

**华中科技大学**

人工智能与自动化学院

**计算机组成原理**

**实验报告**

**专业班级：人工智能2304班**

**学号：U202315265**

**姓名：杜辰宇**

**实验名称：存储器实验**

**实验时间：2025年4月20日**

目录

[1、实验目的 3](#_Toc196409762)

[1.1、存储器扩展实验 3](#_Toc196409763)

[1.1.1、理解存储系统进行位扩展、字扩展的基本原理 3](#_Toc196409764)

[1.2、MIPSRAM设计实验 3](#_Toc196409765)

[1.2.1理解主存储器地址基本概念 3](#_Toc196409766)

[1.2.2理解存储位扩展基本思想 3](#_Toc196409767)

[1.3、MIPS寄存器文件实验 3](#_Toc196409768)

[1.3.1了解MIPS寄存器文件基本概念 3](#_Toc196409769)

[1.3.2熟悉多路选择器、译码器、解复用器等组件 3](#_Toc196409770)

[1.4、Cache硬件设计实验 3](#_Toc196409771)

[1.4.1掌握Cache实现的关键技术 3](#_Toc196409772)

[1.4.2熟悉译码器、多路选择器、寄存器的使用 4](#_Toc196409773)

[2、实验内容 4](#_Toc196409774)

[2.1、汉字字库存储芯片扩展实验 4](#_Toc196409775)

[2.1.1、实验内容： 4](#_Toc196409776)

[2.1.2、实验要求： 4](#_Toc196409777)

[2.1.3、设计原理 4](#_Toc196409778)

[2.1.4、电路测试 5](#_Toc196409779)

[2.2、MIPSRAM设计 5](#_Toc196409780)

[2.2.1、实验内容： 5](#_Toc196409781)

[2.2.2、设计原理 6](#_Toc196409782)

[2.2.3、电路测试 8](#_Toc196409783)

[2.3、MIPS寄存器文件设计 9](#_Toc196409784)

[2.3.1、实验内容： 9](#_Toc196409785)

[2.3.2、原理 9](#_Toc196409786)

[2.3.3、测试 10](#_Toc196409787)

[2.4、全相联映射电路实现 10](#_Toc196409788)

[2.4.1、设计原理 10](#_Toc196409789)

[2.4.2、测试 15](#_Toc196409790)

[2.5、直接相联映射电路实现 16](#_Toc196409791)

[2.5.1、设计原理 16](#_Toc196409792)

[2.5.2、测试 18](#_Toc196409793)

[3、遇到的问题及解决办法 18](#_Toc196409794)

[问题一：存储器扩展电路中的地址线冲突 18](#_Toc196409795)

[问题二：MIPSRAM设计中的数据选择错误 18](#_Toc196409796)

[问题三：Cache设计中的命中率问题 19](#_Toc196409797)

[问题四：直接相联映射中的地址解析错误 19](#_Toc196409798)

1、实验目的

1.1、存储器扩展实验

1.1.1、理解存储系统进行位扩展、字扩展的基本原理

•位扩展（数据总线扩展、字长扩展）

•字扩展（地址总线扩展、字数扩展）

•字位同时扩展（综合扩展）

1.1.2、利用相关原理解决实验中汉字字库的存储扩展问题

•能设计汉字字库存储扩展电路

•能使用正确的字库数据填充

1.2、MIPSRAM设计实验

1.2.1理解主存储器地址基本概念

•内存访问地址都是字节地址

•字节/半字/字访问

lb/sb（load/storebyte）

lh/sh（load/storehalf）

lw/sw（load/storeword）

1.2.2理解存储位扩展基本思想

•能设计同时支持字节、半字、字访问的存储子系统

1.3、MIPS寄存器文件实验

1.3.1了解MIPS寄存器文件基本概念

•寄存器文件（寄存器堆）：通用寄存器的集合

•MIPS32指令集支持32个通用寄存器（32位）

•通过对应寄存器编码访问

1.3.2熟悉多路选择器、译码器、解复用器等组件

•能设计MIPS寄存器文件电路

1.4、Cache硬件设计实验

1.4.1掌握Cache实现的关键技术

•数据查找（如何快速判断数据是否在Cache中）

•地址映射（主存中的数据块应如何放置在Cache中）

•全相联、直接相连、组相联

•替换算法（选择什么样的Cache数据块进行替换或淘汰）

1.4.2熟悉译码器、多路选择器、寄存器的使用

•能设计实现Cache机制

2、实验内容

2.1、汉字字库存储芯片扩展实验

2.1.1、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。

2.1.2、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GB231216\*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图2.1所示：

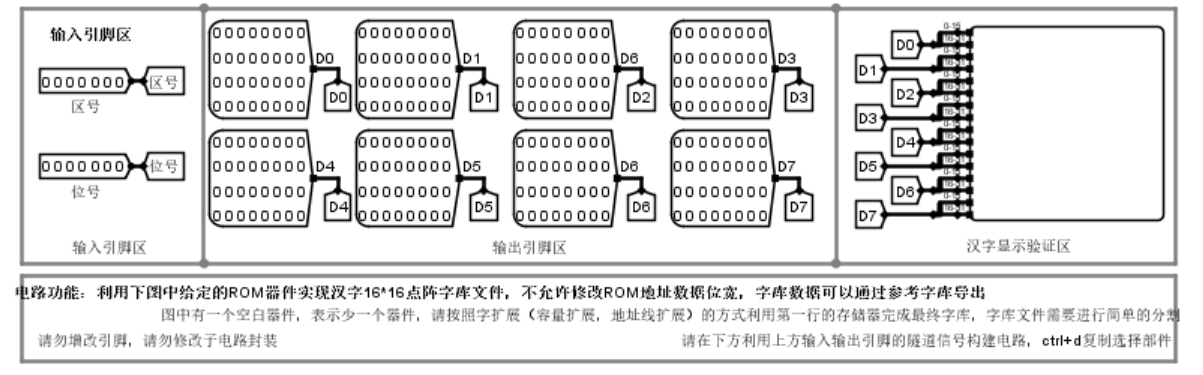


图2.1 输入输出引脚

2.1.3、设计原理

根据实验要求，要将4片4K\*32位ROM通过字扩展扩展成一片16K\*32位ROM。下方电路中汉字的区号和位号的计算得到14位的汉字地址线。将低12位地址通过分线器引出，作为地址将4片8K32位的ROM并联，数据端D1分别与数据线相连。将14位地址线的高两位通过分线器引出连接多路选择器的选择端，四个4K32位ROM的数据端与多路选择器数据端相连，控制D1的输出；同时经过译码器译码，控制片选，进行存储器的选择。电路设计如图2.2所示。

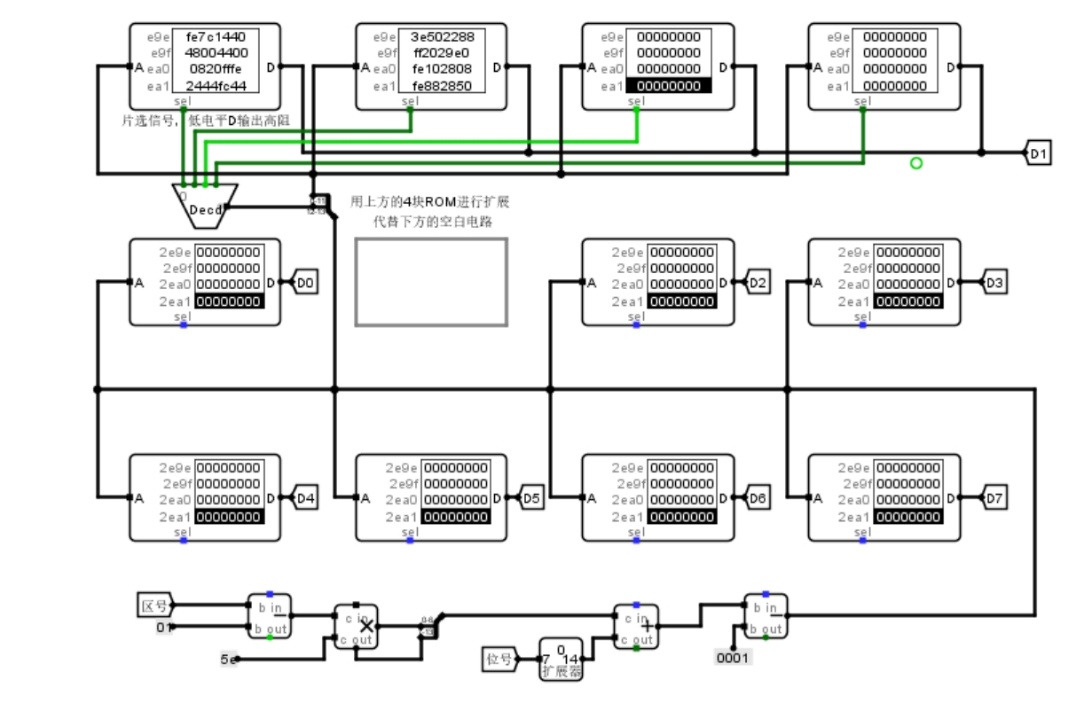


图2.2存储芯片扩展实电路

2.1.4、电路测试

在“字库测试”模块测试电路的正确性，如图2.3所示。

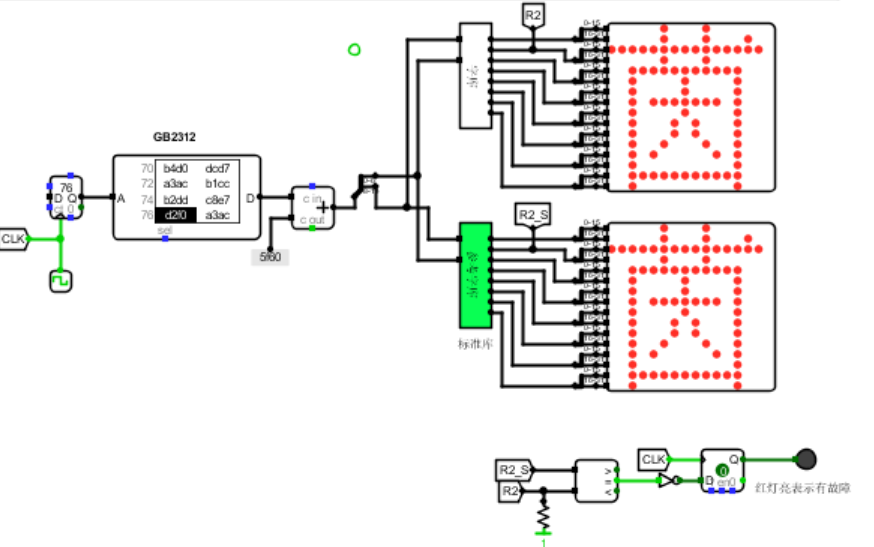


图2.3 字库测试结果

2.2、MIPSRAM设计

2.2.1、实验内容：

计算机中主存储器通常既能按照字节访问也能按照半字访问，还能按照字进行访问，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: 访问模式控制位，2位，00表示字访问，01表示半字访问，10表示字节访问，；

④WE：位宽1位，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）

2.2.2、设计原理

（1）在不同的模式下，根据Mode信合和字节地址，得到四个寄存器的选中信号：

①在字访问模式（Mode=00）下，四个寄存器均被选中；

②在字节访问模式（Mode=01）下，有1个寄存器被选中，结合字节地址的低两位，便可以使得被选中的寄存器对应的选中信号为1；

③在半字访问模式（Mode=10）下，有两个寄存器被选中，结合字节地址的倒数第二位信号，便可以使得被选中的寄存器对应的选中信号为1；

电路图如2.4所示。

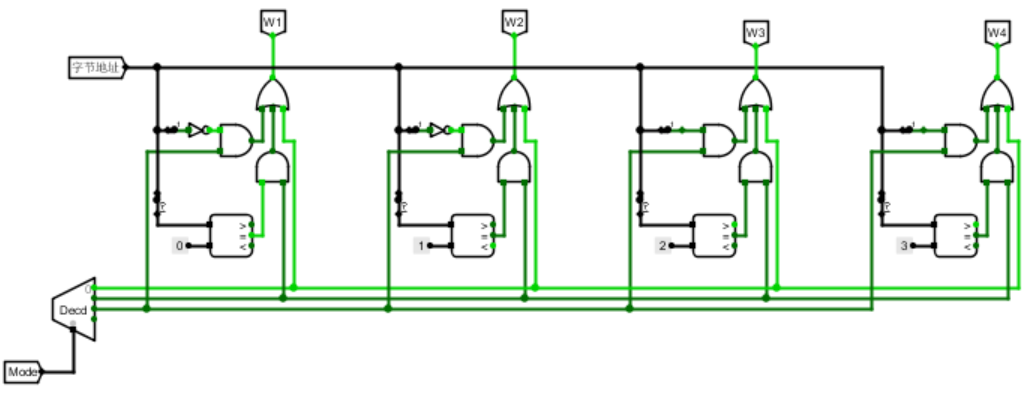


图2.4 存储器选择电路

（2）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；

③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；

电路图如2.5所示。

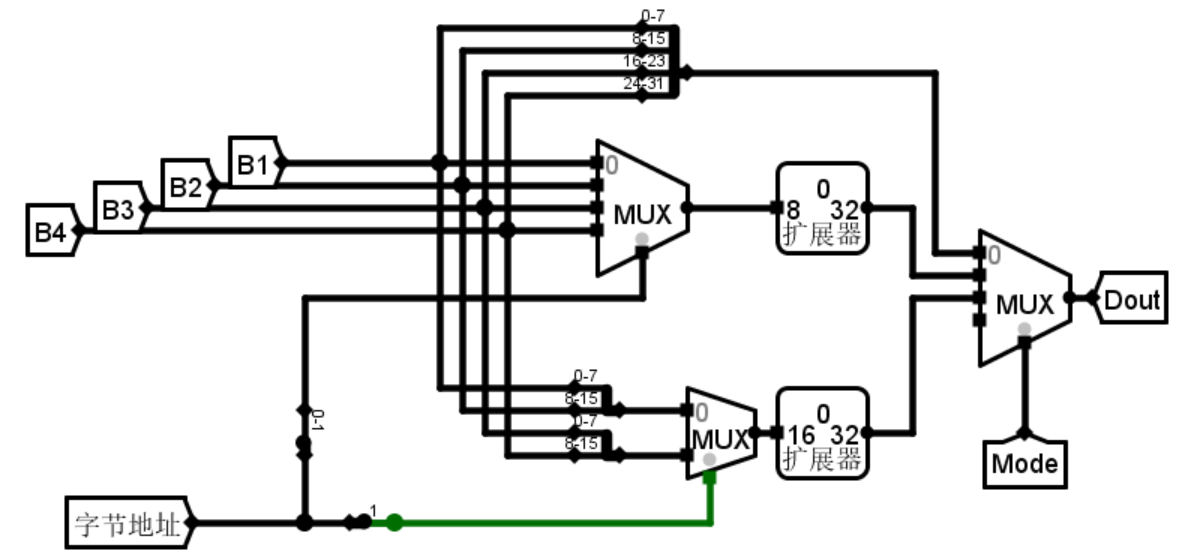


图2.5 读数据选择电路

（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，D1得到输入数据的低八位，D2得到输入数据的8-15位，D3得到输入数据的16-23位，D4得到输入数据的高八位；

②在字节访问模式下，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，D4得到输入数据的低八位；

③在半字访问模式下，D1得到输入数据的低八位，D2得到输入数据的8-15位，D3得到输入数据的低八位，D4得到输入数据的8-15位；

电路图如2.6所示。

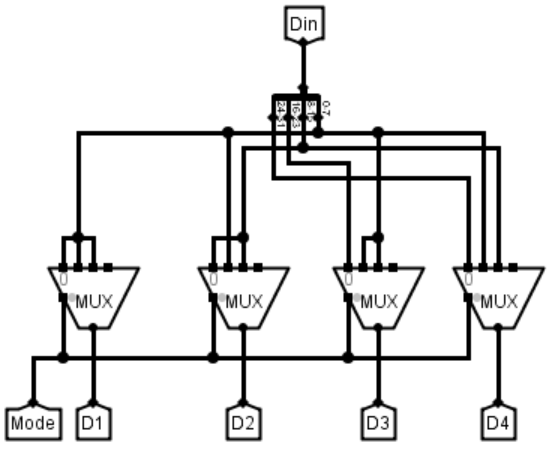


图2.6 写入数据选择电路

（5）存储器电路

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。

电路图如2.7所示。

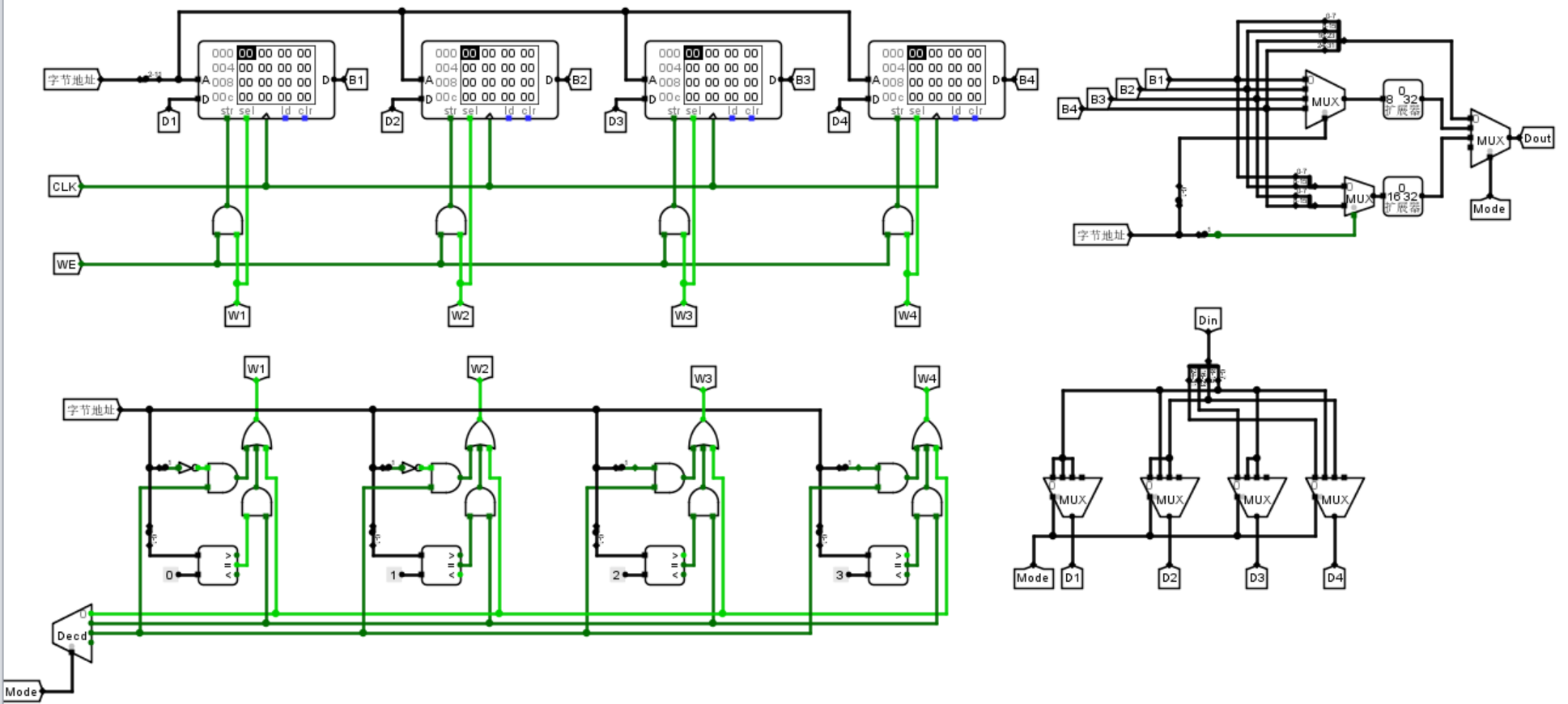


图2.7 MIPSRAM总电路

2.2.3、电路测试

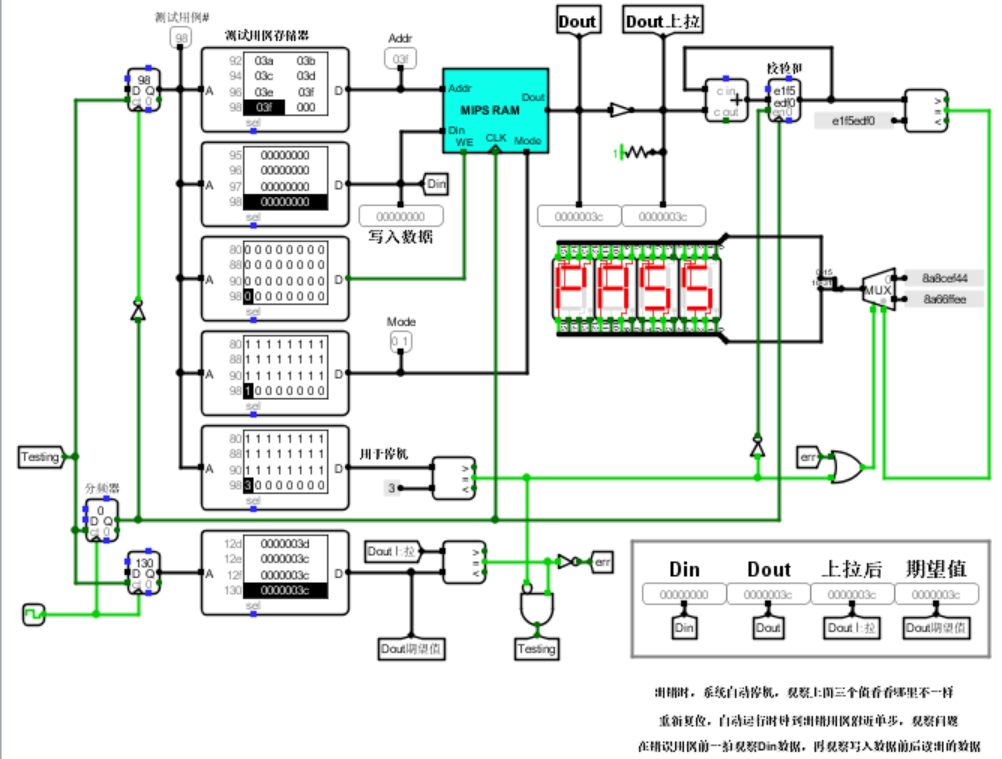


图2.8 MIPSRAM电路测试结果

2.3、MIPS寄存器文件设计

2.3.1、实验内容：

设计完成满足如下规格要求的MIPS通用寄存器组。

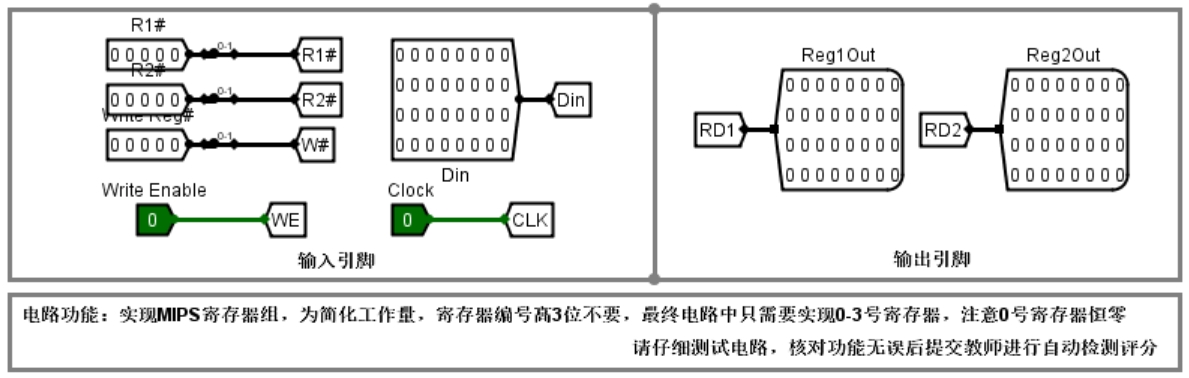


图2.8 MIPS寄存器实验内容

2.3.2、原理

通过一个解码器解码选择对应的寄存器存入数据，解码器输出的值还需要与WE信号相与，当与的结果=1，把Din数据存入寄存器中。然后用两个数据选择器控制输出。电路如图2.9所示。

图示

描述已自动生成

图2.9 MIPS寄存器设计电路

2.3.3、测试

图示

描述已自动生成

图2.10 MIPS寄存器测试结果

2.4、全相联映射电路实现

2.4.1、设计原理

（1）根据32位的BlockDataIn可以得到块内数据为32位，根据8位的Dataout可以得到数据位8位，计算的到offset为2位。故将16位的字节地址分出块内偏移量偏移offset、区地址tag。如图2.11所示。

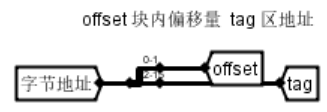


图2.11 字节地址划分

（2）Cache行封装如图2.11所示。

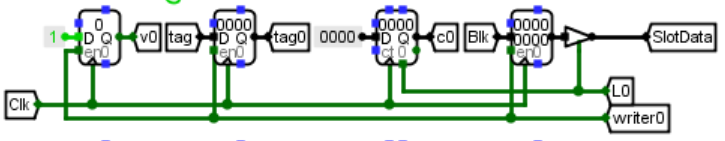


图2.12 Cache行电路

1. 其中第一个为1位寄存器，用来标记这个cache行是否有效(是否存有数据)；
2. 第二个为14位寄存器，存储字节地址中的区号（tag）；
3. 由于采用淘汰算法是近期最少使用（Least Recently Used，LRU），需要一个计数器，记录数据未被使用的次数（C0），用来看谁要被替换；
4. 第四个为32位寄存器，存储块内数据(SlotData)；
5. Writer为写入控制信号，为0时，将区号，块数据输入（BLK）写入，并

将有效位（V0）置1；

1. L为输出信号，为1时，将计数器的计数（C0）清零，并且控制三态门将块内数据输出。

（3）得到读信号：

直接比较字节地址中的tag与各cache行中存储的tagi，为防止寄存器中无数据，加入1与Vi的比较。电路设计如图2.13所示。

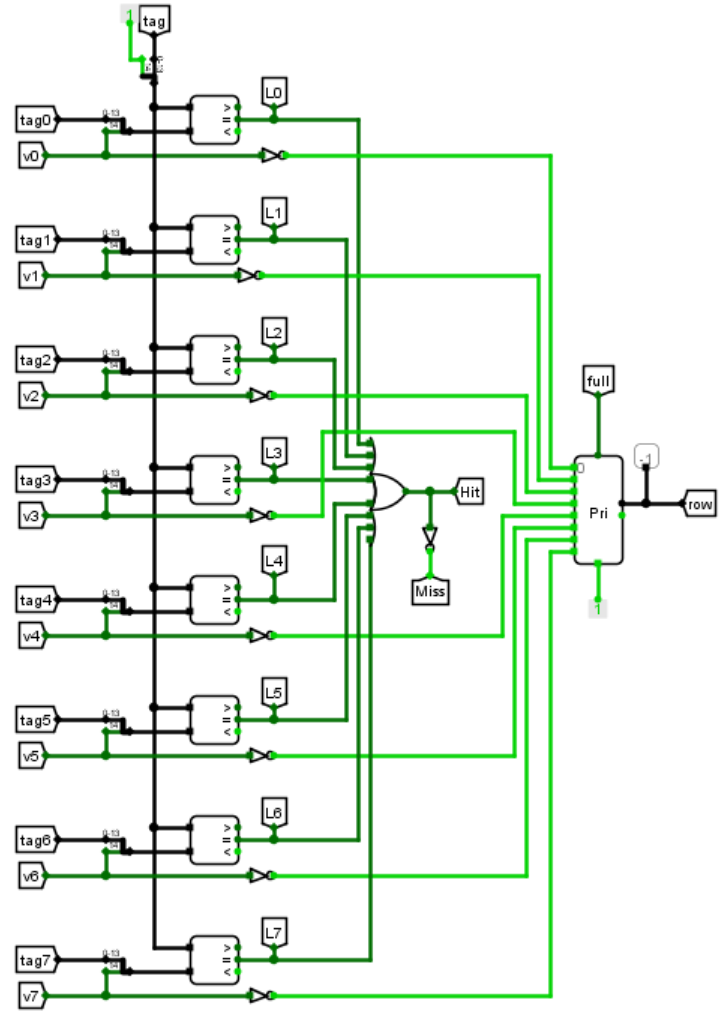


图2.13 读信号L，Miss，row获得电路

（4）得到miss信号：当没有任何tag相等时，Miss信号为1，即未命中时，反之，有一个相等miss信号为0，表明命中。如图2.13所示。

（5）得到为空cache行编号row：采用一个优先编码器（Pri），取反各个cache行的有效位Vi后输入。若Vi均为1，则输出full为1，表示cache已满；若有Vi为0，则row会输出位置最大的为0的Vi的位置，即最大空cache的行数。

（6）数据淘汰算法采取近期最少使用算法，需要比较各个cache行未被使用的次数C，运用3位归并比较器，输入进行比较的两个cache行的C以及行编号，输出较大的C以及行编号，进行逐步比较，最终得到未被使用次数最大被替换的cache行编号out。电路图2.1所示。

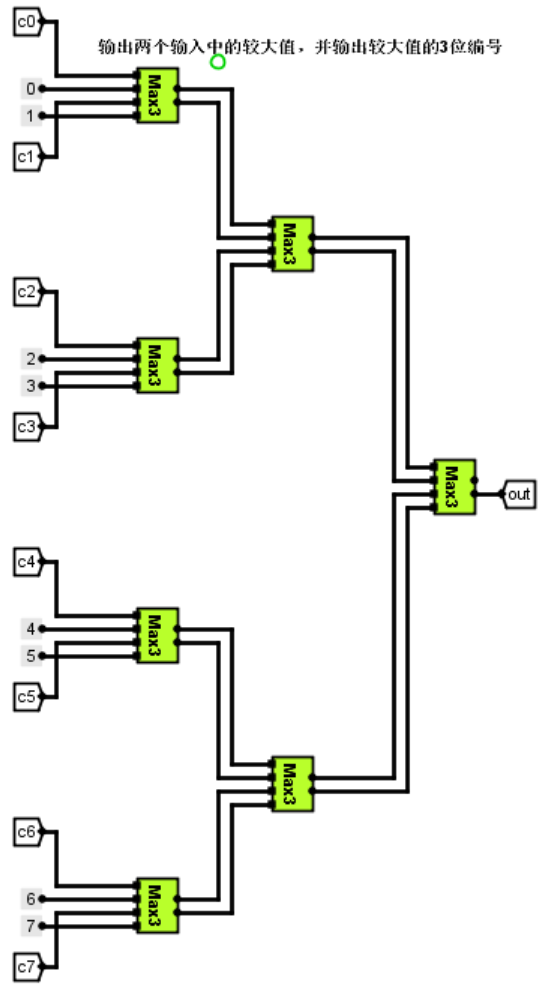


图2.14 归并比较电路

（7）得到写信号writer：采用解复用器（DMX），输入的Miss和BLKReady相与表示仅在数据发生缺失（Miss=1）和块数据准备就绪（BLKReady=1）时，得到的writer为1。下方的数据选择后的信号为DMX提供选择信号，判断哪一路需要输出。

Cache满时full=1，选择被替换的cache行编号out；Cache未满时full=0，选择为空cache行编号row。

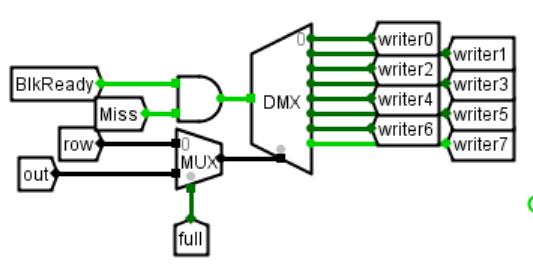


图2.15 写信号输出电路

（8）读数据输出：SlotData为块内数据，根据offset块内偏移量选择输出数据

电路图如2.16所示。

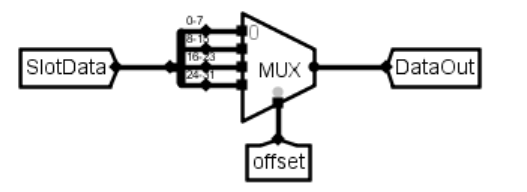


图2.16 读数据输出电路

总电路如图2.17所示。

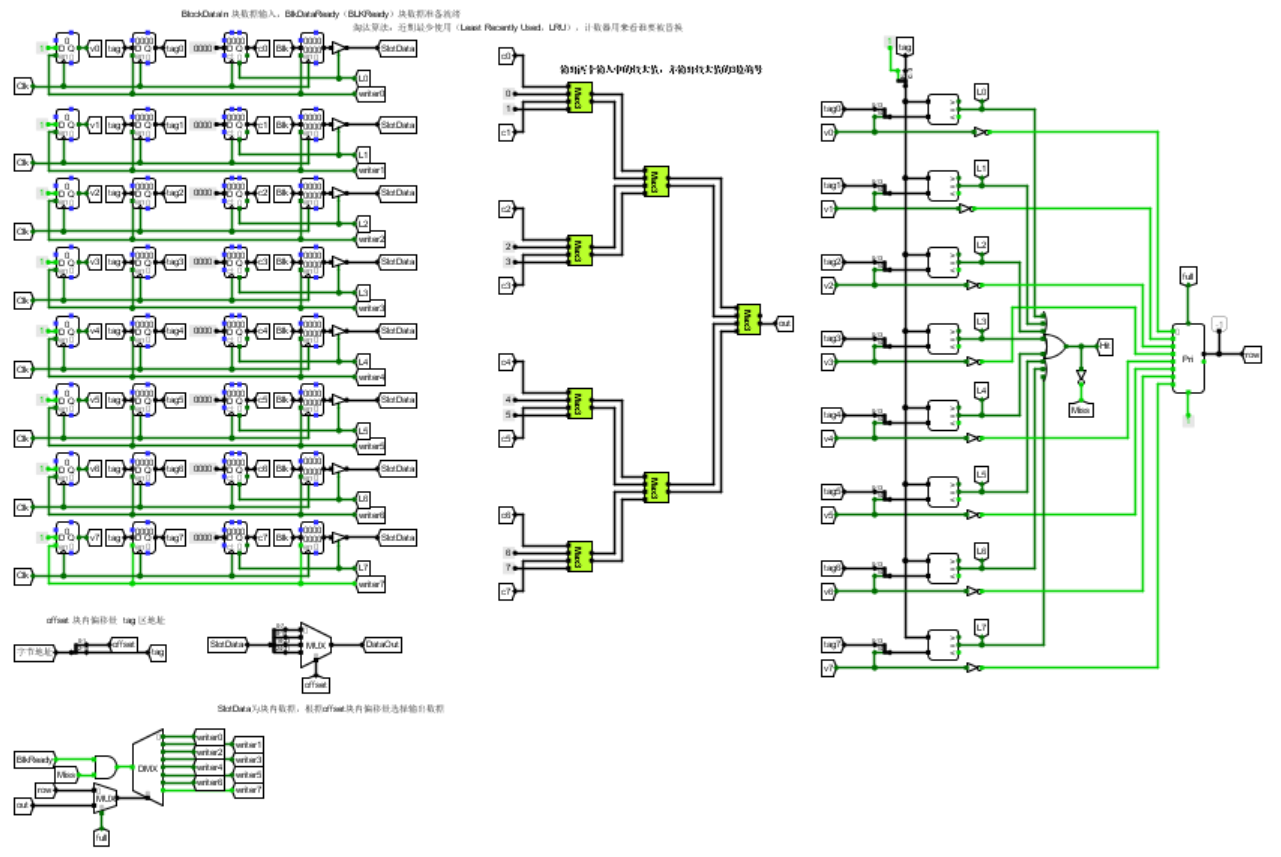


图2.17 全相联cache总电路

2.4.2、测试

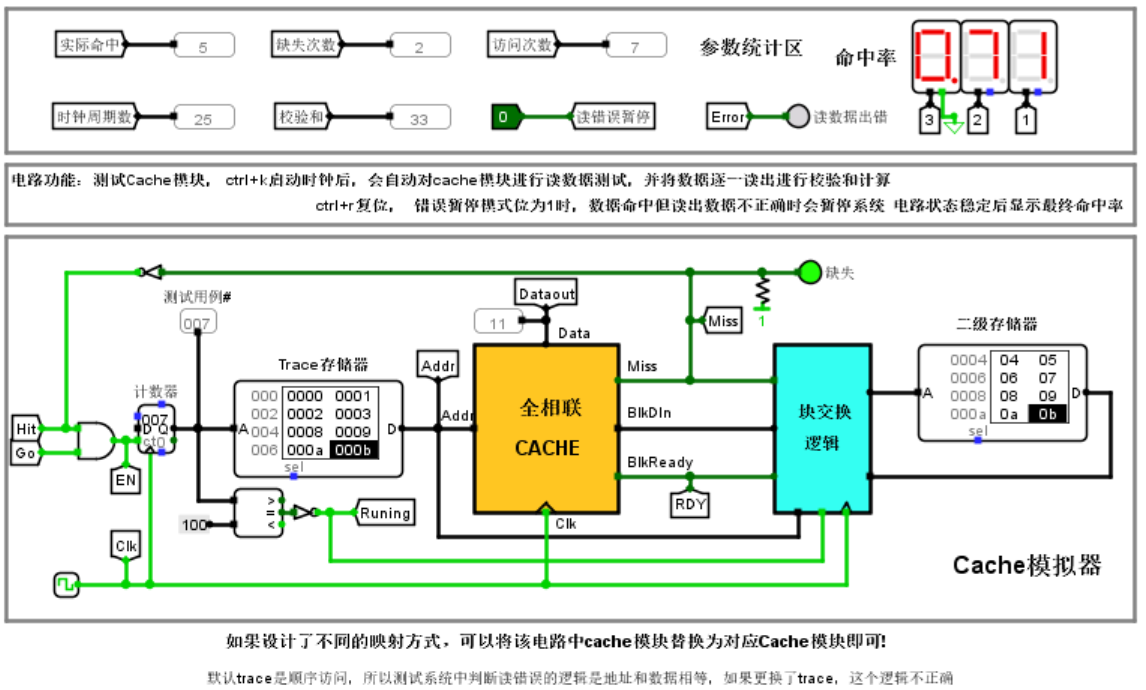


图2.18 全相映cache测试结果

2.5、直接相联映射电路实现

2.5.1、设计原理

（1）与全相连一样可以得到2位的块内偏移量offset ，而直接相联为一一映射，有3行，故行索引index为3位，剩下的11位为区地址tag。电路图如图2.19所示

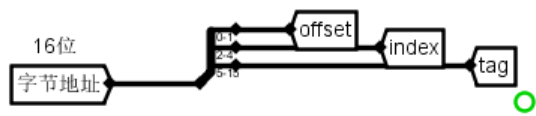


图2.19

（2）Cache行封装和全相联映射基本一致，只是储存tag的寄存器变为11位。

（3）写信号的获得也与全相连基本一致，选择信号变成了更简单的index。

（4）数据输出也是一摸一样。

（5）缺失信号Miss，读信号Li

由于是一一映射，直接通过index选择MUX判断对应的cache行中有效位是否为1，tag是否相等。若相等，则输出Miss为0，表示命中，并且输出对应读信号Li为1。电路图如图2.20所示

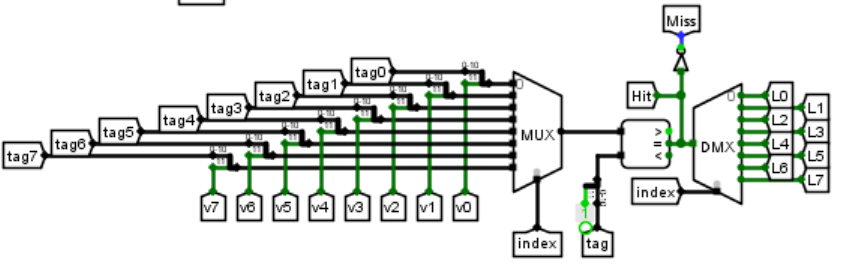


图2.20

总电路如图2.21所示。

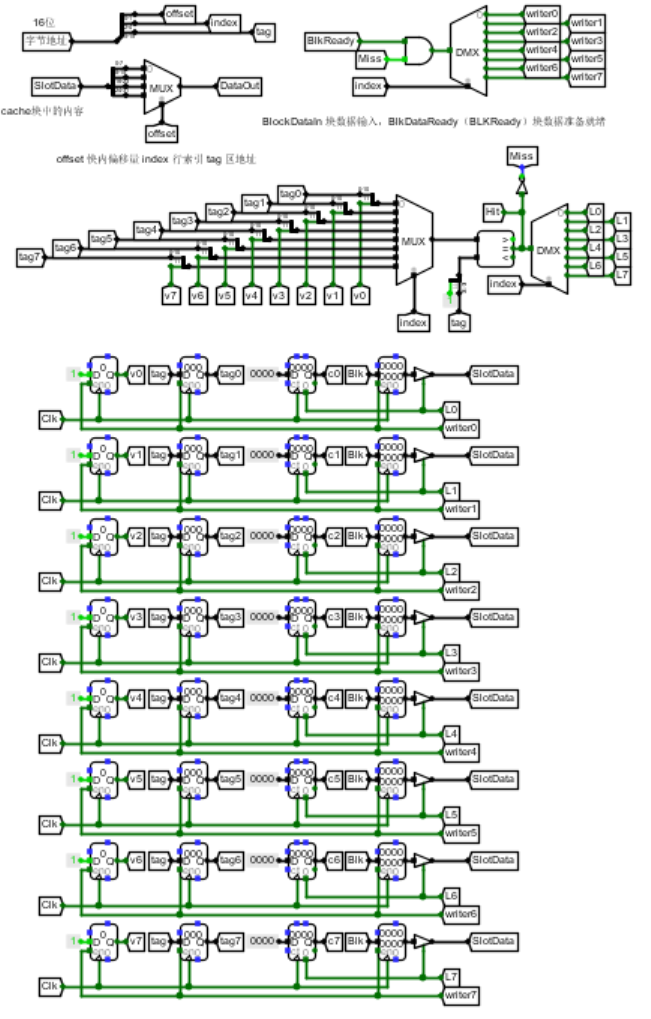


图2.20 直接相联映射总电路

2.5.2、测试

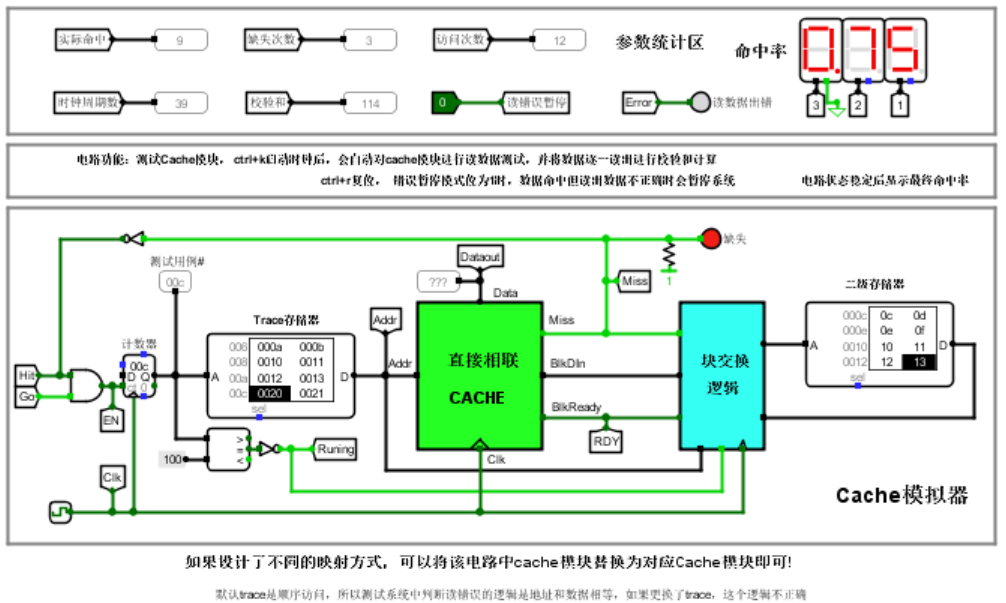


图2.22 直接相联cache测试结果

3、遇到的问题及解决办法

问题一：存储器扩展电路中的地址线冲突

问题描述：

在汉字字库存储芯片扩展实验中，当连接多个ROM芯片时，发现地址线之间存在冲突，导致存储器无法正确读取数据。具体表现为某些地址的数据输出错误，或者部分地址无法访问。

解决办法：

1.检查地址线连接：重新检查所有地址线的连接，确保没有短路或断路的情况。特别是高两位地址线的连接，因为它们用于控制多路选择器。

2.重新设计地址分配：重新分配地址线，确保每个ROM芯片的地址范围互不重叠。例如，将4个4K×32位ROM的地址线重新分配，确保它们的地址范围分别为0-4K、4K-8K、8K-12K、12K-16K。

问题二：MIPSRAM设计中的数据选择错误

问题描述：

在MIPSRAM设计实验中，当进行字节、半字和字访问时，发现读出的数据不正确。例如，在字节访问模式下，读取的数据总是包含无效的高位，而在半字访问模式下，数据的组合方式错误。

解决办法：

1.检查数据选择逻辑：重新检查读数据选择电路的逻辑设计，确保在不同访问模式下，数据选择器能够正确选择对应的寄存器数据。

2.验证地址解析：检查字节地址的解析逻辑，确保在不同访问模式下，地址的低两位或倒数第二位能够正确控制寄存器的选择。

问题三：Cache设计中的命中率问题

问题描述：

在全相联映射Cache设计实验中，发现Cache的命中率远低于预期。即使在简单的测试场景下，Miss信号频繁出现，导致Cache的性能不佳。

解决办法：

1.检查地址映射逻辑：重新检查Cache行的地址映射逻辑，确保块内偏移量、区地址和行索引的划分正确。特别是区地址的比较逻辑，确保能够正确匹配。

2.优化替换算法：检查LRU算法的实现，确保计数器能够正确记录每个Cache行的使用次数。如果计数器逻辑有误，可能导致错误的行被替换。

问题四：直接相联映射中的地址解析错误

问题描述：

在直接相联映射Cache设计实验中，发现地址解析逻辑错误，导致Cache行的选择不正确。具体表现为某些地址的块数据无法正确写入或读出。

解决办法：

1.检查行索引逻辑：检查行索引的生成逻辑，确保能够正确解析出行索引。特别是在多路选择器的选择信号中，确保行索引能够正确控制数据的读写。

2.验证写信号逻辑：检查写信号的生成逻辑，确保在Miss信号和BlkReady信号同时有效时，能够正确触发写入操作。