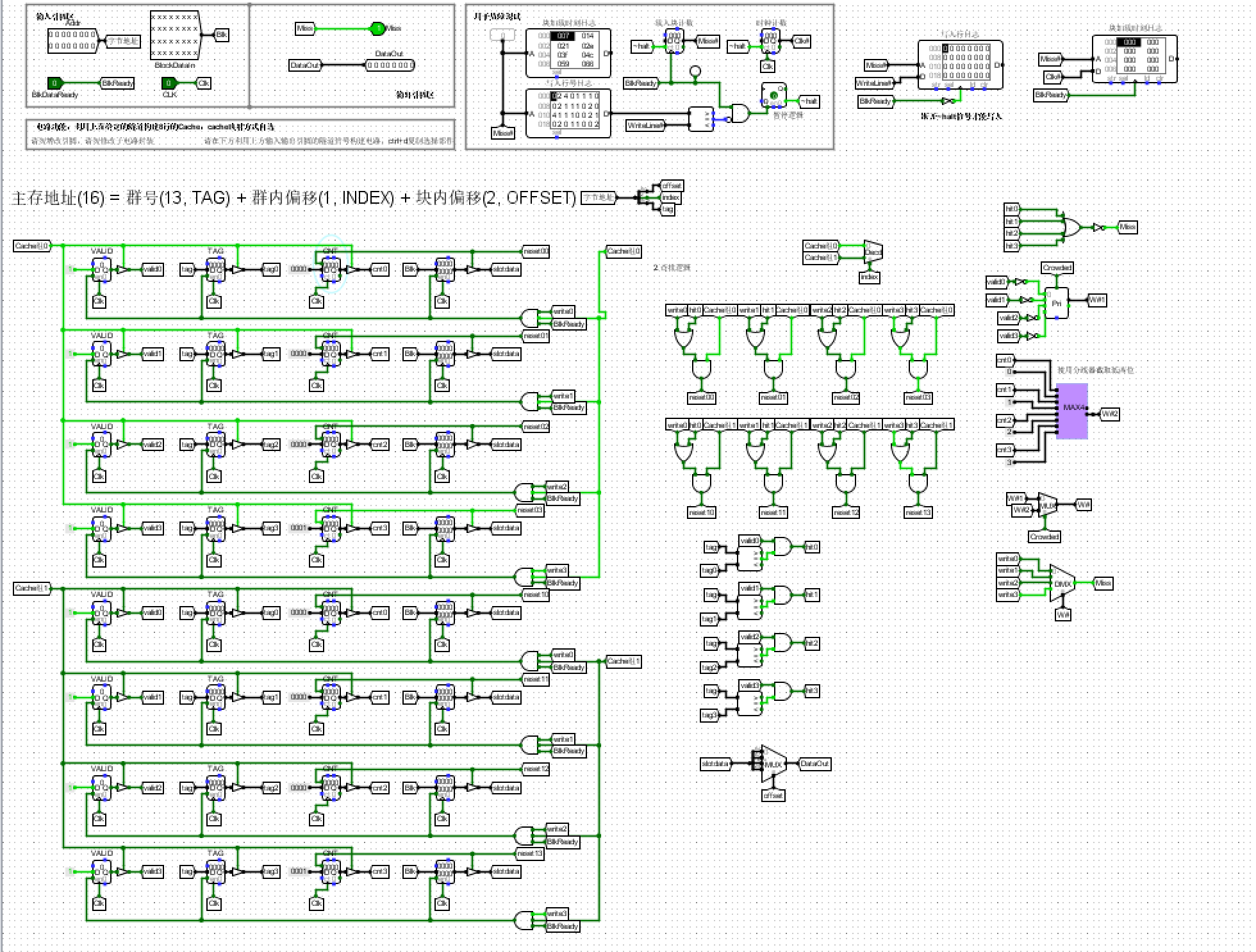
1. 设计cache槽：有效位VALID（1bit）来表示此槽是否为空，VALID=0为空；标记位TAG（13bits）与主存地址群号相对应，在已知组号的情况下通过TAG的比对来判断是否含所寻信息；具有淘汰计数器CNT（16bits）来统计cache槽使用频率，方便找到最少使用的cache槽进行替换；块数据（32bits）四字节。每四个cache槽与一组信号相连。
2. 查找命中逻辑：每个时钟周期，根据主存地址的群内偏移量INDEX确定组数，读取cache此组四个cache槽的当前有效位valid（1bit）与当前标记位tag（13bits），通过将每个标记位tag与所寻主存地址群号进行比较，相等且有效，即当前槽valid=1时为命中，该槽为命中行；若组内四行全未命中，则需要将当前主存信息重新载入同组。但由于此处hit只能确认组内第几行，不能确定cache内第几行，所以还需要通过hit和组号双条件来确定具体的行号。
3. 空行载入逻辑：当未命中时，首选同组空行进行主存信息直接载入，当valid=0时为空行，首次出现的空行为载入行，若valid全为1，则无空行。

（4）替换载入逻辑：无空行情况，需要找到同组内使用频率最低行，用计数器统计调用频率，每个时钟周期计数器加一，每次调用，包括命中和载入，计数器清零，计数器值越大，使用频率越低，选择计数器最大值槽为载入行，进行替换。

（5）数据读取逻辑：当命中时，组内行号结合组号找到并输出当前数据块，当未命中时，先载入主存信息，再输出当前新数据块，再根据主存地址的块内偏移量选中数据块内对应字节为输出数据。

## 5、电路图及相关说明

1. 总电路图展示



**图2.1四路组相联cache图 总电路图**

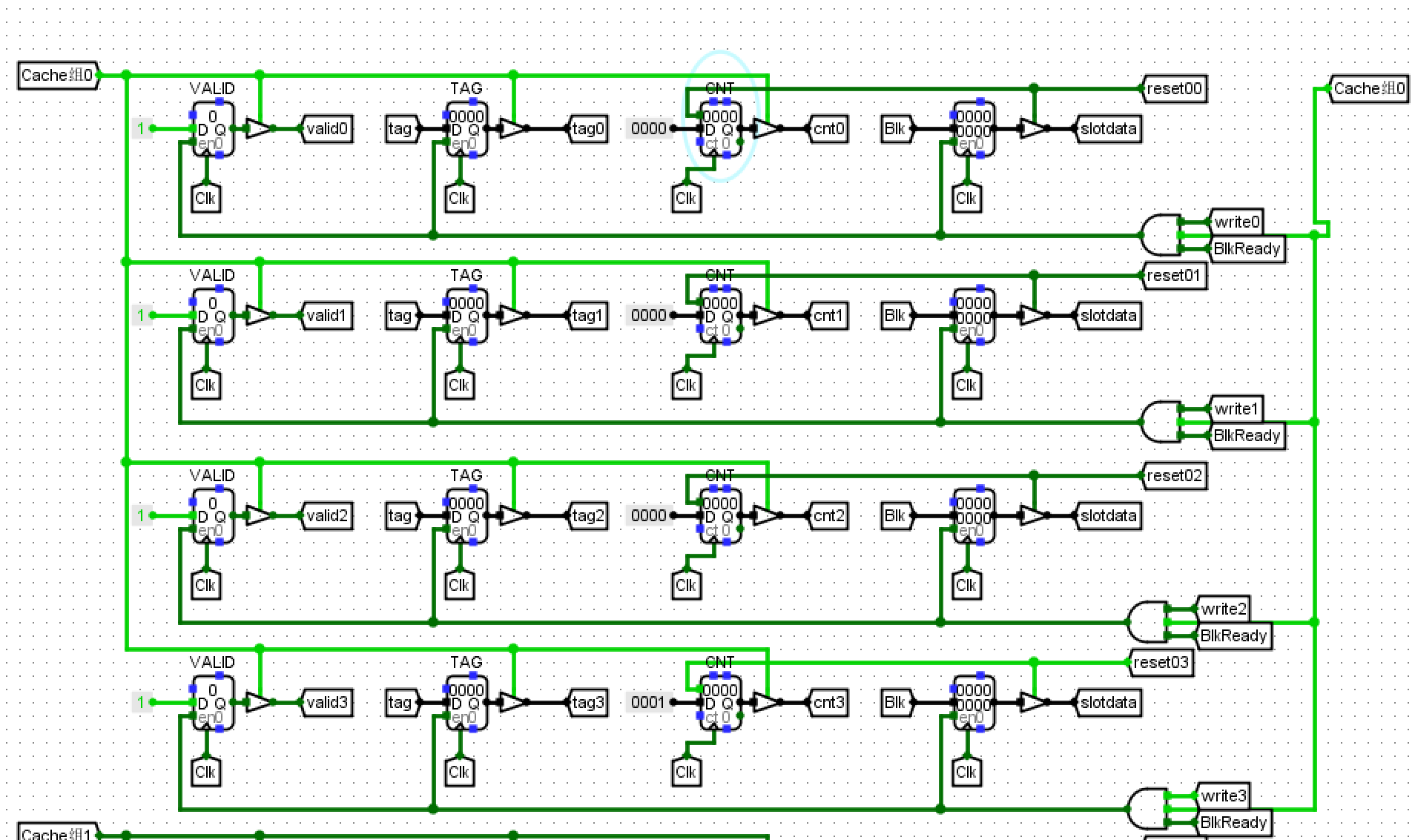
1. 细节图说明

通过分线器对主存字节地址（16bits）进行分线，前两位为块内偏移地址，代表块内四个字节中的其中一个字节对应序号，后一位为群内偏移地址，代表两个组中的其中一组，剩下的为群号（13bits），将主存按两个组分别一个群分配，群号代表第几群，每群内两个快。分线后更方便读取和查找。具体电路如图2.2所示。



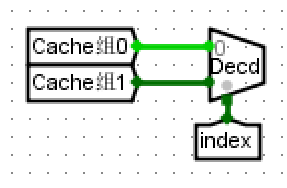
**图2.2四路组相联cache图 字节地址分线图**

一位寄存器存放有效位VALID；十三位寄存器存放标记位TAG；十六位计数器存放淘汰计数位，每个时钟周期，计数器加一；三十二位寄存器存放数据副本；当当前组号被选中并传入信号时，三态门联通，VALID寄存器当前数据输入valid，TAG寄存器当前数据输入tag，CNT计数器当前数据输入cnt；在组号被选中情况下，当write信号与数据准备信号同时为1，即被选为载入行且数据准备好时，将1传入有效位寄存器，将主存地址中块号传入标记位寄存器，将Blk数据传入数据寄存器；当reset为1，即命中或被选为载入行时，被调用一次，淘汰计数器清零，三态门联通，数据副本寄存器将存放内容传给slotdata。最终四个cache槽通过组号类似并联形式连接形成一个组。具体电路如图2.3所示。



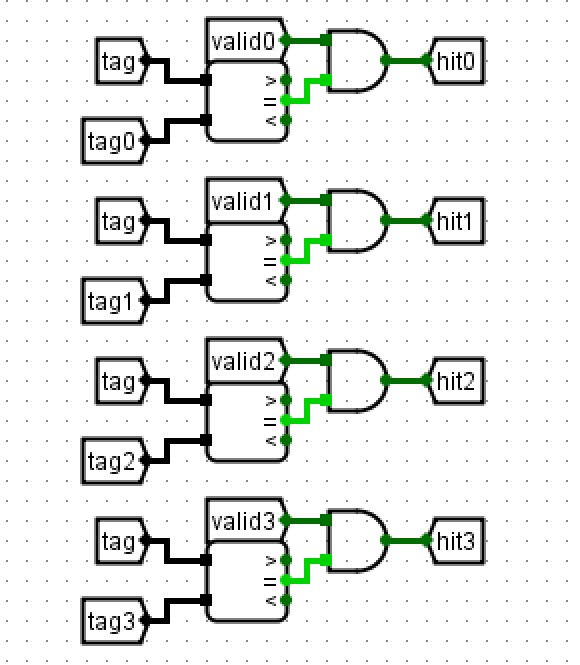
**图2.3四路组相联cache图 组cache槽图**

通过一位译码器，将主存地址中的群内偏移量INDEX转成信号传入对应组号，将范围缩小到cache其中四行。具体电路如图2.4所示。



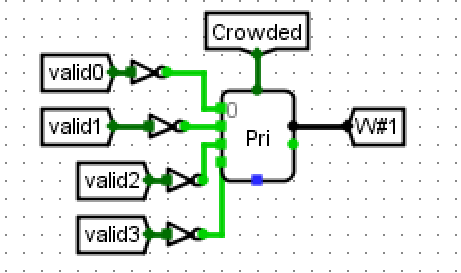
**图2.4四路组相联cache图 群内偏移量译码图**

主存地中每个组都有各自的群序列号，所以只需要比对是否为同群号（13bits）来确定是否命中。通过十三位比较器，比较主存地址群号与组内每行传出的四个tag是否存在相同，通过与门，确定每行传出的有效位valid=1及tag匹配这两个条件同时成立时为命中组内的此行。具体电路如图2.5所示。



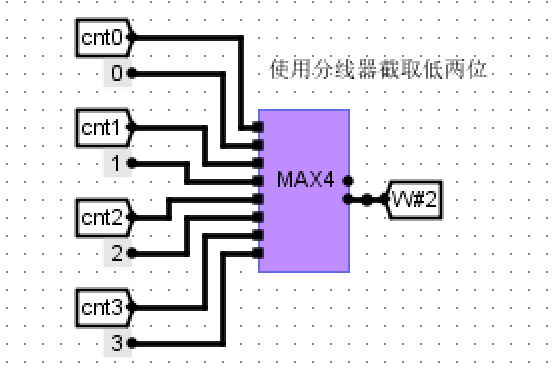
**图2.5四路组相联cache图 单命中查找图**

通过二位优先编码器，输入通过非门后的四行valid值的非值，输出从下至上首次出现1（valid=0），即为空的值的组内序号W#1（2bits），若输入无1，即无空行，输出1至crowded。具体电路如图2.6所示。



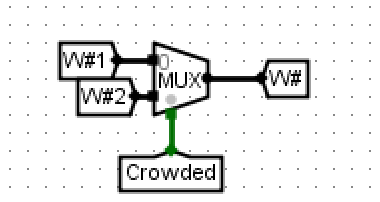
**图2.6四路组相联cache图 空行查找图**

通过一个Max4比较出四行输出的淘汰计数器的最大值所在组内行号W#2（2bits）。具体电路如图2.7所示。



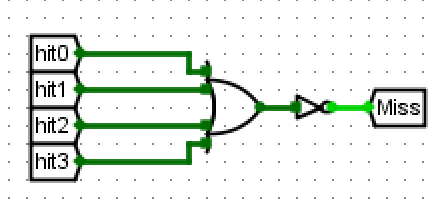
**图2.7四路组相联cache图 cnt最大值查找图**

通过多路选择器，以crowded为选择条件，当crowded为0时，即不为满时，选择第0行的空行W#1为最终载入行，当crowded为1时，即为满时，选择第1行的替换行W#2为最终组内的载入行。具体电路如图2.8所示。



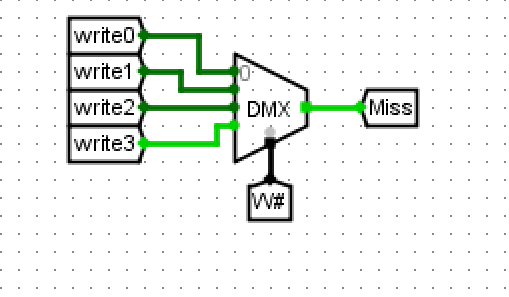
**图2.8四路组相联cache图 最终载入组内行选择图**

通过或门，在hit全部为0时输出0，其他全为1，经过一个非门后传给Miss。具体电路如图2.9所示。



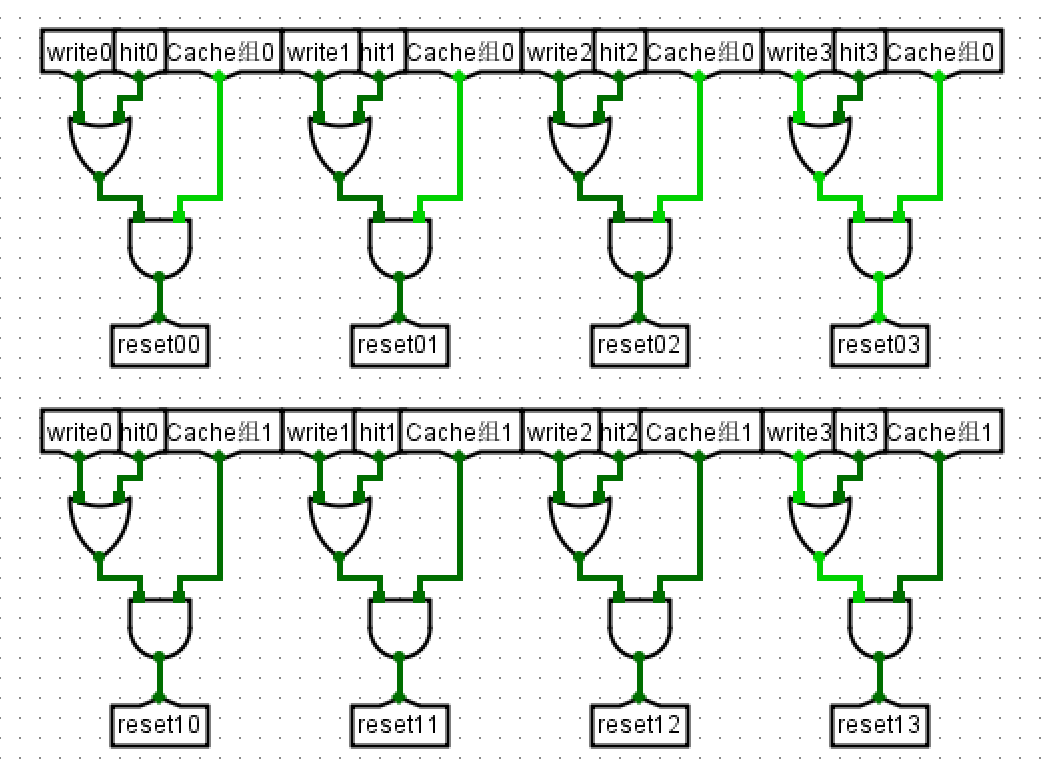
**图2.9四路组相联cache图 Miss值确定图**

当确定最终载入组内行后，需要给对应行的write信号赋值1，这里通过解复用器，当Miss为1时，将1传给W#所指向序号的write。具体电路如图2.10所示。



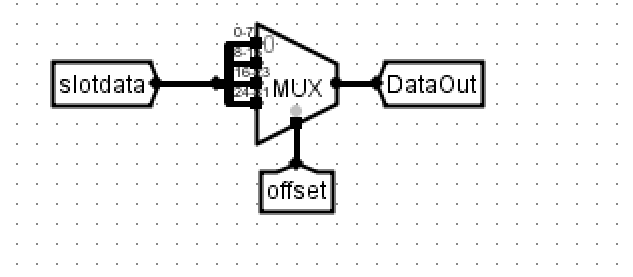
**图2.10四路组相联cache图 赋值write信号图**

此处用的hit和write都是组内的行数，只有结合组号和组内行数，才能确定具体的cache行数。通过或门表示满足hit和write任意一项，通过与门表示组号和组行数的同时存在，得到调用的具体cache行数（3bits）。具体电路如图2.11所示。



**图2.11四路组相联cache图 调用cache行确定图**

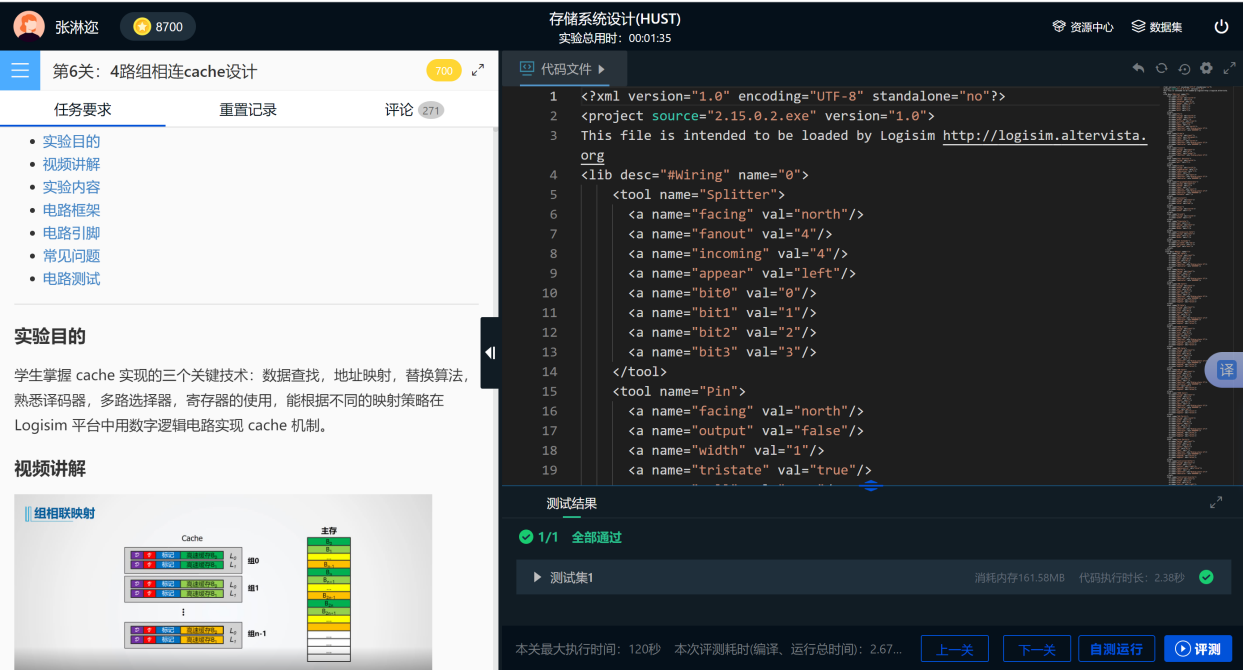
通过分线器，将调用cache行的块数据（32位）按字节分为四段，再通过多路选择器根据主存地址块内偏移选择出最终输出字节。具体电路如图2.12所示。



**图2.12四路组相联cache图 数据输出图**

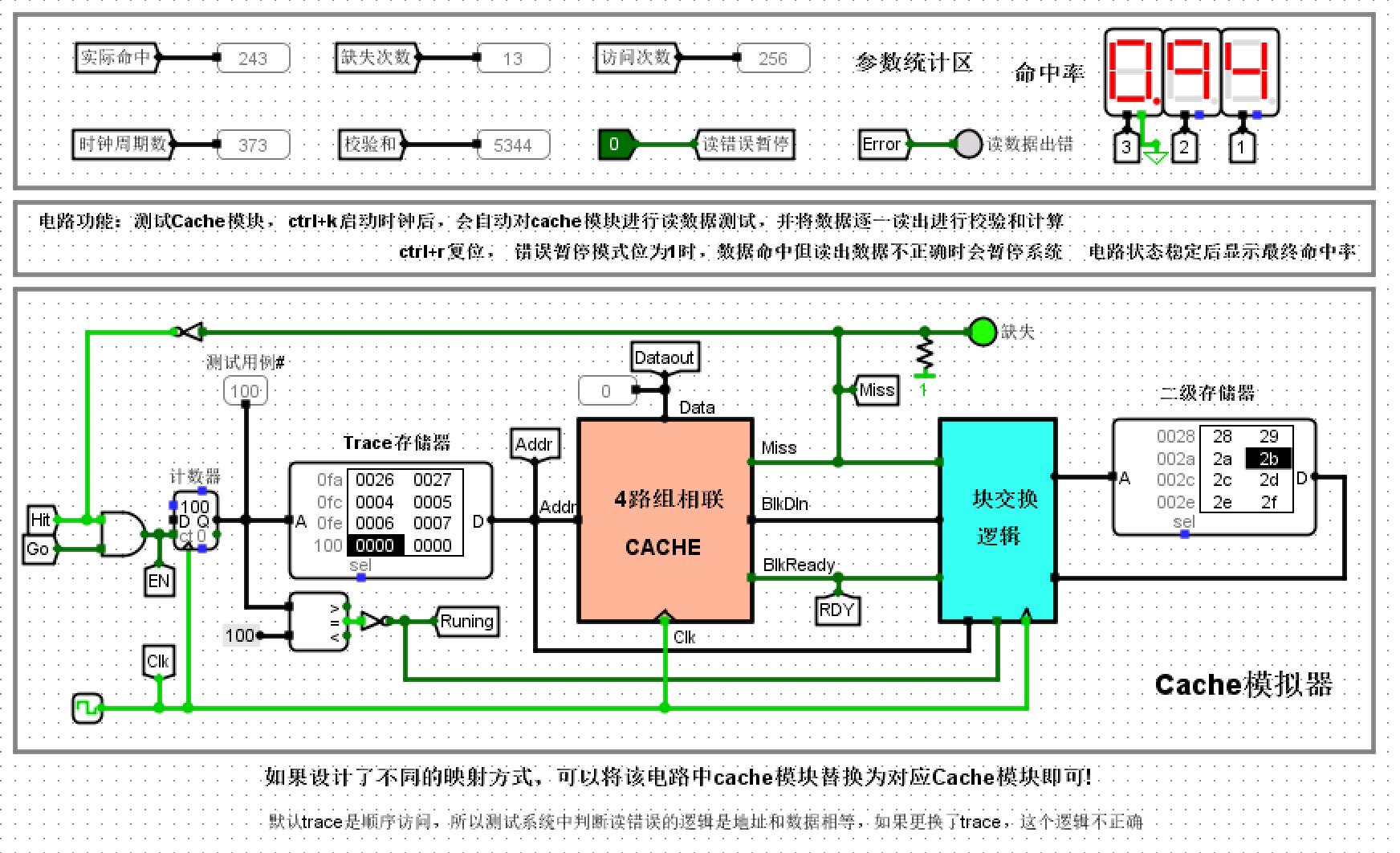
## 6、实验数据测试

（1）头歌平台实验通过截图。



**图2.13四路组相联cache图 头歌平台验证通过截图**

（2）程序自动测试截图。



**图2.14四路组相联cache图 程序自动测试截图**

# 实验三、2路组相联cache设计

## 1、实验目的

（1）理解2路组相联缓存的工作原理：通过实验，加深对2路组相联缓存如何根据地址信息直接定位到缓存行的理解。

（2）掌握缓存映射算法：通过实践，学会如何实现2路组相联映射算法，并理解其与直接映射和全相联映射的区别。

（3）分析缓存性能：通过实验，学会如何评估和比较不同缓存策略对系统性能的影响，包括命中率和访问时间。

（4）培养问题解决能力：通过解决实验过程中遇到的问题，提高分析问题和解决问题的能力。

（5）增强理论联系实际的能力：将课堂上学到的理论知识应用到logisim的实际电路设计中，加深对计算机组成原理的掌握。

## 2、主要任务

（1）构建一个存储系统，输入字节地址（16bits）、块数据Blk（32bits），数据准备信号BlkReady（1bit）、时钟周期信号Clk（1bit），输出是否命中信号Miss（1bit）、目标字节数据DataOut（8bits）。

（2）构建一个核心模块cache子电路，该 cache 模块共包括8个 cache 行，每两行为一个组，每个数据块包含4个字节共32位数据，无写入机制，只读（类似指令 cache）。

（3）涉及cache 实现的三个关键技术：数据查找，地址映射，替换算法。

## 3、实验方案的总体设计

1. 需求分析：地址映射，每两块为一个组，一个cache中有四个组，主存按组号被分为四个群，根据主存的群偏移量找到组号后，cache槽要有对应的标记位与主存地址的群号进行比对；数据查找，子电路要能够通过主存地址中的块地址偏移（2bits）找到cache块数据（4B）中的对应字节；替换算法，当cache槽全部未命中时，要优先将主存信息载入空槽，没有空槽则载入使用频率最低槽。
2. 实验步骤：设计cache槽；查找命中逻辑；空行载入逻辑；替换载入逻辑；数据读取逻辑。

## 4、实验的实现过程

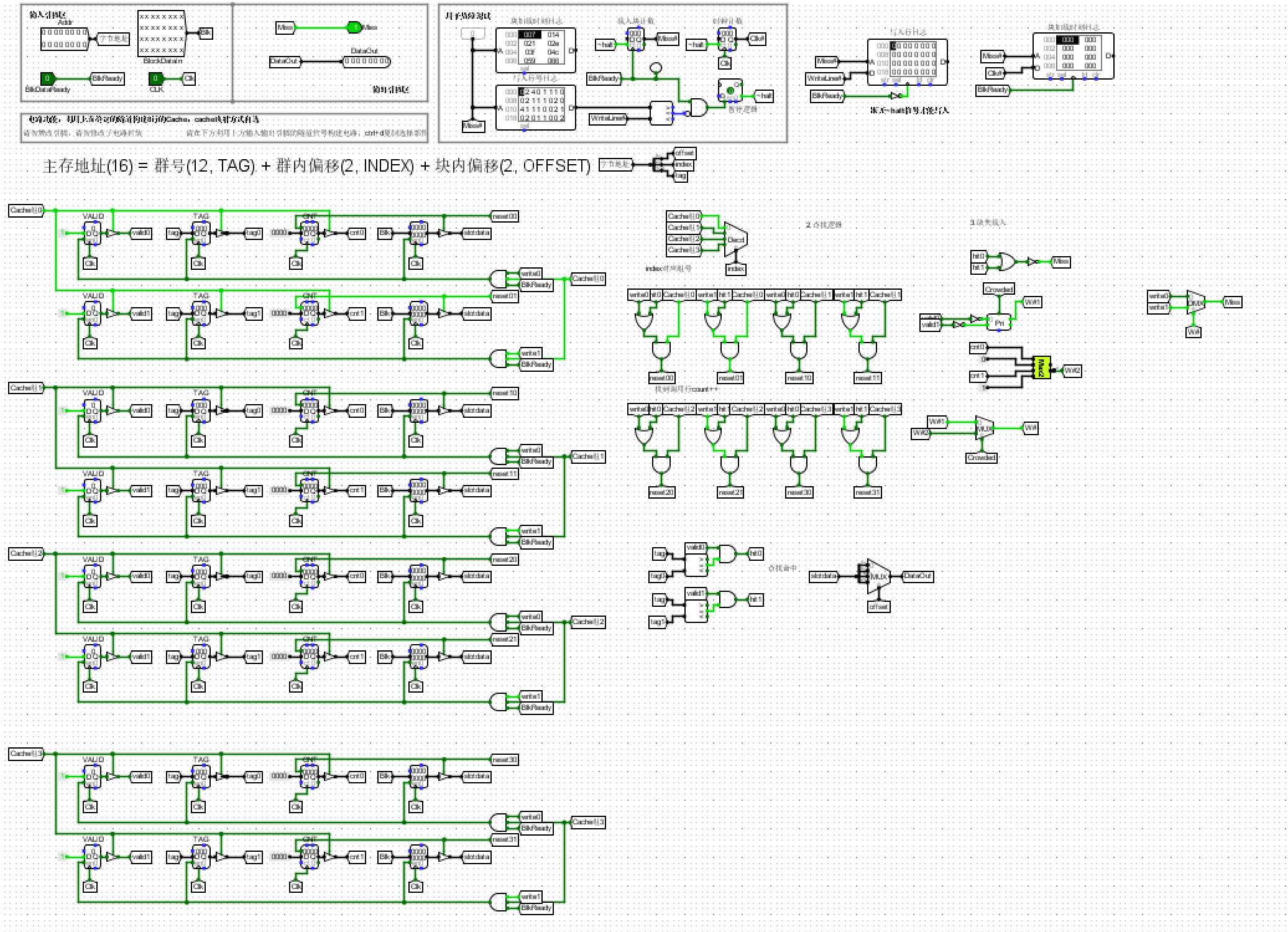
1. 设计cache槽：有效位VALID（1bit）来表示此槽是否为空，VALID=0为空；标记位TAG（12bits）与主存地址群号相对应，在已知组号的情况下通过TAG的比对来判断是否含所寻信息；具有淘汰计数器CNT（16bits）来统计cache槽使用频率，方便找到最少使用的cache槽进行替换；块数据（32bits）四字节。每四个cache槽与一组信号相连。
2. 查找命中逻辑：每个时钟周期，根据主存地址的群内偏移量INDEX确定组数，读取cache此组四个cache槽的当前有效位valid（1bit）与当前标记位tag（12bits），通过将每个标记位tag与所寻主存地址群号进行比较，相等且有效，即当前槽valid=1时为命中，该槽为命中行；若两行全未命中，则需要将当前主存信息重新载入同组。但由于此处hit只能确认组内第几行，不能确定cache内第几行，所以还需要通过hit和组号双条件来确定具体的行号。
3. 空行载入逻辑：当未命中时，首选同组空行进行主存信息直接载入，当valid=0时为空行，首次出现的空行为载入行，若valid全为1，则无空行。

（4）替换载入逻辑：无空行情况，需要找到同组内使用频率最低行，用计数器统计调用频率，每个时钟周期计数器加一，每次调用，包括命中和载入，计数器清零，计数器值越大，使用频率越低，选择计数器最大值槽为载入行，进行替换。

（5）数据读取逻辑：当命中时，组内行号结合组号找到并输出当前数据块，当未命中时，先载入主存信息，再输出当前新数据块，再根据主存地址的块内偏移量选中数据块内对应字节为输出数据。

## 5、电路图及相关说明

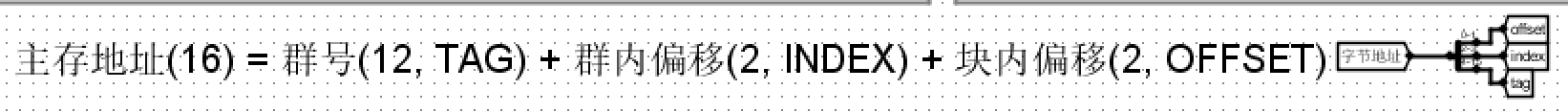
1. 总电路图展示



**图3.1二路组相联cache图 总电路图**

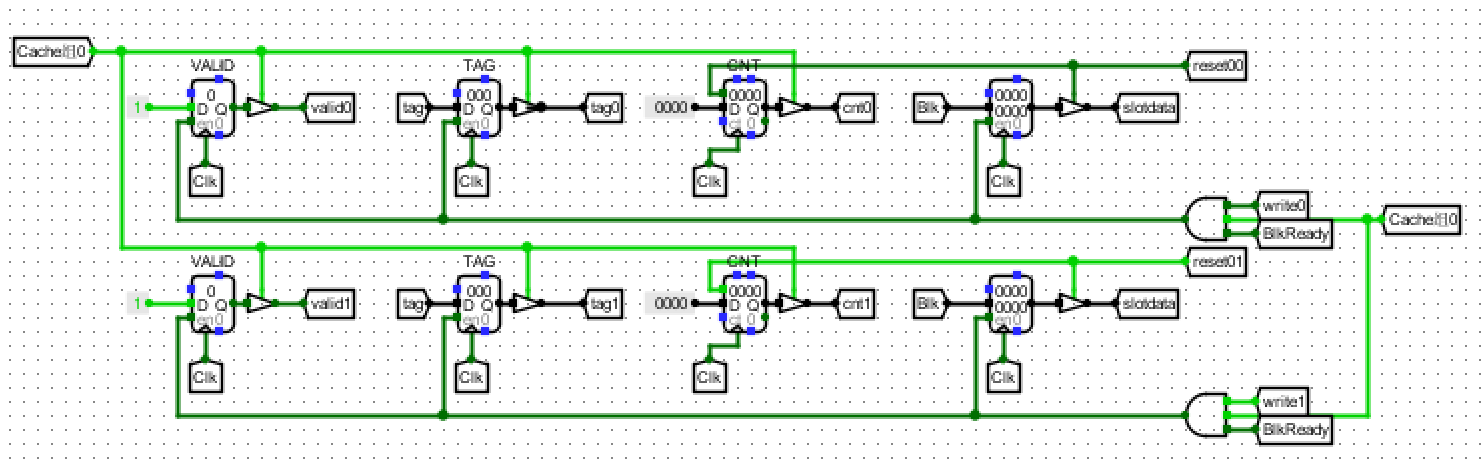
1. 细节图说明

通过分线器对主存字节地址（16bits）进行分线，前两位为块内偏移地址，代表块内四个字节中的其中一个字节对应序号，后两位位为群内偏移地址，代表四个组中的其中一组，剩下的为群号（13bits），将主存按两个组分别一个群分配，群号代表第几群，每群内两个快。分线后更方便读取和查找。具体电路如图3.2所示。



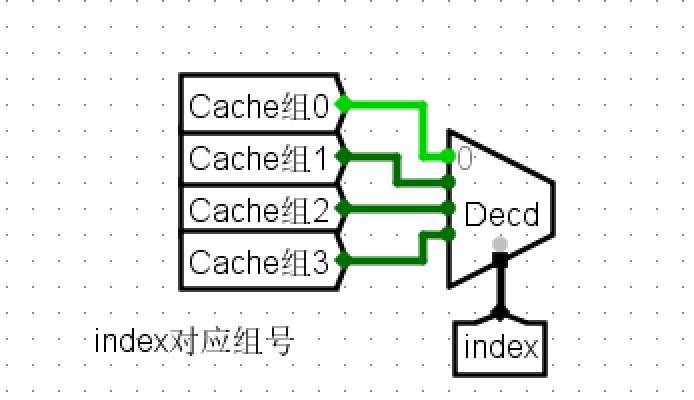
**图3.2二路组相联cache图 字节地址分线图**

一位寄存器存放有效位VALID；十二位寄存器存放标记位TAG；十六位计数器存放淘汰计数位，每个时钟周期，计数器加一；三十二位寄存器存放数据副本；当当前组号被选中并传入信号时，三态门联通，VALID寄存器当前数据输入valid，TAG寄存器当前数据输入tag，CNT计数器当前数据输入cnt；在组号被选中情况下，当write信号与数据准备信号同时为1，即被选为载入行且数据准备好时，将1传入有效位寄存器，将主存地址中块号传入标记位寄存器，将Blk数据传入数据寄存器；当reset为1，即命中或被选为载入行时，被调用一次，淘汰计数器清零，三态门联通，数据副本寄存器将存放内容传给slotdata。最终两个cache槽通过组号类似并联形式连接形成一个组。具体电路如图3.3所示。



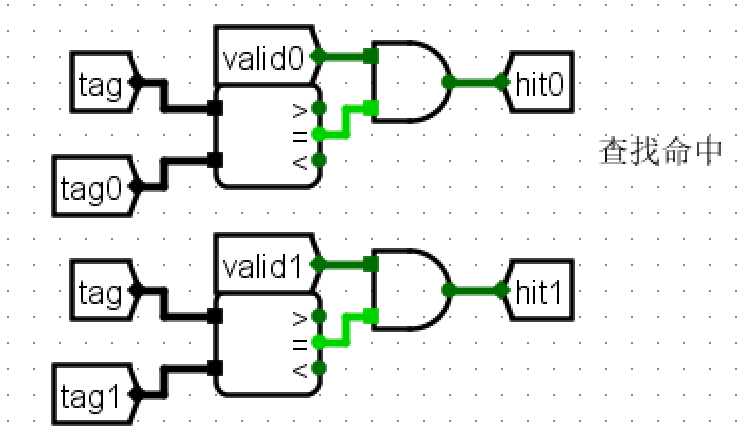
**图3.3二路组相联cache图 组cache槽图**

通过二位译码器，将主存地址中的群内偏移量INDEX转成信号传入对应组号，将范围缩小到cache其中两行。具体电路如图3.4所示。



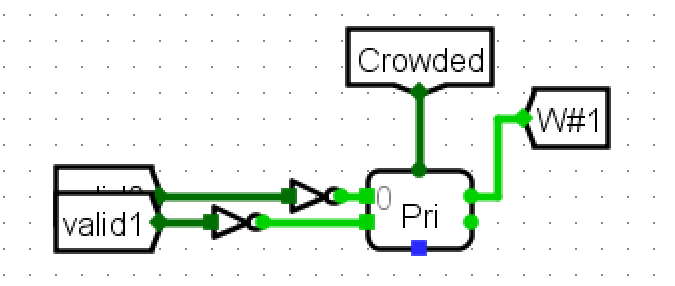
**图3.4二路组相联cache图 群内偏移量译码图**

主存地中每个组都有各自的群序列号，所以只需要比对是否为同群号（12bits）来确定是否命中。通过十二位比较器，比较主存地址群号与组内每行传出的两个tag是否存在相同，通过与门，确定每行传出的有效位valid=1及tag匹配这两个条件同时成立时为命中组内的此行。具体电路如图3.5所示。



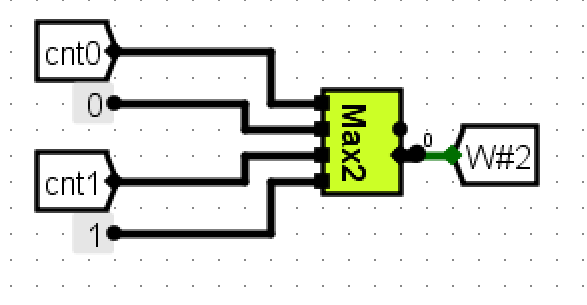
**图3.5二路组相联cache图 命中查找图**

通过一位优先编码器，输入通过非门后的两行valid值的非值，输出从下至上首次出现1（valid=0），即为空的值的组内序号W#1（1bit），若输入无1，即无空行，输出1至crowded。具体电路如图3.6所示。



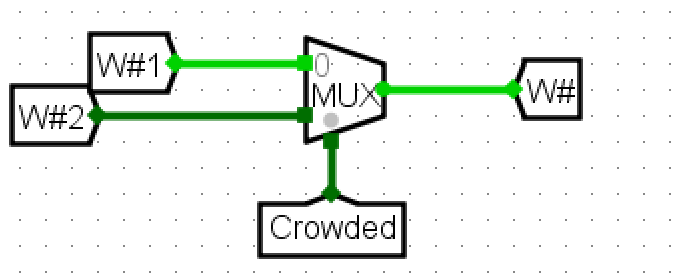
**图3.6二路组相联cache图 空行查找图**

通过一个Max2比较出两行输出的淘汰计数器的最大值所在组内行号W#2（1bit）。具体电路如图3.7所示。



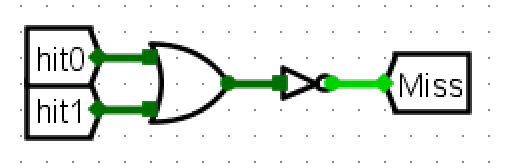
**图3.7二路组相联cache图 cnt最大值查找图**

通过多路选择器，以crowded为选择条件，当crowded为0时，即不为满时，选择第0行的空行W#1为最终载入行，当crowded为1时，即为满时，选择第1行的替换行W#2为最终组内的载入行。具体电路如图3.8所示。



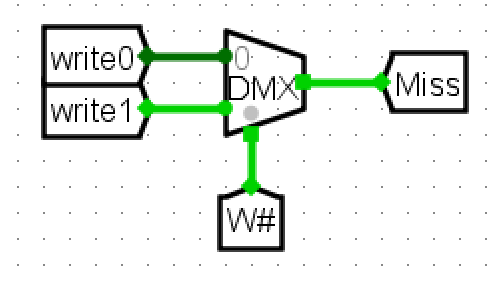
**图3.8二路组相联cache图 最终载入组内行选择图**

通过或门，在hit全部为0时输出0，其他全为1，经过一个非门后传给Miss。具体电路如图3.9所示。



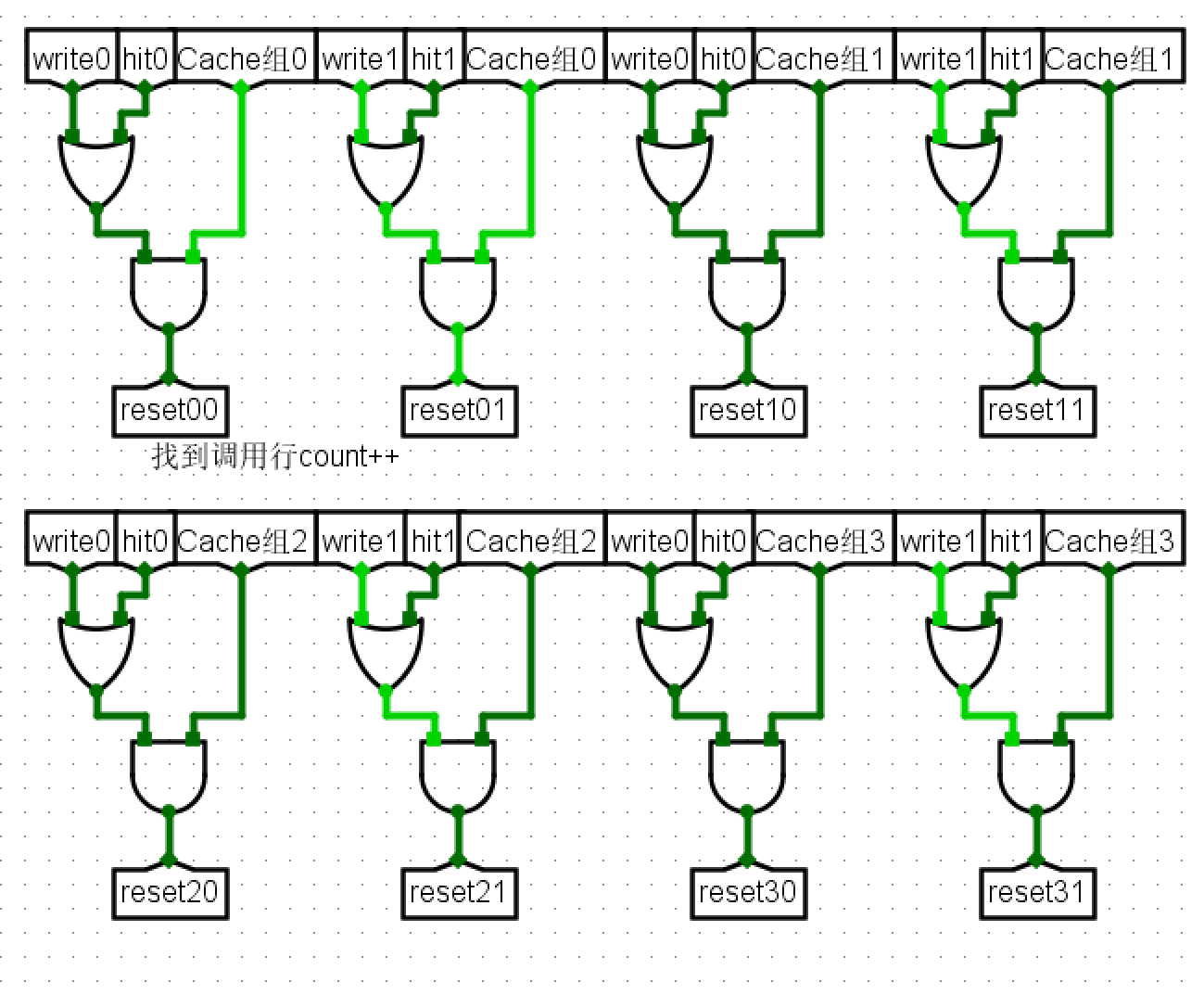
**图3.9二路组相联cache图 Miss值确定图**

当确定最终载入组内行后，需要给对应行的write信号赋值1，这里通过解复用器，当Miss为1时，将1传给W#所指向序号的write。具体电路如图3.10所示。



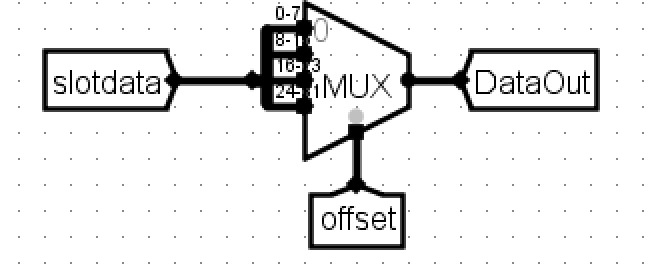
**图3.10二路组相联cache图 赋值write信号图**

此处用的hit和write都是组内的行数，只有结合组号和组内行数，才能确定具体的cache行数。通过或门表示满足hit和write任意一项，通过与门表示组号和组行数的同时存在，得到调用的具体cache行数（3bits）。具体电路如图3.11所示。



**图3.11二路组相联cache图 调用cache行确定图**

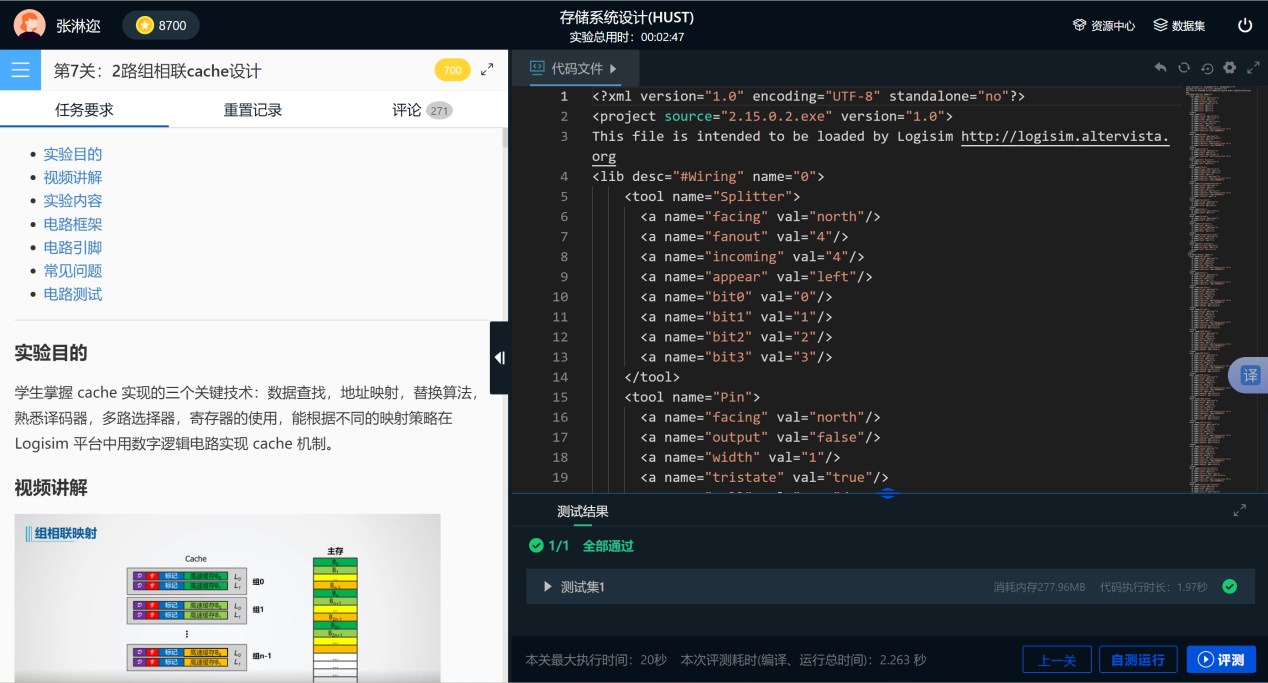
通过分线器，将调用cache行的块数据（32位）按字节分为四段，再通过多路选择器根据主存地址块内偏移选择出最终输出字节。具体电路如图3.12所示。



**图3.11二路组相联cache图 数据输出图**

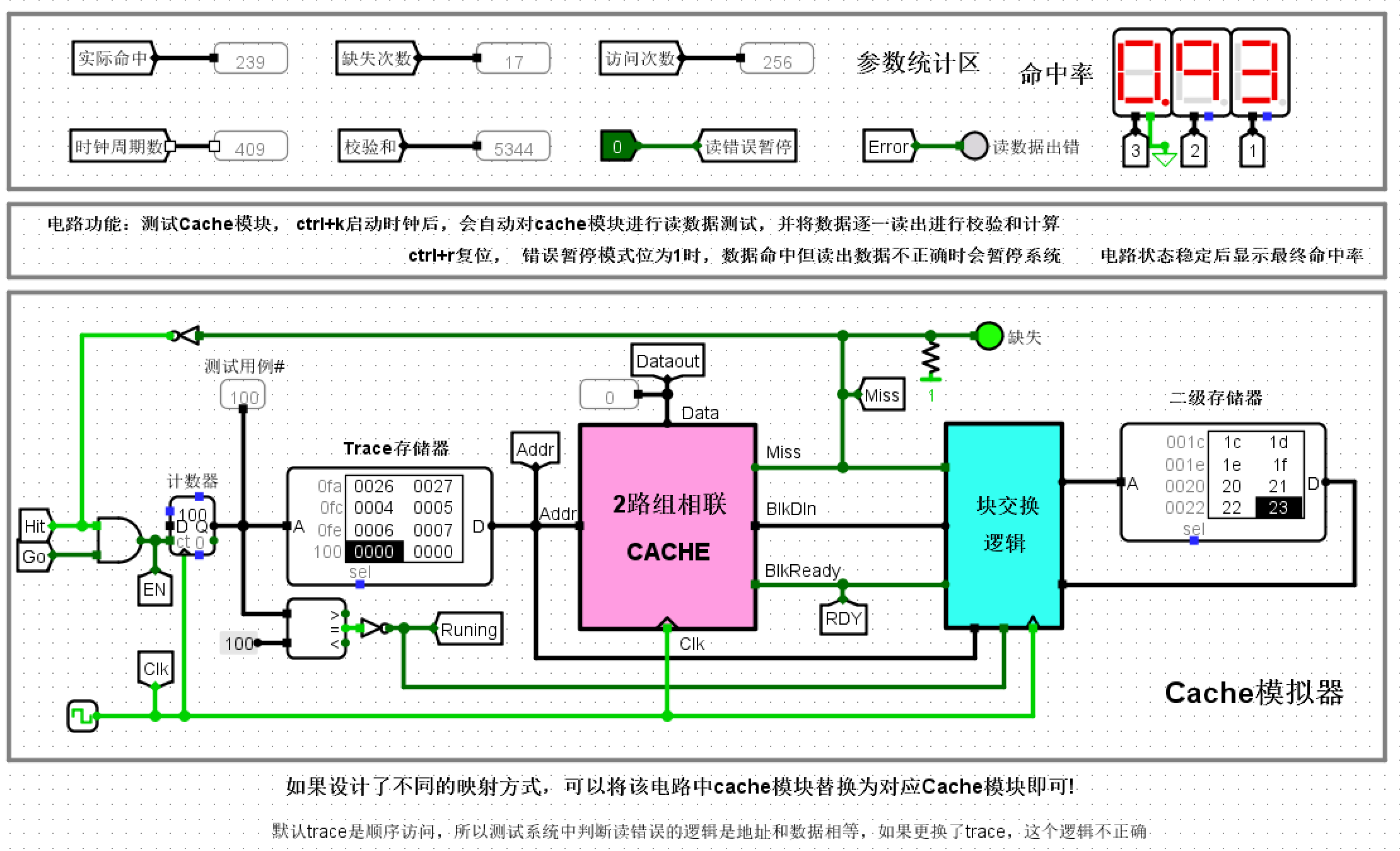
## 6、实验数据测试

1. 头歌平台实验通过截图。



**图3.12二路组相联cache图 头歌平台实验通过截图**

1. 程序自动测试截图。



**图3.12二路组相联cache图 程序自动测试截图**

# 实验四、单周期MIPS CPU设计

## **1、**实验目的

（1）理解CPU工作原理：深入掌握单周期CPU的工作原理和设计方法。

（2）学习指令执行过程：通过设计，理解MIPS指令集的执行过程和控制逻辑。

（3）掌握硬件设计技能：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。

（4）提升问题解决能力：在设计和调试过程中，培养分析和解决问题的能力

（5）实践理论知识：将计算机组成原理的理论知识应用到实际的硬件设计中。

## 2、主要任务

（1）构建一个32位 MIPS CPU 单周期处理器，输出每个周期的PC地址（32bits）、指令IR（32bits）、寄存器文件写入端口RDin（32bits）、内存写入端口MDin（32bits）等。

（2）构建一个单周期硬布线控制器，具有判断R指令与I指令功能，输出控制CPU处理器中多个多路选择器输出的信号。

## 3、实验方案的总体设计

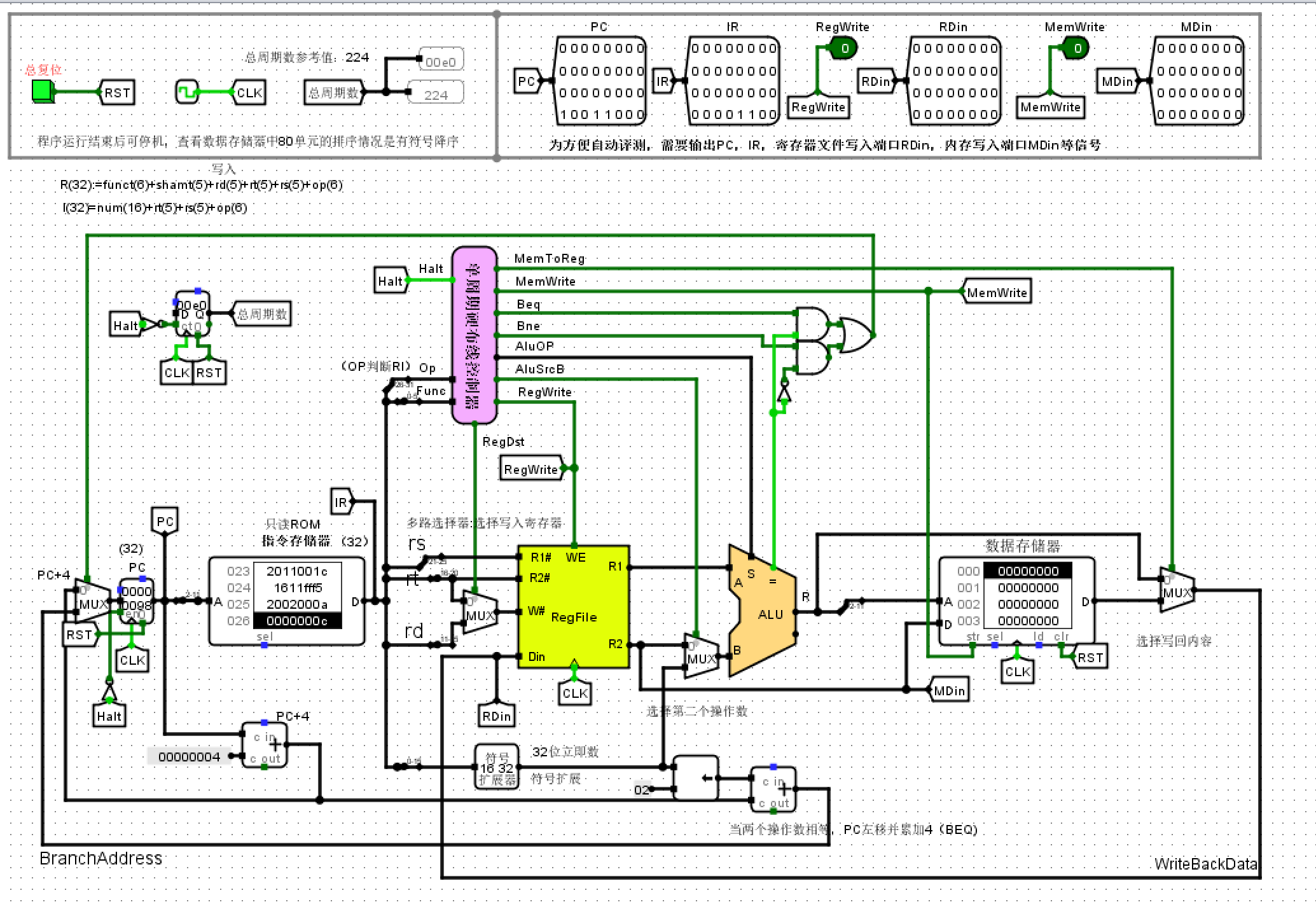
1. 需求分析：PC地址常规更替，每一个周期，PC地址要往后四字节（32bits）来执行下一条指令；指令执行，针对不同指令的要求，对ALU运算器的操作数和具体操作，通过多个多路选择器及硬布线控制器的输出信号对其进行多重选择，使均能实现通路；单周期硬布线控制器，通过OP值与FUNC值确定不同的指令类型，并赋值指令类型所对应的多路选择器的控制信号；计总周期数，通过时钟周期信号和停止信号对周期数进行统计。
2. 实验步骤：设计多路选择器；CPU处理器连线；硬布线控制器中指令译码逻辑；硬布线控制器中ALU控制逻辑；硬布线控制器中输出信号逻辑。

## 4、实验的实现过程

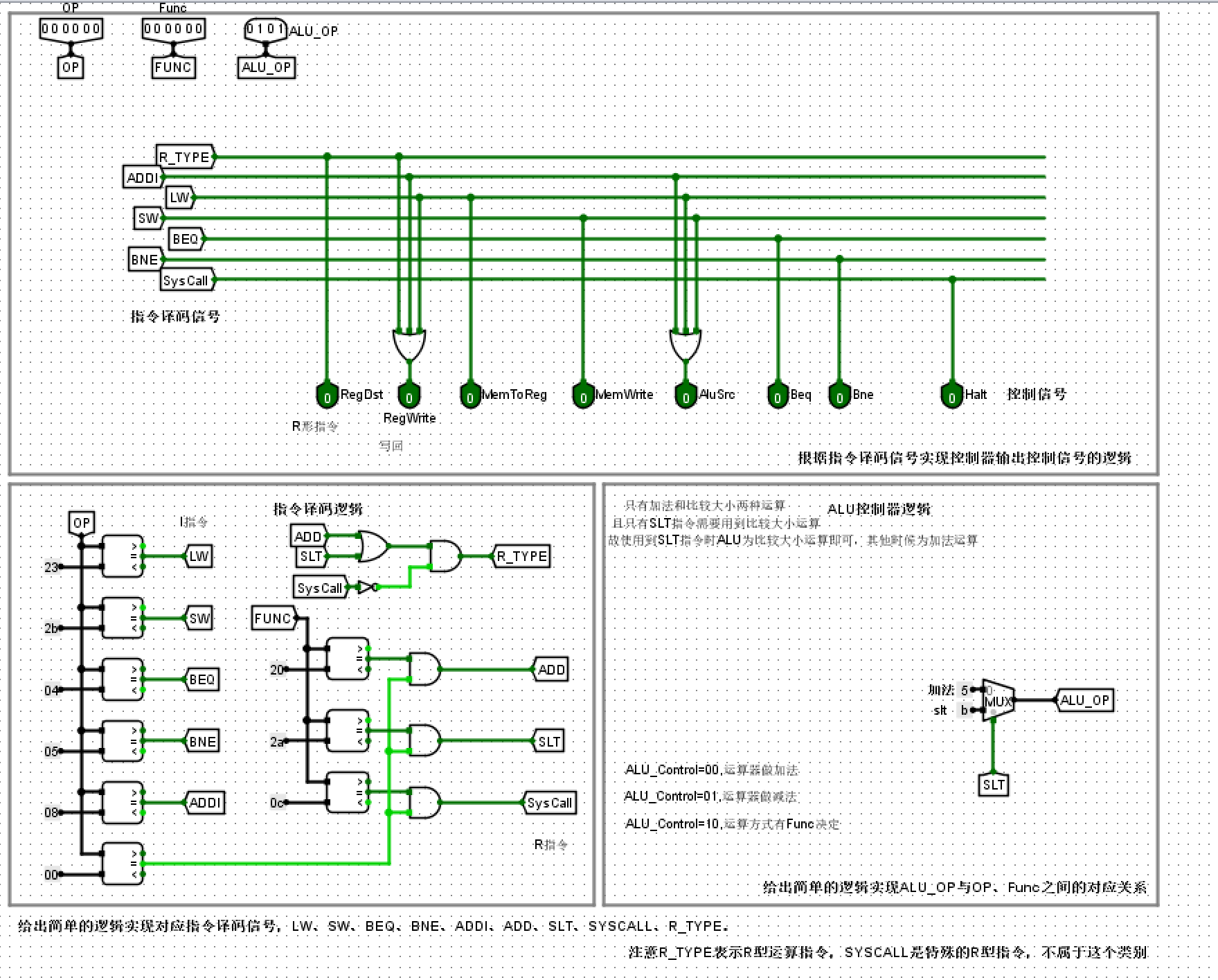
1. 清晰指令：R指令(32)=funct(6)+shamt(5)+rd(5)+rt(5)+rs(5)+op(6)，I指令(32)=num(16)+rt(5)+rs(5)+op(6)；R指令包含add(R[rd]<—R[rs+rt])，slt(R[rd]<—R[rs-rt])；I指令包含lw(R[rt]<—[(R[rs]+num)对应数据存储器数据])， sw(R[rs]+num<—R[rt])，addi(R[rt]<—(R[rs]+num))，beq，bne。
2. 设计多路选择器：因为R指令最终都将写回R[rd]，但I指令不具有rd，I指令在lw和addi指令将写回R[rt]，所以为确定在RegFile的写回地址W#，需要设置一个多路选择器，通过RegDst信号进行选择；因为R指令操作数A是R[rs]，另一个操作数B一定是R[rt]，但I指令操作数A是R[rs]，另一个操作数在lw指令、sw指令及addi指令中为立即数num，所以为确定进入运算器ALU的第二个操作数，需要设置一个多路选择器，通过AluSrcB信号进行选择；因为R指令的写回数据都直接是运算器ALU的输出结果，但I指令中的lw写回数据需要将ALU的输出结果再经过一遍数据存储器，所以为确定写回数据，需要设置一个多路选择器，通过MemToReg信号进行选择；因为I指令中的beq指令与bne指令会对PC指令进行左移操作，不同于PC地址的常规更替，所以为确定下一个PC地址，需要设置一个多路选择器，通过beq与bne成立信号进行选择。
3. 硬布线控制器中指令译码逻辑：通过OP段是否为全0判断是否为R指令，如果不是R指令，则根据与已知的不同指令对应OP进行比较判断具体I指令；如果是R指令，则结合FUNC段与已知的不同指令所对应FUNC进行比较判断具体R指令；根据是否为R指令其中一种且不是结束指令来判断是否为有效R指令。
4. 硬布线控制器中ALU控制逻辑：除了R指令中的slt指令在ALU中做减法运算，其他所有指令都在ALU中做加法运算，所以只用判断是否为slt，是slt则做减法运算，不是slt则做加法运算。
5. 硬布线控制器中输出信号逻辑：通过是否是有效的R指令输出RegDst信号来控制写回地址选择；通过是否是LW指令输出MemToReg信号来控制写回数据选择；通过是否是addi、lw、sw指令其一输出AluSrc信号来控制ALU第二个操作数选择；通过是否是有效的R指令、addi、lw指令其一输出RegWrite信号来控制是否需要寄存器文件进行写回操作；bne指令和beq指令对应其信号；SysCall指令对应halt信号

## 5、电路图及相关说明

（1）整体电路图展示。



**图4.1单周期硬布线MIPS CPU图 总电路图**

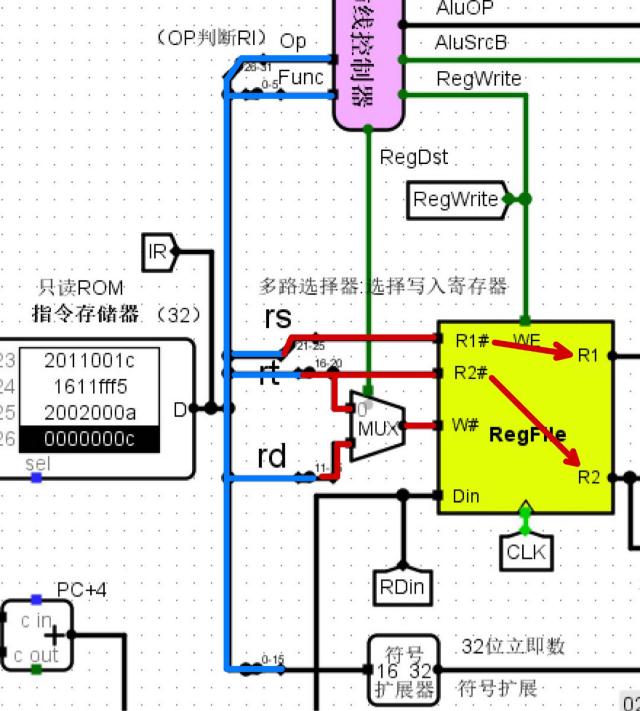


**图4.2单周期硬布线MIPS CPU图 硬布线控制器内部图**

（2）细节图说明。

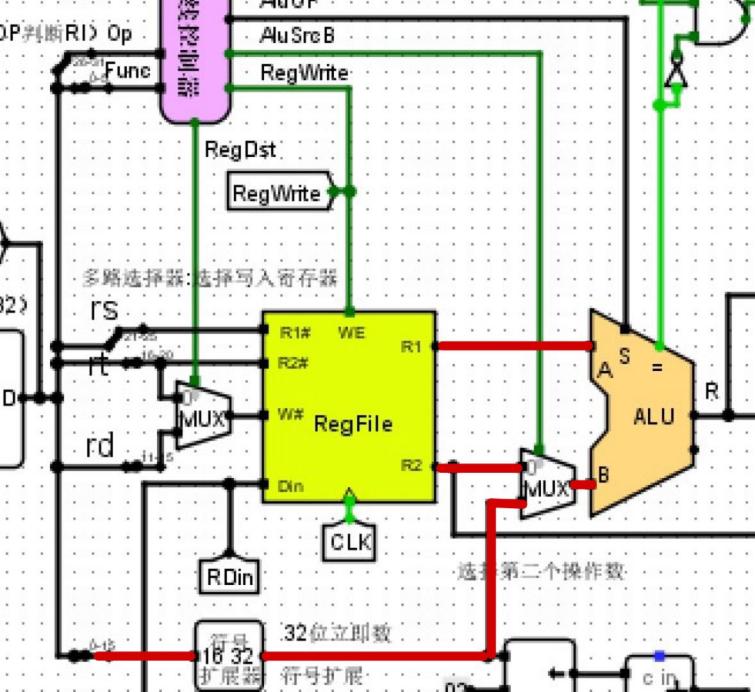
将32位的指令用分线器，第26位到第31位是OP，第21位到25位是rs，第16到20位是rt，顾及R指令和I指令，第0到第15位既要作为立即数num整体作为一段，也要按从第11到第15位是rd和第0到第5位是Func分段，后续根据多重信号保留有效。具体电路如图4.3蓝色线条所示。

rs直接传入R1#，经题目提供的寄存器RegFile电路输出rs对应数据R1，rt直接传入R2#，经题目提供的寄存器RegFile电路输出rt对应数据R2的同时，作为第零行进入多路选择器，rd作为第一行进入多路选择器，根据RegDst信号选择对应的行内容传入W#，当RegDst在硬布线控制器中被传入信号，将选择第一行rd，否则选择地领航rt，作为写回地址保留。具体电路如图4.3红色线条所示。



**图4.3单周期硬布线MIPS CPU图 第一个多路选择器图**

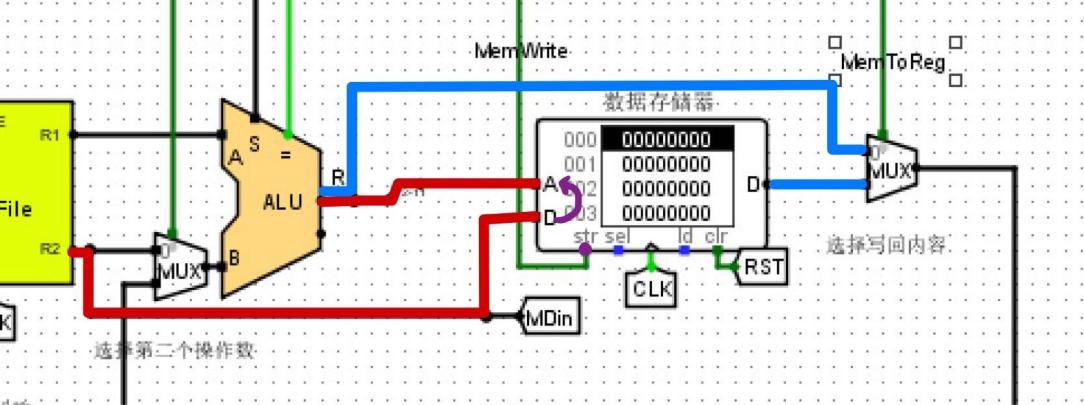
R1直接作为第一个操作数传入运算器ALU的A端入口，R2作为第零行传入多路选择器，立即数num（16bits）为适应32位的ALU及适应32位的CPU需要先通过扩展器将其扩展为32位，再作为第一行传入多路选择器，根据AluSrcB信号选择对应的行作为另一操作数输入运算器ALU的B段入口，当AluSrcB被传入信号时，选择立即数作为操作数，否则选择R2，即rt地址对应文件数据作为操作数。具体电路如图4.4红色线条所示。

****

**图4.4单周期硬布线MIPS CPU图 第二个多路选择器图**

R2，即rt同时作为输入的数据值传入并保留在数据储存器的D端口，运算器ALU的计算结果R作为目标地址传入数据存储器的A端口，由MemWrite信号控制数据存储器的str端，当MemWrite被传入信号时，将D端口的数据存入数据存储器的A端口地址。具体电路如图4.5红色线条所示。

R2同时直接作为第零行传入多路选择器，数据存储器的输出，即A端口地址对应数据作为第一行传入多路选择器，由MemToReg控制，当MemToReg被传入信号时，选择第一行数据输出，否则选择第零行数据输出，输出数据即为写回的数据。具体电路如图4.5蓝色线条所示。

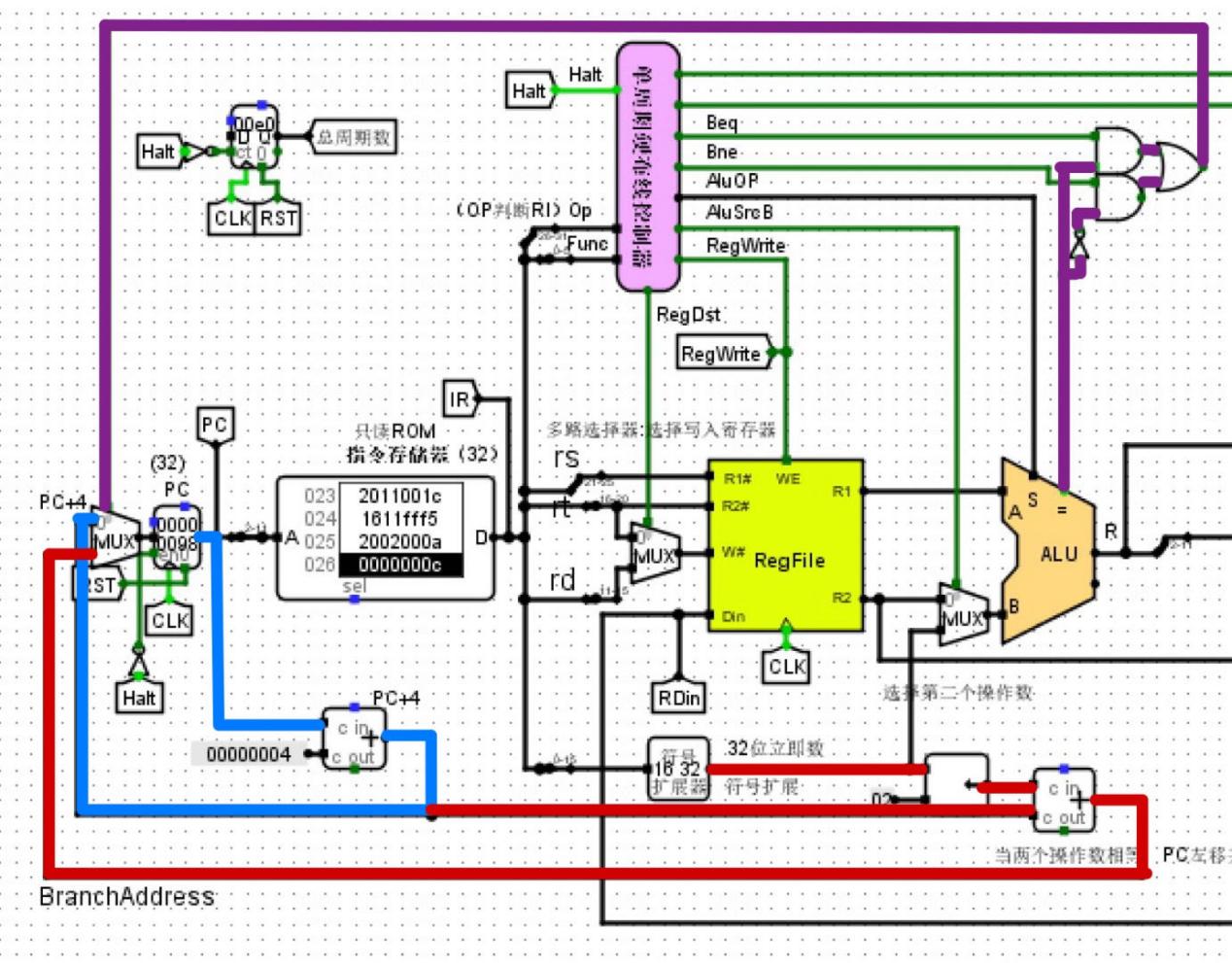
****

**图4.5单周期硬布线MIPS CPU图 第三个多路选择器图**

PC要做常规的迭代，每个时钟周期PC要通过加法器加4，即加四个字节（32bits），跳转到下一条指令，这个结果作为第零行数据传入多路选择器，作为默认选项。具体电路如图4.6蓝色线条所示。

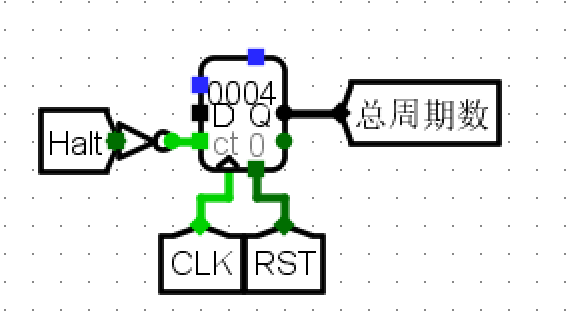
立即数num扩展成32位后传入移位器，将其左移两位后传入加法器加上常规的PC+4，作为新的下一条指令地址，改变读取顺序，这个结果作为第一行数据传入多路选择器。具体电路如图4.6红色线条所示。

当运算器ALU的A端口为rs，B端口为rt，通过二端口数据是否相等，若Beq且AB相等，即Beq被传入信号且ALU比较输出端为1时，或Bne且AB不相等，即Bne被传入信号且ALU比较输出端取非值后为1时，选中该多路选择器的第一行。具体电路如图4.6紫色线条所示。

****

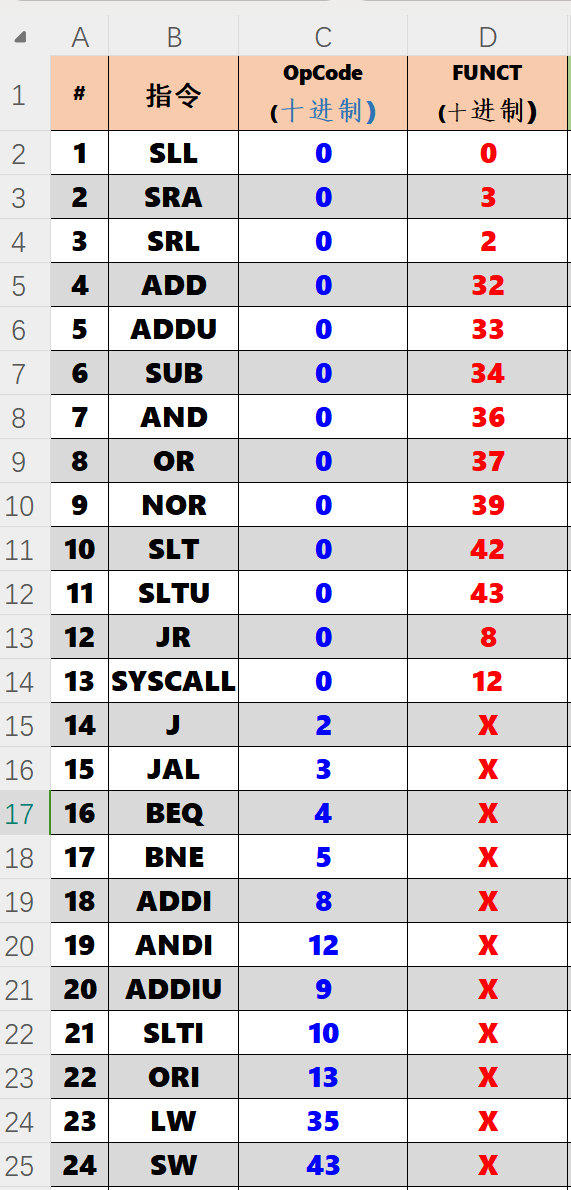
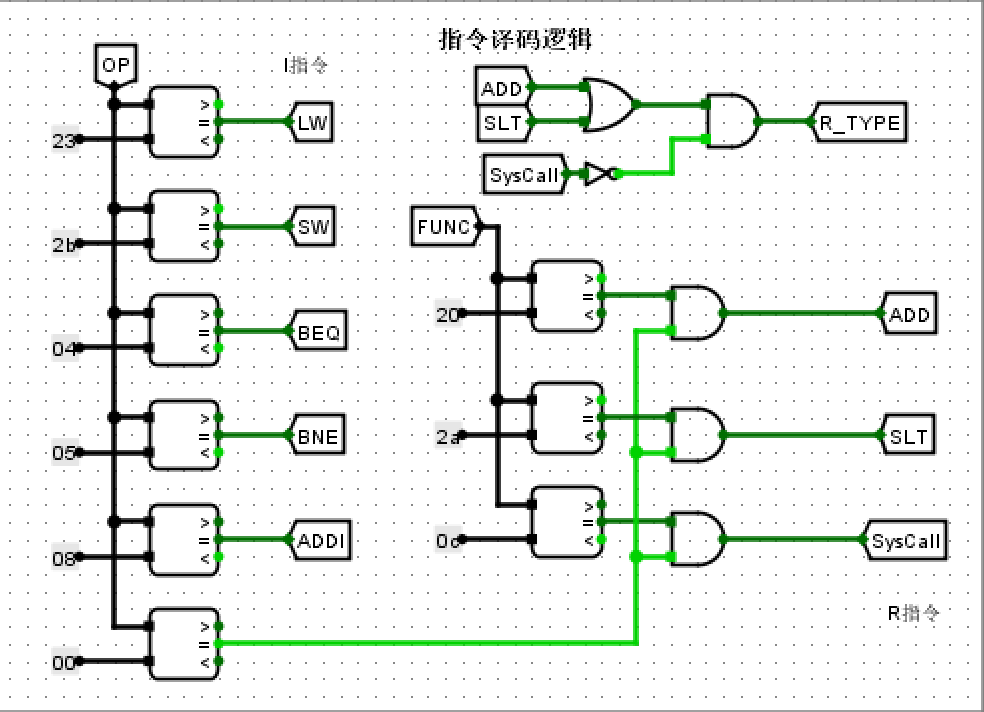
**图4.6单周期硬布线MIPS CPU图 第三个多路选择器图**

通过一个十六位的计数器统计总周期数，当halt未被传入信号时，取非为1，计数器加一，当halt被传入信号时，取非为0，计数器不进行操作，每个周期更新一次总周期数，当RST信号传入时，计数器清零。具体电路如图4.7所示。



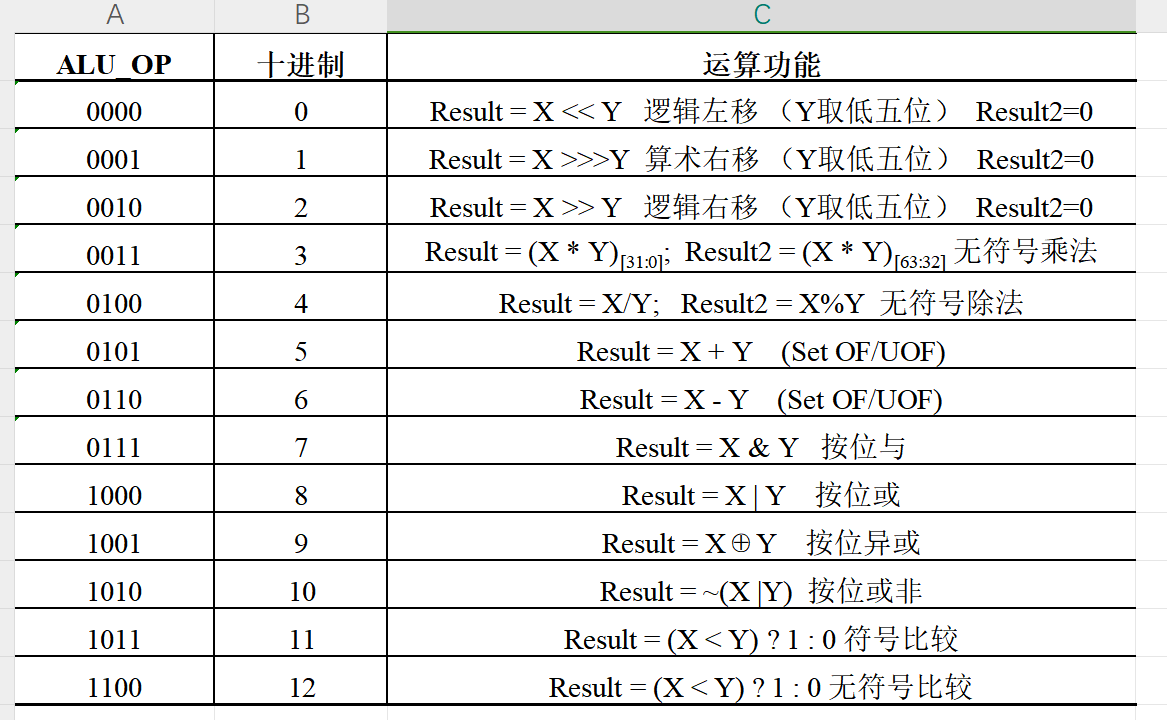
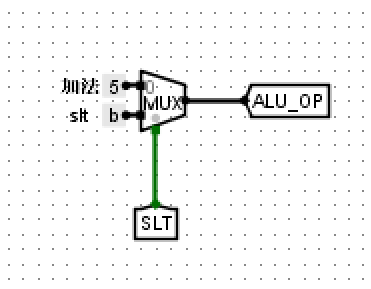
**图4.6单周期硬布线MIPS CPU图 总周期统计图**

指令译码逻辑中，先根据OP字段判断是否为R指令，通过一个比较器，当OP与0相等时，为R指令，否则再并列的根据五个比较器，与已知的不同I指令对应OP字段进行比较，是否相同，判断具体的I指令，信号传至对应隧道，当为R指令时，再通过三个比较器与已知的不同R指令的OP字段1进行比较，是否相同，判断具体的R指令，信号传至对应隧道，最后根据是否为除了syscall的R指令判断是否为有效的R指令并传出信号。具体OP及Func字段查找表如图4.7左所示。具体电路如图4.7右所示。

**图4.7单周期硬布线MIPS CPU图 指令逻辑译码图**

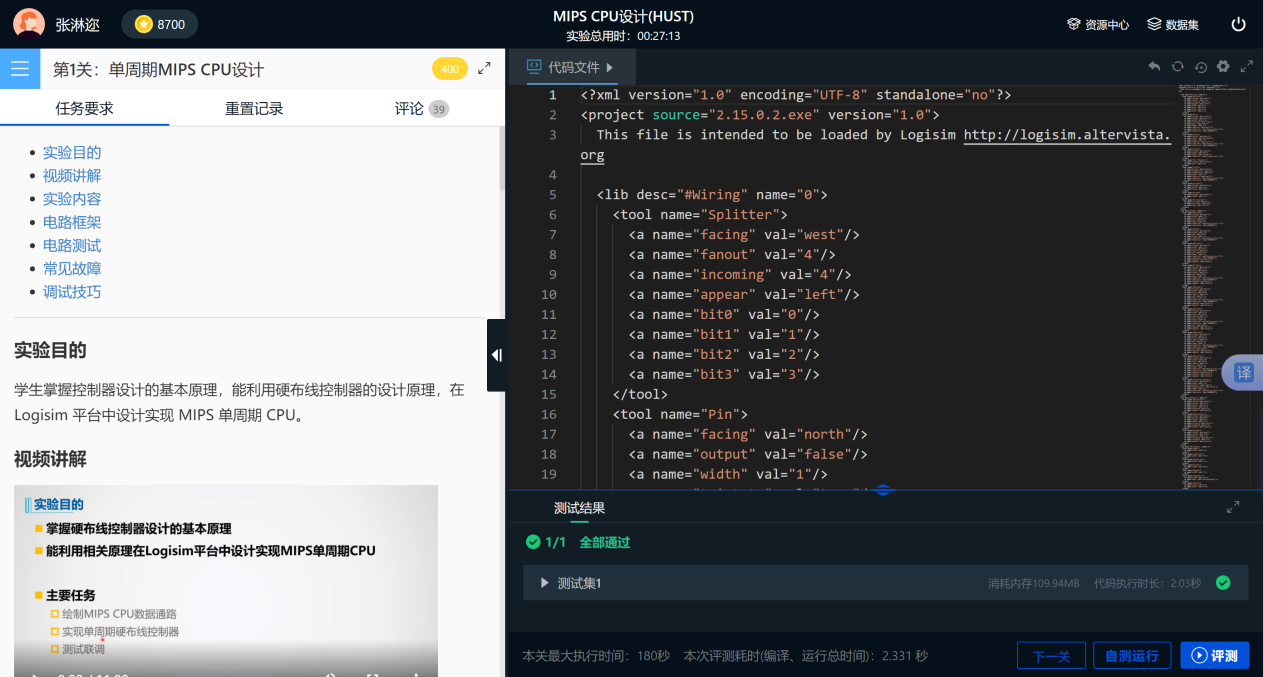
在选择运算器ALU的运算符号时，除了R指令中的slt指令，其他指令都是进行加法运算，所以只需要通过一个多路选择器，将slt设置为选择条件，当slt被传入信号时，选择第一行，已知的ALU减法运算的代码字段，否则为已知的ALU加法运算的代码字段。具体ALU运算符字段查找表如图4.8左所示。具体电路如图4.8右所示。

**图4.8单周期硬布线MIPS CPU图 ALU控制器图**

## 6、实验数据测试

1. 头歌平台实验通过截图。

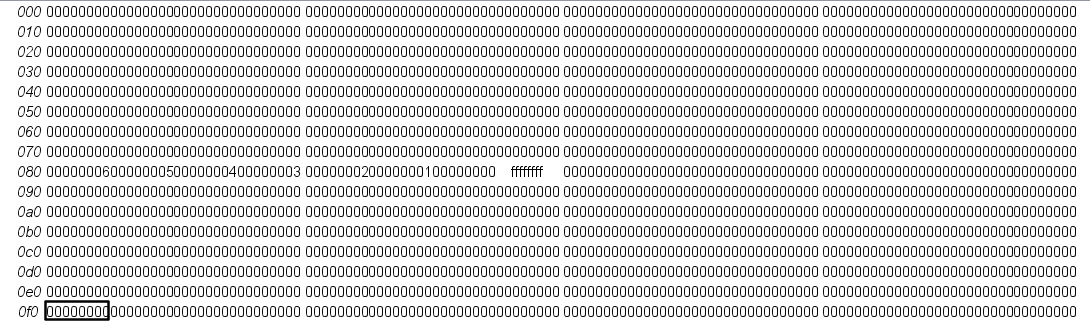


**图4.8单周期硬布线MIPS CPU图 头歌平台实验通过截图**

1. 程序自动测试—sort.hex



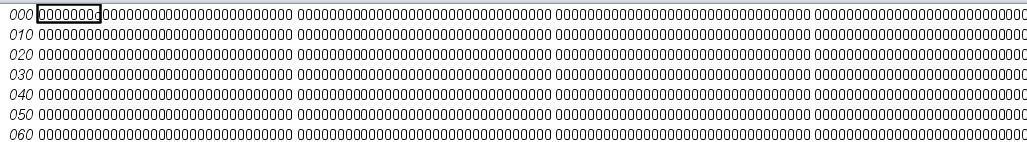
**图4.9单周期硬布线MIPS CPU图 sort.hex指令集**



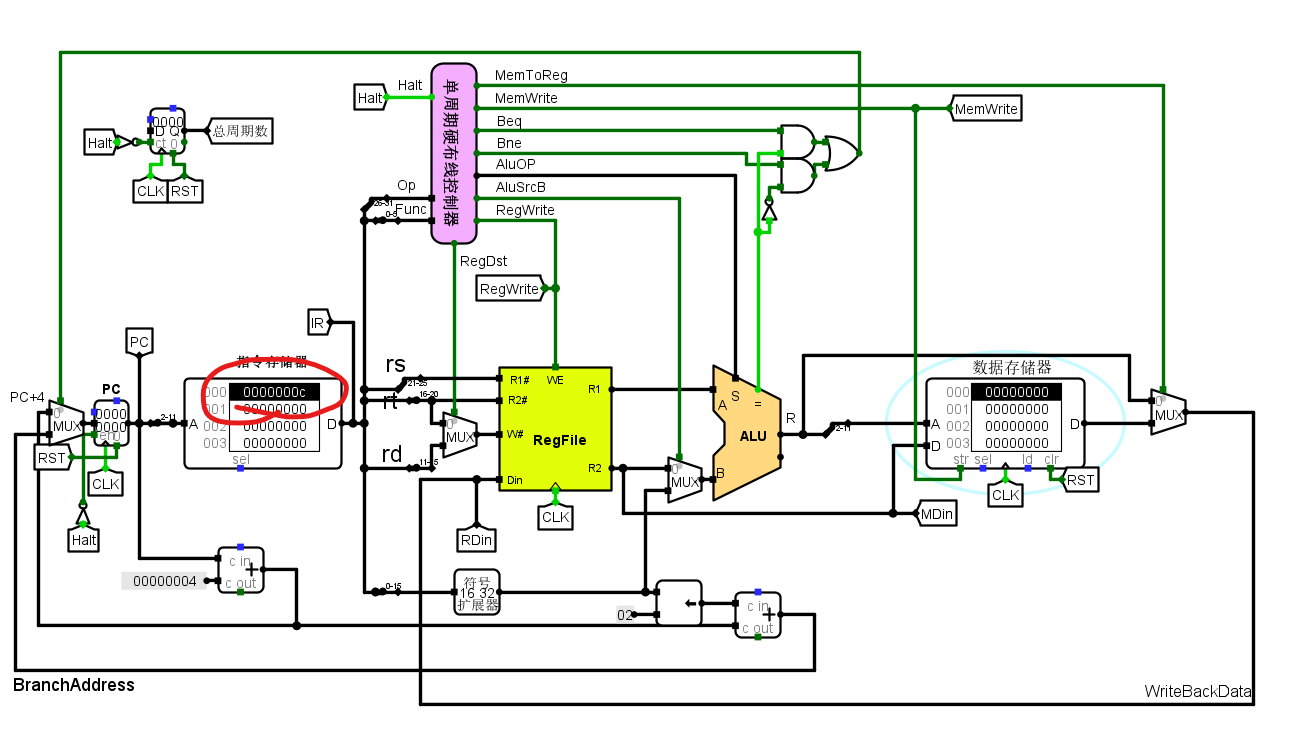
**图4.10单周期硬布线MIPS CPU图 sort.hex执行结果（数据寄存器）**

1. 程序自动测试—syscall\_halt.hex

syscall指令为0000000c，运行后程序终止运行。



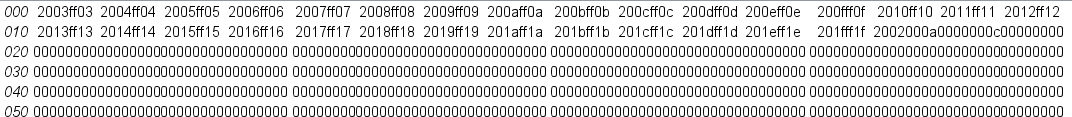
**图4.11单周期硬布线MIPS CPU图 syscall\_halt.hex指令集**



**图4.12单周期硬布线MIPS CPU图 syscall\_halt.hex执行结果（电路图）**

1. 程序自动测试—addi.hex

由已知的OP字段查找表可知addi指令的第26至第31位为20（十六进制），即addi指令从左往右前两位数为20。因为addi(R[rt]<—(R[rs]+num))，所以addi指令执行结果在文件寄存器RegFile中查看。该指令每个周期都会将文件寄存器的R1与立即数相加，并将结果存储在文件寄存器的目标地址中。



**图4.13单周期硬布线MIPS CPU图 addi.hex指令集**

**图4.14单周期硬布线MIPS CPU图 addi.hex第一条指令执行结果（文件寄存器）**



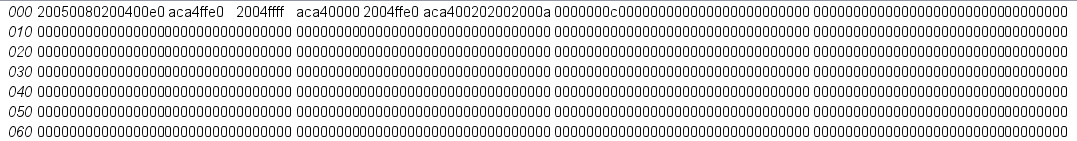
**图4.15单周期硬布线MIPS CPU图 addi.hex全部指令执行结果（电路）**



**图4.16单周期硬布线MIPS CPU图 addi.hex全部指令执行结果（文件寄存器）**

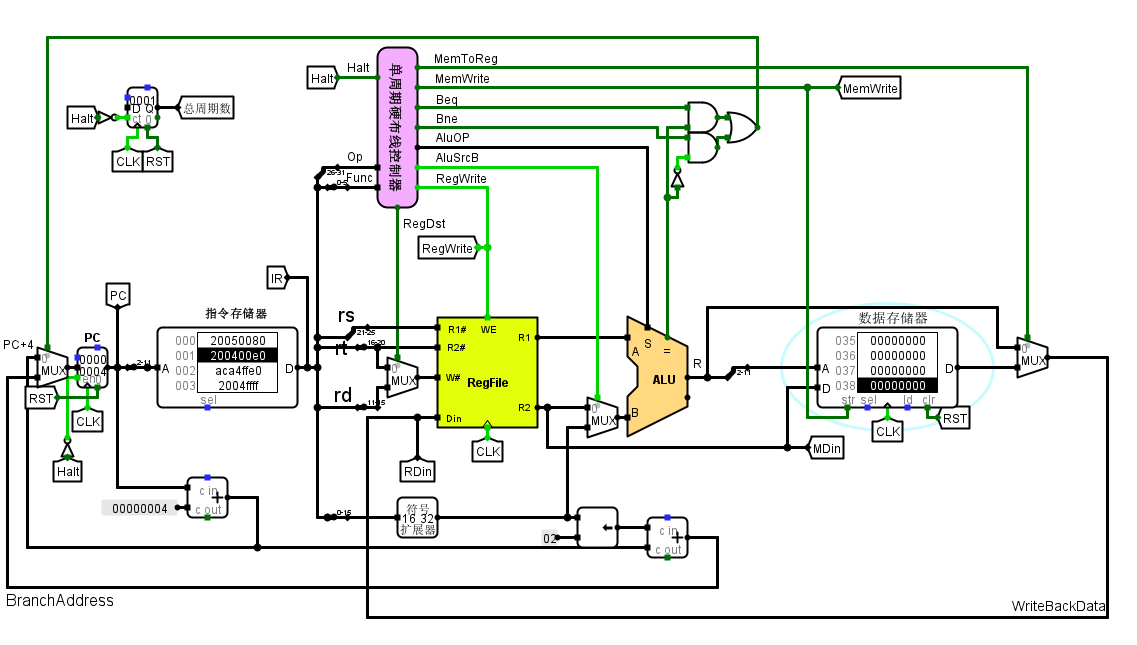
1. 程序自动测试—sw.hex

因为sw(R[rs]+num<—R[rt])，所以lw指令执行结果在数据寄存器中查看。

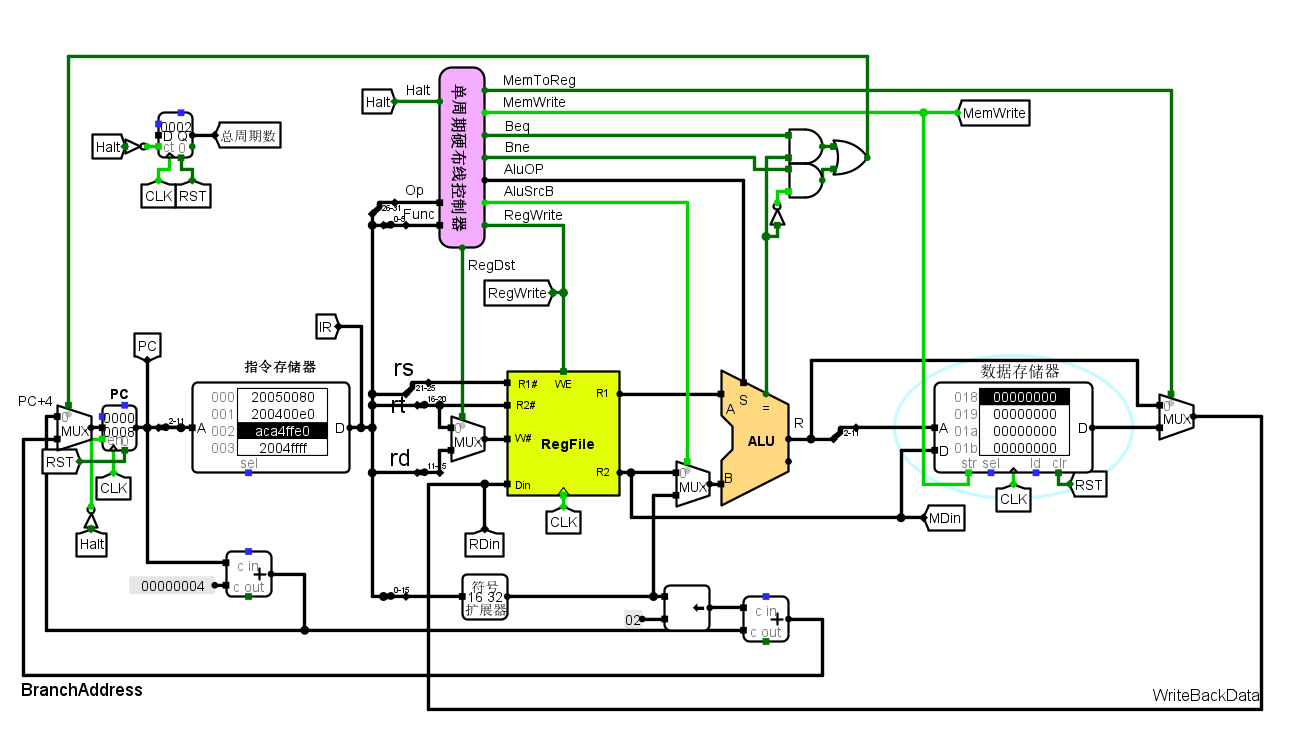


**图4.17单周期硬布线MIPS CPU图 sw.hex指令集**

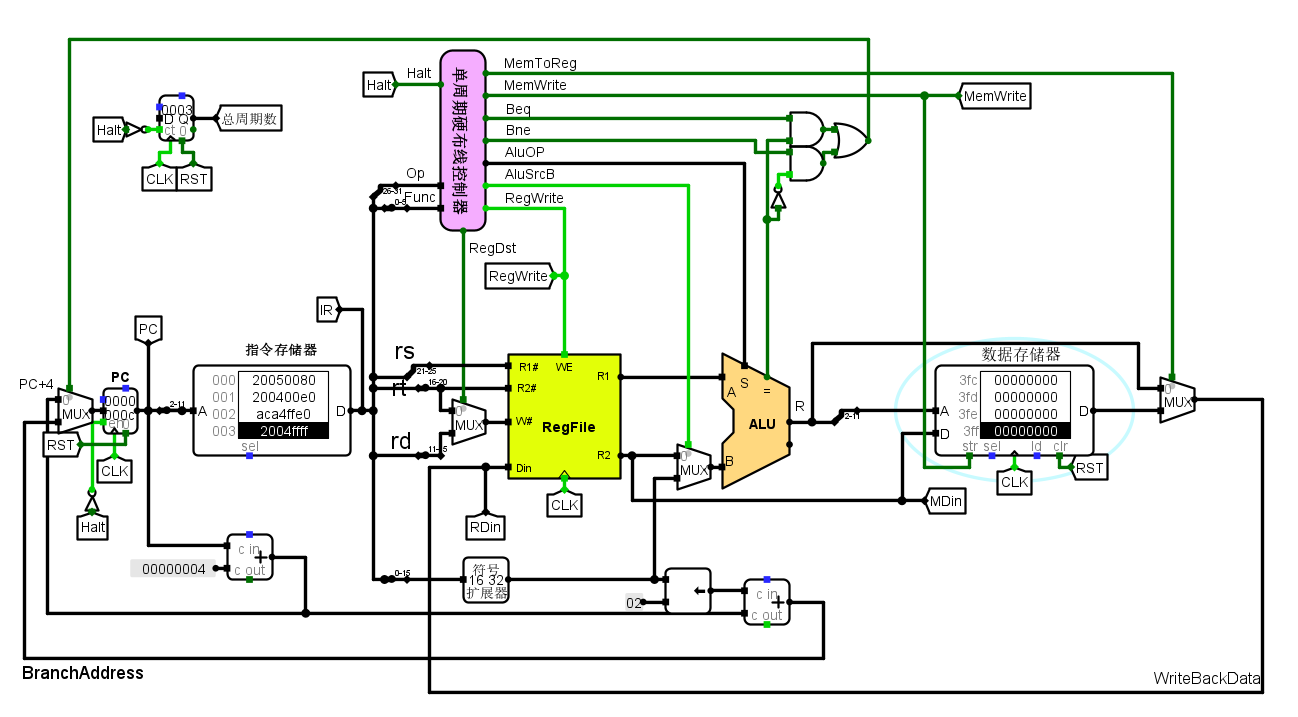
由于当初始文件寄存器数据全部为零，所以R[rt]恒为0，写入数据寄存器的数据始终为0，无法检验是否执行了正确的操作，所以在指令集开头添加两个addi指令，为文件寄存器其中两个地址赋值后，再进行sw指令的调用，所以在第三个周期后数据寄存器中才出现首个结果。



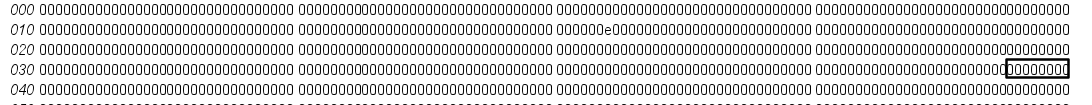
**图4.18单周期硬布线MIPS CPU图 sw.Hex第一个周期执行结果（电路图）**



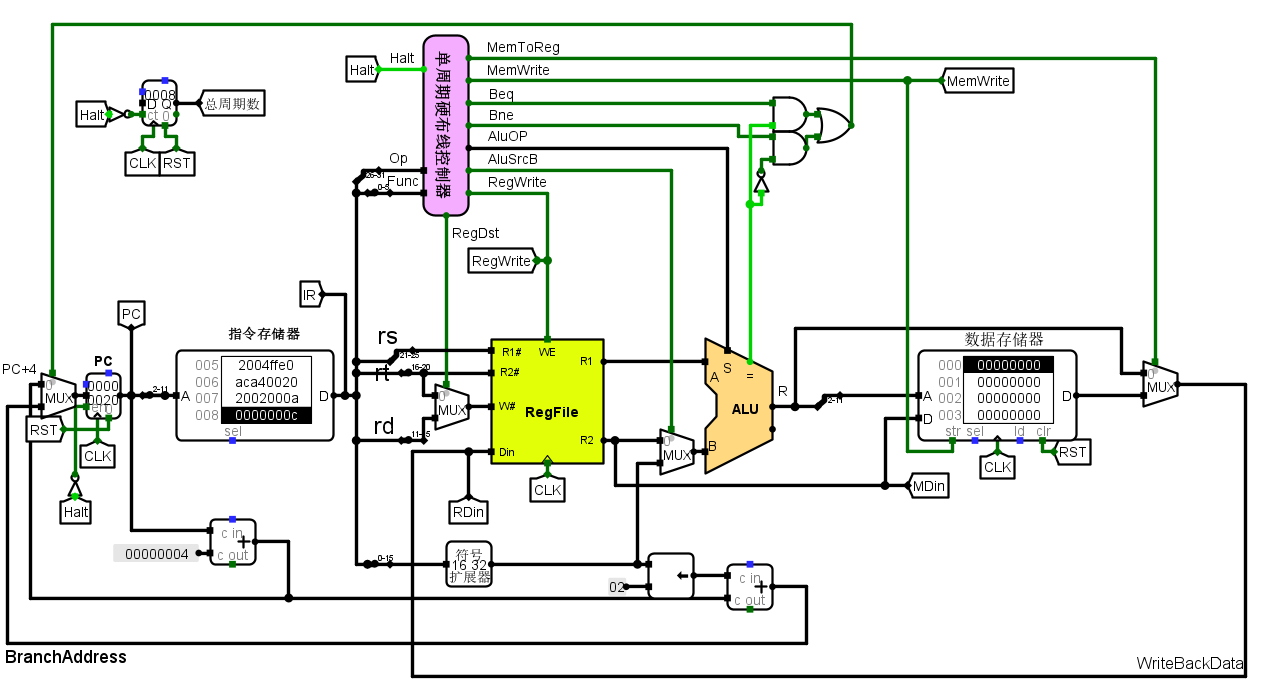
**图4.19单周期硬布线MIPS CPU图 sw.Hex第二个周期执行结果（电路图）**



**图4.20单周期硬布线MIPS CPU图 sw.Hex第三个周期执行结果（电路图）**



**图4.21单周期硬布线MIPS CPU图 sw.Hex第三个周期执行结果（数据寄存器）**



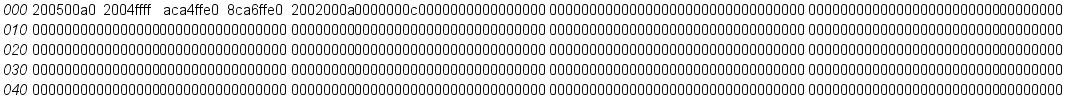
**图4.21单周期硬布线MIPS CPU图 sw.hex全部指令执行结果（电路图）**



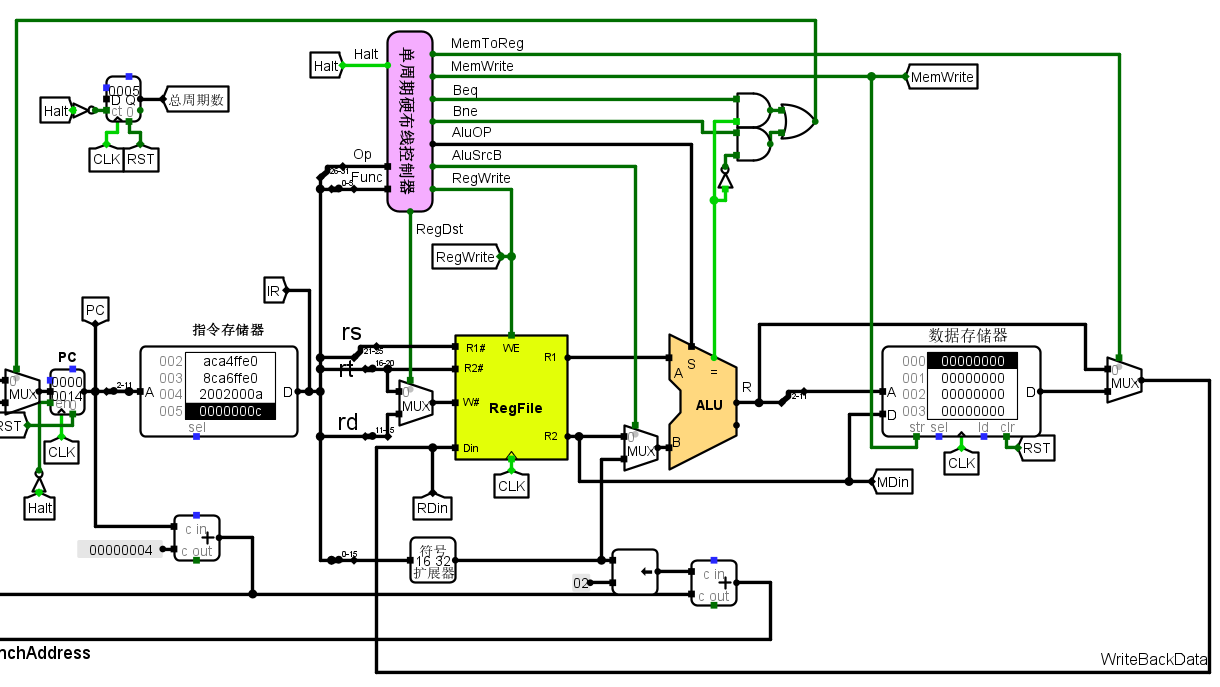
**图4.21单周期硬布线MIPS CPU图 sw.hex全部指令执行结果（数据寄存器）**

1. 程序自动测试—lw.hex

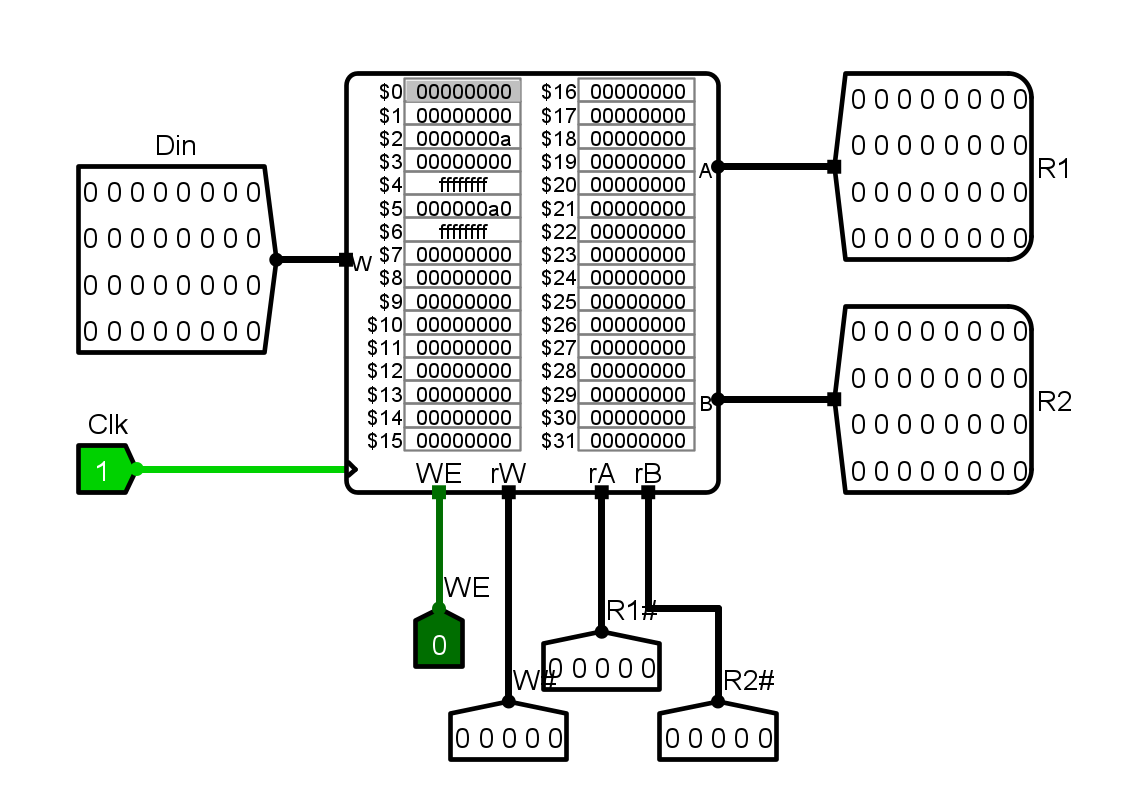
因为lw(R[rt]<—[(R[rs]+num)对应数据存储器数据])，类似sw指令，数据寄存器初始时全部为0，写入文件寄存器也会为0，无法判断是否进行了正确的操作，于是需要先进行addi操作和sw操作为数据寄存器地址赋值，再进行lw操作。



**图4.22单周期硬布线MIPS CPU图 lw.hex指令集**

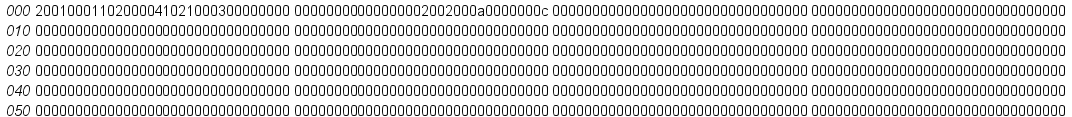


**图4.23单周期硬布线MIPS CPU图 lw.hex全部指令执行结果（电路图）**

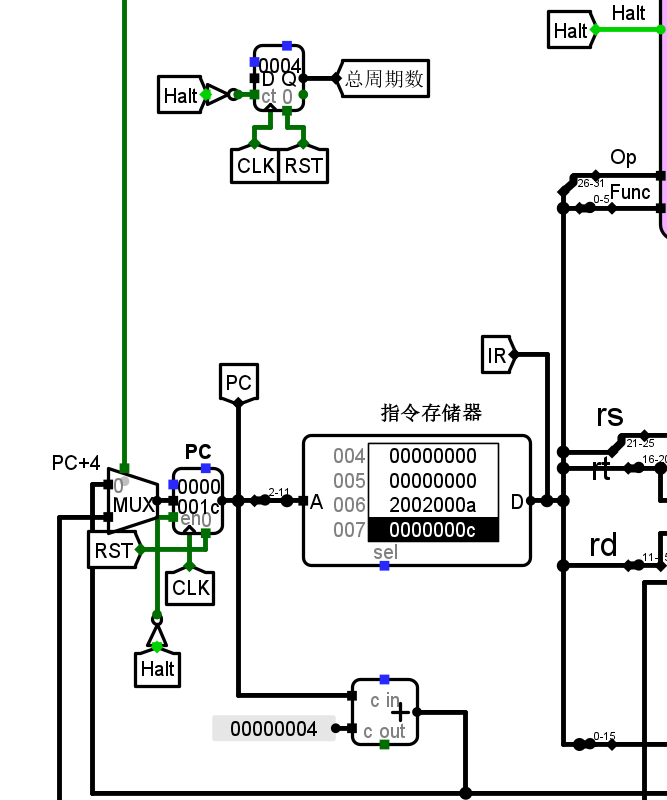


**图4.24单周期硬布线MIPS CPU图 lw.hex全部指令执行结果（文件寄存器）**

1. 程序自动测试—beq.hex

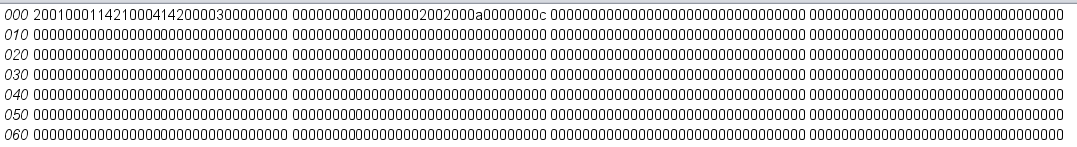


**图4.25单周期硬布线MIPS CPU图 beq.hex指令集**

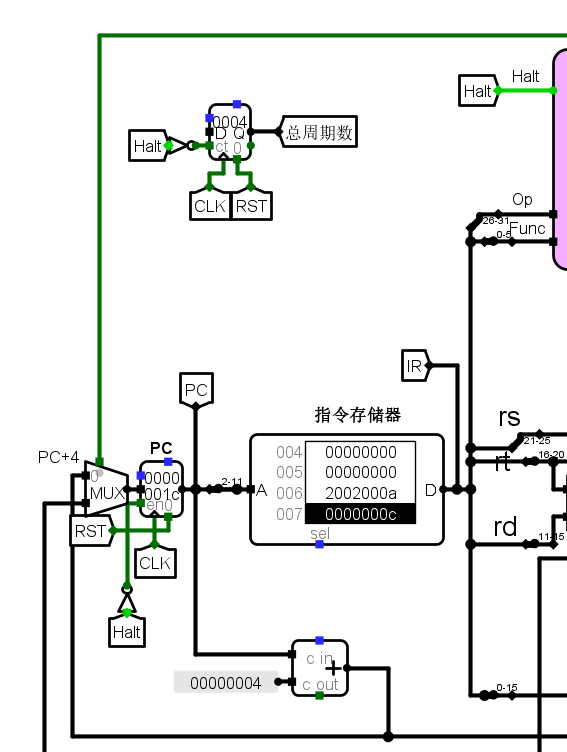


**图4.26单周期硬布线MIPS CPU图 beq.hex全部指令执行结果（PC）**

1. 程序自动测试—bne.hex



**图4.27单周期硬布线MIPS CPU图 bne.hex指令集**



**图4.28单周期硬布线MIPS CPU图 bne.hex全部指令执行结果（PC）**

# 实验五、微程序地址转移逻辑设计

## 1、实验目的

（1）理解微程序控制：深入理解微程序在CPU控制中的作用和实现原理。

（2）设计微程序序列：通过实验在logisim设计微程序序列，实现复杂指令的控制。

（3）培养硬件设计能力：提升硬件设计和逻辑电路设计的能力。

（4）实践编程和调试：通过编程和调试微程序，提高编程技巧和调试能力。

## 2、主要任务

（1）构建一个微程序控制器，输入Opcode、Func 字段、时钟信号、复位信号，输出多路选择器的选择信号、存储器的访问控制信号、寄存器的写使能信号、微指令中的微命令。

（2）涉及微程序地址转移逻辑实现的三个关键技术：指令译码逻辑；ALU控制器逻辑；根据指令进行微程序地址转移。

## 3、实验方案的总体设计

（1）实验步骤：指令译码逻辑；ALU控制器逻辑；根据指令进行微程序地址转移。

## 4、实验的实现过程

（1）指令译码逻辑：通过OP段是否为全0判断是否为R指令，如果不是R指令，则根据与已知的不同指令对应OP进行比较判断具体I指令；如果是R指令，则结合FUNC段与已知的不同指令所对应FUNC进行比较判断具体R指令；根据是否为R指令其中一种且不是结束指令来判断是否为有效R指令。

（2）运算器的控制逻辑：输出微指令中的微命令 ALU\_Control，其值决定运算器的运算选择控制信号 ALU\_OP 的值，第一行为减法运算，第三行与Func字段相关，其他行均为加法运算。

（3）微程序地址转移逻辑设计：利用 Excel 文件得到四位的微程序入口地址的逻辑表达式“微程序地址转移逻辑自动生成 ext(2020-9-1).xlsx”。

（4）根据状态图构建微程序：利用 Excel 文件自动得到控制存储器中存储的微指令“微指令自动生成(2019-4-22).xlsx”。

## 5、电路图及相关说明

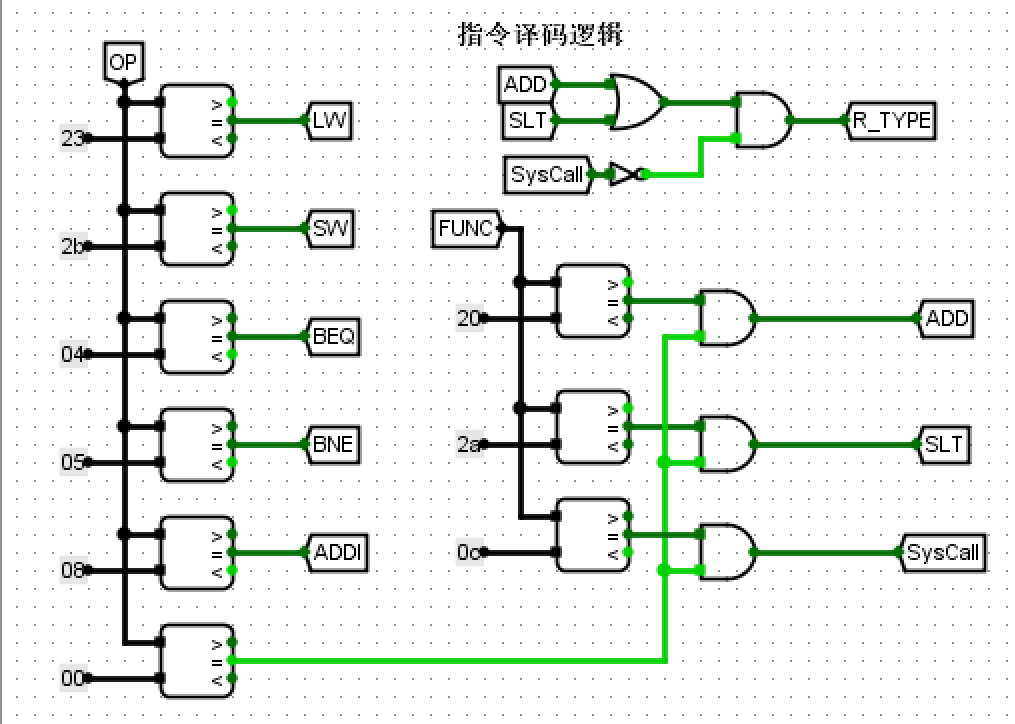
1. 整体电路图展示



**图5.1微程序地址转移逻辑设计 总电路图**

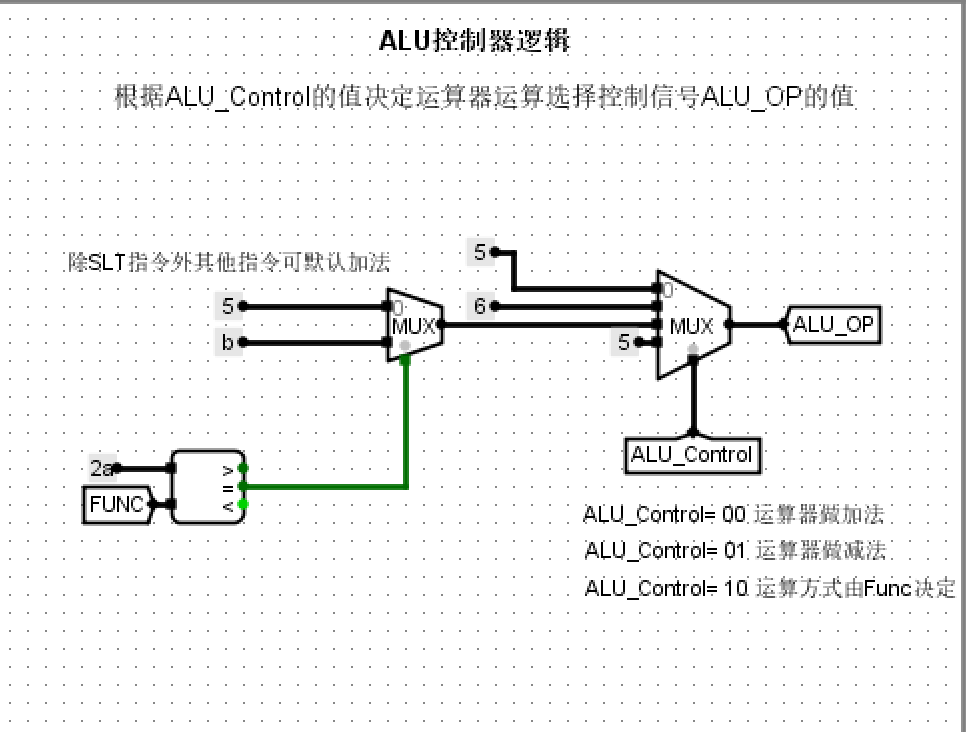
1. 细节图说明

指令译码逻辑中，通过一个比较器，当OP与0相等时，为R指令，否则再并列的根据五个比较器，与已知的不同I指令对应OP字段进行比较，是否相同，判断具体的I指令，信号传至对应隧道，当为R指令时，再通过三个比较器与已知的不同R指令的OP字段1进行比较，是否相同，判断具体的R指令，信号传至对应隧道，最后根据是否为除了syscall的R指令判断是否为有效的R指令并传出信号。具体电路如图5.2所示。



**图5.2微程序地址转移逻辑设计 指令译码图**

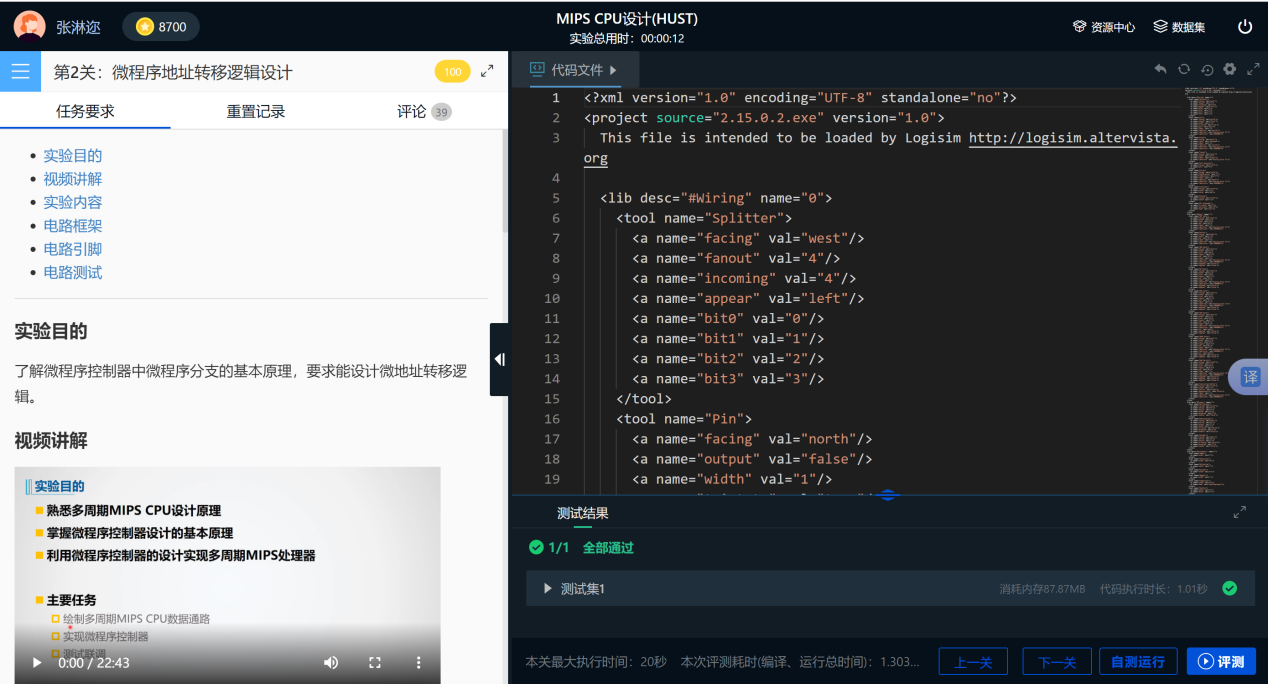
左边的数据首先都要根据已知ALU运算符字段查找表换成ALU特定字段，第零行为加法运算直接为5；第一行为减法运算直接为6；第二行要由Func决定，R指令中除了slt也是加法运算为5，slt指令在此处为比较字符大小运算为b，所以先用一个比较器判断Func是否为slt指令字段，如果是，则选择多路选择器中的第一行b，否则默认情况下选择第零行5；最后第四行为默认的5。具体电路如图5.3所示。



**图5.3微程序地址转移逻辑设计 ALU控制器图**

## 6、实验数据测试

1. 头歌平台实验通过截图。



**图5.4微程序地址转移逻辑设计 头歌平台实验通过截图**

# 实验六、MIPS微程序CPU设计

## 1、实验目的

（1）理解微程序控制：深入理解微程序控制原理，掌握微指令的执行过程。

（2）学习MIPS架构：熟悉MIPS指令集架构，了解其基本指令和操作。

（3）掌握硬件设计：学习如何使用logisim设计和实现CPU。

（4）学习指令执行：理解指令如何在CPU中被执行，包括取指、译码、执行、访存和写回等阶段。

（5）清晰多周期CPU与单周期CPU的区别。

## 2、主要任务

（1）构建一个多周期 MIPS 主机通路，包括存储器、寄存器文件、运算器 ALU 和专用寄存器。

（2）绘制多周期 MIPS CPU 数据通路，确保数据在 CPU 内部正确流动。

## 3、实验方案的总体设计

（1）实验步骤：设计多路选择器；CPU处理器连线。

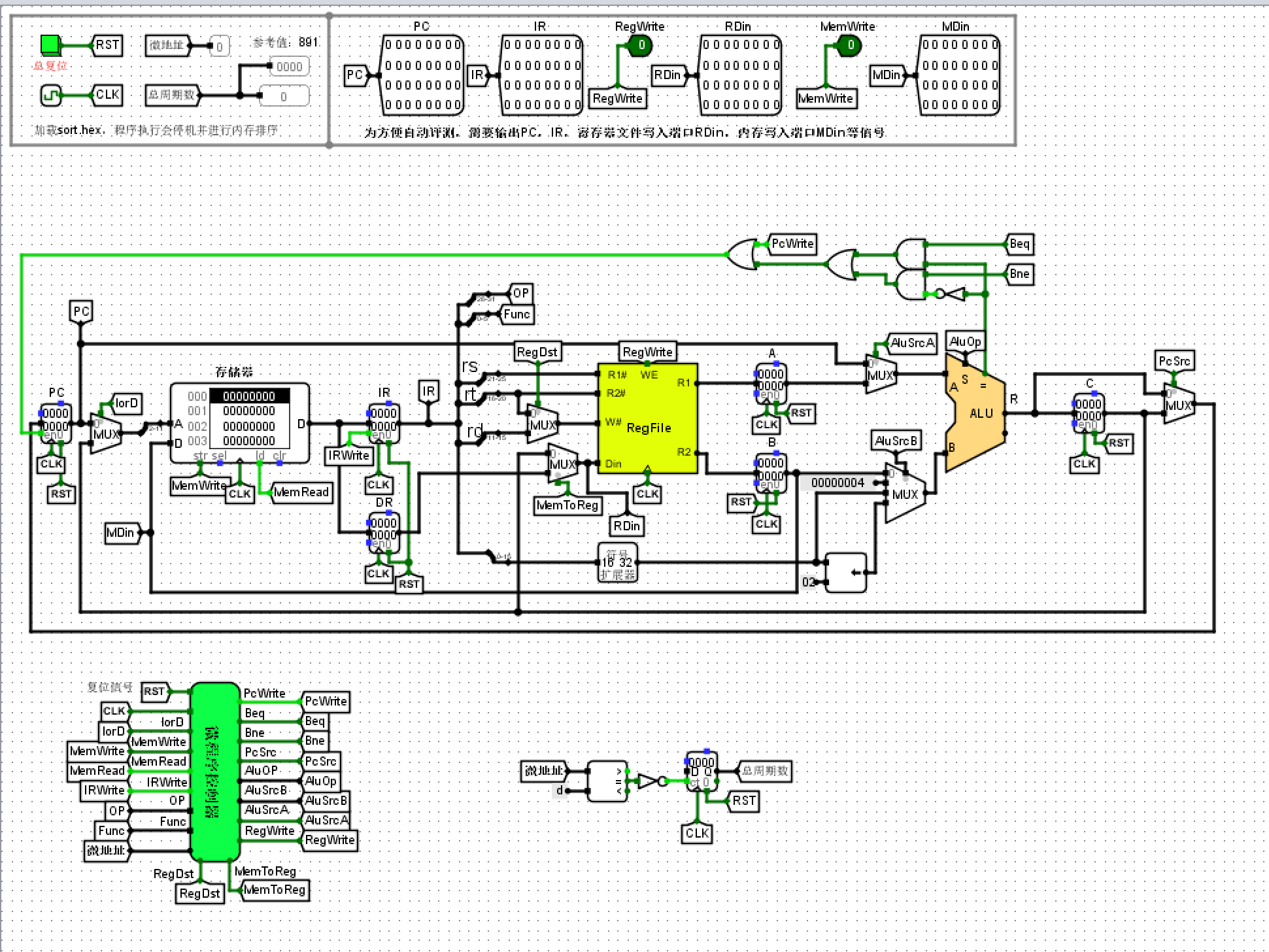
## 4、实验的实现过程

（1）添加专用寄存器（均为 32 位）：PC，IR，DR，A，B，C

（2）设计多路选择器

## 5、电路图及相关说明

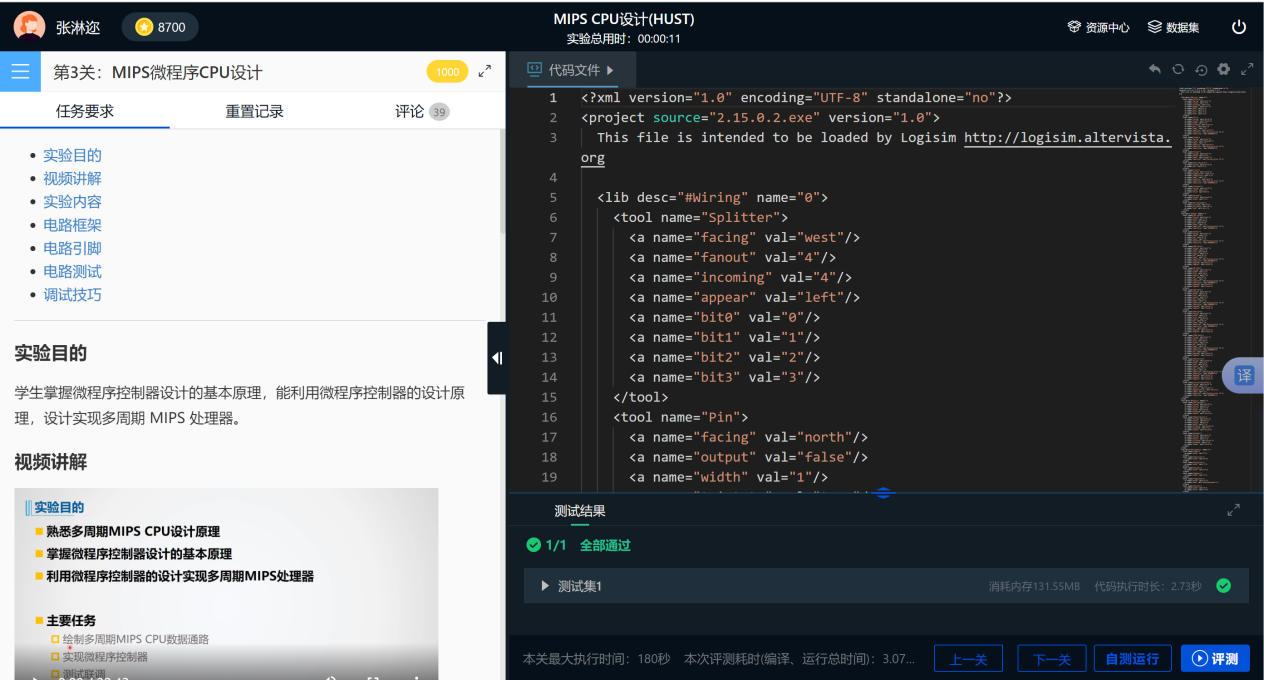
（1）总电路图展示



**图6.1MIPS微程序CPU 总电路图**

## 6、实验数据测试

（1）头歌平台实验通过截图。



**图6.2MIPS微程序CPU 头歌平台实验通过截图**

# 实验心得

不仅仅是看mooc视频和电路连接视频，更多需要自己思考，脱离了视频能否将电路走通，才能真正明白实验每一步的原因。

在自己琢磨电路的过程中，才真正把之前课堂上的理论知识理解清楚，经常有恍然大悟的感觉，连写报告的过程都在不断的发现问题，思考问题，解决问题，学到新的知识。

多周期微指令CPU一块是真的还没有弄太明白，所以写的比较潦草。