***2017***



**计算机组成原理 ·实验报告·**

j0242087[1]

|  |  |
| --- | --- |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS1501 |
| 学 号： | U201514477 |
| 姓 名： | 华龙 |
| 电 话： | 15927598966 |
| 邮 件： | [709603507@qq.com](mailto:709603507@qq.com) |
| 完成日期： | 2018-01-10 |



**目 录**

[1 数据表示实验 1](#_Toc503545522)

[1.1 设计要求 1](#_Toc503545523)

[1.2 方案设计 5](#_Toc503545524)

[1.3 实验步骤 9](#_Toc503545525)

[1.4 测试与分析 10](#_Toc503545526)

[2 CPU设计实验 13](#_Toc503545527)

[2.1 设计要求 13](#_Toc503545528)

[2.2 方案设计 14](#_Toc503545529)

[2.3 实验步骤 17](#_Toc503545530)

[2.4 故障与调试 21](#_Toc503545531)

[2.5 测试与分析 22](#_Toc503545532)

[3 总结与心得 24](#_Toc503545533)

[3.1 实验总结 24](#_Toc503545534)

[3.2 实验心得 24](#_Toc503545535)

[**参考文献** 25](#_Toc503545536)

# 数据表示实验

## 设计要求

### 汉字编码

1. 设计国标码转区位码电路

输入：GB2312 16位国标码；输出：区号，行号（区号行号均从1开始计数），电路引脚定义如图1.1所示。

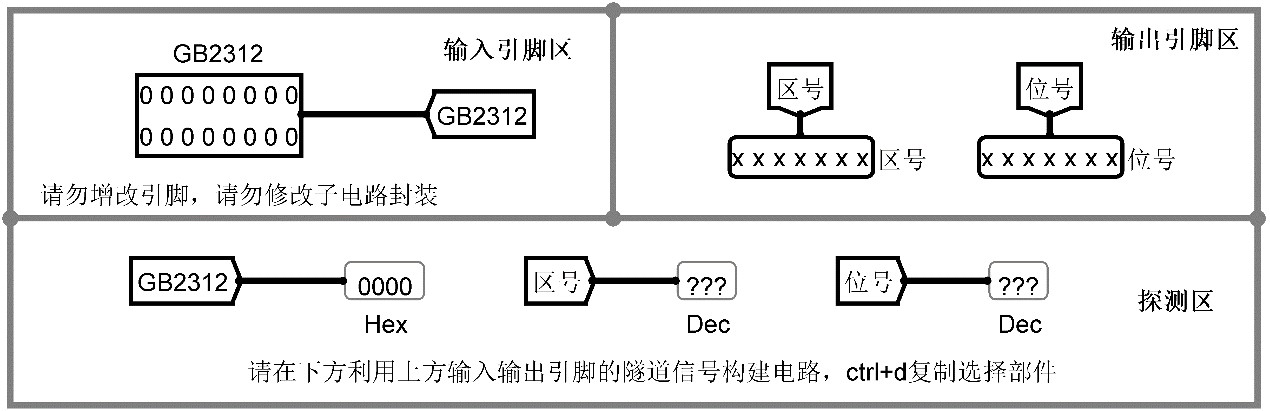


图1.1 汉字国标码转区位码电路引脚定义

1. 汉字GB2312编码实验

完成国标码到区位码的转换电路后，在汉字显示电路中进行测试，尝试在图1.2所示电路中的ROM存储器中存入100个成句的汉字，ROM存储器使用方法见Logisim参考手册。

该电路启用时钟仿真后可以依次显示ROM中的事先预存的所有汉字（时钟仿真快捷键：Ctrl+k）。

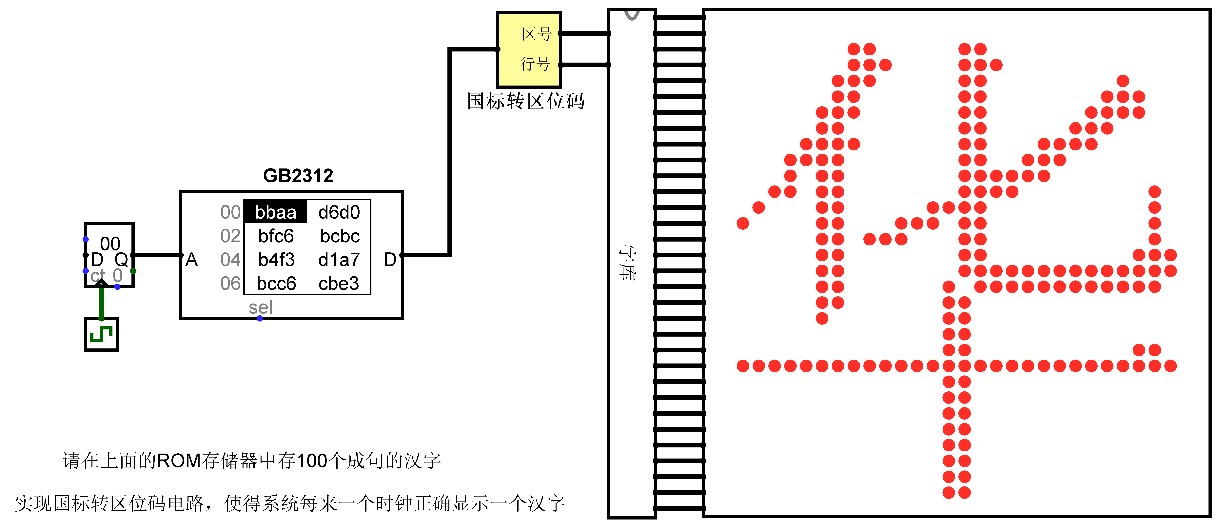


图1.2 汉字字模码显示电路

### 偶校验

1. 设计16位数据编码的偶校验编码电路

输入：16位原始数据；输出：17位校验码（16位数据位+1位校验位），输入输出引脚定义如图1.3所示。

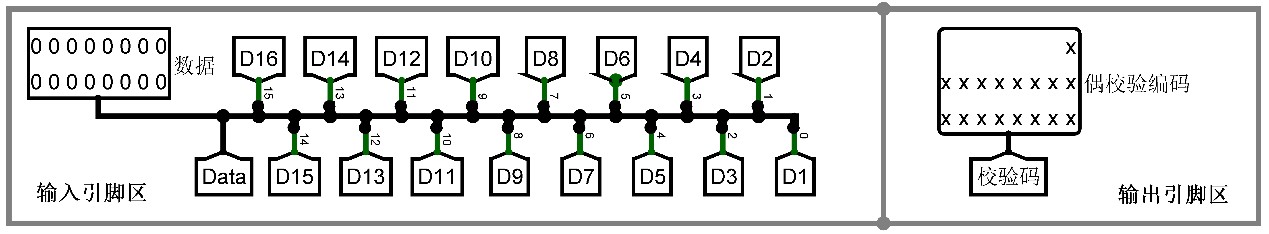


图1.3 偶校验编码电路引脚定义

1. 设计17位偶校验编码的检错电路

输入：17位校验码；输出：16位原始数据，1位检错位；输入输出引脚定义如图1.4所示。

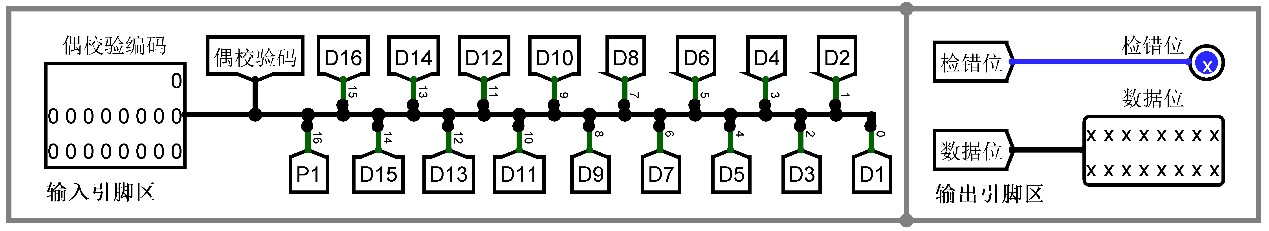


图1.4 偶校验解码电路引脚定义

1. 偶校验传输测试

在偶校验传输测试1电路中测试偶校验编解码电路的正确性，并观察数据传输过程中何时会出现误报情况，分析奇偶校验传输的性能，如果已经实现汉字显示模块，可直接使用偶校验传输测试2电路。

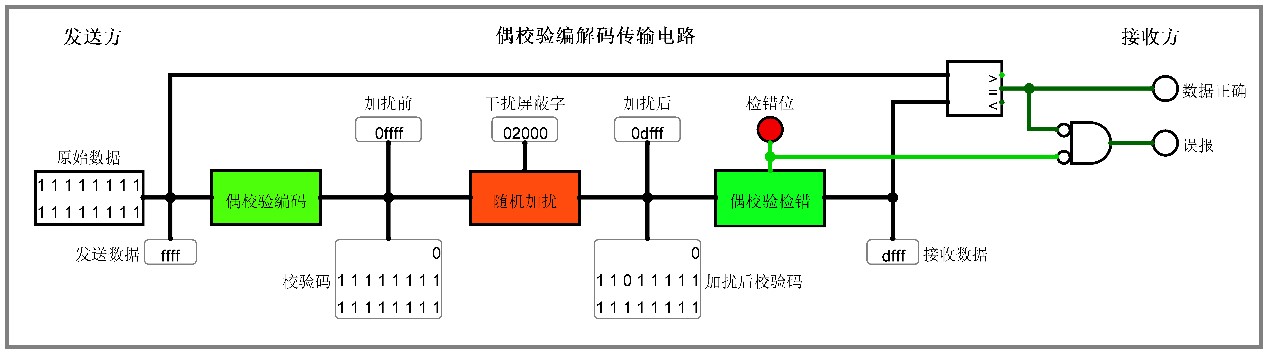


图1.5 偶校验传输测试电路1

在偶校验传输测试2电路中测试偶校验编解码电路实现是否正确，测该电路引入了汉字显示模块，可以直接显示接收端和发送端的编码的汉字，通过汉字显示可以很直观观察传输是否发生错误，从而观察采用偶校验进行数据传输时传输的可靠性，在logisim中可以使用Ctrl+T快捷键开启时钟自动仿真测试，具体测试电路如图1.6所示。

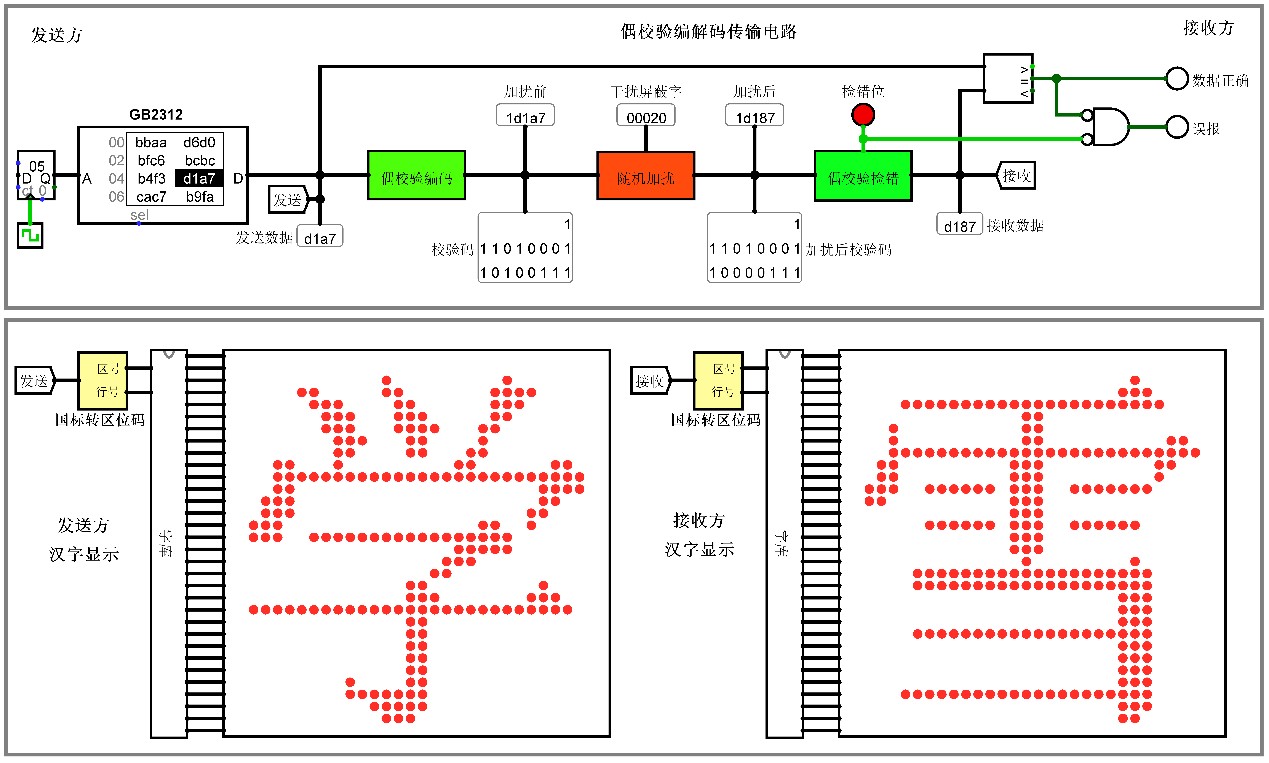


图1.6 偶校验传输测试电路2

### 海明校验

1. 设计16位数据编码的海明校验编码电路

输入：16位原始数据；输出：22位校验码（16位数据位+5位校验位），输入输出引脚定义如图1.7所示。

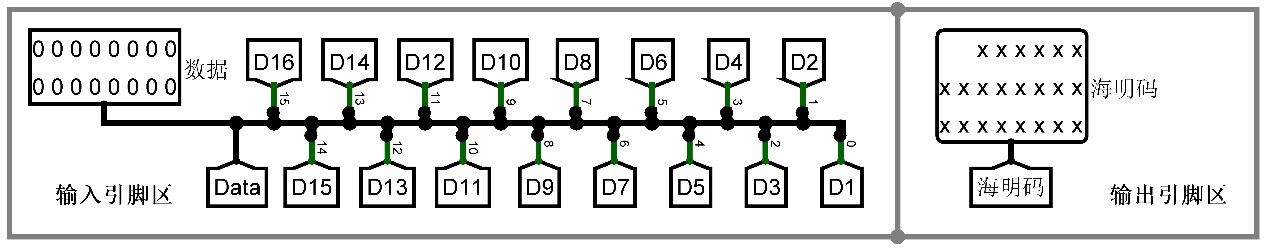


图1.7 海明编码电路引脚定义

1. 设计22位海明校验码的解码电路

输入：22位校验码；输出：16位原始数据，1位检错位；2位检错位；无错误状态位；输入输出引脚定义如图1.8所示。

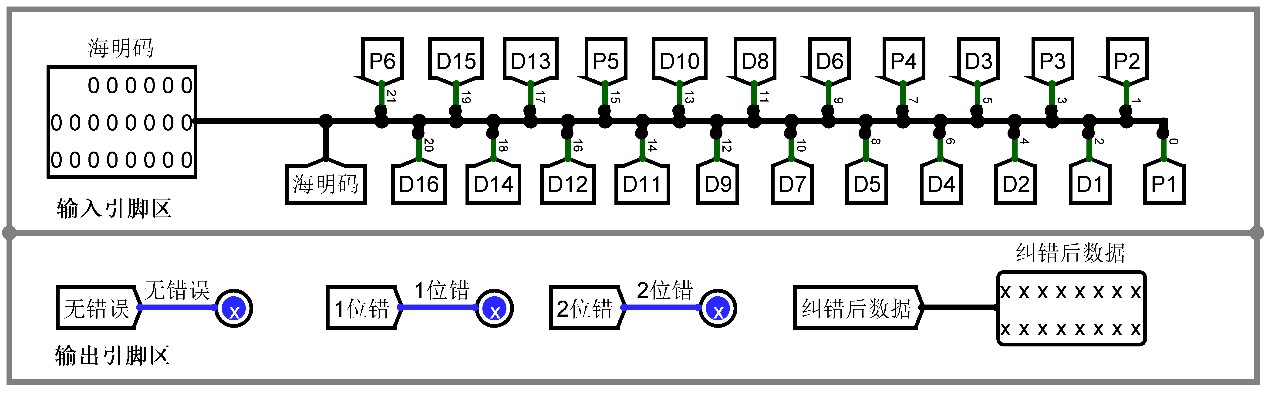


图1.8 海明解码电路引脚定义

1. 海明校验传输测试

在海明校验传输测试1子电路中测试海明校验编解码电路的正确性，注意随机干扰电路只能产生最多2位错误，具体测试电路如图1.9所示。

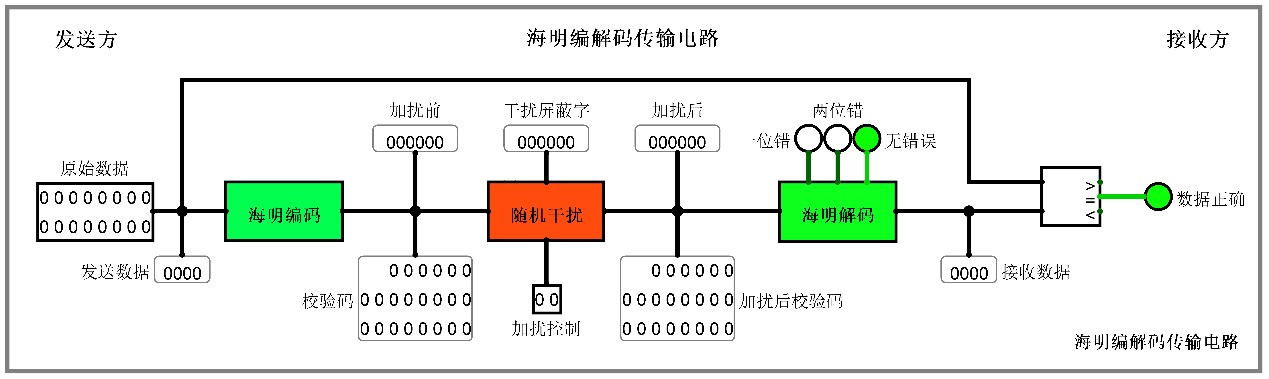


图1.9 海明编码传输电路1

在海明校验传输测试2子电路中测试海明校验编解码电路的正确性，该电路引入了汉字显示模块，可以直接显示接收端和发送端的编码的汉字，通过汉字显示可以很直观的看出海明纠错的效果，可以使用 Ctrl+T 快捷键开启时钟自动仿真测试，具体测试电路如图1.10所示。

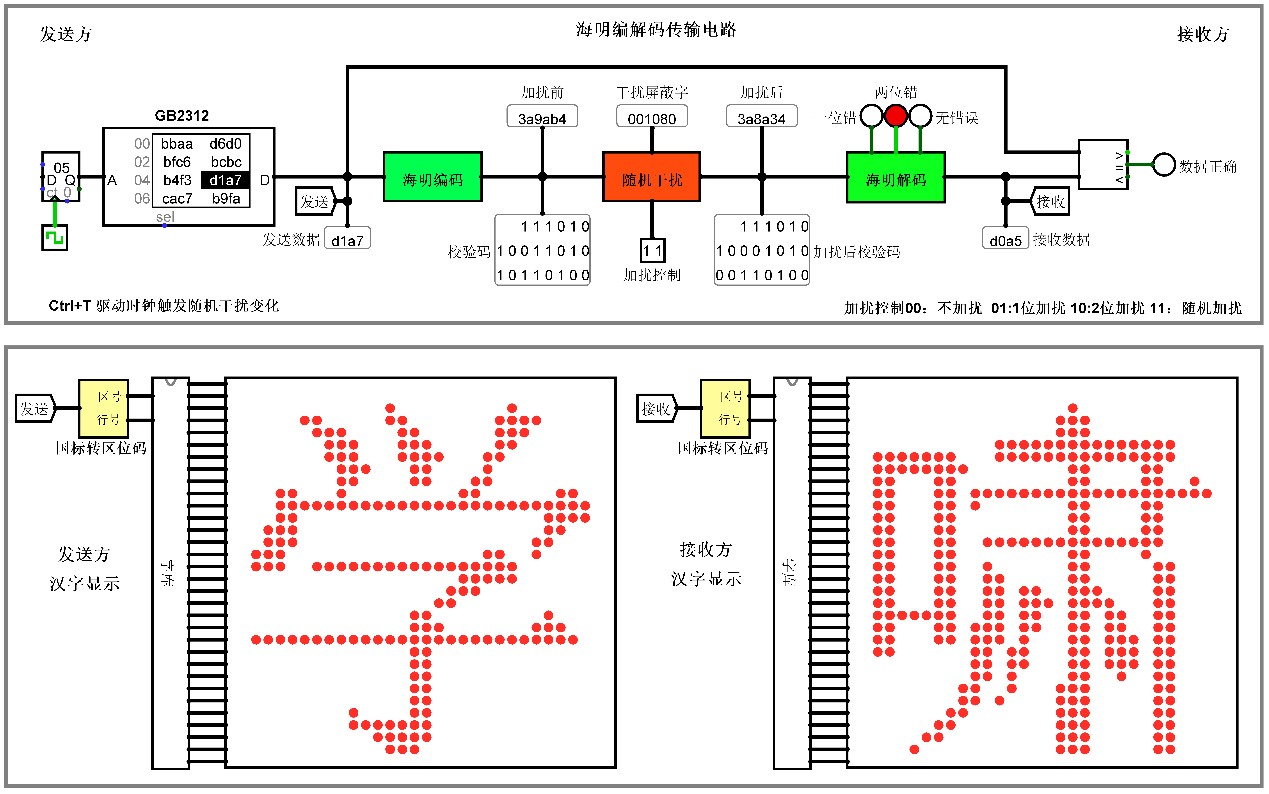


图1.10 海明编码传输电路2

1. 海明编码流水传输测试

图1.11中将海明编码传输过程分成了5个阶段（取数，编码，传输，解码，显示）类似 CPU指令流水线的处理过程，中间蓝色长条为流水接口部件（内部实际是若干寄存器，用于传输数据和控制信号），流水接口部件提供同步清零控制信号，试启用时钟自动仿真运行该电路，观察接收方接受到的信息，当发生两位错时，将会发生错误，尝试简单修改该电路，使得解码阶段出现两位错时，系统能自动重传对应编码（类似指令流水线中的分支跳转），从而使得该电路能正确传输所有数据。

适当修改图1.11中的电路，实现出现两位错的情况后自动重传对应编码。

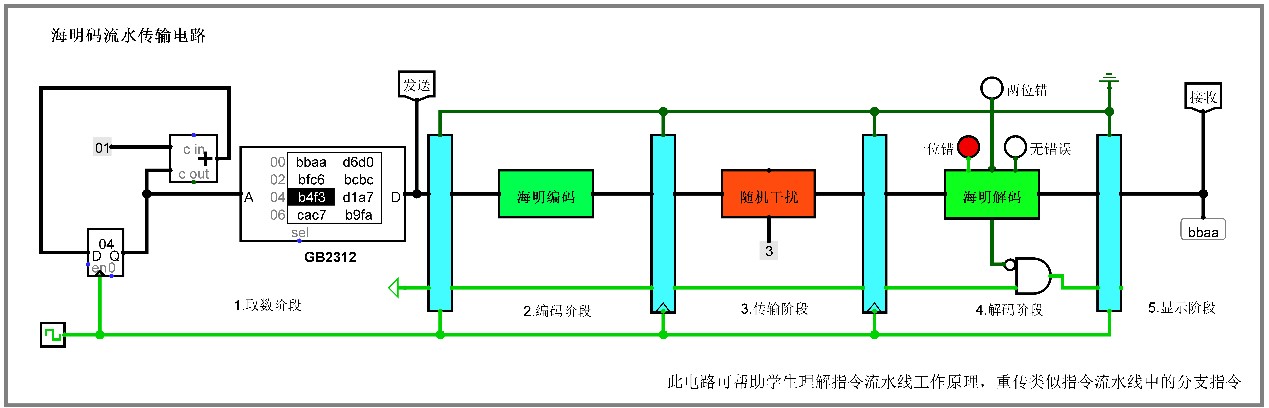


图1.11 海明编码流水传输电路

## 方案设计

### 国标码转区位码电路

国标码与区位码的转换关系是，16位的国标码减去2020H即可得到对应的区位码，所以电路设计如图1.12所示，用一个减法器将国标码减去2020H，在用一个分线器选出位号和区号即可。

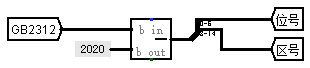


图1.12 总体结构图

### 16位数据偶检验编码电路

由偶校验的校验位是所有数据位相异或后的结果，而校验码前16位为数据位，第17位为校验位，所以用分线器将16位数据与1位校验合成17位的校验码即可，电路设计如图1.13所示。

### 17位偶检验编码检错电路

数据位输出即是偶检验码的原始数据位，而检错位由原始数据位相异后得到新校验位后，再与原检验位相异或得到的，电路设计如图1.14所示。检错位为1，则表示偶校验码有错误（可能原始数据位出错，也可能是校验位出错），检错位为0，则可以认为偶检验码没有错误。

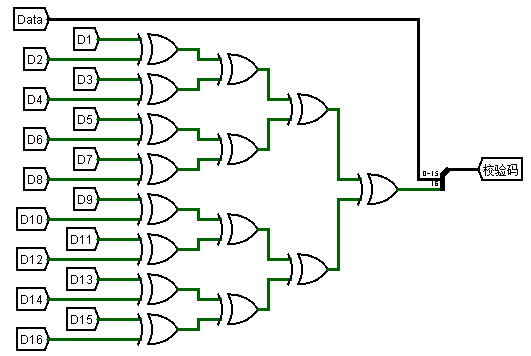


图1.13 16位数据偶检验编码电路

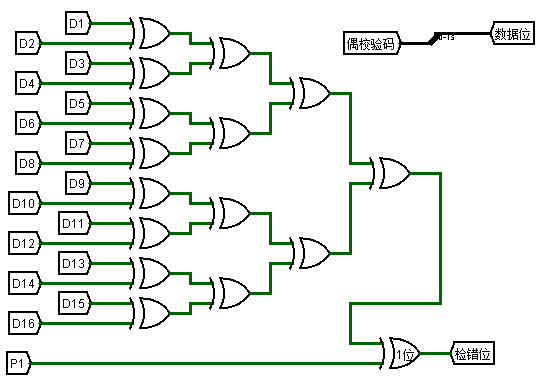


图1.14 17位偶检验码检错电路

### 16位数据海明校验编码电路

将16位数据编码为能检测并纠正1位错的海明校验码至少需要5位校验位，而要实现能检测两位错，必须在最后增加一位偶校验位，所以输出的校验码位宽为22位，其中，6位校验位所校验的数据分组情况见表1.1。

表1.1 校验位分组表



按照表1.1所示的分组，编码电路如图1.15所示。用分线器将相应的数据位选择出来做偶校验，得到P1-P5后，所有数据位和P1-P5做偶校验得到P6，最后将数据位和校验位合成海明码输出。

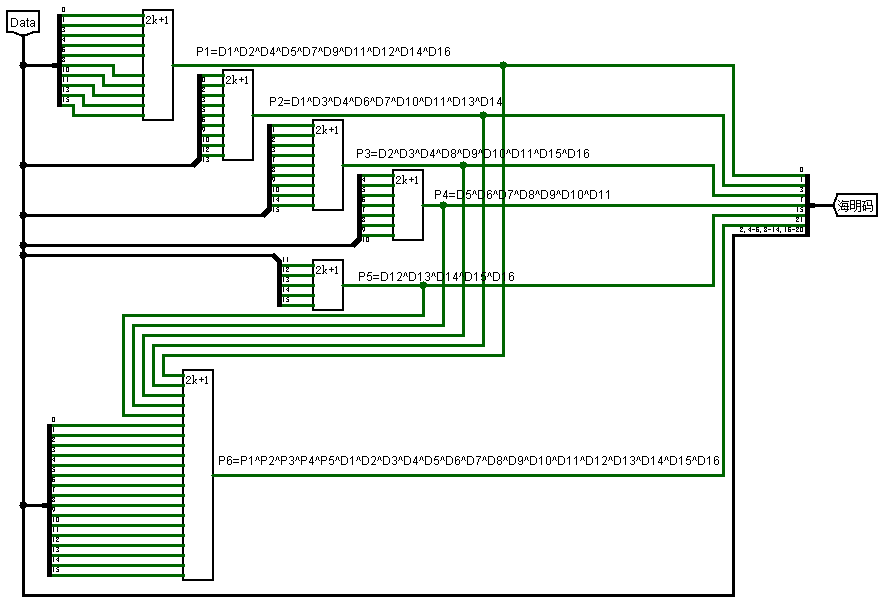


图1.15 16位数据编码的海明校验编码电路

### 22位海明校验码解码电路

由海明校验的原理可以得出，指错字G1-G6可由下面表达式表示：

G1=P1^D1^D2^D4^D5^D7^D9^D11^D12^D14^D16

G2=P2^D1^D3^D4^D6^D7^D10^D11^D13^D14

G3=P3^D2^D3^D4^D8^D9^D10^D11^D15^D16

G4=P4^D5^D6^D7^D8^D9^D10^D11

G5=P5^D12^D13^D14^D15^D16

G6=P6^P5^P4^P3^P2^P1^D1^D2^D3^D4^D5^D6^D7^D8^D9^D10^D11^D12^D13^D14^D15^D16

在logisim中画出相应电路，电路设计如图1.16所示。图中用几个奇校验模块代替了大部分异或门做偶校验操作，得到G1-G5后，通过一个译码器将G5G4G3G2G1指出的错误位表示，并让其与原海明码做逐位相异或，而得到纠错后的数据；而当G5-G1逗为0时，如果G6也为0，说明无错误发生或有两个以上且是偶数个错误发生，如果G6为1，说明有奇数个错误发生。因此，得到检错位和无错误状态位的电路如图1.16中所示。

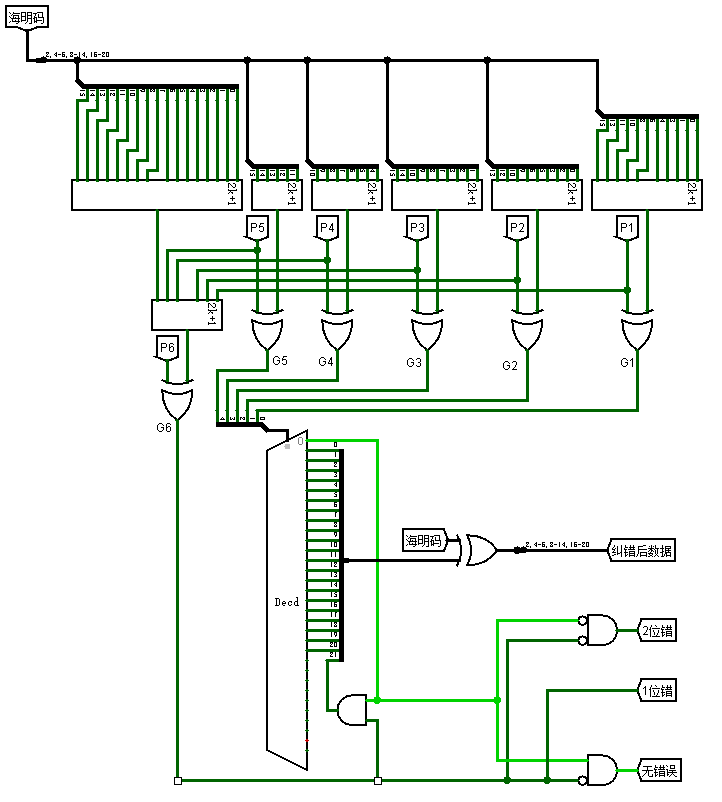


图1.16 22位海明校验码解码电路

### 流水传输2位错重传

流水线传输实现重传的电路如图1.17所示，没有发生错误或只发生一位错时，流水控制单元的同步清零端接地，传输状态输入都为1，流水线继续传输工作，当发生两位错时，最后一级流水线的传输状态变为0，同步清零端置1，于此同时，通过多选择器，ROM的地址计数器由正常的加一，变为减四，从而重新传输出错的数据。

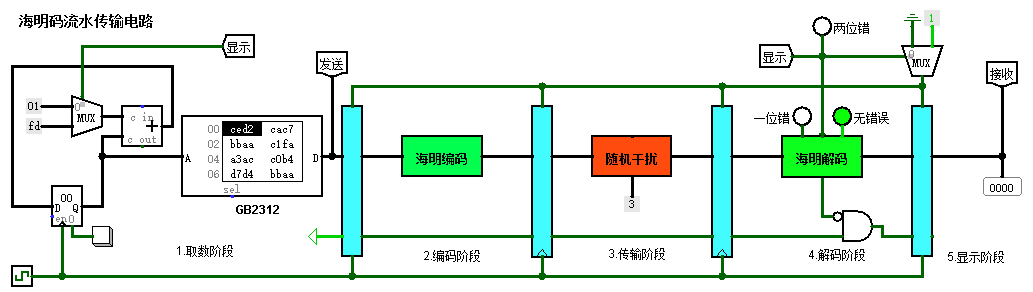


图1.17 流水传输2位错重传电路

## 实验步骤

### 汉子编码

1. 设计国标码转区位码电路

按照图1.12所示的电路图，在logisim中画出该电路即可。

1. 汉字GB2312编码实验

1. 把一段成句的文字转换成国标码表示，存放在一个文本文件中。

2. 在图1.2所示的电路图中，右键点击ROM，选择Edit Contents选项，然后将文本文件中的国标码粘贴到logisim的16进制编辑器中，最后保存。

3. 将计数器清零，按Ctrl+K启动时钟，观察显示的文字是否有误。

### 偶检验

1. 设计16位数据编码的偶校验编码电路

按图1.13所示的电路图画出电路，并检查功能是否正确。

1. 设计17位偶校验编码的检错电路

按图1.14所示的电路图画出电路，并检查功能是否正确。

1. 偶校验传输测试

打开偶校验传输测试1子电路，输入测试数据观察偶校验编码和解码功能是否正常。

### 海明校验

1. 设计16位数据编码的海明校验编码电路

按图1.15所示的电路图画出电路，用一个测试用例测试编码功能是否正确。

1. 设计22位海明校验码的解码电路

按图1.16所示的电路图画出电路，用一个测试用例测试编码功能是否正确。

1. 海明校验传输测试

打开海明传输测试1子电路，输入测试数据，随机干扰分别设成无错误、一位错和两位错，观察编码和解码电路功能是否正常。

1. 海明编码流水传输测试

按图1.17所示的电路图修改图1.11中的电路，启动时钟仿真看重传功能是否正确。

## 测试与分析

### 偶检验传输测试

输入的测试用例为1111111100000000，图1.18、图1.19和图1.20分别是检错电路检查出错误、检错电路没有检查出错误和数据传输正确时的测试结果，从结果来看，编码电路和解码电路功能都正常。

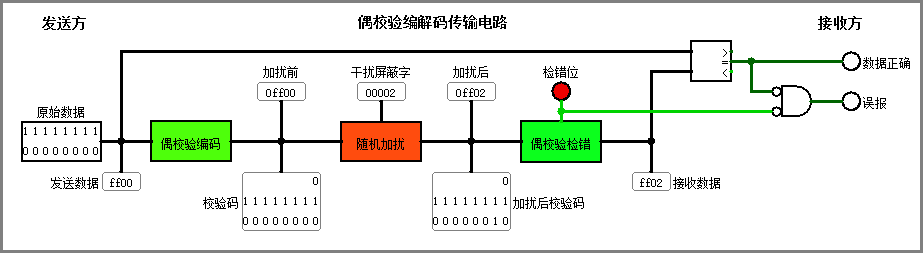


图1.18 偶检验传输检测到错误

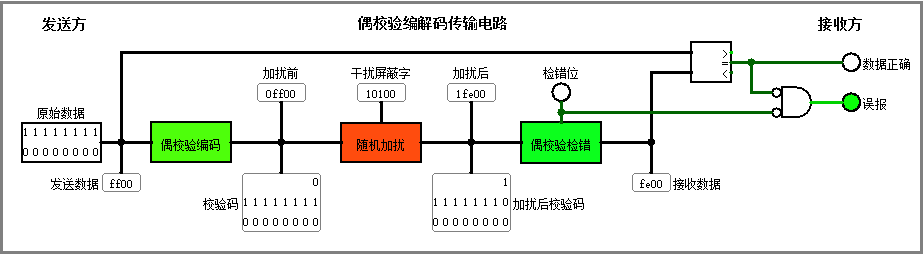


图1.19 检错电路误报的情况

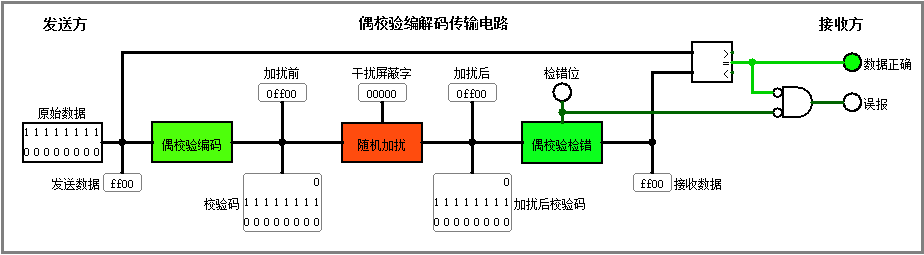


图1.20 数据传输正确的情况

### 海明校验传输测试

输入的测试用例为1111111100000000，图1.21、图1.22和图1.23分别是干扰电路不干扰、加1位干扰和加2位干扰的测试结果，从结果来看，海明编码电路和海明解码电路功能都正常。

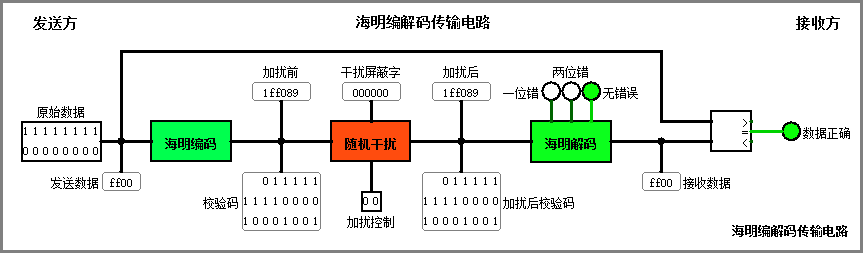


图1.21 不加干扰时传输结果

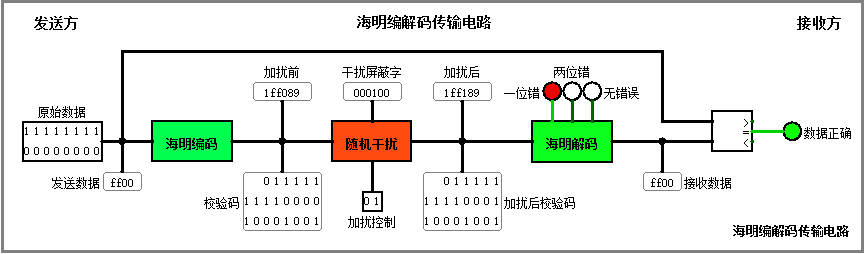


图1.22 加1位干扰时的传输结果

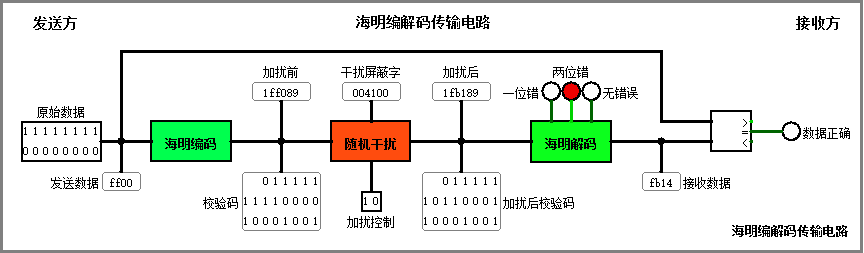


图1.23 加2位干扰时的传输结果

### 海明流水重传测试

打开修改好的流水传输电路，按Ctrl+K开始时钟仿真，观察发现传输的文字是通顺的句子，说明重传功能正确。

# CPU设计实验

## 设计要求

利用运算器实验，存储系统实验中构建的运算器、寄存器文件、存储系统等部件以及Logisim中其他功能部件构建一个32位MIPS CPU单周期处理器，该处理器应支持表2.1中列出的所有指令，另外还必须支持表2.2中的2条C类运算指令、1条M类存储指令和1条B类分支指令，共4条扩展指令。最终设计完成的CPU应能运行标准测试程序，程序存储在Logisim ROM模块中（指令存储器、数据存储器分开）。

表2.1 基础指令集



表2.2 扩展指令集



其中，syscall的功能为：如果$v0等于10，则停机，否则在数码管中显示$a0值。其他指令功能参考MIPS32指令集，最终功能以MARS模拟器为准。

## 方案设计

### PC+1模块

PC+1模块的电路如图2.1所示，因为PC的位宽是32位，所以PC值实际上加4，所以用一个加法器对输入的原PC加4输出即可。

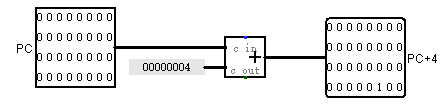


图2.1 PC+1模块电路图

### ALU模块

ALU内部的电路图如图2.2所示，图中的隧道各表示一个引脚，各引脚的定义和功能描述见表2.3，其中，操作码S对应的运算功能说明见表2.4。在电路中，操作数传入后，就传到各运算部件计算所有运算的结果，但最终结果通过操作码配合多路选择器进行输出。

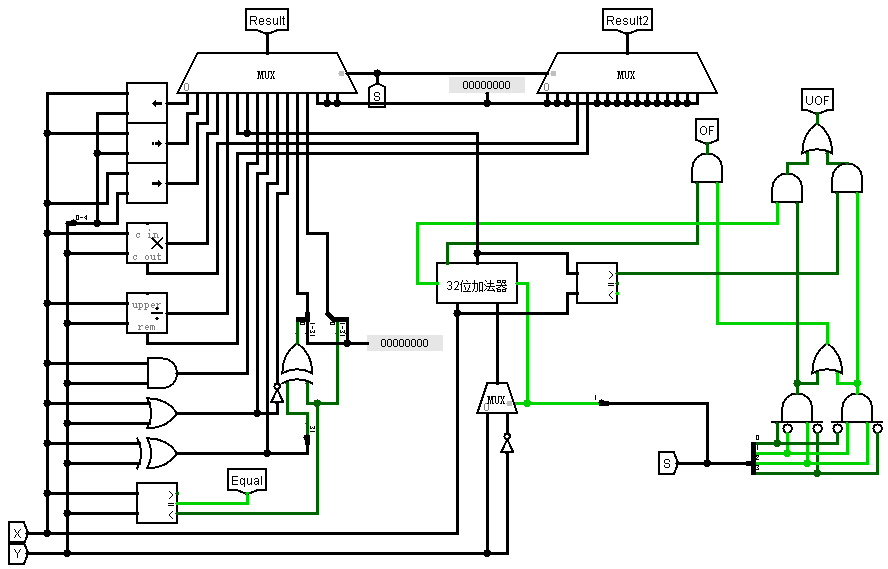


图2.2 ALU内部电路图

表2.3 ALU引脚与功能描述



表2.4 ALU功能表



### 控制器模块

控制器的主要功能是，通过IM输出的指令中的OP字段和funct字段，判别该指令具体那哪种指令，从而生成该指令对应的控制信号并输出。控制器的电路设计如图2.3所示，先通过29个比较器，分辨出指令的种类，然后把各控制信号被用到的各指令相加（即相或），即可得到对应的控制信号（具体得出各控制信号的表达式的过程将在下一节中详细说明），所有控制信号用分线器整合成一个输出引脚。在构建CPU系统电路时，再把控制信号按照图2.4所示而分解成各控制信号，传到相应的控制部件。

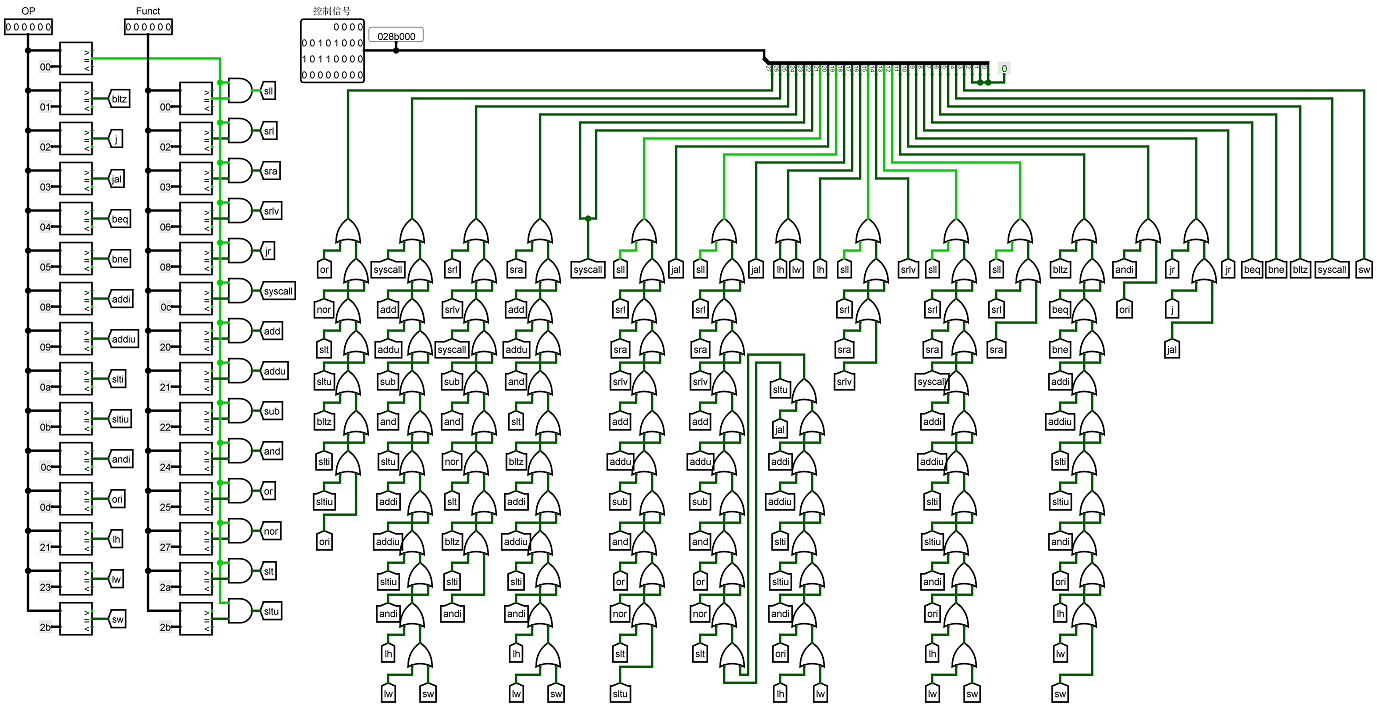


图2.3 控制器电路图

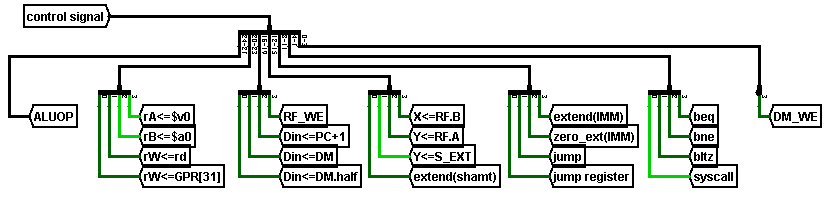


图2.4 控制信号分解图

### 数码管显示模块

数码管显示模块电路图如图2.5所示，显示部分用8个Hex数码管显示，他们的输入通过一个寄存器来满足锁存功能，当执行syscall时，如果$v0等于10，则display为1，寄存器的值在时钟上升沿把RF.B中的$a0的值存到寄存器并输出到Hex数码管，即可实现显示$a0寄存器的值。

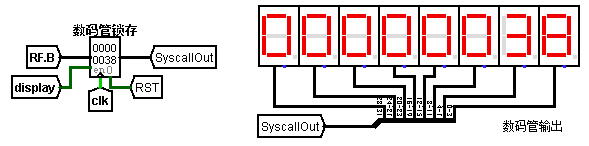


图2.5 数码管显示电路

### 其他模块

其他模块都可以直接用logisim自带的组件和CS3410.jar库中的组件来实现，例如：指令存储器可用ROM实现，且因为ROM有锁存功能，因此不用添加IR暂存指令；数据存储器可用RAM实现，符号扩展器就用logisim自带的扩展器，寄存器堆用CS3410.jar库中的Register File组件来实现。

### CPU整体电路设计

整个CPU的电路设计如图2.6所示，该电路把各功能部件连接通过数据通路连接起来，数据通路的构建过程将在下一节中详细说明。

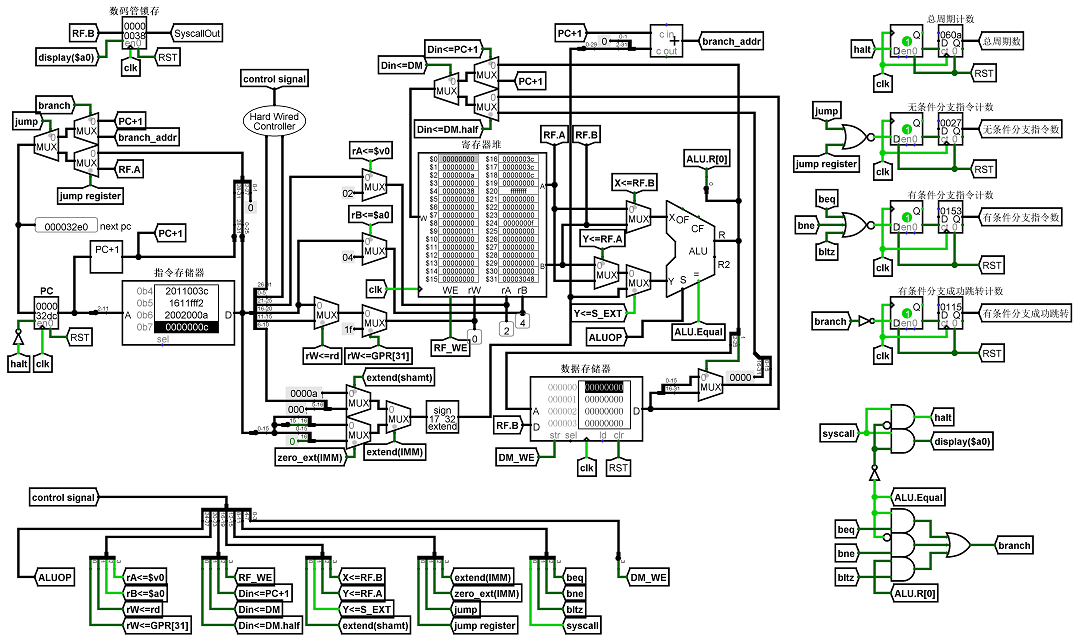


图2.6 CPU整体电路图

## 实验步骤

**1. 构建主要功能部件。**

按照上一节中各模块的设计，在logisim中构建出各个模块，并把每个模块都复制到CPU主电路中

**2. 一次性构建所有数据通路。**

绘制主要功能部件输入来源表，该表主要用于描述各部件之间的连接关系，记录各部件输入端数据来源，这里忽略控制类信号，仅保留数据类信号。

**3. 输入源合并。**

将主要功能部件输入来源表中部件输入按列进行合并，对于多输入来源的输入，引入多路选择器，同时新增多路选择器选择控制信号，对于空输入，或其他输入不影响指令执行，可直接忽略，否则需要增加额外的控制电路。合并后的主要功能部件输入来源表见表2.5。

**4. 构建数据通路。**

根据合并后的输入来源表，在logisim中连接各主要功能部件，实现MIPS CPU所有数据通路。

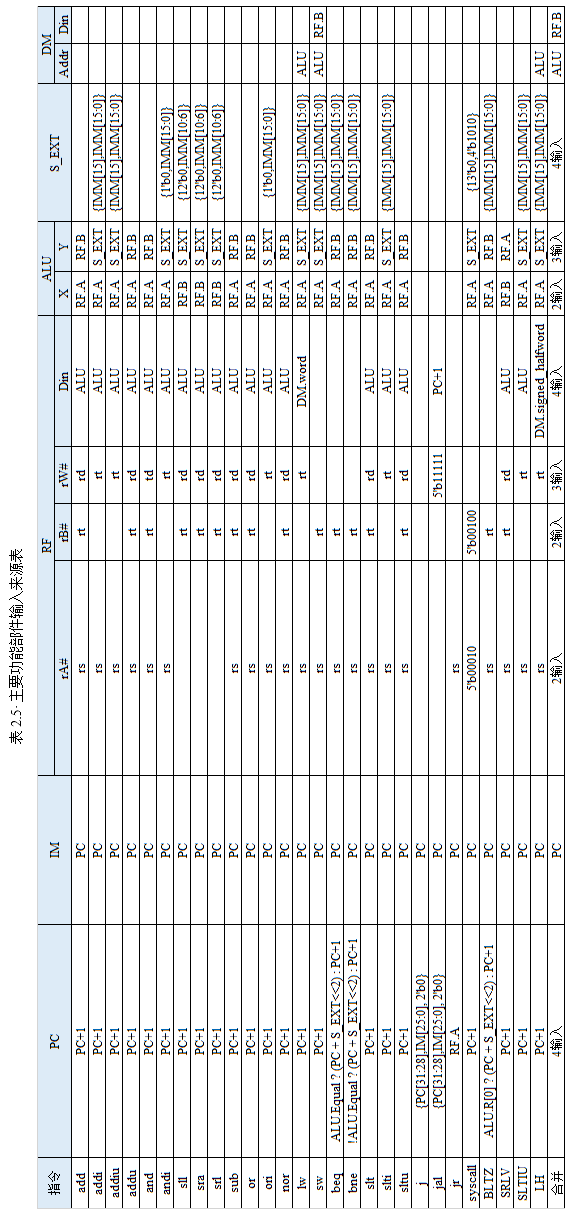
**5. 控制信号综合。**

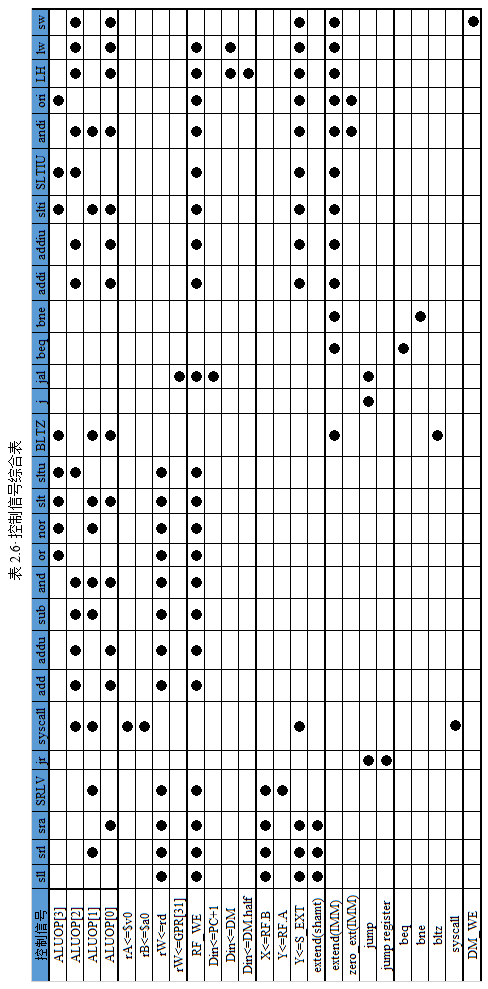
构建控制信号综合表，列出所有控制信号的产生条件，如表2.6所示，各控制信号的表达式即是该可控制信号所在行的有标记的所有指令相加。然后根据相应逻辑表达式画出控制器，并封装。

**6. 系统调试。**

a. 将实验提供的benchmark程序加载如指令存储器中，重置所有部件，然后启动时钟仿真，观察数码管的动态看是否有跑马灯的效果，程序运行完后，看数据存储器中的数据是否按顺序排序好，以及各指令计数和总周期计数是否正确。

b. 编写一个汇编程序，并用MARS编译成二进制指令加载到CPU的指令存储器中，测试扩展指令的功能是否正确。





## 故障与调试

### 存储器地址位宽过大

**故障现象：**指令存储器地址的位宽为24位，将benchmark程序加载如指令存储器后，运行完第二条跳转指令时，PC值正常，但是IM中对应地址内的指令为空的，如图2.7所示。

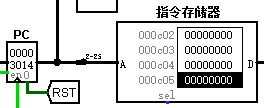


图2.7 PC指向的指令为空

**原因分析：**因为用mars编译的汇编程序，其代码段是从0x3000开始，而第二条指令地址是0x3004，跳转到0x3014位置只需PC+0x10，但是图中的CPU的PC值是从0开始的，指令存储的位置也是从第0个单元开始存放，因此跳转时，PC实际增加了0x3010，因此程序执行不正确。

**解决方案：**因为该程序的程度不需要用到24位宽的地址线，可将PC的前20位忽略，只用10位宽的地址，这样相当于代码段是从0x000处开始，跳转后，PC增加0x3010，但是处于高位的3被忽略，PC有效值相当于只增加0x010，如此一来，执行玩第二条指令后，PC将指向第6条指令，如图2.8所示。修改后，程序可以正常运行了。

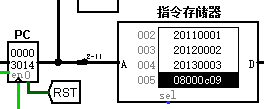


图2.8 修改后PC指向正常的指令

### 总周期计数少1次

**故障现象：**测试程序benchmark运行完后，显示的总周期数为1545，但是正确的总周期数是1546，实际测试的周期数计数比理论值少1。

**原因分析：**总周期数的计数电路如图2.9所示，当程序运行结束，最后一条syscall指令周期的上升沿到来后，CPU就开始执行停机指令，计数器要在下一个时钟上升沿到来时开始计数，但是执行停机指令的周期内，CPU把停机信号置位1了，所以计数器的count输入变为0，因此在下一个时钟上升沿到来时不能加一。

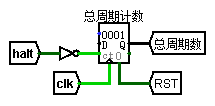


图2.9 总周期计数电路

**解决方案：**如图2.10所示，停机信号halt先连接一个D促发器，其起到锁存和延后一个时钟周期的作用，然后将D促发器的Q非输出端接到计数器的count控制端。如此一来，当halt信号变为1的时候，这个时钟周期内计数会加一，下个时钟周期才会停止计数。

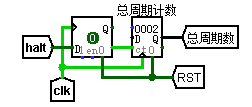


图2.10 修改后的计数电路

## 测试与分析

### benchmark程序测试

将benchmark程序加载进指令存储器后，按Ctrl+K开启时钟仿真，从数码管显示模块可以明显看到跑马灯效果，程序运行完后，CPU处于停机状态，显示面板显示的结果如图2.x所示。其中的数码管最终状态和总周期数、无条件分支指令数、有条件分支指令数以及有条件分支成功跳转计数的结果都与理论的结果相符。

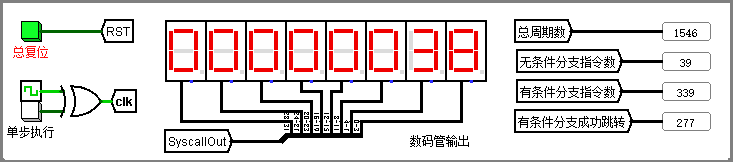


图2.x benchmark程序运行结束后显示结果

### CCMB程序测试

1. CCMB测试程序的汇编代码：

.text

#测试sltiu

sltiu $s0,$zero,0 #此时$zero不小于，$s0不置1

sltiu $s0,$zero,1 #此时$zero小于1，$s0置1

#测试srlv

addi $s0,$zero,0xf1 #给$s0赋初值0xf1

sll $s0,$s0,16 #将$s0左移16位

addi $s1,$zero,4 #给$s0赋初值4

srlv $s0,$s0,$s1 #将$s0after,$s0=0x00ffee00

#测试lh

sw $s0,0 #将$s0的值存到数据存储器的0号单元

lh $s0,0 #将数据存储器0号单元的低半字加载到$s0,此时$s0=0x0000ee00

add $a0,$0,$s0

addi $v0,$0,34 # display hex

syscall

lh $s0,2 #将数据存储器0号单元的高半字加载到$s0,此时$s0=0x0000ffee

add $a0,$0,$s0

addi $v0,$0,34 # display hex

syscall

#测试bltz

bltz $s0,exit #此时$s0不小于0，跳转

addi $s0,$zero,-1 #给$s0赋值-1，使其小于0

bltz $s0,exit #此时$s0不小于0，跳转到退出

addi $s0,$zero,0xeee

add $a0,$0,$s0

addi $v0,$0,34 # display hex

syscall

exit: #程序退出代码部分

addi $v0,$zero,10 # system call for exit

syscall # exit

2. 将上面描述的程序汇编成机器指令后，载入CPU的指令存储器中单步运行，可以观察到每一步的运行结果与上面程序中注释描述的情况都相符，且运行结束后，数码管将显示0x0000000F，程序共执行20条指令，其中有条件分支指令2条但只有一次成功跳转。图2.x为程序运行完后显示的结果，结果表明CPU的扩展指令功能正确。

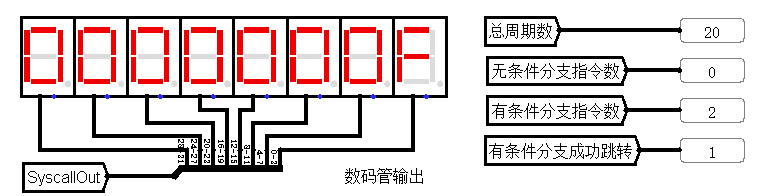


图2.x 扩展指令测试程序结束后显示结果

# 总结与心得

## 实验总结

1. 数据表示实验中，主要工作有：

a. 实现将国标码转换成区位码的电路；

b. 了解从区位码到字库查询并显示汉子的大概原理；

c. 实现16位数据的偶校验编码电路和解码电路；

d. 实现16位数据的海明校验码的编码电路和解码电路；

e. 实现海明码流水传输的2位错重传功能。

2. CPU设计实验中，主要工作有：

a. 实现ALU模块的设计；

b. 构建主要模块输入来源表；

c. 构建CPU数据通路；

d 构建控制信号综合表；

e. 实现控制器电路的设计；

f. 连接其他模块，完成整个CPU。

3. CPU的功能包括：

a. 能够执行表2.1中的所有基础指令；

b. 能够执行表2.2中的扩展指令；

c. 用数码管显示寄存器的值；

d. 统计指令周期的数目。

## 实验心得

组成原理实验非常贴合课程的内容，实验的内容都是上课讲过的重点内容。

通过组成原理实验，我不仅巩固了对课本的知识点的掌握，还增加了学习组成原理的兴趣和信心，尤其是本次的CPU设计实验，除了让我学习到工程化设计CPU的方法外，还给我带来很大的成就感，让我感觉到设计CPU这种极其复杂的工作并不是不可能完成。

在整个实验过程中，我几乎没有遇到什么大问题，就是感觉画图比较繁琐，因为不仅要把图画出来，跟要把各部件和导线拜访整齐，达到美观的效果，这是我对自己的要求，这个习惯也让我受益良多，比如那个部件出现错误，我很快就能找出出错的位置和原因。

总之，实验并不是特别难，但是一定要花心思认真做，前面的实验是后面实验的基础，就算简单也不要马虎。比如CPU实验要用到前几次实验设计的ALU，如果设计的ALU效率特别低，那么组装成的CPU效率也会非常低，会给调试带来很大麻烦。

参考文献

[1] DAVID A.PATTERSON(美).计算机组成与设计硬件/软件接口(原书第4版).北京：机械工业出版社.

[2] David Money Harris(美).数字设计和计算机体系结构（第二版）. 机械工业出版社

[3] 秦磊华，吴非，莫正坤.计算机组成原理. 北京：清华大学出版社，2011年.

[4] 袁春风编著. 计算机组成与系统结构. 北京：清华大学出版社，2011年.

[5] 张晨曦，王志英. 计算机系统结构. 高等教育出版社，2008年.

|  |
| --- |
| 一、原创性声明 |
| 本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。  特此声明！  **作者签字:** |
| 二、对课程实验的学术评语（教师填写） |
|  |
| 三、对课程实验的评分（教师填写） |
| |  |  |  |  | | --- | --- | --- | --- | | 评分项目  （分值） | 报告撰写  （30分） | 课设过程  （70分） | 最终评定  （100分） | | 得分 |  |  |  | |
| **指导教师签字: 2017-11-30** |