

**计算机组成原理-实验报告**

院 系 软件学院

专业班级 软工2003班

姓 名 刘铭宸

学 号 U202010783

指导教师 黄浩

2022年 12月 23 日

**目录**

[1 课程实验概述 1](#_Toc122781683)

[1.1 实验目的 1](#_Toc122781684)

[1.2 实验环境 1](#_Toc122781685)

[1.3 实验内容 2](#_Toc122781686)

[2 运算器组成实验 3](#_Toc122781687)

[2.1 八位串行可控加减法器电路设计 3](#_Toc122781688)

[2.2 四位先行进位电路 6](#_Toc122781689)

[2.3 4位快速加法器设计 8](#_Toc122781690)

[2.4 16位快速加法器设计 9](#_Toc122781691)

[2.5 32位快速加法器设计 11](#_Toc122781692)

[2.6 32位MIPS运算器设计 13](#_Toc122781693)

[3 存储系统综合实验 17](#_Toc122781694)

[3.1 存储扩展实验 17](#_Toc122781695)

[3.2 MIPS寄存器文件设计 19](#_Toc122781696)

[4 心得体会 28](#_Toc122781697)

# 课程实验概述

## 实验目的

### 运算器组成实验

* 熟悉Logisim软件平台；
* 掌握运算器基本工作原理；
* 掌握运算溢出检测的原理和实现方法；
* 理解有符号数和无符号数运算的区别；
* 理解基于补码的加/减运算实现原理；
* 熟悉运算器的数据传输通路。

### 存储系统综合实验

* 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。

## 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。

<http://www.cburch.com/logisim/docs.html>

## 实验内容

### 运算器组成实验

* 八位串行可控加减法器电路设计
* 四位先行进位电路
* 4位快速加法器设计
* 16位快速加法器设计
* 32位快速加法器设计
* 32位MIPS运算器设计

### 存储系统综合实验

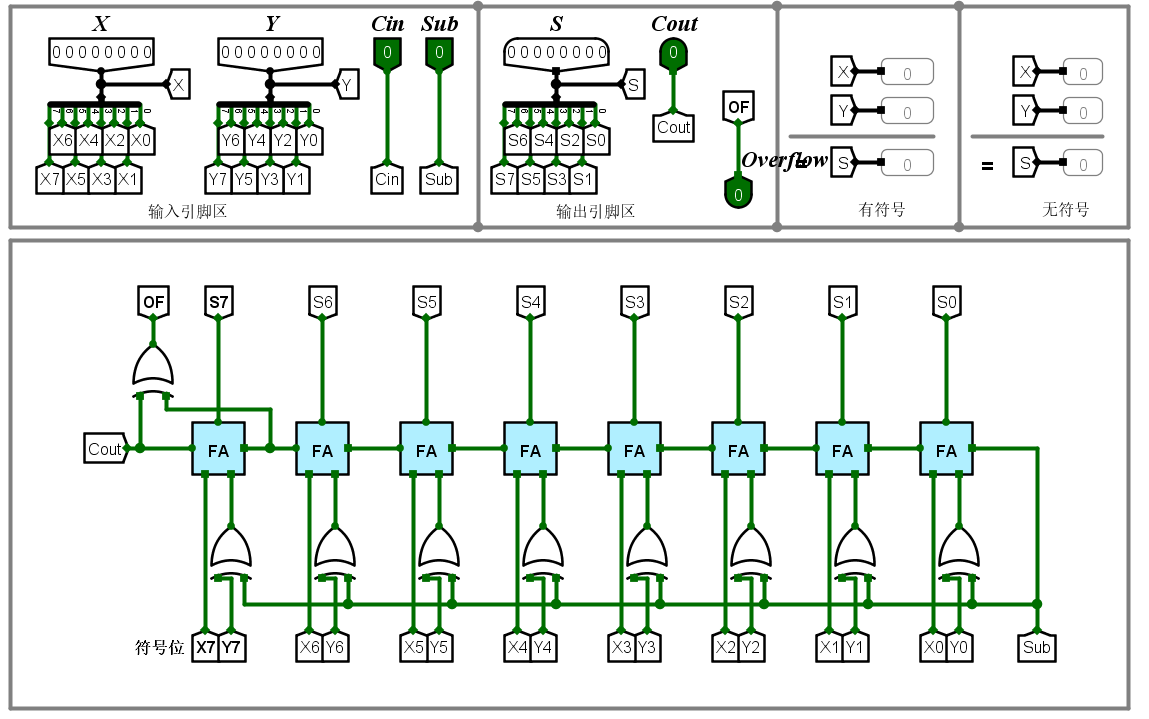
* 存储扩展实验
* MIPS寄存器文件设计

# 运算器组成实验

## 八位串行可控加减法器电路设计

### 设计思路

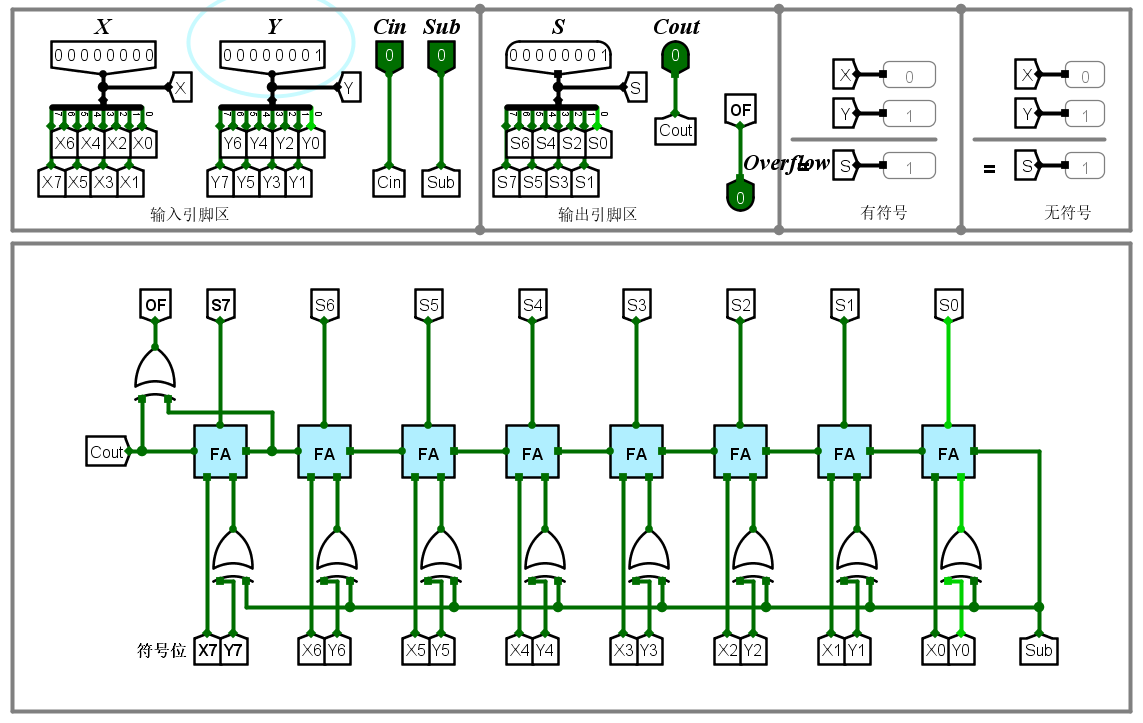
该电路与8位串行加法器的不同之处在于要通过Sub输入引脚实现控制加减法的功能。设计思路是将减Y的补码转换成加-Y的补码，将Y的每一位都与Sub引脚进行异或运算后再与X的各位相加，当Sub为0时值不改变，进行加法运算；当Sub为1时相当于将Y的各位取反，并将Sub作为低位的进位，这样就实现了将Y各位取反，末位加一的操作，将Y的补码转换成-Y的补码，相当于进行减法运算。溢出检测方法与8位串行加法器相同，根据最高数据位的进位与符号位的进位是否一致进行检测。电路设计如下图所示：



### 实验结果

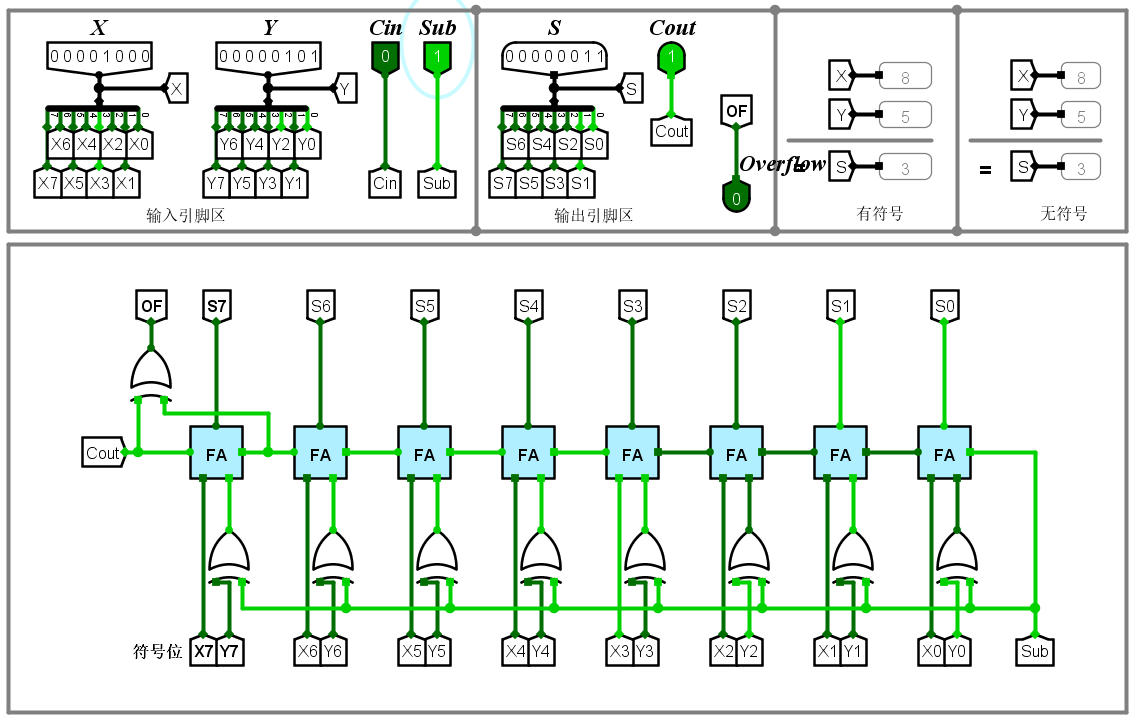
1. x=0000 0000，y=0000 0001，sub=0；

s=0000 0001，Cout=0，OF=0



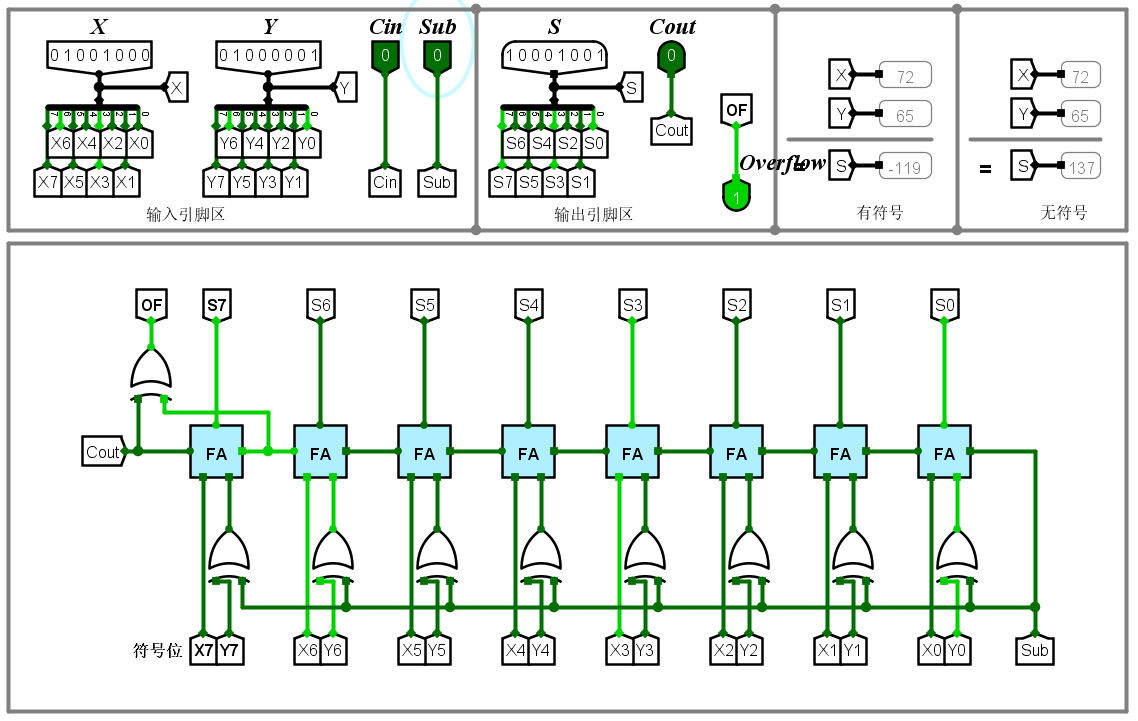
1. x=0000 1000，y=0000 0101，sub=1；

s=0000 0011，Cout=1，OF=0



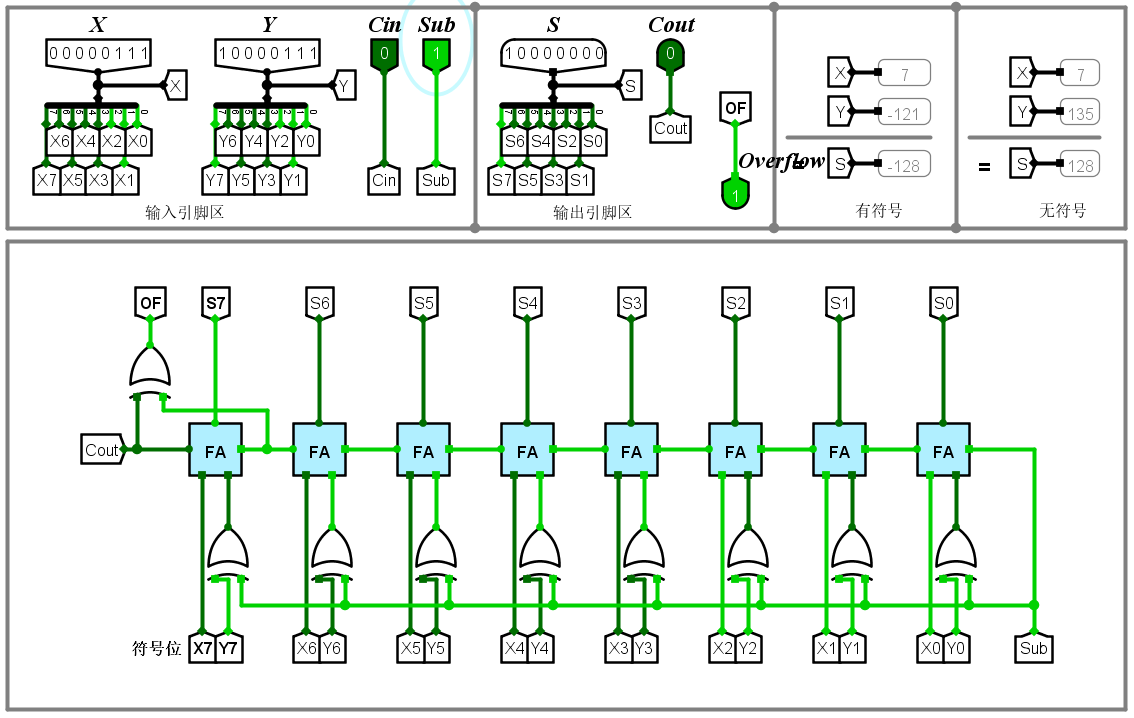
1. x=0100 1000，y=0100 0001，sub=0；

s=1000 1001，Cout=0，OF=1



1. x=0000 0111，y=1000 0111， sub=1；

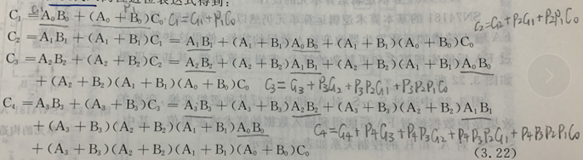
s=1000 0000， Cout=0，OF=1



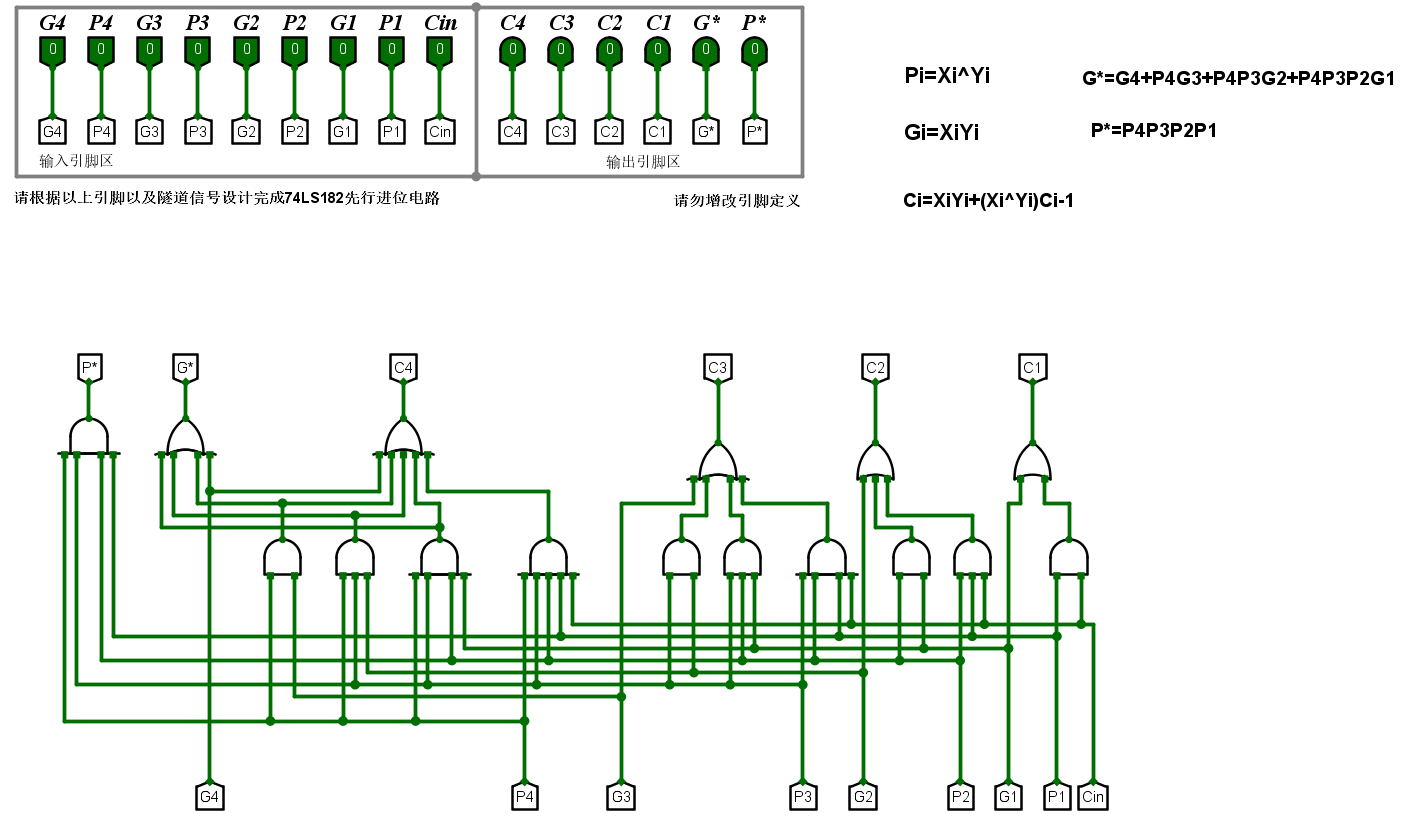
## 四位先行进位电路

### 设计思路

根据公式，各位的进位可以独立根据两个加数的各位计算出来，以此实现四位先行进位电路。公式如下：



电路设计如下图所示：

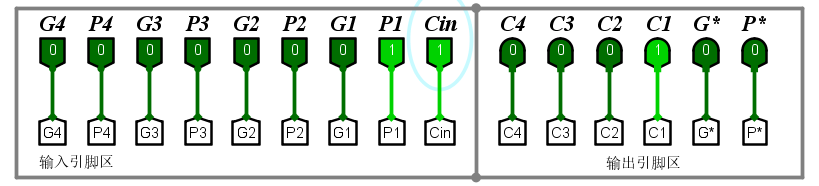


G\*称为成组进位生成函数，P\*称为成组进位传递函数，通过这两个函数，4位一组的进位信号可以采用类似的原理进行成组的先行进位，可以方便级联构成多级的先行进位系统。

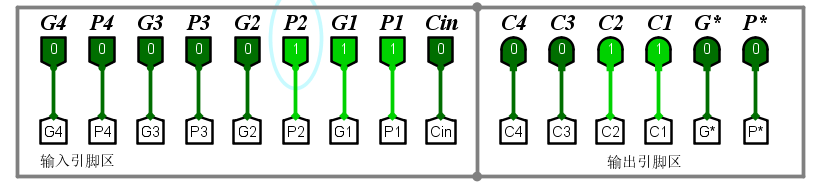
### 实验结果

输入序列G4 P4 G3 P3 G2 P2 G1 P1 Cin，输出序列C4 C3 C2 C1 G\* P\*

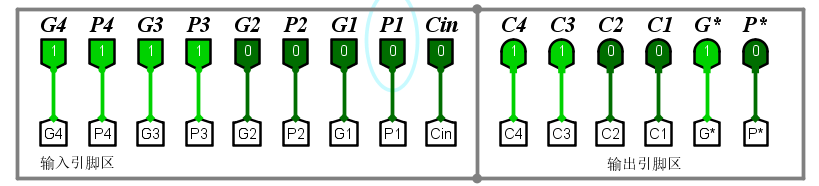
1. 输入：0000 0001 1 输出：0001 00



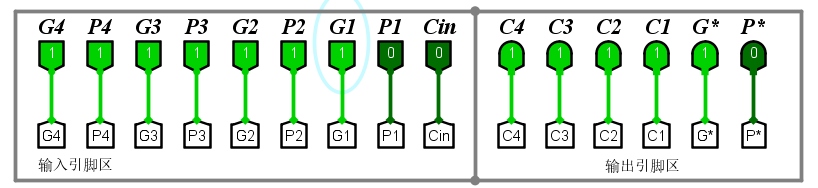
1. 输入：0000 0111 0 输出：0011 00



1. 输入：1111 0000 0 输出：1100 10



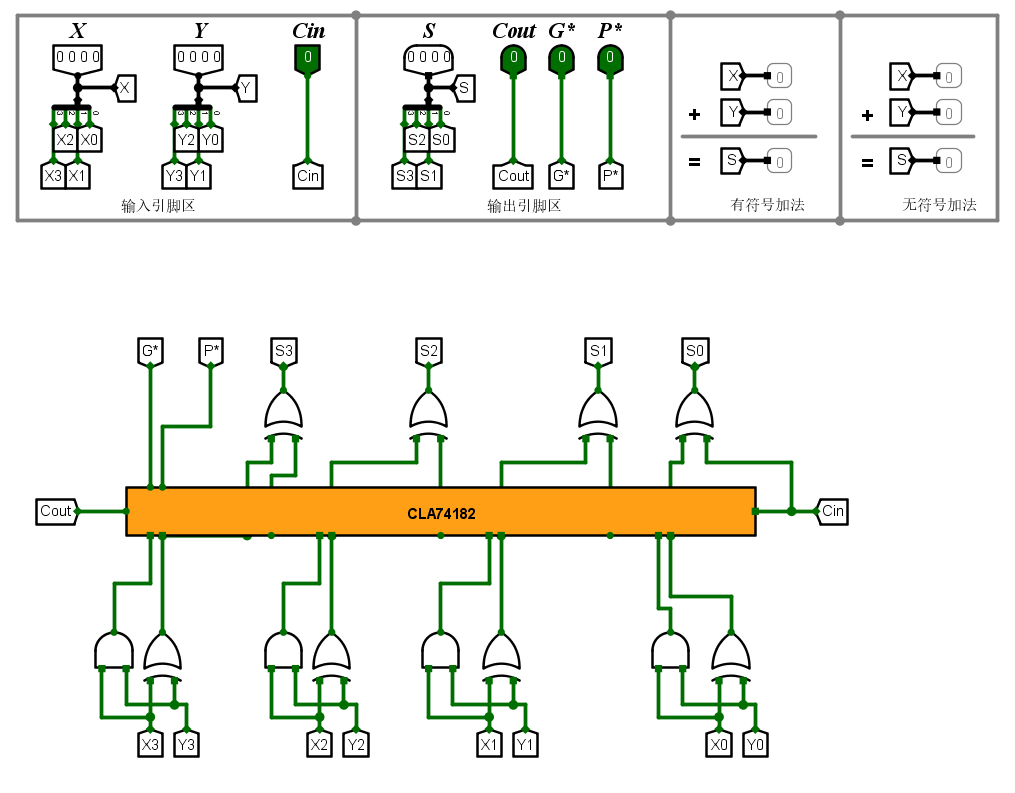
1. 输入：1111 1110 0 输出：1111 10



## 4位快速加法器设计

### 设计思路

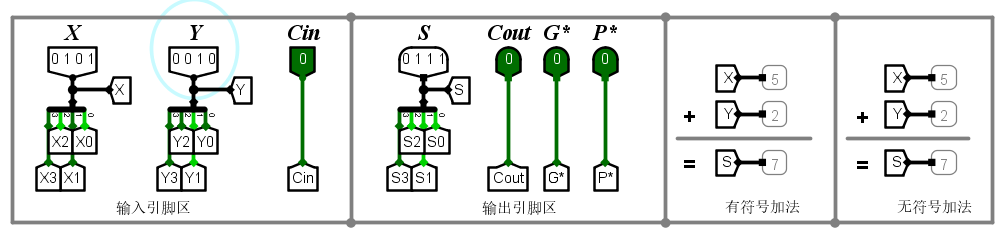
根据进位生成函数和进位传递函数，可以得到公式：⊕。首先用X、Y的各位计算出和，然后通过CLA74182计算出各位的进位，将其与进行异或运算即可以得到结果。电路设计如下图所示：



### 实验结果

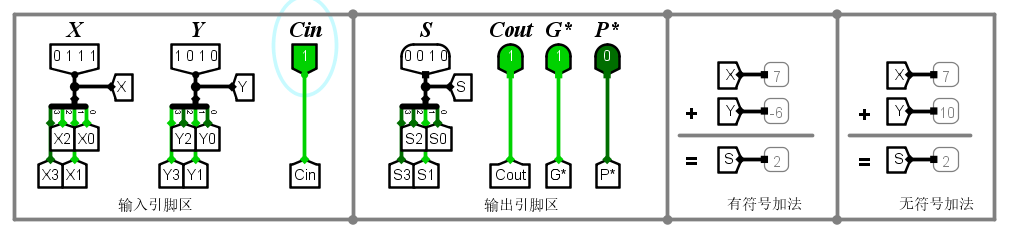
1. X=0101，Y=0010，Cin=0

S=0111，Cout=0，G\*=0，P\*=0



1. X=0111，Y=1010，Cin=1

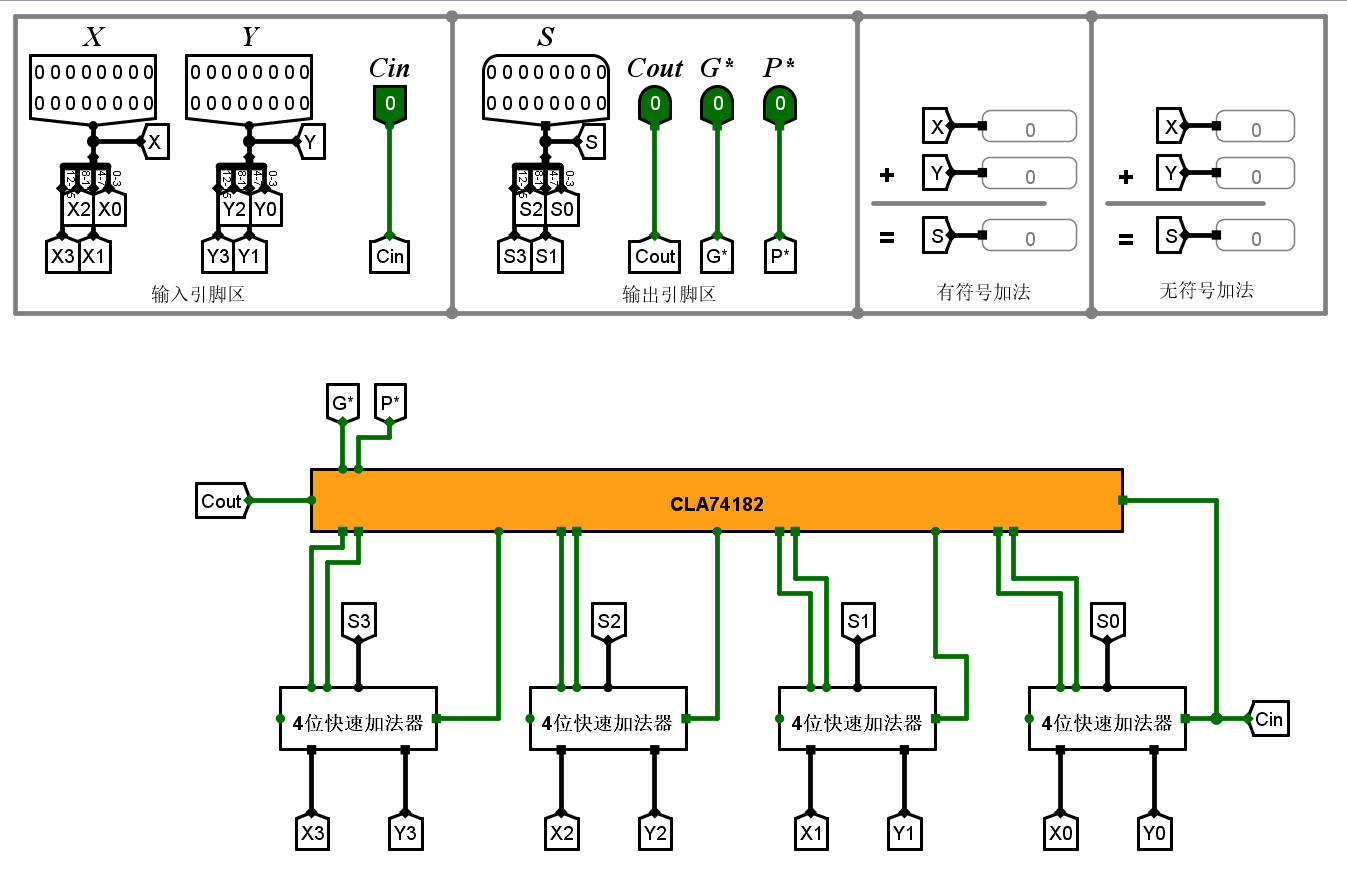
S=0010，Cout=1，G\*=1，P\*=0



## 16位快速加法器设计

### 设计思路

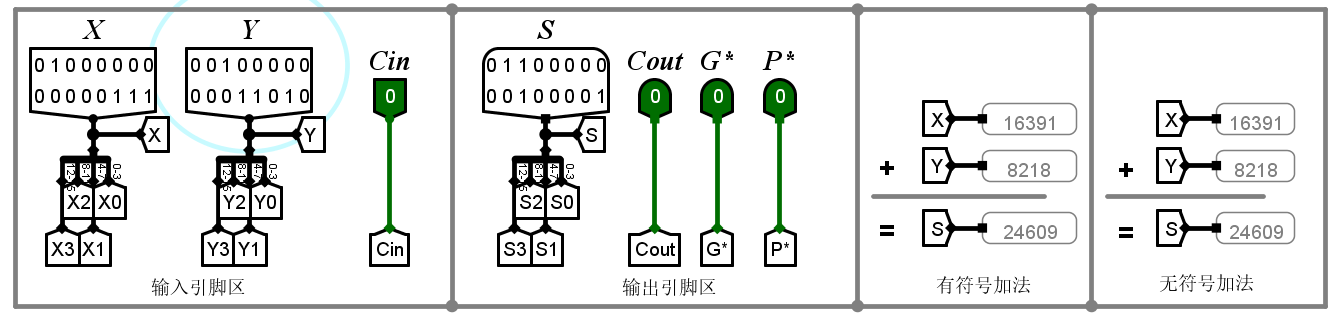
将4个4位快速加法器输出的成组进位生成、传递函数G\*和P\*及Cin连接到CLA74182的输入端，即可先行产生、、、4个进位信号。再将对应信号连接到相应的快速加法器的进位输入端即可构成16位组内并行进位、组间并行进位的快速加法器。电路设计如下图所示：



### 实验结果

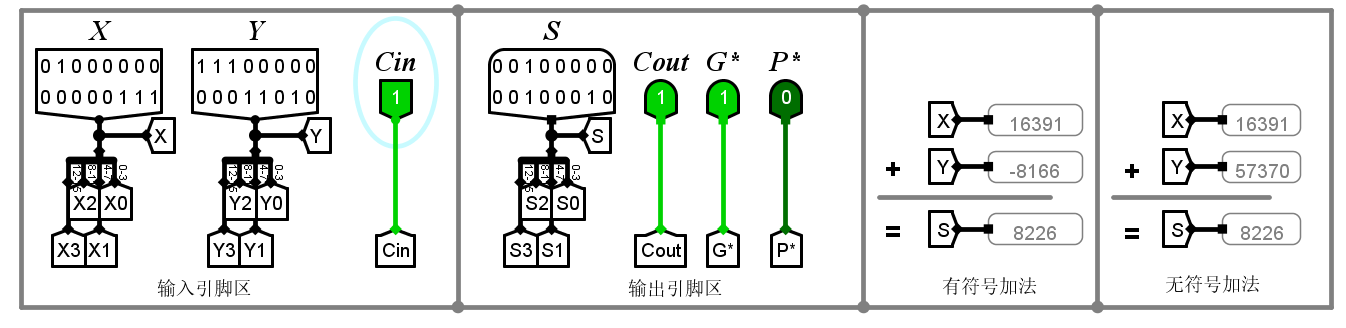
1. X=0100000000000111，Y=0010000000011010，Cin=0

S=0110000000100001，Cout=0，G\*=0，P\*=0



1. X=0100000000000111，Y=1110000000011010，Cin=1

S=0010000000100010，Cout=1，G\*=1，P\*=0



## 32位快速加法器设计

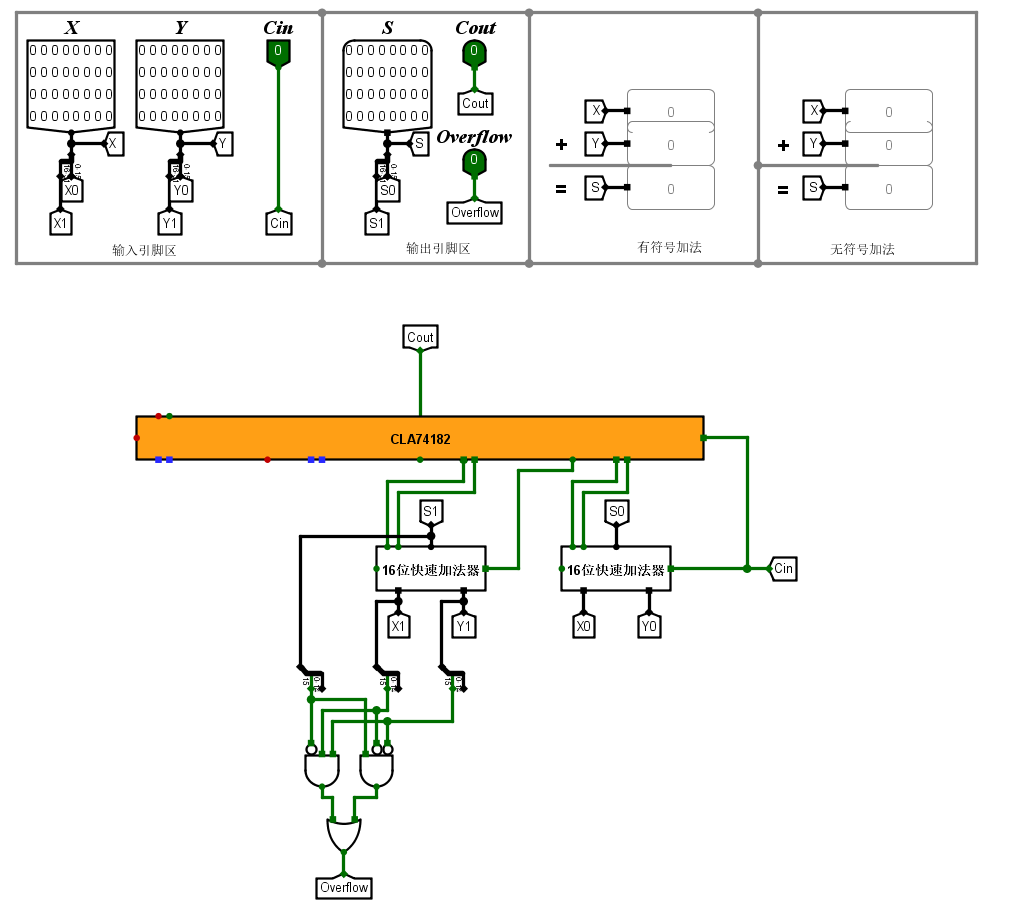
### 设计思路

加法功能的实现与16位快速加法器类似，用两个16位快速加法器与先行进位电路实现32位快速加法器。难点在于溢出判断，由于该电路无法单独获取最高数据位的进位，所以不能使用之前的方法进行溢出判断。此处根据操作数和运算结果的符号位是否一致来进行检测。

设Xf, Yf分别为两个操作数的符号位，Sf为结果的符号位，V为溢出标志位，V=1时即表示溢出，那么就有逻辑表达式：



通过分线器可以分别获取两个操作数和结果的符号位，经逻辑电路进行溢出判断。电路设计如下图所示：



### 实验结果

1. X=01111000000000000000001000000100

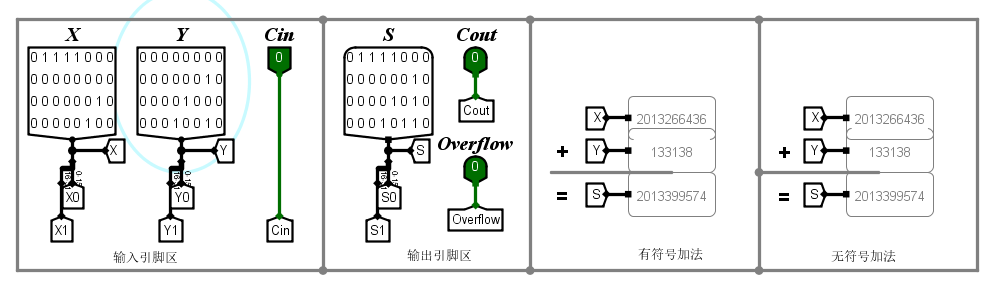
Y=00000000000000100000100000010010

Cin=0

S=01111000000000100000101000010110

Cout=0

Overflow=0



1. X=01111000000000000000001000000100

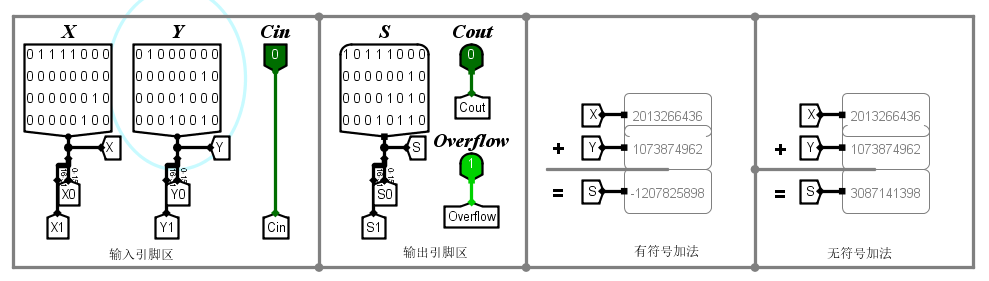
Y=01000000000000100000100000010010

Cin=0

S=10111000000000100000101000010110

Cout=0

Overflow=1



## 32位MIPS运算器设计

### 设计思路

表 1芯片引脚与功能描述

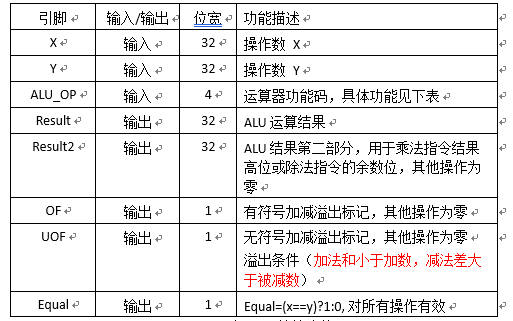
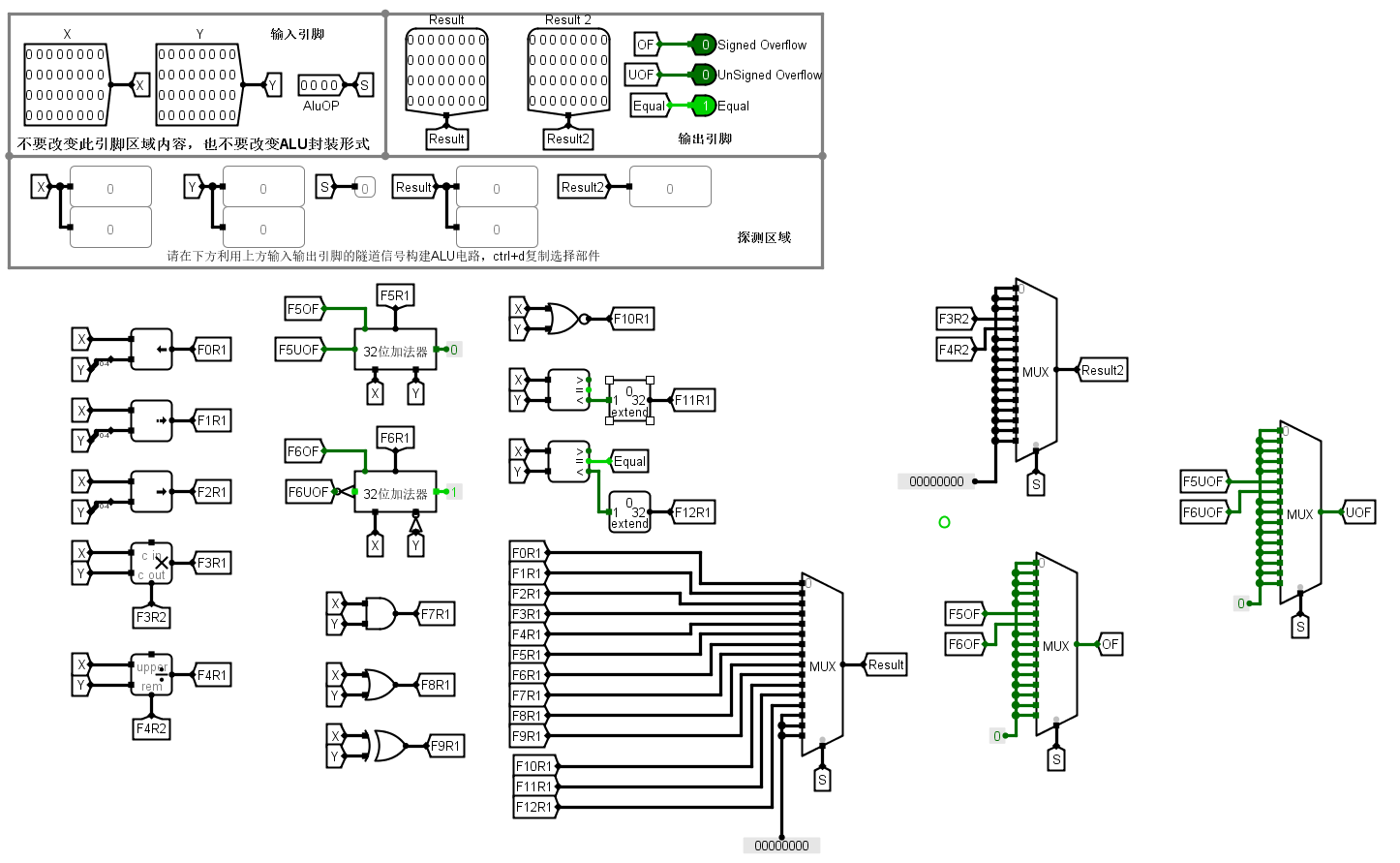


表 2运算符功能

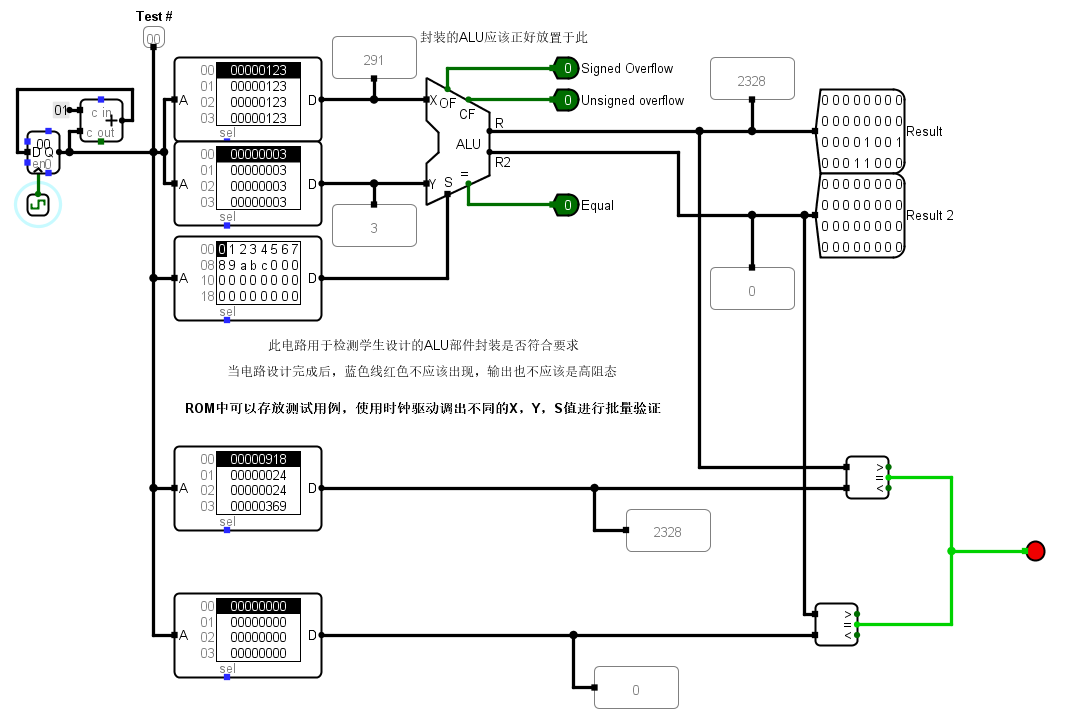


根据上表，分别实现各个功能。使用数据选择器Multiplexer，将各个功能得到的结果连接到输入端，将AluOP连接到选择端实现运算器功能控制。难点在于对于32位加减法要分别进行有符号溢出判断和无符号溢出判断，由于在32位快速加法器实验中已经实现了对有符号溢出的判断，而对于无符号数加法，如果最高位产生进位即出现溢出，所以仅需判断Cout是否为1；而对于无符号减法则相反，如果最高位不产生进位则溢出；在判断X和Y的大小时，要注意比较器得到的结果是1位，而输出的结果要求是32位，所以需要进行位扩展。电路设计如下图所示：

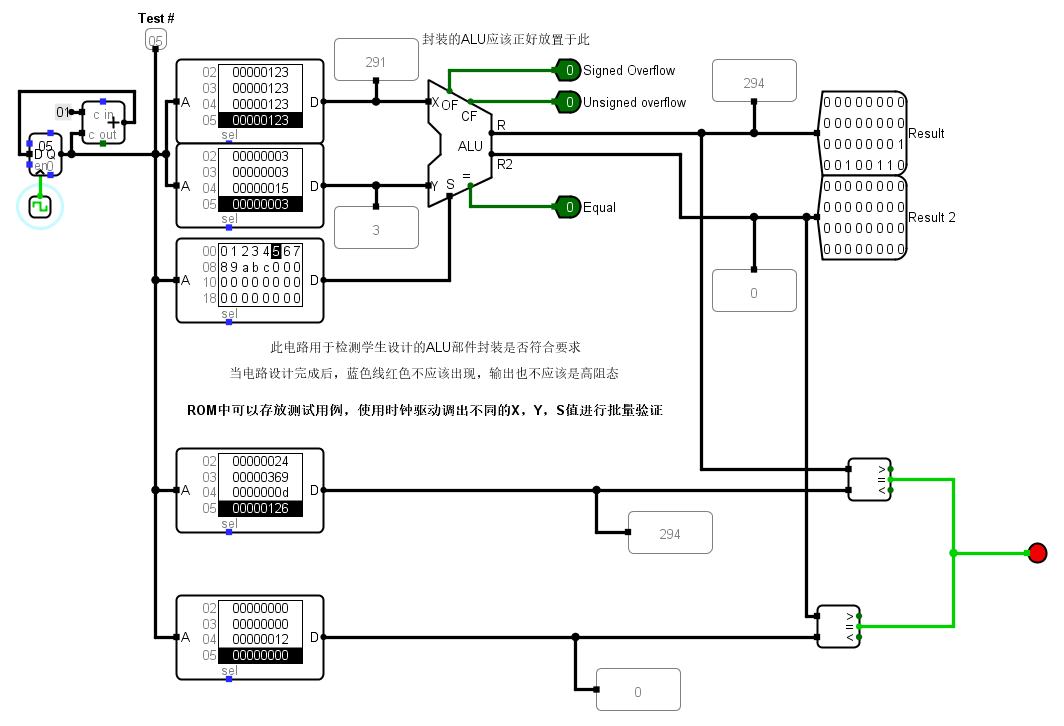


### 实验结果

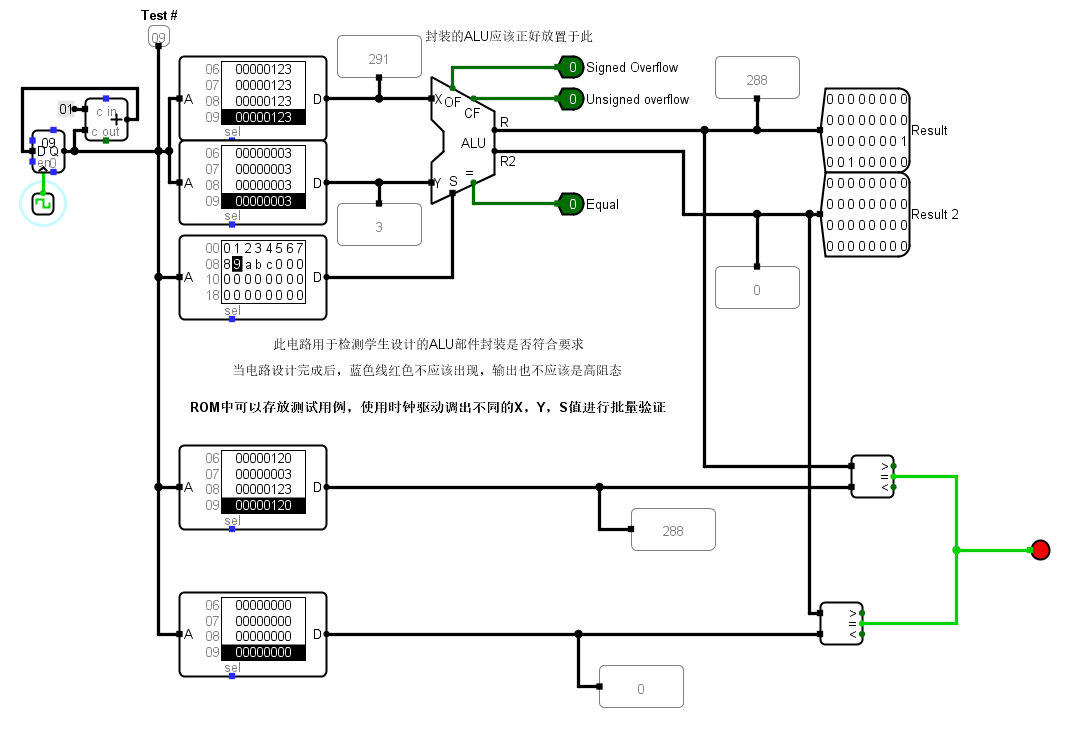
运行运算器测试电路，12个功能都通过测试，如下图所示（仅展示部分）：



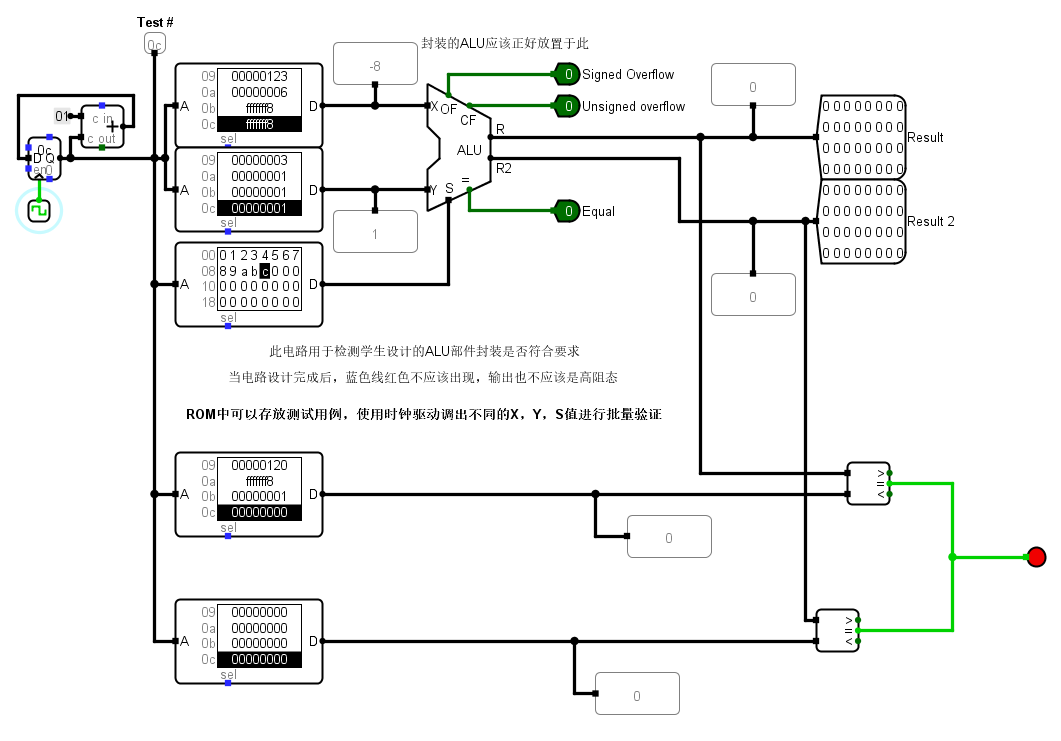
功能1



功能5



功能9



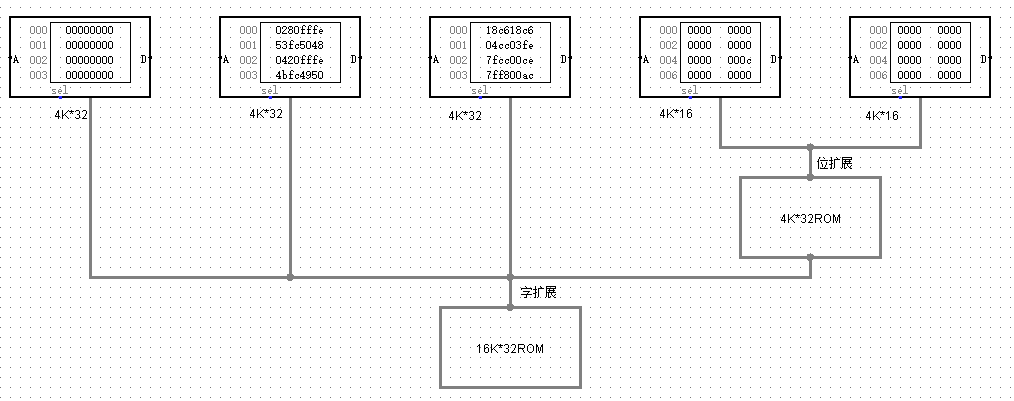
功能12

# 存储系统综合实验

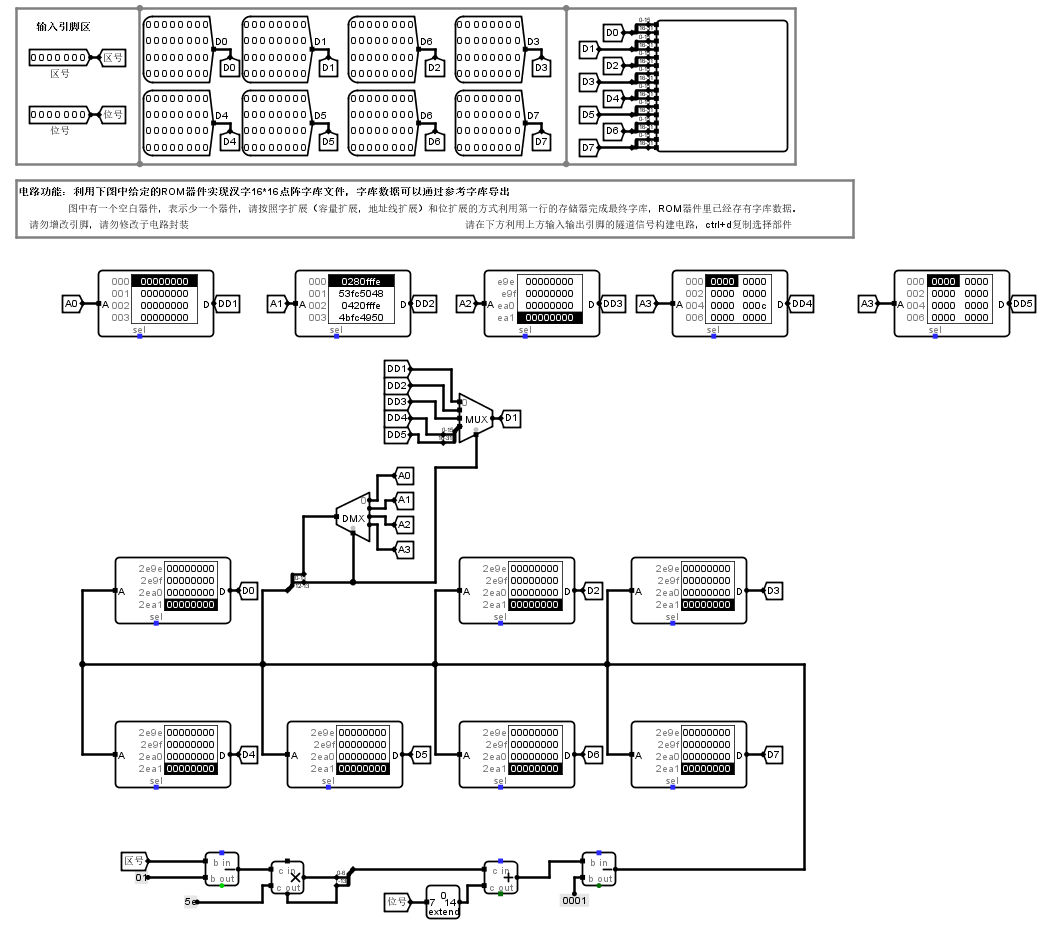
## 存储扩展实验

### 设计思路

为了一次获得在16\*16点阵屏上表示单个汉字所需的256位点阵码，我们需要8个16k\*32位的ROM进行位扩展，每次使用同一个存储器偏移地址同时对8个32位ROM进行寻址，这也是参考电路的实现思路。现在我们有2个4K\*16位ROM、3个4K\*32位ROM和7个16K\*32位ROM，与参考电路相比，我们缺少一个16K\*32位ROM，多了2个4K\*16位ROM和3个4K\*32位ROM。所以可以利用字扩展（容量扩展，地址线扩展）和位扩展，将现有的2个4K\*16位ROM和3个4K\*32位ROM组织成一个16K\*32位ROM。



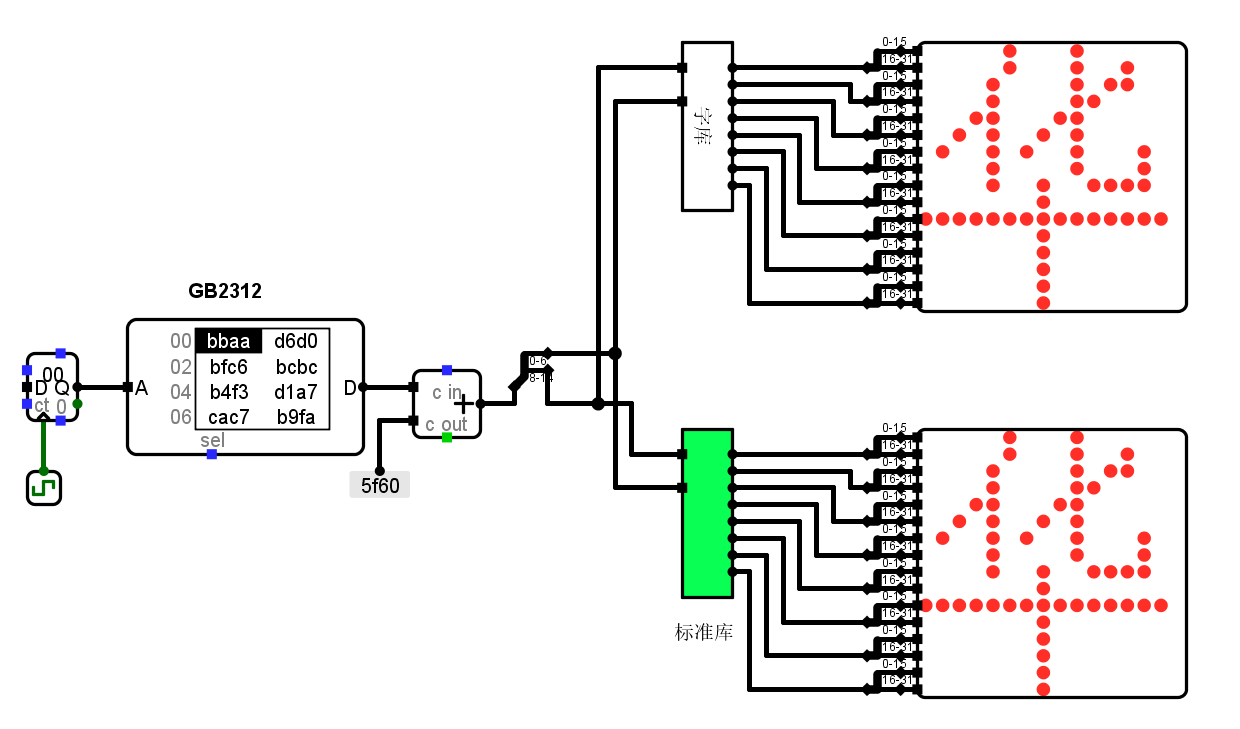
电路设计如下图所示：

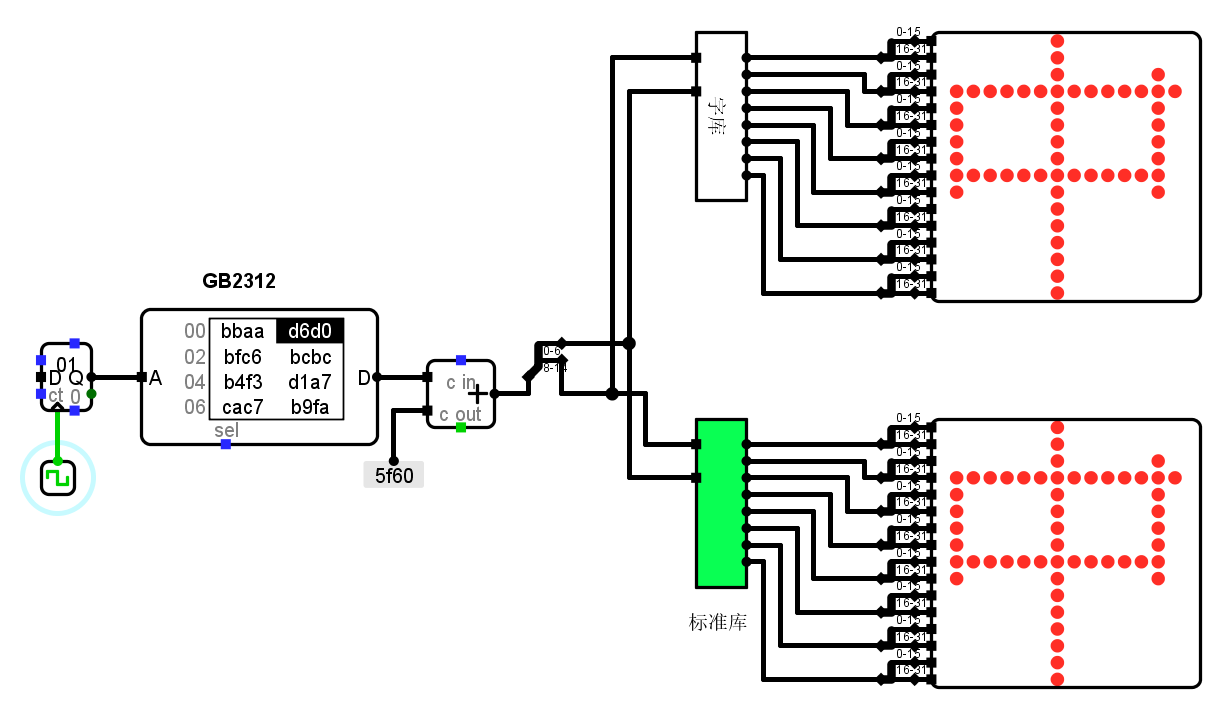


将14位地址的高两位作为片选信号连接到多路分配器的选择端以此来确定3个4K\*32位ROM和由2个4K\*16位ROM位扩展形成的ROM的地址；将片选信号作为数据选择器的选择端，2个4K\*16位ROM的数据通过分线器合并成32位的数据实现位扩展，以此来实现字扩展。

### 实验结果

运行字库测试电路，可见字库电路输出的汉字和标准库输出的汉字一致，如下图所示（仅展示部分）：





## MIPS寄存器文件设计

### 设计思路

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 |

根据实验要求，只需实现4个寄存器（0-3号），0号寄存器恒为0。

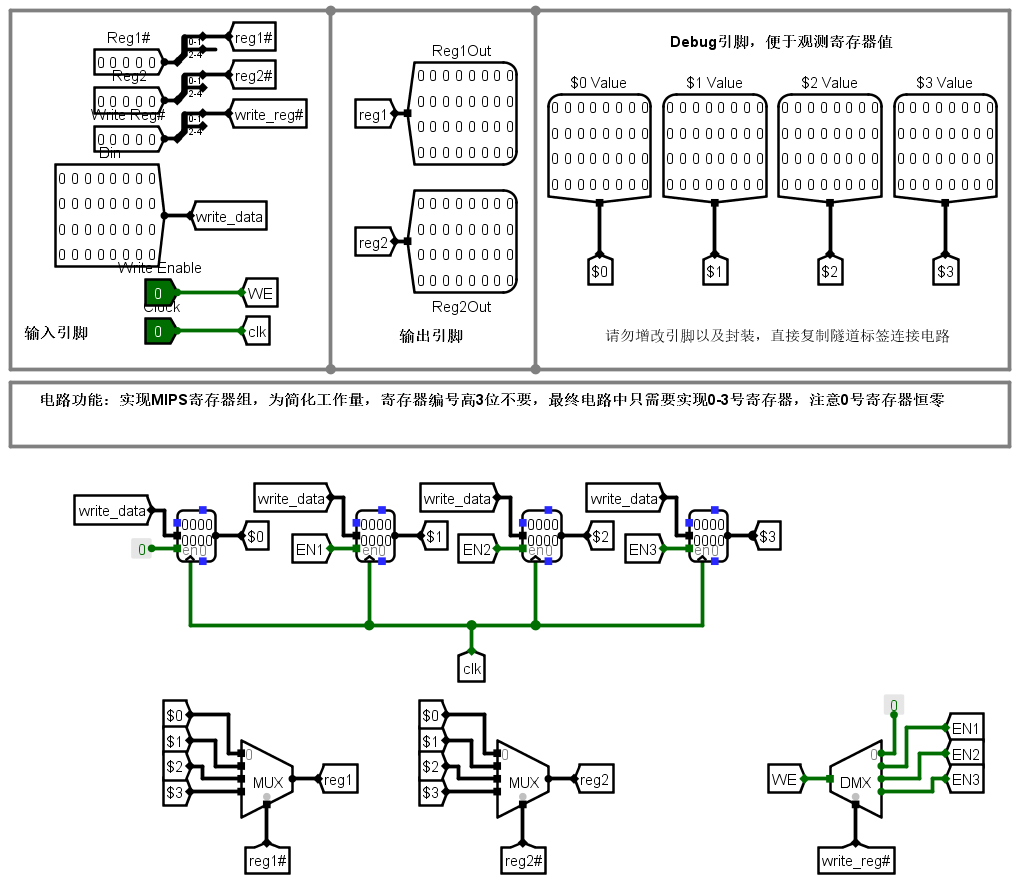
1. 如何才能实现只将编号为reg1#和reg2#的寄存器的值输出到reg1和reg2输出引脚呢？

利用多路复用器（multiplexer），多路复用器可以通过select bits选择多个输入中的一个进行输出。因此，我们可以将四个寄存器的输出引脚同时接到多路复用器的输入引脚上，将reg1#和reg2#作为多路复用器的select bits，从而选择指定输入进行输出。

1. 如何才能实现只将Din引脚输入的数据写入到编号为write\_reg#的寄存器中呢？

利用解复用器（demultiplexer），解复用器可以将输入只通过select bits指定的线路输出。因此，我们可以将write\_data同时连接到四个寄存器的data引脚（由于0号寄存器恒为0，可以将en置0），将we引脚作为解复用器的输入，将write\_reg#作为解复用器的select bits，通过解复用器将要写入数据的寄存器的enable引脚置1，下个时钟信号到来时，只有该写入寄存器的enable引脚为1，因此wrtie\_data数据只会写入到该寄存器中（其余寄存器enable引脚为0，时钟信号不会触发寄存器写入），从而实现寄存器选择写入。

电路设计如下图所示：

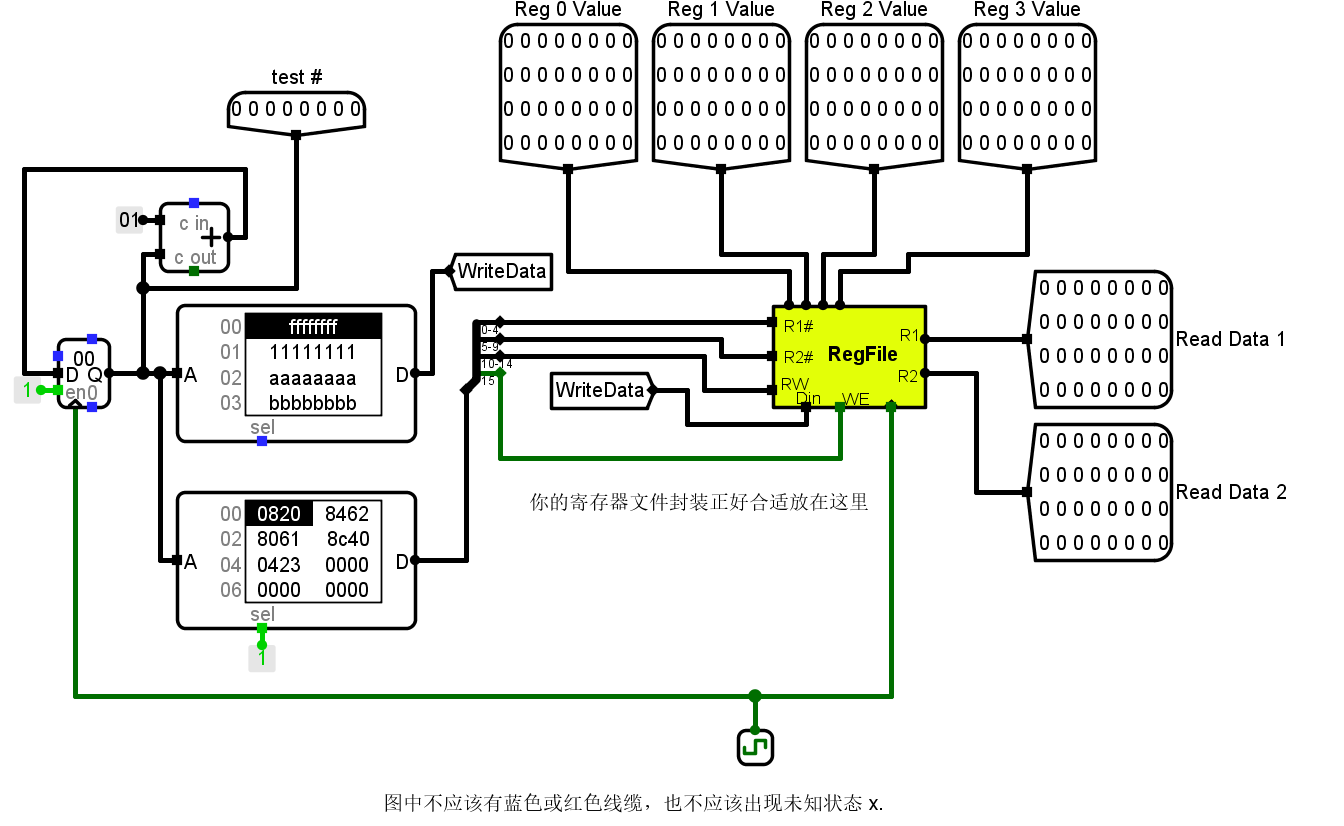


### 实验结果

导入测试数据后，运行MIPS寄存器文件测试电路，如下图所示：

注意：时钟信号不直接影响R1和R2，R1和R2始终为当前存储器输出指令指定的编号为R1#和R2#的寄存器值。

1. 执行指令前



1. 读0号和1号寄存器

R1# = 0

R2# = 1

RW = 2

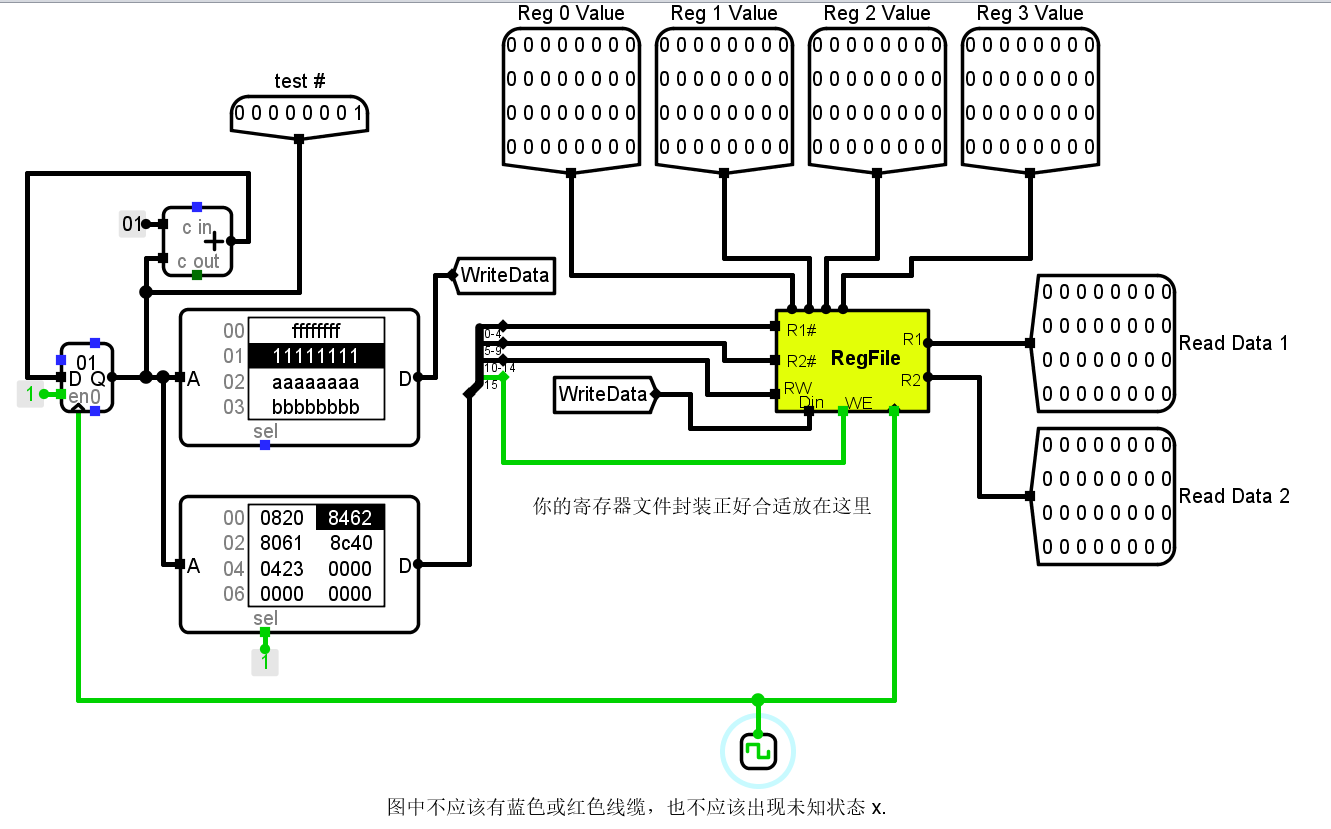
Din = FFFFFFFF

WE = 0

对应数据存储器值：FFFFFFFF

对应指令存储器值：0820

执行完该条指令之后，结果如下图所示：



1. 读2和3号寄存器 ，写1号寄存器

R1# = 2

R2# = 3

RW = 1

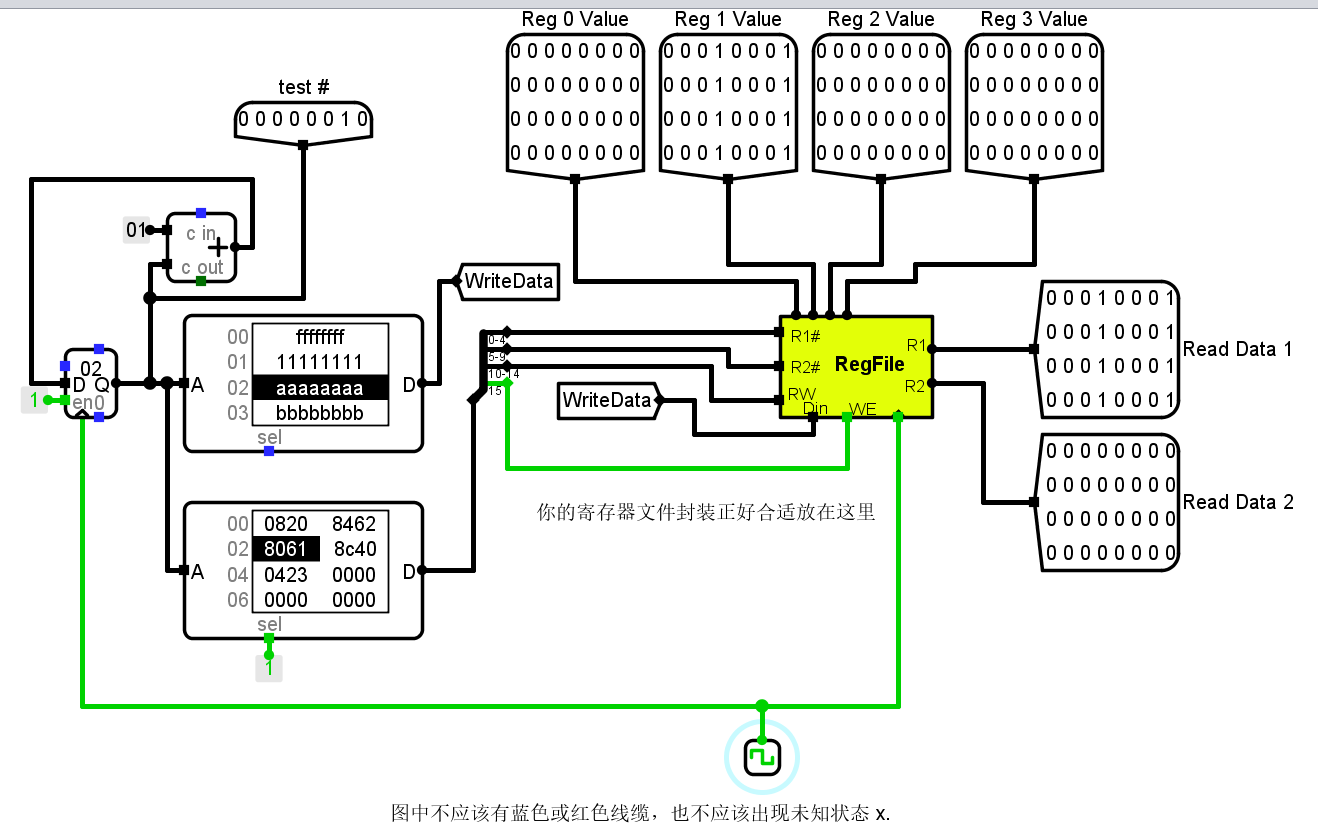
Din = 11111111

WE = 1

对应数据存储器值：11111111

对应指令存储器值：8462

执行完该条指令之后，结果如下图所示：



1. 读1号寄存器和3号寄存器，写0号寄存器

R1# = 1

R2# = 3

RW = 0

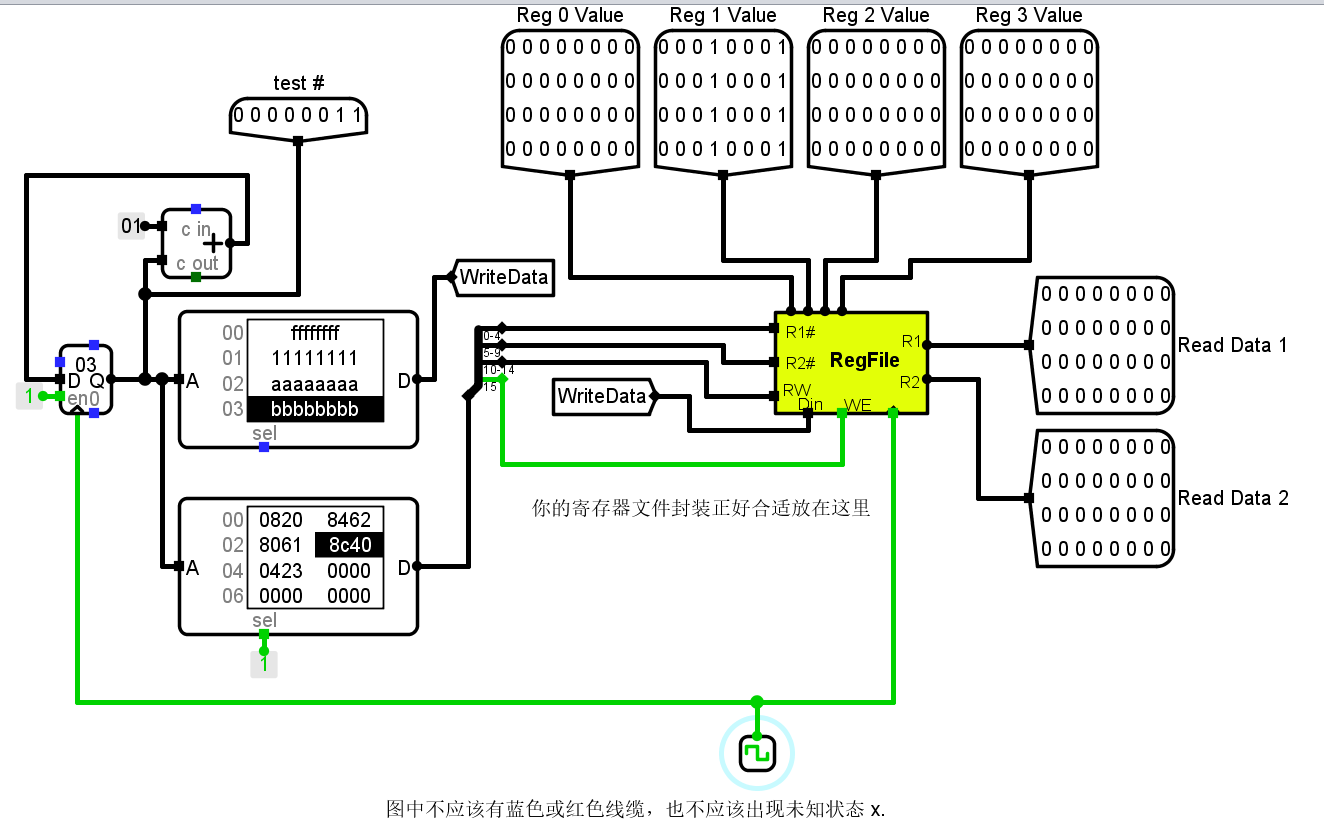
Din = AAAAAAAA

WE = 1

对应数据存储器值：AAAAAAAA

对应指令存储器值：8061

执行完该条指令之后，结果如下图所示：



1. 读0号寄存器和2号寄存器，写3号寄存器

R1# = 0

R2# = 2

RW = 3

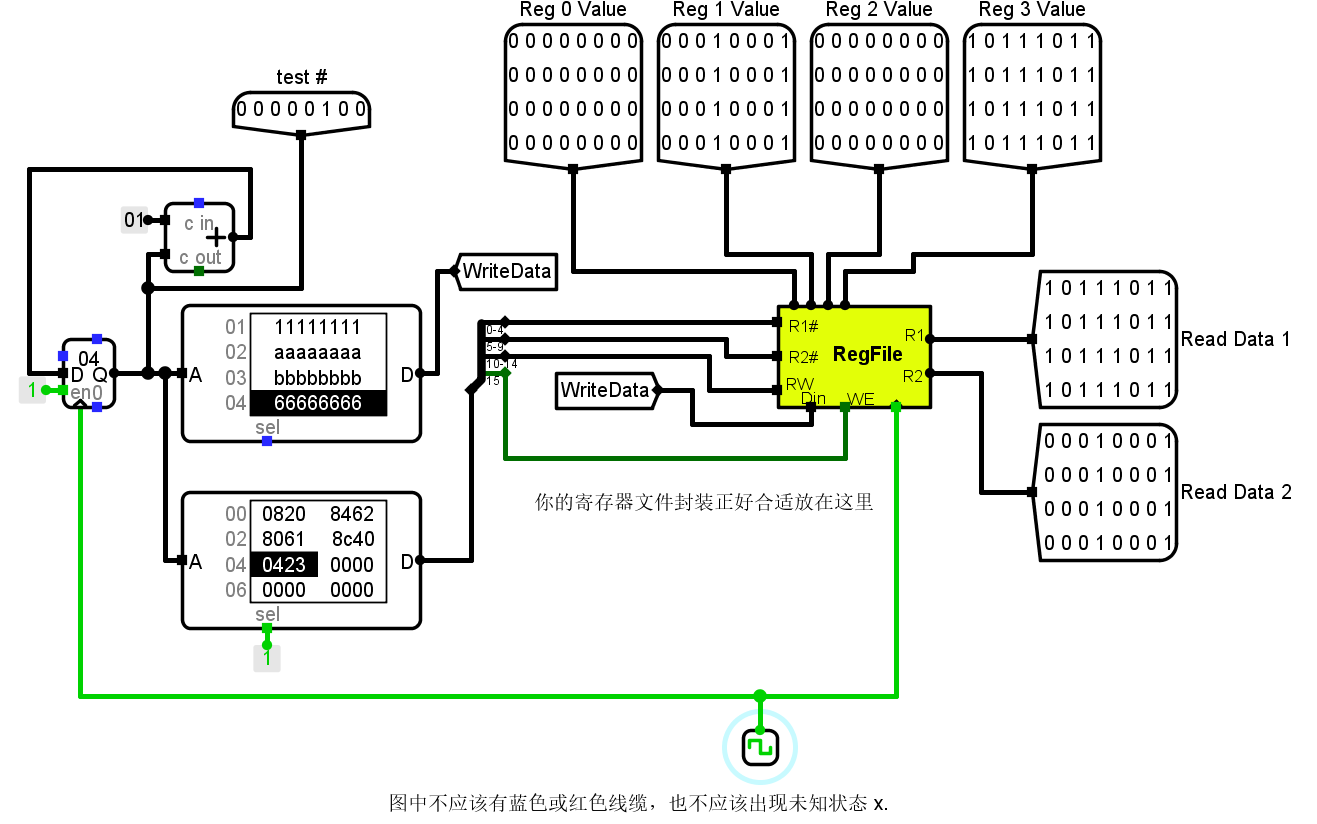
Din = BBBBBBBB

WE = 1

对应数据存储器值：BBBBBBBB

对应指令存储器值：8C40

执行完该条指令之后，结果如下图所示：



1. 读3号寄存器和1号寄存器

R1# = 3

R2# = 1

RW = 1

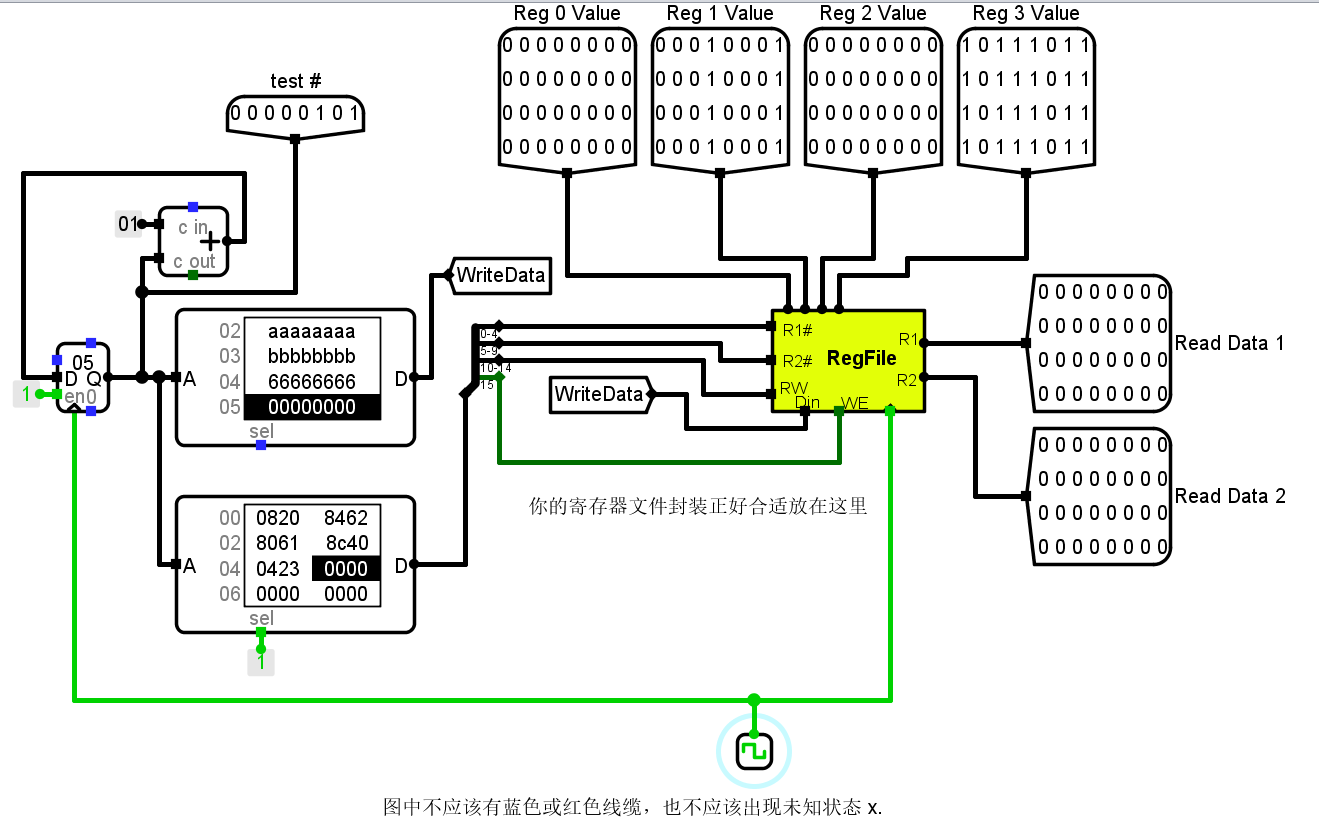
Din = 66666666

WE = 0

对应数据存储器值：66666666

对应指令存储器值：0423

执行完该条指令之后，结果如下图所示：



# 心得体会

在运算器组成实验中，我熟悉了 Logisim软件平台的使用，掌握了运算器基本工作原理与运算溢出检测的原理和实现方法，理解了有符号数和无符号数运算的区别和基于补码的加/减运算实现原理，并熟悉了运算器的数据传输通路。在存储系统综合实验中，我熟悉了 ROM、RAM 存储器的使用，掌握了存储器字扩展，位扩展的基本原理，并为 MIPS CPU 设计了功能部件---寄存器文件。

这几个实验环环相扣，难度适中，让我在动手完成实验内容的同时巩固了课堂所学，加深对于老师所讲授的理论知识的理解。在完成实验的过程中，我也遇到了一些问题，比如在存储扩展实验中，一开始始终无法使字库电路输出的汉字和标准库输出的汉字一致，后来经过不断排查错误发现是在字扩展中分线器的地址高位和地位搞反了，最后成功解决了问题。

感谢老师与助教精心准备的实验指导书和ppt讲解，详细清晰地阐明了实验的原理，对我完成实验产生了很大的帮助。

**最后，感谢黄浩老师一学期的辛苦教学！**