

计算机组成与嵌入式系统

--存储系统实验

老师：徐文辉

QQ: 4127164

电话: 18202799815

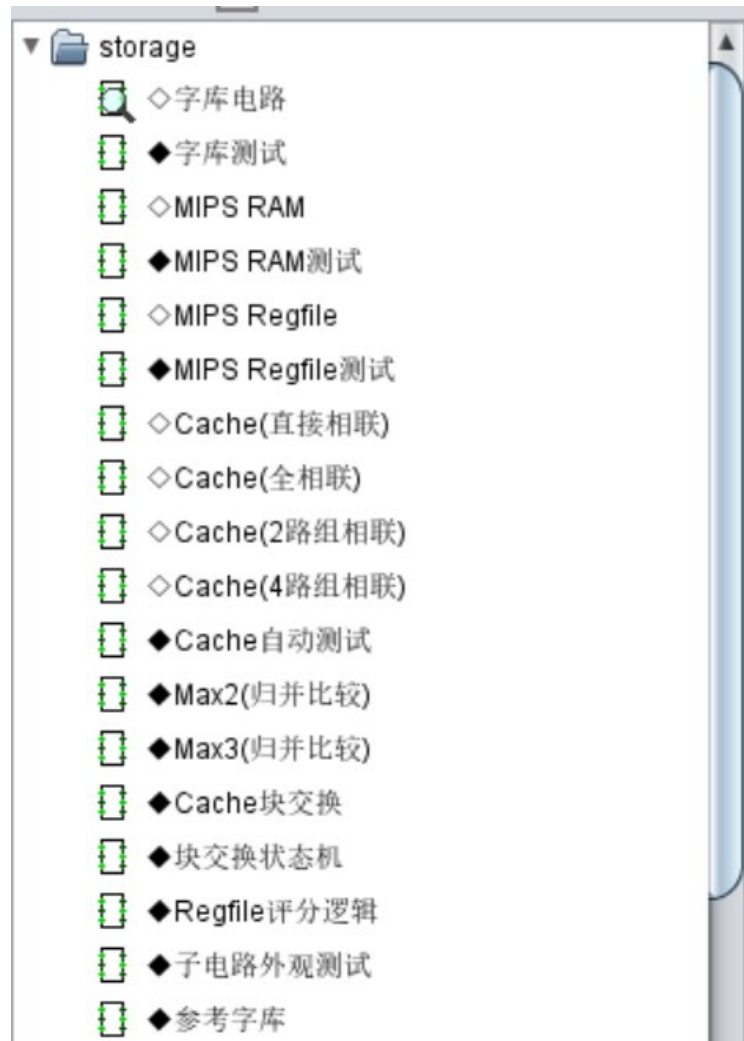
Email: xuwenhui@hust.edu.cn

实验环境

- Logisim仿真软件

实验内容

- 存储器扩展
- MIPS RAM设计
- MIPS 寄存器文件设计
- Cache硬件设计



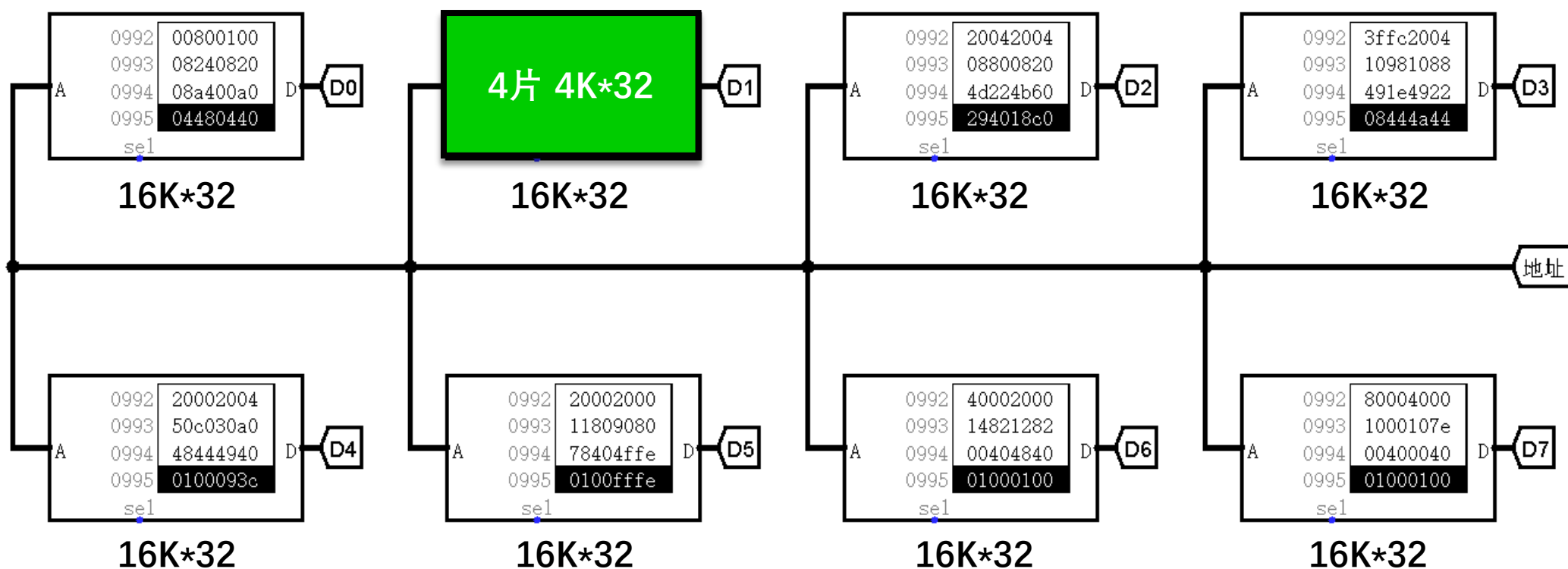
实验一：存储器扩展实验

存储器扩展实验

• 实验目的

- 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）
- 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

存储器扩展实验



- 用4片4K*32位的ROM 替换其中一片16K*32位器件
 - 容量扩展 (地址总线扩展)
 - 原字库文件数据如何分布?

存储器扩展实验

Logisim 2.15.0.2.exe: 字库电路 of storage

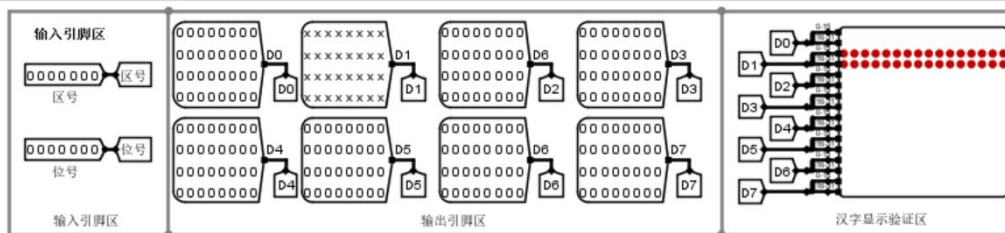
File Edit Project Simulate Window Help



Circuit: 字库电路

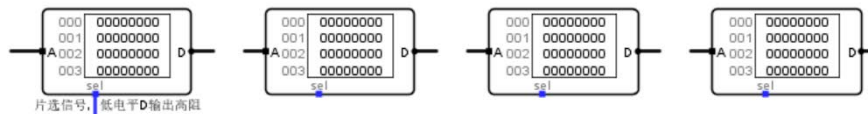
Circuit Name: 字库电路
Shared Label: 字库
Shared Label Facing: East
Shared Label Font: SansSerif Plain 12
Label Color: #000000

Zoom: 81%



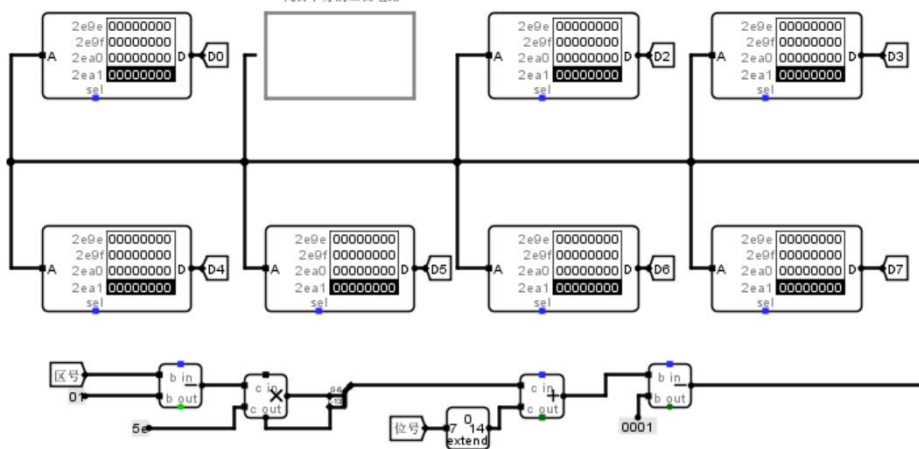
电路功能: 利用下图中给定的ROM器件实现汉字16*16点阵字库文件, 不允许修改ROM地址数据位置, 字库数据可以通过参考字库导出

图中有一个空白器件, 表示少一个器件, 请按照字扩展《容量扩展, 地址线扩展》的方式利用第一行的存储单元完成最终字库, 字库文件需进行简单的分类
请勿修改引脚, 请勿修改子电路封装

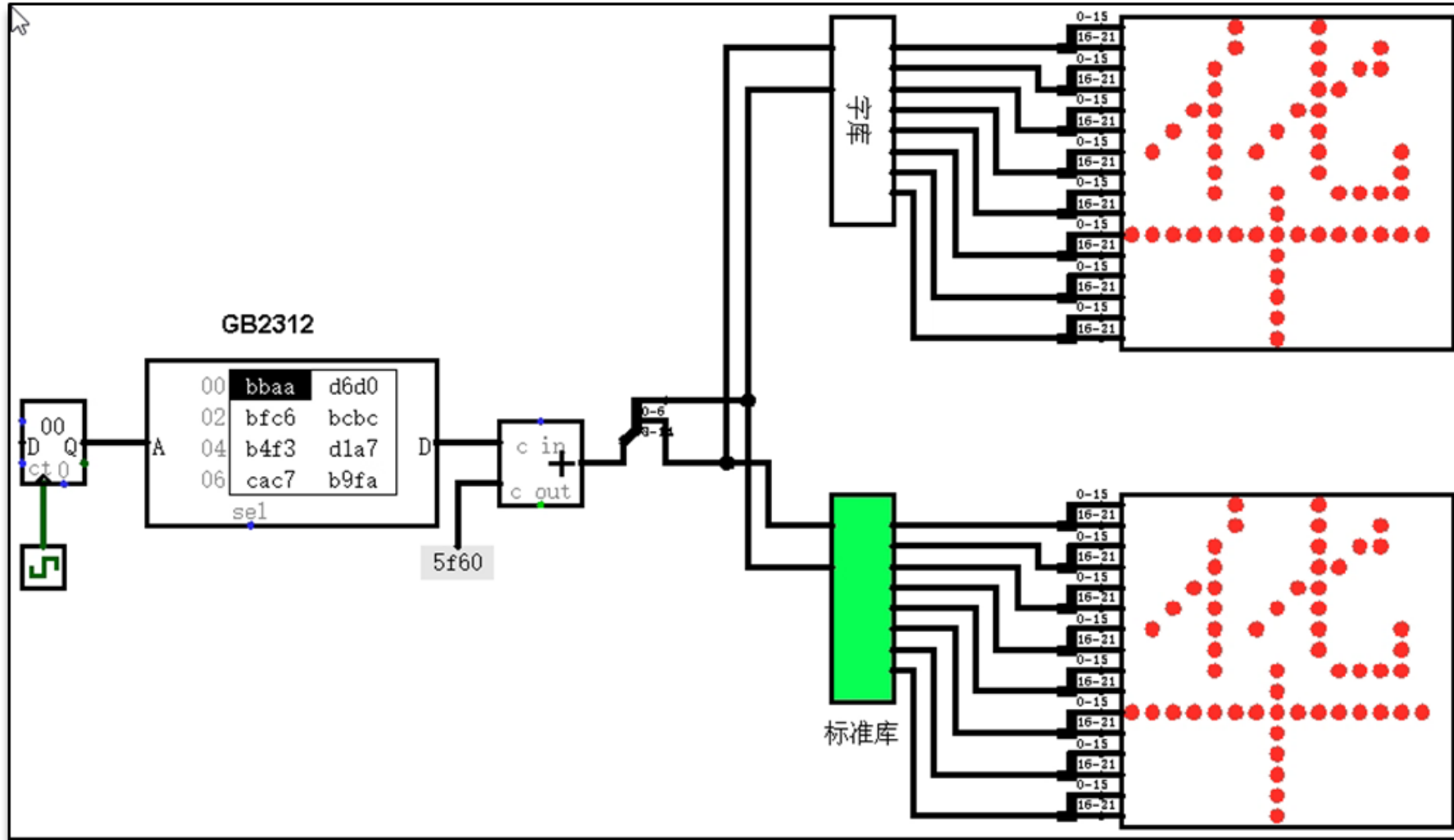


片选信号, 低电平D输出高阻

用上方的4块ROM进行扩展
代替下方的空白电路



存储器扩展实验



时钟频率8Hz, Ctrl+k, Command+k 自动测试, 提交检查

实验二：MIPS RAM设计实验

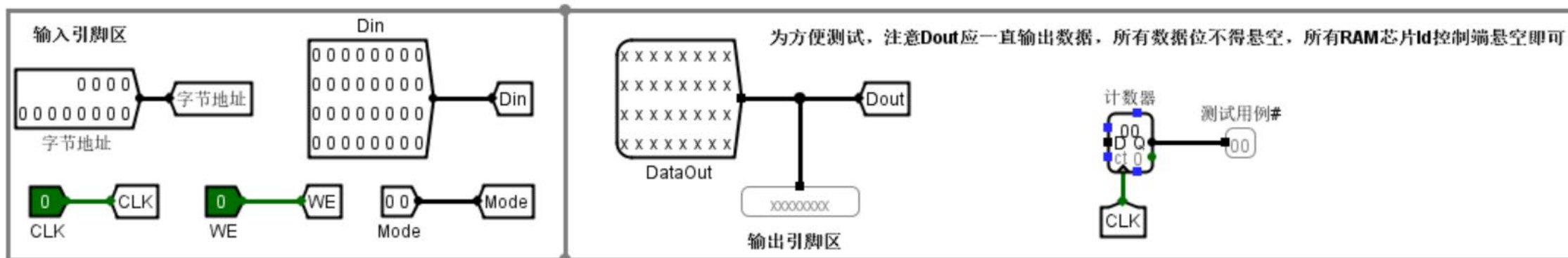
MIPS RAM设计实验

• 实验目的

- 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/store byte)
 - lh/sh (load/store half)
 - lw/sw (load/store word)
- 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

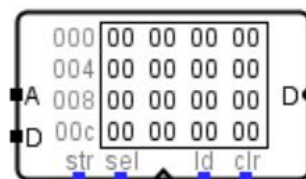
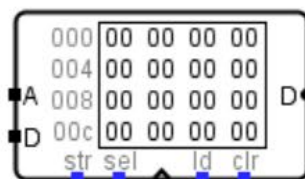
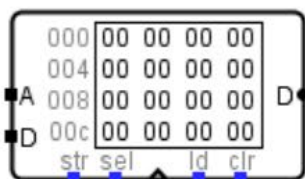
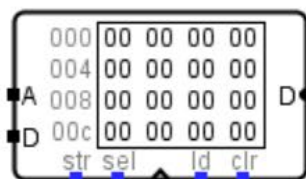
MIPS RAM设计实验

- 4个4KB × 8位的RAM组件进行扩展
- 设计完成能按8/16/32位进行读写访问的32位存储器



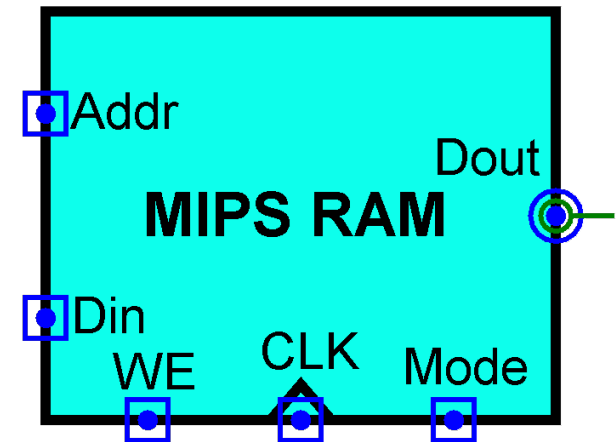
电路功能：利用下图中给定的RAM器件实现既可以按照字节，也可以按照半字，也可以按照32位字访问的MIPS存储器
请勿增减引脚，请勿修改子电路封装

请在下方利用上方输入输出引脚的隧道信号构建电路，ctrl+d复制选择部件

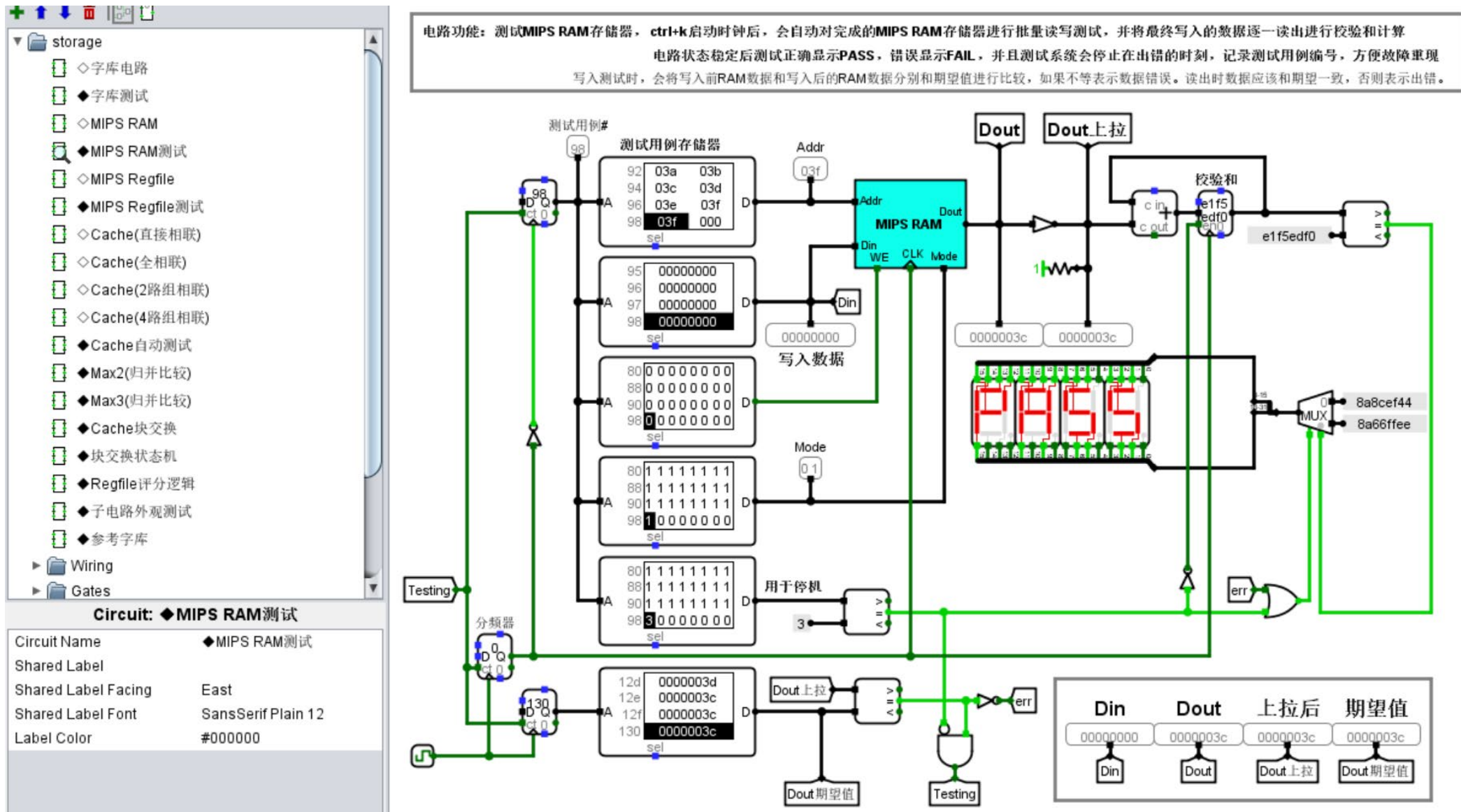


MIPS RAM设计实验

- Addr: 位宽12位, 字节地址输入
 - 字访问 (忽略最低2位)、半字访问 (忽略最低位)、字节访问 (低两位片选)
- Din: 位宽32位, 写入数据 (不同模式下, 有效数据存放在最低位, 高位忽略)
- Dout: 位宽32位, 读出数据 (不同模式下, 有效数据存放在最低位, 高位补零)
- Mode: 位宽2位, 访问模式控制位
 - 00: 字访问
 - 01: 字节访问
 - 10: 半字访问
- WE: 位宽1位, 写使能 (1: 写入; 0: 读出)



MIPS RAM设计实验



实验三：MIPS 寄存器文件设计实验

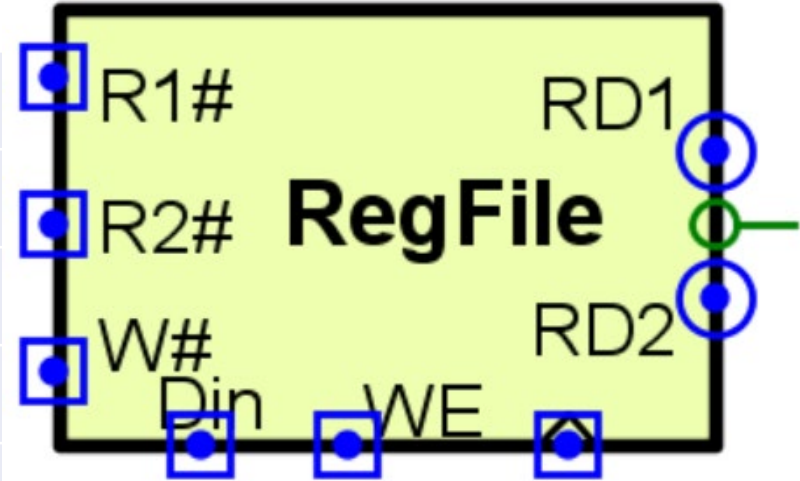
MIPS 寄存器文件设计实验

• 实验目的

- 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
- 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

MIPS 寄存器文件设计实验

引脚	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
D _{in}	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D _{in} 数据写入W#寄存器
CLK	输入	1	时钟信号，上跳沿有效
RD ₁	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
RD ₂	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



MIPS 寄存器文件设计实验

Logisim 2.15.0.2.exe: <MIPS Regfile of storage

File Edit Project Simulate Window Help

storage*

- ◇字库电路
- ◆字库测试
- ◇MIPS RAM
- ◆MIPS RAM测试
- ◇MIPS Regfile
- ◆MIPS Regfile测试
- ◇Cache(直接相联)
- ◇Cache(全相联)
- ◇Cache(2路组相联)
- ◇Cache(4路组相联)
- ◆Cache自动测试
- ◆Max2(归并比较)
- ◆Max3(归并比较)
- ◆Cache块交换
- ◆块交换状态机
- ◆Regfile评分逻辑
- ◆子电路外观测试
- ◆参考字库

Wiring

Gates

Circuit: <MIPS Regfile

Circuit Name <MIPS Regfile

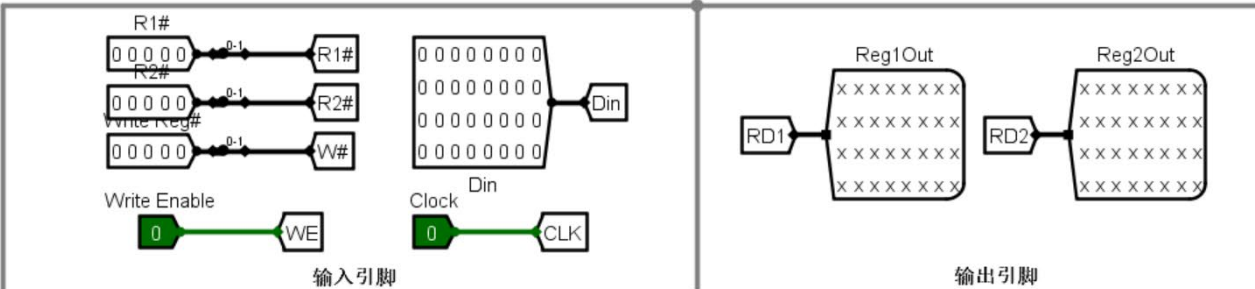
Shared Label

Shared Label Facing East

Shared Label Font SansSerif Plain 12

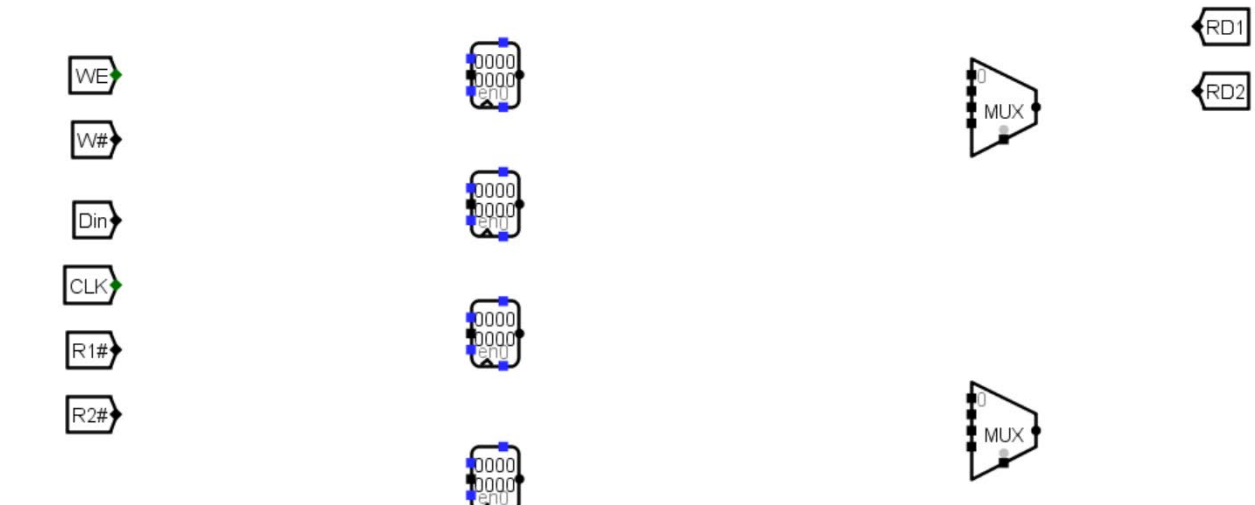
Label Color #000000

Zoom: 123%



The diagram illustrates the MIPS Register File design. It is divided into two main sections: '输入引脚' (Input Pins) and '输出引脚' (Output Pins). The input section shows three 4-bit registers (R1#, R2#, W#) connected to a 4-bit 'Write Reg#' input. A 'Write Enable' (WE) input is connected to a logic '0'. A 'Clock' (CLK) input is connected to a logic '0'. A 'Din' input is connected to a 32-bit register. The output section shows two 32-bit registers (Reg1Out, Reg2Out) connected to 'RD1' and 'RD2' inputs. The output pins are labeled 'Reg1Out' and 'Reg2Out'.

电路功能：实现MIPS寄存器组，为简化工作量，寄存器编号高3位不要，最终电路中只需要实现0-3号寄存器，注意0号寄存器恒零
请仔细测试电路，核对功能无误后提交教师进行自动检测评分



The implementation diagram shows the circuit components. On the left, there are input pins for WE, W#, Din, CLK, R1#, and R2#. In the center, there are four 4-bit multiplexers (MUX) connected to the R1# and R2# inputs. On the right, there are two 32-bit multiplexers (MUX) connected to the RD1 and RD2 inputs. The output pins are labeled RD1 and RD2.

MIPS 寄存器文件设计实验

Logisim 2.15.0.2.exe: MIPS Regfile测试 of storage

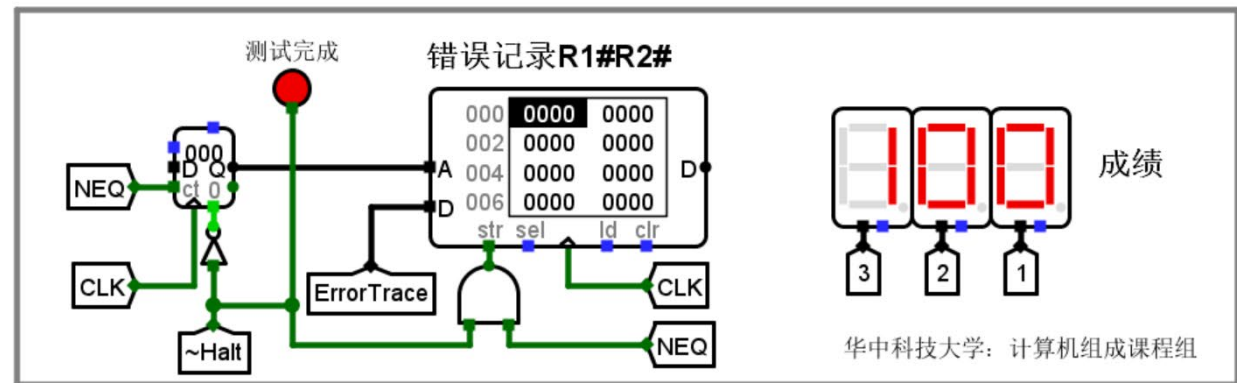
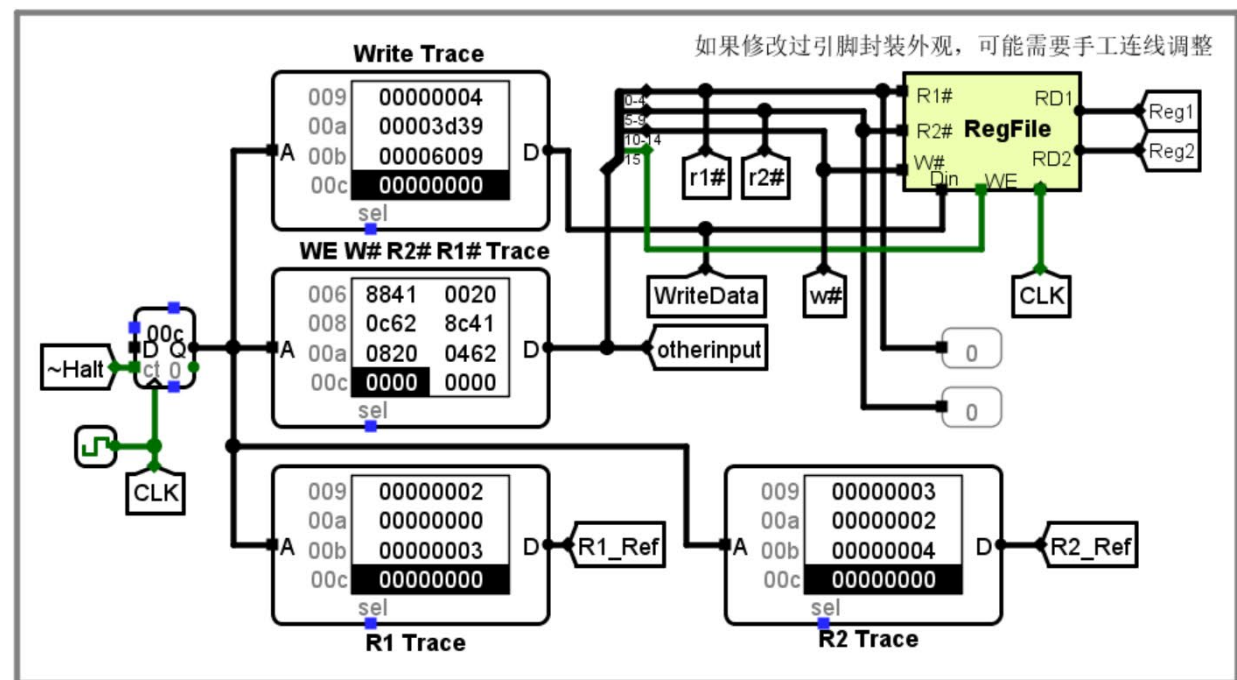
File Edit Project Simulate Window Help



Circuit: ◆MIPS Regfile测试

Circuit Name ◆MIPS Regfile测试
Shared Label
Shared Label Facing East
Shared Label Font SansSerif Plain 12
Label Color #000000

Zoom: 150%



实验四：Cache硬件设计实验

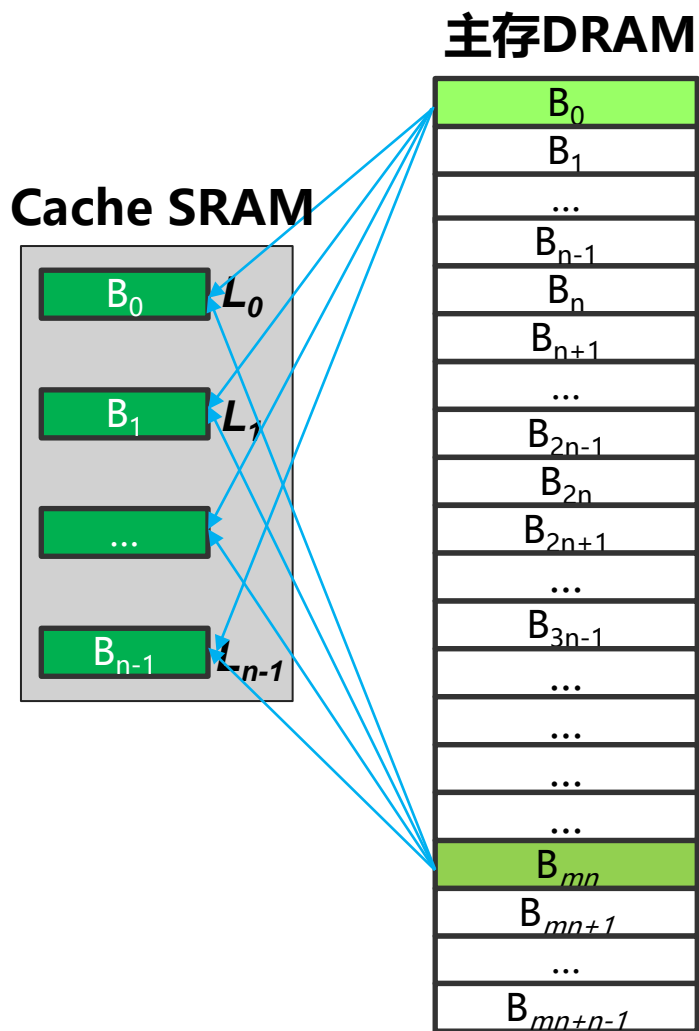
Cache硬件设计实验

• 实验目的

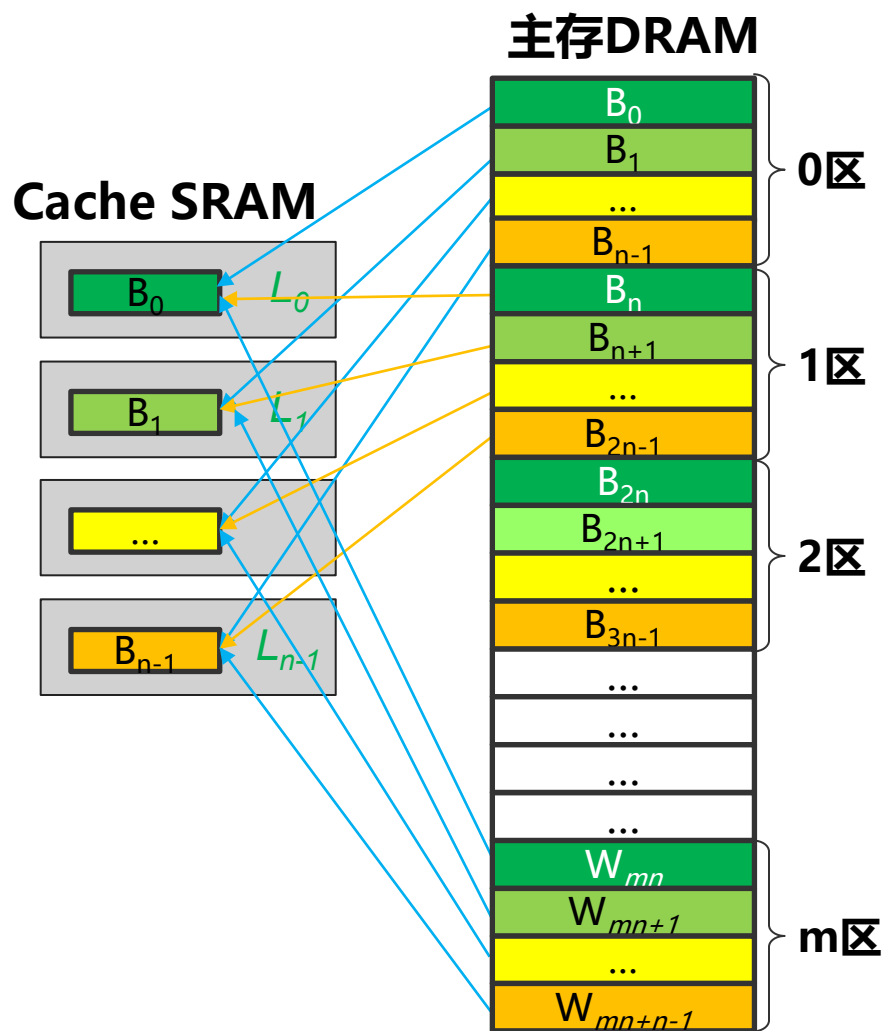
- 掌握Cache实现的关键技术
 - 数据查找（如何快速判断数据是否在Cache中）
 - 地址映射（主存中的数据块应如何放置在Cache中）
 - 全相联、直接相连、组相联
 - 替换算法（选择什么样的Cache数据块进行替换或淘汰）
- 熟悉译码器、多路选择器、寄存器的使用
 - 能设计实现Cache机制

Cache硬件设计实验

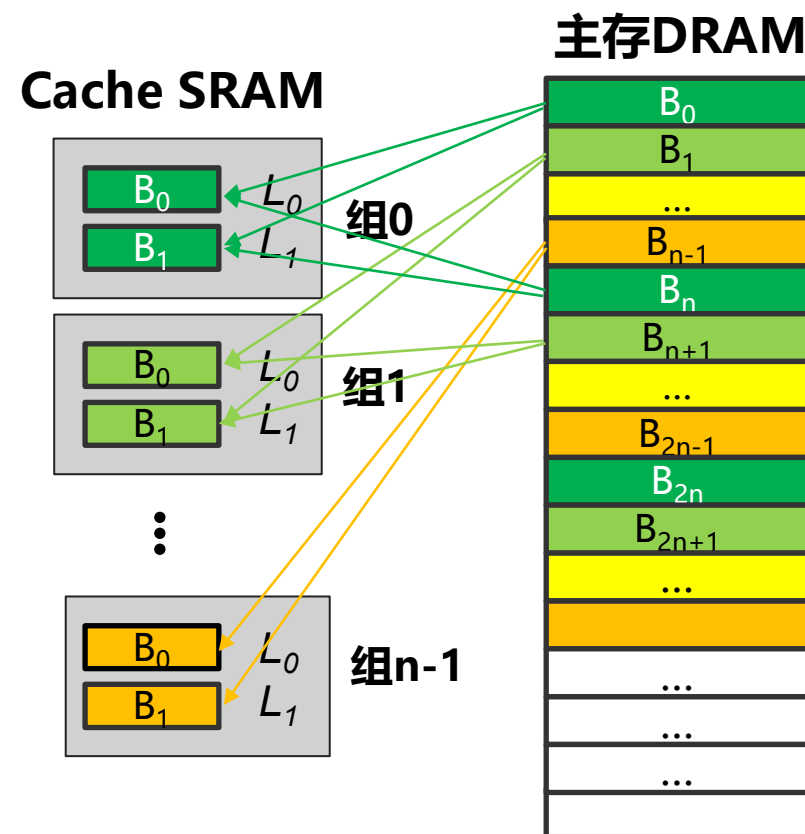
全相联映射



直接相联映射



组相联映射



Cache硬件设计实验

全相联映射逻辑实现

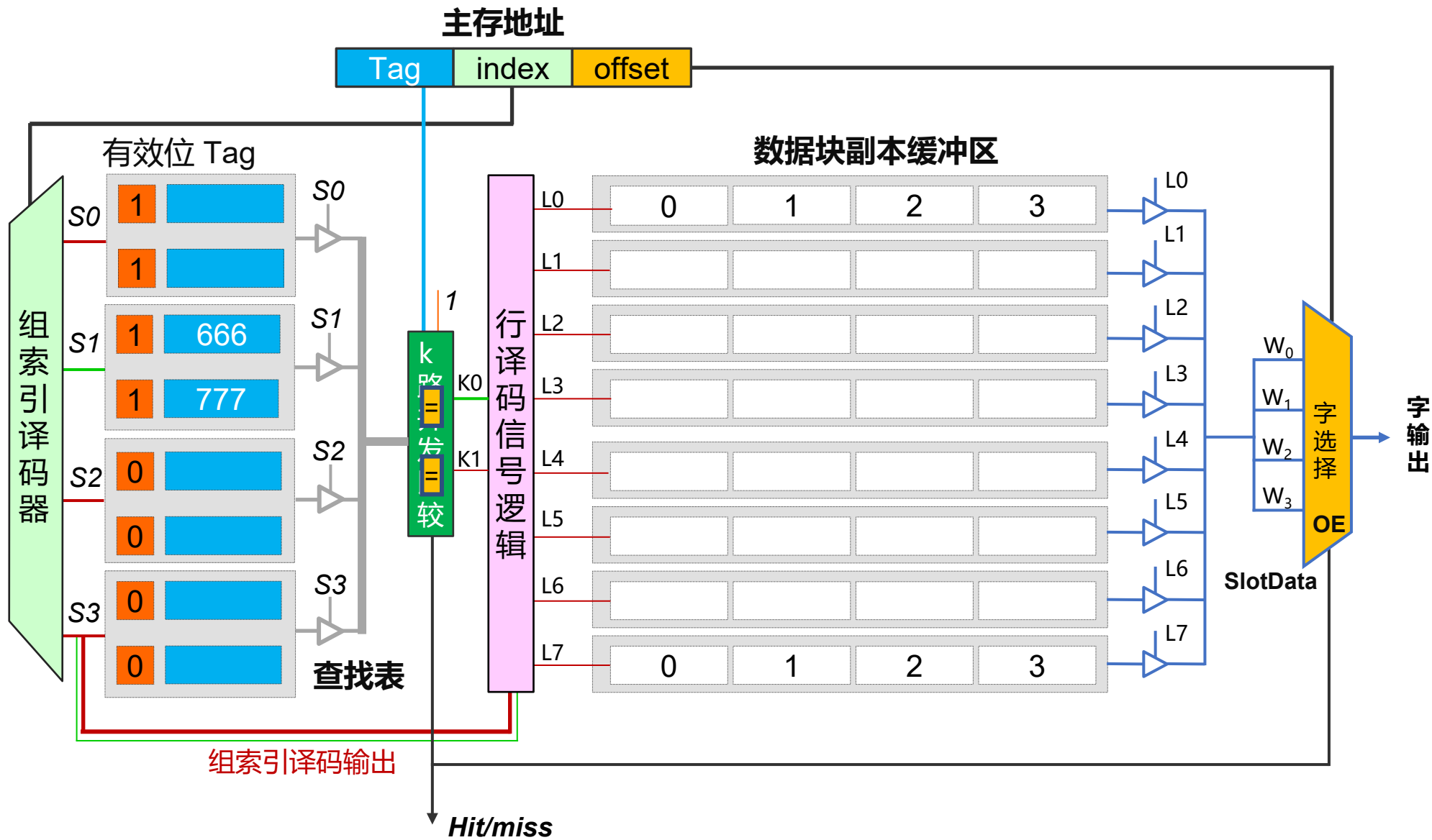


Cache硬件设计实验

直接相联映射逻辑实现



组相联映射逻辑实现



Cache硬件设计实验

Logisim 2.15.0.2.exe: ◊Cache(直接相联) of storage

File Edit Project Simulate Window Help

storage*

- ◊字库电路
- ◆字库测试
- ◊MIPS RAM
- ◆MIPS RAM测试
- ◊MIPS Regfile
- ◆MIPS Regfile测试
- ◊Cache(直接相联)
- ◊Cache(全相联)
- ◊Cache(2路组相联)
- ◊Cache(4路组相联)
- ◆Cache自动测试
- ◆Max2(归并比较)
- ◆Max3(归并比较)
- ◆Cache块交换
- ◆块交换状态机
- ◆Regfile评分逻辑
- ◆子电路外观测试
- ◆参考字库

Wiring

Gates

Circuit: ◊Cache(直接相联)

Circuit Name ◊Cache(直接相联)

Shared Label

Shared Label Facing East

Shared Label Font SansSerif Plain 12

Label Color #000000

Zoom: 150%

输入引脚区

Addr

0 0 0 0 0 0 0 0

字节地址

BlockDataIn

X X X X X X X X

X X X X X X X X

X X X X X X X X

X X X X X X X X

Blk

BlkDataReady

0

BlkReady

CLK

0

Clk

输出引脚区

Miss

Miss

DataOut

DataOut

X X X X X X X X

电路功能: 利用上面给定的隧道构建8行的Cache, cache映射方式自选

请勿增减引脚, 请勿修改子电路封装

请在下方利用上方输入输出引脚的隧道信号构建电路, ctrl+d复制选择部件

16位

字节地址

0-1

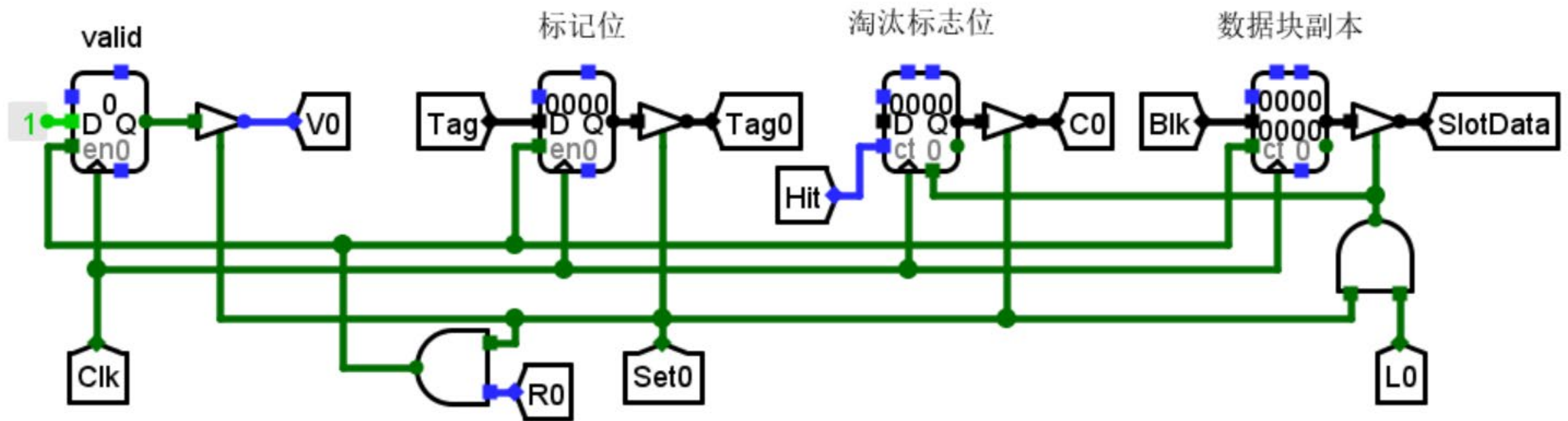
2-3

5-15

SlotData

DataOut

Cache硬件设计实验



- V0: 数据有效信号
- Tag0: 标记位信号
- C0: 淘汰标志位
- Set0: 组索引信号
- L0: 行选中信号
- R0: 当前组被替换信号

Cache硬件设计实验

Logisim 2.15.0.2.exe: ♦Cache自动测试 of storage

File Edit Project Simulate Window Help

storage*

- 字库电路
- 字库测试
- MIPS RAM
- MIPS RAM测试
- MIPS Regfile
- MIPS Regfile测试
- Cache(直接相联)
- Cache(全相联)
- Cache(2路组相联)
- Cache(4路组相联)
- Cache自动测试
- Max2(归并比较)
- Max3(归并比较)
- Cache块交换
- 块交换状态机
- Regfile评分逻辑
- 子电路外观测试
- 参考字库

Wiring

Gates

Circuit: ♦Cache自动测试

Circuit Name: ♦Cache自动测试

Shared Label

Shared Label F...: East

Shared Label F...: SansSerif Plain...

Label Color: #000000

Zoom: 120%

实际命中 243 缺失次数 13 访问次数 256 参数统计区 命中率 0.94

时钟周期数 373 校验和 7337 读错误暂停 0 Error 读数据出错

电路功能: 测试Cache模块, ctrl+k启动时钟后, 会自动对cache模块进行读数据测试, 并将数据逐一读出进行校验和计算
ctrl+r复位, 错误暂停模式位为1时, 数据命中但读出数据不正确时会暂停系统 电路状态稳定后显示最终命中率

Cache模拟器

测试用例# 100

计数器 100

Trace存储器

0fa	0026	0027
0fc	0004	0005
0fe	0006	0007
100	0000	0000

Addr 0000

Data 8c

4路组相联 CACHE

Miss 1

块交换逻辑

二级存储器

0028	28	29
002a	2a	2b
002c	2c	2d
002e	2e	2f

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

请同学们开始实验！