

# **数字逻辑实验报告（1**）

|  |  |  |
| --- | --- | --- |
| **数字逻辑实验1** | | |
| **一、系列二进制加法器设计50%** | **二、小型实验室门禁系统设计50%** | **总成绩** |
|  |  |  |

评语：（包含：预习报告内容、实验过程、实验结果及分析）

教师签名

**姓 名： 刘本嵩**

**学 号： U201614531**

**班 级： CS1601**

**指 导 教 师： 赵贻竹**

**计算机科学与技术学院**

**2018 年 5月 25 日**



**数字逻辑实验报告**

**系列二进制加法器设计预习报告**

一、系列二进制加法器设计

1、实验名称

系列二进制加法器设计。

2、实验目的

要求同学采用传统电路的设计方法，对5种二进制加法器进行设计，并利用工具软件，例如，“logisim”软件的虚拟仿真功能来检查电路设计是否达到要求。

通过以上实验的设计、仿真、验证3个训练过程使同学们掌握传统逻辑电路的设计、仿真、调试的方法。

3、实验所用设备

Logisim2.7.1软件一套。

4、实验内容

对已设计的5种二进制加法器，使用logisim软件对它们进行虚拟实验仿真，除逻辑门、触发器外，不能直接使用logisim软件提供的逻辑库元件，具体内容如下。

**（1）一位二进制半加器**

设计一个一位二进制半加器，电路有两个输入A、B，两个输出S和C。输入A、B分别为被加数、加数，输出S、C为本位和、向高位进位。

**（2）一位二进制全加器**

设计一个一位二进制全加器，电路有三个输入A、B和Ci，两个输出S和Co。输入A、B和Ci分别为被加数、加数和来自低位的进位，输出S和Co为本位和和向高位的进位。

**（3）串行进位的四位二进制并行加法器**

用四个一位二进制全加器串联设计一个串行进位的四位二进制并行加法器，电路有九个输入A3、A2、A1、A0、B3、B2、B1、B0和C0，五个输出S3、S2、S1、S0和C4。输入A= A3A2A1A0、B= B3B2B1B0和C0分别为被加数、加数和来自低位的进位，输出S= S3S2S1S0和Co为本位和和向高位的进位。

**（4）先行进位的四位二进制并行加法器**

利用超前进位的思想设计一个先行进位的四位二进制并行加法器，电路有九个输入A3、A2、A1、A0、B3、B2、B1、B0和C0，五个输出S3、S2、S1、S0和C4。输入A= A3A2A1A0、B= B3B2B1B0和C0分别为被加数、加数和来自低位的进位，输出S= S3S2S1S0和Co为本位和和向高位的进位。

**（5）将先行进位的四位二进制并行加法器封装成一个组件并验证它的正确性**

将设计好的先行进位的四位二进制并行加法器进行封装，生成一个“私有”库元件并验证它的正确性，以便后续实验使用，封装后的逻辑符号参见图1-1所示。

**S3 S2 S1 S0**

**C4 四位二进制并行加法器 C0**

**A3 A2 A1 A0  B3 B2 B1 B0**

图1-1“私有”的先行进位的四位二进制并行加法器

5、实验方案设计

**（1）一位二进制半加器的设计方案**

电路有两个输入A、B，两个输出S和C。输入A、B分别为被加数、加数，输出S、C为本位和、向高位进位。

半加器（half adder）的功能是将两个一位[二进制数](https://zh.wikipedia.org/wiki/%E4%BA%8C%E9%80%B2%E4%BD%8D%E6%95%B8)相加。它具有两个输入和两个输出（分别是[和](https://zh.wikipedia.org/wiki/%E5%92%8C)、[进位](https://zh.wikipedia.org/wiki/%E8%BF%9B%E4%BD%8D)）。输出的进位[信号](https://zh.wikipedia.org/wiki/%E4%BF%A1%E5%8F%B7)代表了输入两个数相加[溢出](https://zh.wikipedia.org/wiki/%E6%BA%A2%E5%87%BA)的高一位数值。因此，这两2个一位[二级制数](https://zh.wikipedia.org/w/index.php?title=%E4%BA%8C%E7%BA%A7%E5%88%B6%E6%95%B0&action=edit&redlink=1)的和等于2C + S。根据两个一位[二进制数](https://zh.wikipedia.org/wiki/%E4%BA%8C%E9%80%B2%E4%BD%8D%E6%95%B8)相加的结果，可以通过[真值表](https://zh.wikipedia.org/wiki/%E7%9C%9F%E5%80%BC%E8%A1%A8)、[卡诺图](https://zh.wikipedia.org/wiki/%E5%8D%A1%E8%AF%BA%E5%9B%BE)得到右图所描绘的简易[半加器](https://zh.wikipedia.org/wiki/%E5%8D%8A%E5%8A%A0%E5%99%A8)设计。它使用了一个[异或门](https://zh.wikipedia.org/wiki/%E5%BC%82%E6%88%96%E9%97%A8)来产生和S，并使用了一个[与门](https://zh.wikipedia.org/wiki/%E4%B8%8E%E9%97%A8)来产生进位[信号](https://zh.wikipedia.org/wiki/%E4%BF%A1%E5%8F%B7)C。如果再添加一个[或门](https://zh.wikipedia.org/wiki/%E6%88%96%E9%97%A8)来接收低位的进位输出信号，则两个[半加器](https://zh.wikipedia.org/wiki/%E5%8D%8A%E5%8A%A0%E5%99%A8)就构成了一个[全加器](https://zh.wikipedia.org/wiki/%E5%85%A8%E5%8A%A0%E5%99%A8)。

半加器将两个输入位加和，产生进位与和，是半加器的两个[输出](https://zh.wikipedia.org/w/index.php?title=%E8%BE%93%E5%87%BA&action=edit&redlink=1)。半加器的[输入](https://zh.wikipedia.org/w/index.php?title=%E8%BE%93%E5%85%A5&action=edit&redlink=1)[变量](https://zh.wikipedia.org/wiki/%E5%8F%98%E9%87%8F_(%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1))叫做[被加数](https://zh.wikipedia.org/w/index.php?title=%E8%A2%AB%E5%8A%A0%E6%95%B0&action=edit&redlink=1)或[被加位](https://zh.wikipedia.org/w/index.php?title=%E8%A2%AB%E5%8A%A0%E4%BD%8D&action=edit&redlink=1)。输出[变量](https://zh.wikipedia.org/wiki/%E5%8F%98%E9%87%8F)为[和](https://zh.wikipedia.org/wiki/%E5%92%8C)与[进位](https://zh.wikipedia.org/wiki/%E8%BF%9B%E4%BD%8D)。[半加器](https://zh.wikipedia.org/wiki/%E5%8D%8A%E5%8A%A0%E5%99%A8)的[真值表](https://zh.wikipedia.org/wiki/%E7%9C%9F%E5%80%BC%E8%A1%A8)如表1-1所示。

表1-1 半加器的真值表

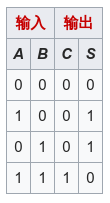


图1-2是 一位二进制半加器。

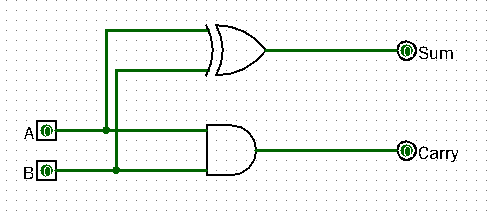


图1-2 一位二进制半加器

**（2）一位二进制全加器的设计方案**

电路有三个输入A、B和Ci，两个输出S和Co。输入A、B和Ci分别为被加数、加数和来自低位的进位，输出S和Co为本位和和向高位的进位。

全加器（full adder）将两个一位二进制数相加，并根据接收到的低位进位信号，输出和、进位输出。全加器的三个输入信号为两个加数A、B和低位进位Cin。[全加器](https://zh.wikipedia.org/wiki/%E5%85%A8%E5%8A%A0%E5%99%A8)通常可以通过[级联](https://zh.wikipedia.org/wiki/%E7%BA%A7%E8%81%94)（cascade）的方式，构成多位（如8位、16位、32位）[二进制数](https://zh.wikipedia.org/wiki/%E4%BA%8C%E9%80%B2%E4%BD%8D%E6%95%B8)加法器的基本部分。全加器的[输出](https://zh.wikipedia.org/w/index.php?title=%E8%BE%93%E5%87%BA&action=edit&redlink=1)和半加器类似，包括向高位的进位信号Cout和本位的和信号S，相加结果的总和表达为{\displaystyle \mathrm {sum} =2\times C\_{out}+S}。一位全加器的[真值表](https://zh.wikipedia.org/wiki/%E7%9C%9F%E5%80%BC%E8%A1%A8)如表1-2所示。

表1-2 全加器的真值表

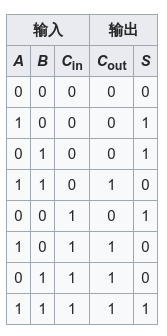


图1-3是一位二进制全加器。

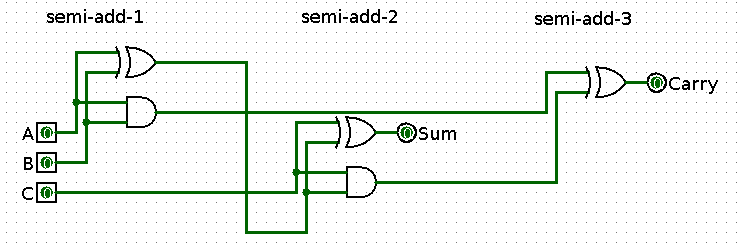


图1-3 一位二进制全加器

**（3）串行进位的四位二进制并行加法器的设计方案**

电路有九个输入A3、A2、A1、A0、B3、B2、B1、B0和C0，五个输出S3、S2、S1、S0和C4。输入A= A3A2A1A0、B= B3B2B1B0和C0分别为被加数、加数和来自低位的进位，输出S= S3S2S1S0和Co为本位和和向高位的进位。

可以使用多个一位全加器来构成N位加法器，其中对应低位的全加器将其进位输出信号Cout连接到高一位的全加器的进入输入端Cin。这种构成多位加法器的形式被称为“串行进位加法器”。如果不需要连接其他进位信号，则最低位的全加器可以用半加器替换。

图1-4是串行进位的四位二进制并行加法器。

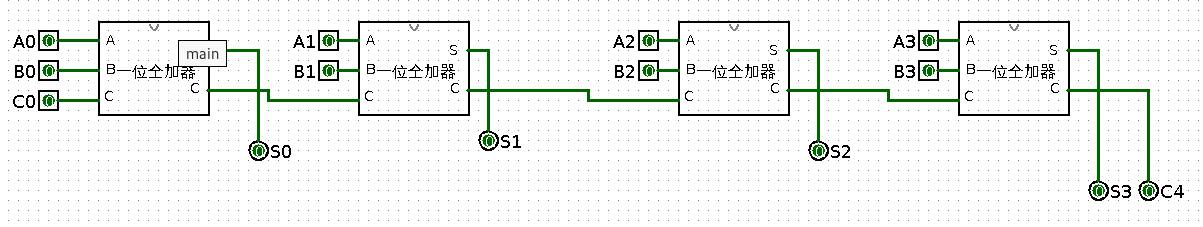


图1-4串行进位的四位二进制并行加法器

**（4）先行进位的四位二进制并行加法器的设计方案**

电路有九个输入A3、A2、A1、A0、B3、B2、B1、B0和C0，五个输出S3、S2、S1、S0和C4。输入A= A3A2A1A0、B= B3B2B1B0和C0分别为被加数、加数和来自低位的进位，输出S= S3S2S1S0和Co为本位和和向高位的进位。

为了减少多位[二进制数](https://zh.wikipedia.org/wiki/%E4%BA%8C%E9%80%B2%E4%BD%8D%E6%95%B8)[加减计算](https://zh.wikipedia.org/w/index.php?title=%E5%8A%A0%E5%87%8F%E8%AE%A1%E7%AE%97&action=edit&redlink=1)所需的时间，[工程师](https://zh.wikipedia.org/wiki/%E5%B7%A5%E7%A8%8B%E5%B8%88)设计了一种比[脉动](https://zh.wikipedia.org/w/index.php?title=%E8%84%89%E5%8A%A8&action=edit&redlink=1)[进位](https://zh.wikipedia.org/wiki/%E8%BF%9B%E4%BD%8D)加法器速度更快的加法器[电路](https://zh.wikipedia.org/wiki/%E7%94%B5%E8%B7%AF)，这种加法器被称为“[超前进位加法器](https://zh.wikipedia.org/w/index.php?title=%E8%B6%85%E5%89%8D%E8%BF%9B%E4%BD%8D%E5%8A%A0%E6%B3%95%E5%99%A8&action=edit&redlink=1)”（carry-lookahead adder）。

下面简述超前进位加法器的主要原理。我们先来考虑构成多位加法器的单个全加器从其低一位获得的进位[信号](https://zh.wikipedia.org/wiki/%E4%BF%A1%E5%8F%B7)，我们可以将它变换为。现在为[二级制数](https://zh.wikipedia.org/w/index.php?title=%E4%BA%8C%E7%BA%A7%E5%88%B6%E6%95%B0&action=edit&redlink=1)的每一位[构建](https://zh.wikipedia.org/w/index.php?title=%E6%9E%84%E5%BB%BA&action=edit&redlink=1)两个新[信号](https://zh.wikipedia.org/wiki/%E4%BF%A1%E5%8F%B7)：

* [生成](https://zh.wikipedia.org/w/index.php?title=%E7%94%9F%E6%88%90&action=edit&redlink=1)（Generate）[信号](https://zh.wikipedia.org/wiki/%E4%BF%A1%E5%8F%B7)：
* [传输](https://zh.wikipedia.org/wiki/%E4%BC%A0%E8%BE%93)（Propagate）信号：

则有，









将Ci分别展开，有









其中Gi和Pi容易被计算。

图1-5是先行进位的四位二进制并行加法器。

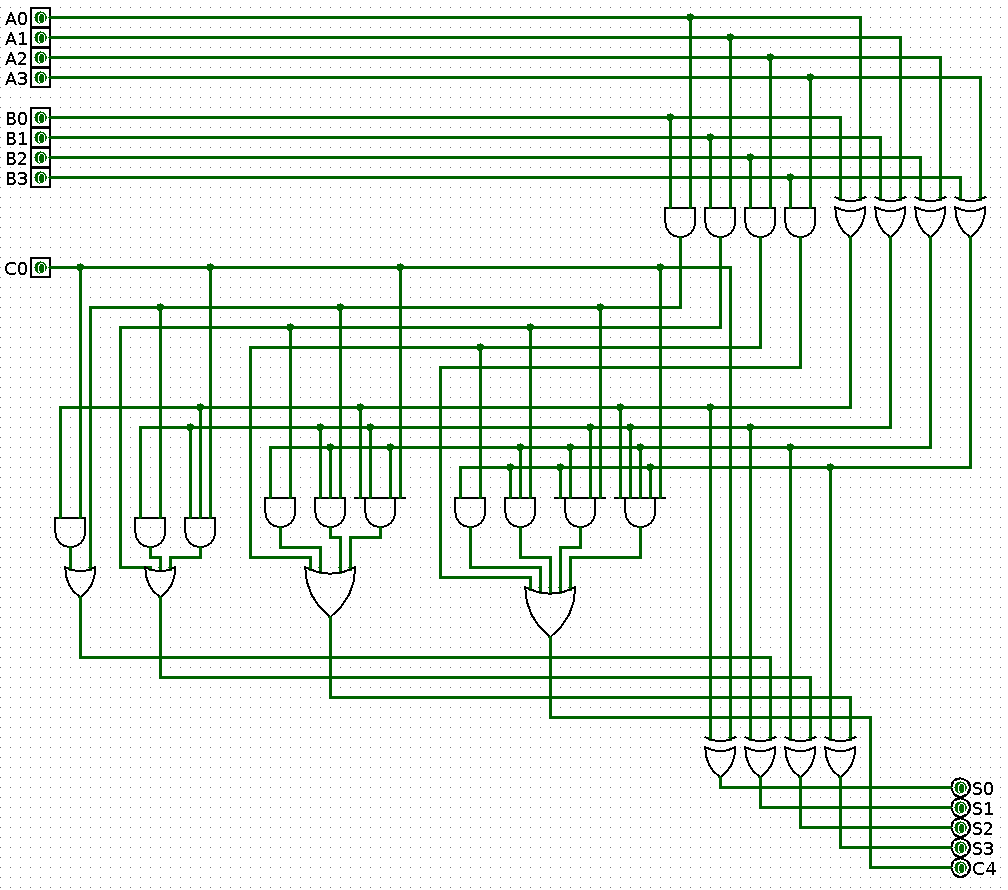
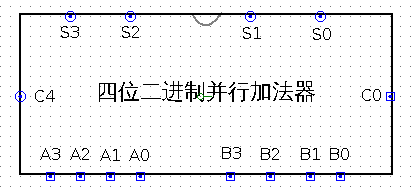


图1-5先行进位的四位二进制并行加法器

**（5）封装先行进位的四位二进制并行加法器电路**

对“第4步”完成的电路进行封装，然后对它设计的正确性进行验证。



封装之后，经过几次试验操作不难验证，其结果是正确的。



**数字逻辑实验报告**

小型实验室门禁系统设计实验报告

二、小型实验室门禁系统设计

1、实验名称

小型实验室门禁系统设计。

2、实验目的

要求同学采用传统电路的设计方法，对一个“设计场景”进行逻辑电路的设计，并利用工具软件，例如，“logisim”软件的虚拟仿真来检查这个小型实验室门禁系统的设计是否达到要求。

通过以上实验的设计、仿真、验证3个训练过程使同学们掌握小型电路系统的设计、仿真、调试方法以及电路模块封装的方法。

3、实验所用设备

Logisim2.7.1软件一套。

4、实验内容

设计场景：某小型保密实验室需要安装一个门禁系统，用于监测、控制和显示该实验室内上班人数，该实验室只有一个门，最多只能容纳15人。假设员工进出实验室都要刷校园卡，并且保证一次刷卡后有且只有一人能进出。实验室空置时人数显示为0，刷卡进入时实验室人数加1，刷卡离开时实验室人数减1。当实验室满员时，还有员工在门外刷卡进入时，门禁系统“不”动作，系统报警提示满员。

使用logisim软件对小型电路进行虚拟实验仿真，除逻辑门、触发器、7段数码显示管外，不能直接使用logisim提供的逻辑元件库，具体要求如下。

**（1）设计一个四位二进制可逆计数器电路并进行封装和验证它的正确性**

用D触发器设计一个四位二进制可逆计数器，并进行封装。该计数器有一个清零端CLR、一个累加计数脉冲端CPU（输入刷卡进入请求）、一个累减计数脉冲端CPD（输入刷卡离开请求），四个计数输出端QDQCQBQA记录当前实验室人数。

将设计好的4位二进制可逆计数器进行封装，生成一个“私有”库元件，以便后续实验使用，4位二进制可逆计数器逻辑符号参见图2-1所示。

**SD SC SB SA**

**CPU**

**CLR 四位二进制可逆计数器**

**CPD**

图2-1 “私有”的一个4位二进制可逆计数器

**（2）用实验1中已封装的“先行进位的四位二进制并行加法器”设计一个将实验室内人数转换成8421BCD码的电路**

用实验一中已封装的“先行进位的四位二进制并行加法器”和适当的逻辑门将二进制数表示的实验室人数转换成两位十进制数的8421BCD码。

**（3）设计7段译码器，并采用“7段数码显示管”显示人数的电路**

设计一个7段译码器（参考书的7448芯片），将两位十进制数的8421BCD码表示的实验室人数用“7段数码显示管”显示出来。

该7段译码器有四个输入A3A2A1A0和七个输出abcdefg, A3A2A1A0为8421BCD码，abcdefg为7段数码显示管对应的段。

**（4）设计当实验室满员时，门禁“不”动作，系统报警提示满员的电路**

当实验室满员时，在累加计数脉冲端CPU输入刷卡进入请求，计数输出端数据保持不变，门禁“不”动作，系统报警提示满员。当实验室空时，逻辑上不会有实验室内累减计数脉冲端CPD输入刷卡离开请求。为防止信号干扰，在计数输出为0时，若CPD端有脉冲，也应使计数输出端数据保持不变，门禁“不”动作，但不用报警。

**（5）设计小型实验室门禁系统电路并进行封装和验证它的正确性**

设计满足要求的小型实验室门禁系统电路并进行封装，生成一个小型实验室门禁系统芯片，封装后的小型实验室门禁系统逻辑符号参见图2-2所示。

**十位**：7段数码显示管

**个位：**7段数码显示管

**a b c d e f g a b c d e f g**

**CPU**

**CLR 封装后的门禁系统逻辑符号 报警**

**CPD**

图2-2 封装后的小型实验室门禁系统

5、实验方案设计

**（1）设计一个四位二进制可逆计数器电路**

**加法计数时的真值表**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 现态 | | | | 激励函数 | | | | | | | | 次态 | | | |
| Q3n | Q2n | Q1n | Q0n | D3 | C3 | D2 | C2 | D1 | C1 | D0 | C0 | Q3n+1 | Q2n+1 | Q1n+1 | Q0n+1 |
| 0 | 0 | 0 | 0 | d |  | d |  | d |  | 1 | ↓ | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | d |  | d |  | 1 | ↓ | 0 | ↓ | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | d |  | d |  | d |  | 1 | ↓ | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | d |  | 1 | ↓ | 0 | ↓ | 0 | ↓ | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | d |  | d |  | d |  | 1 | ↓ | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | d |  | d |  | 1 | ↓ | 0 | ↓ | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | d |  | d |  | d |  | 1 | ↓ | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | ↓ | 0 | ↓ | 0 | ↓ | 0 | ↓ | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | d |  | d |  | d |  | 1 | ↓ | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | d |  | d |  | 1 | ↓ | 0 | ↓ | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | d |  | d |  | d |  | 1 | ↓ | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | d |  | 1 | ↓ | 0 | ↓ | 0 | ↓ | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | d |  | d |  | d |  | 1 | ↓ | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | d |  | d |  | 1 | ↓ | 0 | ↓ | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | d |  | d |  | d |  | 1 | ↓ | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | ↓ | 0 | ↓ | 0 | ↓ | 0 | ↓ | 0 | 0 | 0 | 0 |

**减法计数时的真值表**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 现态 | | | | 激励函数 | | | | | | | | 次态 | | | |
| Q3n | Q2n | Q1n | Q0n | D3 | C3 | D2 | C2 | D1 | C1 | D0 | C0 | Q3n+1 | Q2n+1 | Q1n+1 | Q0n+1 |
| 0 | 0 | 0 | 0 | 1 | ↓ | 1 | ↓ | 1 | ↓ | 1 | ↓ | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | d |  | d |  | d |  | 0 | ↓ | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | d |  | d |  | 0 | ↓ | 1 | ↓ | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | d |  | d |  | d |  | 0 | ↓ | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | d |  | 0 | ↓ | 1 | ↓ | 1 | ↓ | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | d |  | d |  | d |  | 0 | ↓ | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | d |  | d |  | 0 | ↓ | 1 | ↓ | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | d |  | d |  | d |  | 0 | ↓ | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | ↓ | 1 | ↓ | 1 | ↓ | 1 | ↓ | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | d |  | d |  | d |  | 0 | ↓ | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | d |  | d |  | 0 | ↓ | 1 | ↓ | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | d |  | d |  | d |  | 0 | ↓ | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | d |  | 0 | ↓ | 1 | ↓ | 1 | ↓ | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | d |  | d |  | d |  | 0 | ↓ | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | d |  | d |  | 0 | ↓ | 1 | ↓ | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | d |  | d |  | d |  | 0 | ↓ | 1 | 1 | 1 | 0 |

不难得到表达式：

[](about:blank)

[](about:blank)

[](about:blank)

[](about:blank)

下面的电路图由74191直接修改而成。

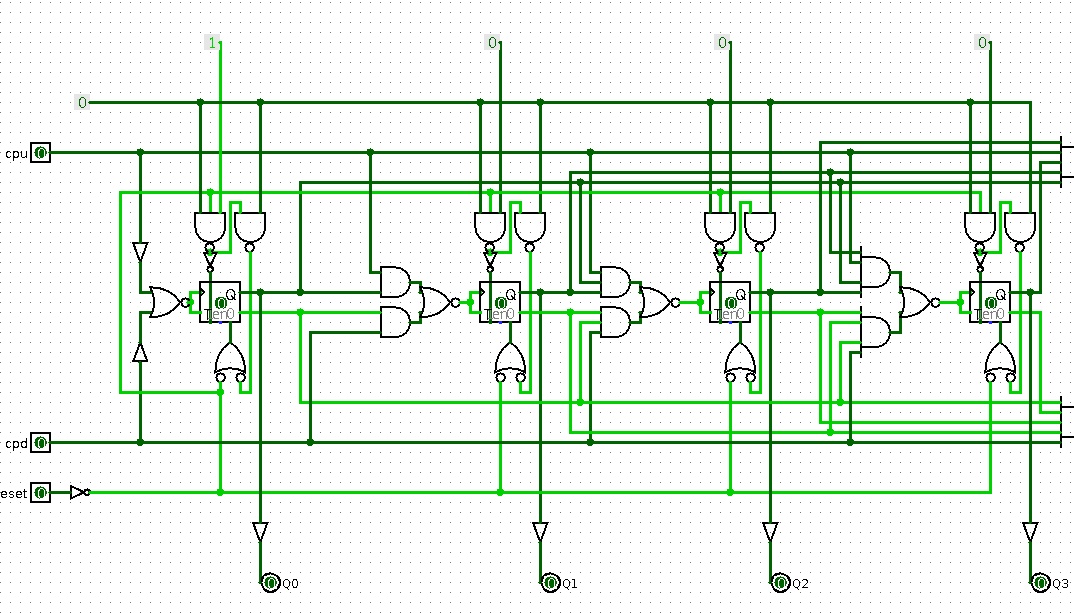


图2-3 一个四位二进制可逆计数器

**（2）用实验一中已封装的“先行进位的四位二进制并行加法器”设计将实验室内人数转换成8421BCD码的电路**

若输入的数Q3Q2Q1Q0小于10，则直接输出即可。若输入的数Q3Q2Q1Q0大于等于10，则输Q3Q2Q1Q0 + 6。下面用logisim进行仿真。

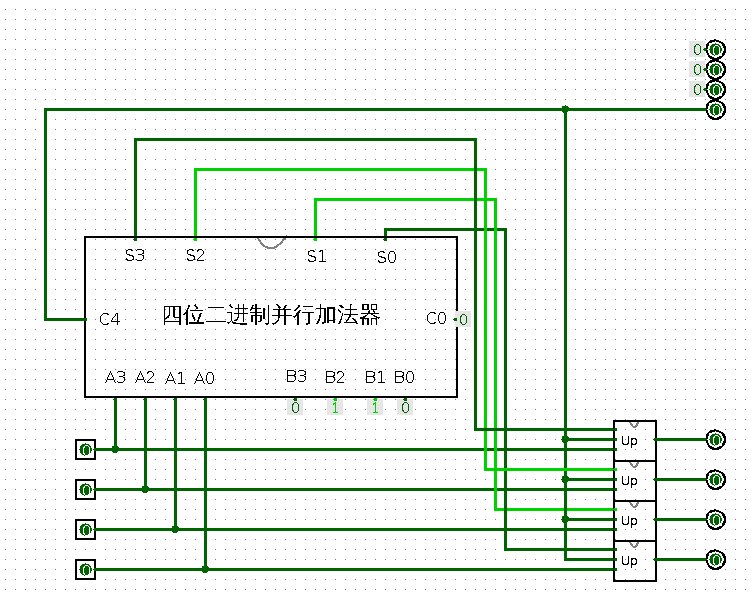


图2-4 一位16进制数转2位8421码

**（3）设计7段译码器，并采用“7段数码显示管”显示人数的电路**

**（A）设计一个7段译码器**

七段数码管一般由8个发光二极管组成，其中由7个细长的发光二极管组成数字显示，另外一个圆形的发光二极管显示小数点。 当发光二极管导通时，相应的一个点或一个笔画发光。控制相应的二极管导通，就能显示出各种字符。发光二极管的阳极连在一起的称为共阳极数码管，此时译码器的输出应该是低电平有效；阴极连在一起的称为共阴极数码管，此时译码器的输出应该是高电平有效。其真值表如表2-1所示。

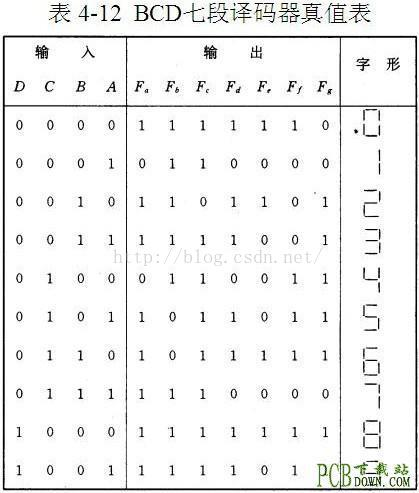


表2-1 BCDto7真值表

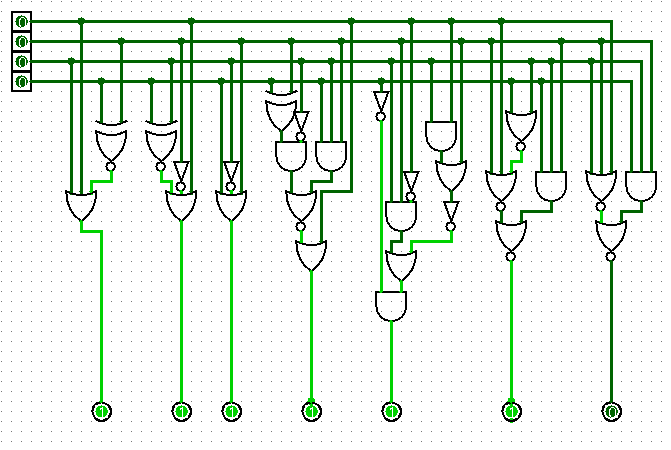


图2-5 7段译码器

**（B）设计用“7段数码显示管”显示人数的逻辑电路**

logisim提供了7 segment display，只需将引脚分别接上即可。

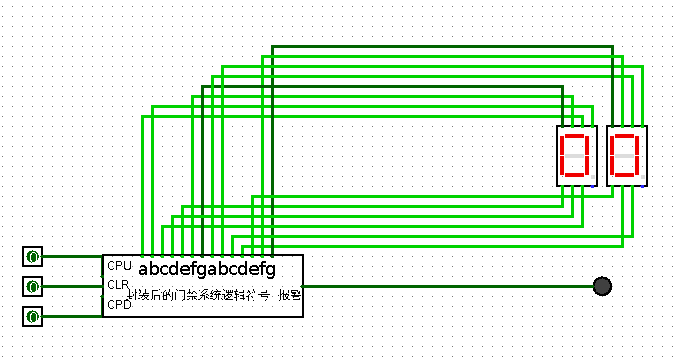


图2-6 用7段数码显示管显示实验室人数的电路

**（4）设计当实验室满员时，门禁“不”动作，系统报警提示满员的电路**

电路设计要求要采用本次实验1中由4位二进制可逆计数器所封装的“私有”库元件。

在输入时钟端增加一个旁路，在人满时通过与非门和与门的组合禁止时钟输入，并电量发光二极管；在输出时钟端也增加一个旁路，在人空时通过或门和与门的组合禁止时钟输入。

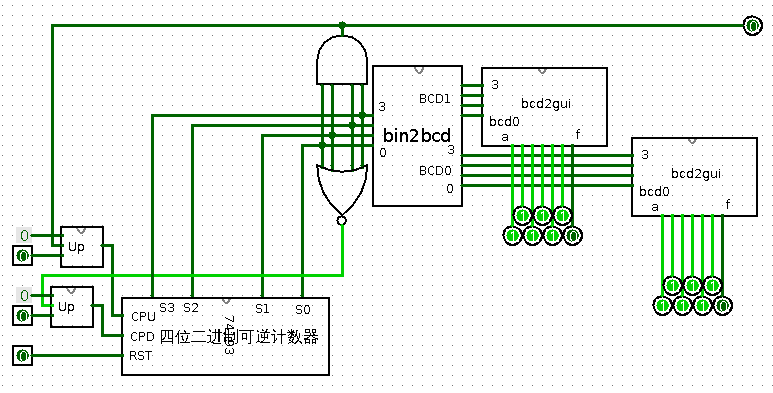


图2-7 报警电路

**（5）设计小型实验室门禁系统电路**

将以上各元件进行组合，得到如下的电路图：

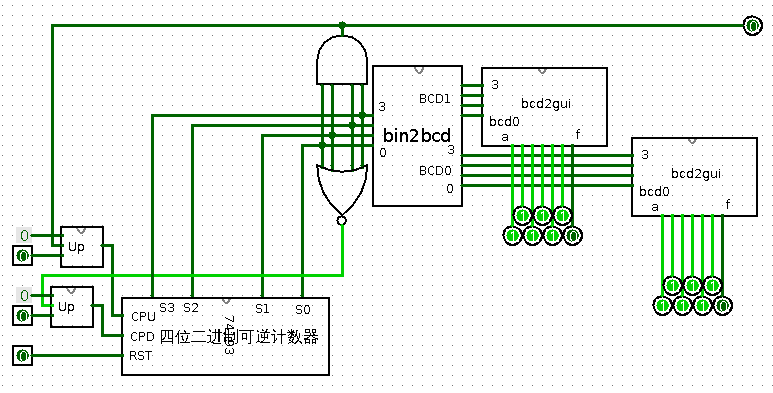


图2-8 门禁系统电路

6、实验结果记录

**（1）给出“私有”库元件（采用一个四位二进制可逆计数器进行封装）的测试电路**

对四位二进制可逆计数器进行封装并加载库，得到如下的测试电路，经过测试验证结果正确。

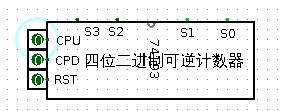


图2-9 一个四位二进制可逆计数器

**（2）给出采用实验1中已封装的“先行进位的四位二进制并行加法器”设计的将实验室内人数转换成8421BCD码的电路**

“logisim”软件绘制的电路图如下（经过仿真验证基本正确）

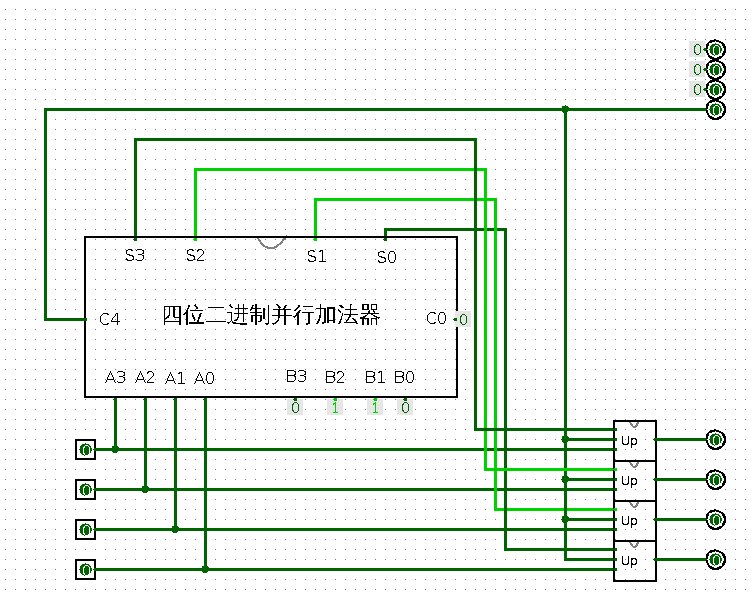


图2-10 二进制数转换成8421BCD码的电路

**（3）给出采用“7段数码显示管”显示人数的电路**

“logisim”软件绘制的电路图如下（经过仿真验证基本正确）

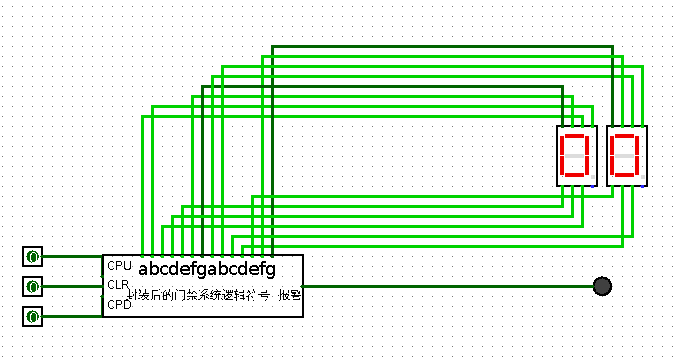


图2-11 人数显示的电路

**（4）给出当实验室满员时，门禁不动作，系统报警提示满员的电路**

“logisim”软件绘制的电路图如下（经过仿真验证基本正确）

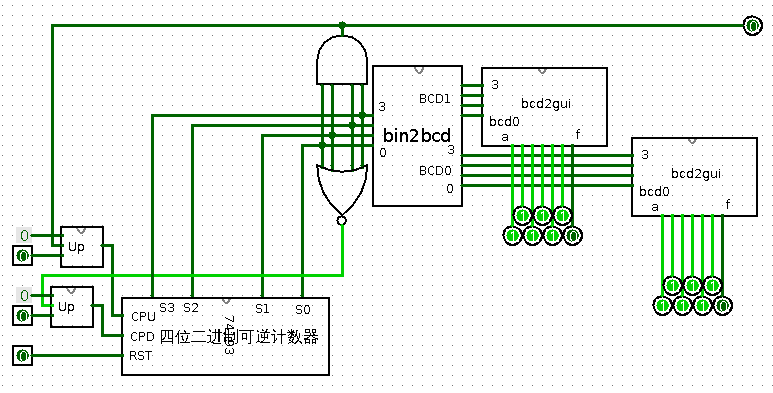


图2-12 系统报警电路

**（5）给出“私有”库元件（采用小型实验室门禁系统电路进行封装）的测试电路**

“logisim”软件绘制的电路图如下（经过仿真验证基本正确）

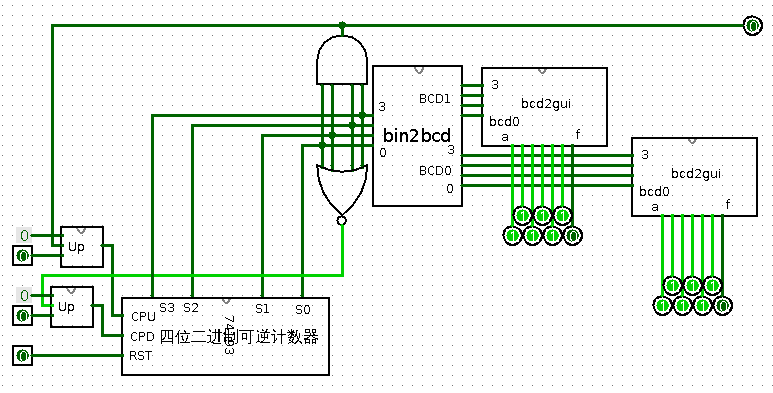


图2-13 实验室门禁系统电路

7、实验后的思考

**（1）这两次实验的难点你认为在哪些方面？**

使用logisim正确地绘制复杂逻辑。学会使用logisim。

**（2）你是如何解决的？**

使用类似prototype-based object-oriented programming的良好封装。使用Google。

**（3）意见和建议**

无。