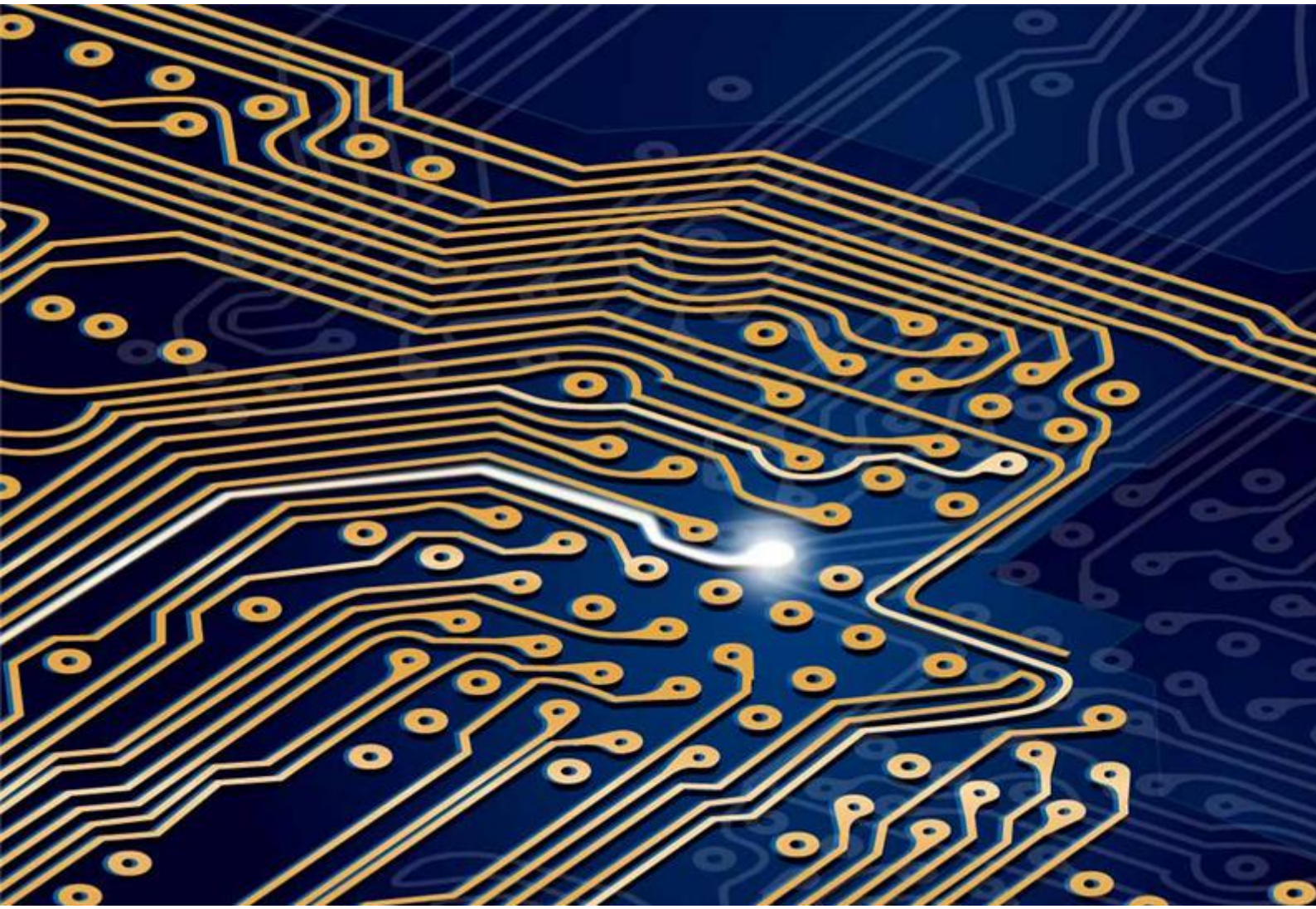
数字电路与逻辑设计

实验指导



目录

Dream Logic 简介

获得人生中的成功需要的专注与坚持不懈多过天才与机会。
——C.W.Wendte

Dream Logic 是一款集成了数字电路、模拟电路以及数字系统、数模混合系统设计与仿真的 EDA 软件，Dream Logic 具有元器件库丰富、原理图绘制方便、交互式仿真、可视化程度高、扩展性强等特点。Dream Logic 还配有精心设计的实验教学方案以及大量的电路设计图样例，便于读者学习和研究。

Dream Logic 主要具有以下特色：

- **与 CodeCode.net 平台深度整合。**Dream Logic 已经完全接入了 CodeCode.net 平台。首先，用户可以使用在 CodeCode.net 注册的账号登录 Dream Logic 软件，获取该软件的使用授权；其次，Dream Logic 需要用到的所有实验项目模板，都托管在 CodeCode.net 上的 Git 远程库中，用户可以使用 Dream Logic 集成的 Git 功能，十分方便的将这些 Git 远程库克隆到本地使用；同时，用户使用 Dream Logic 设计的电路图，也可以推送到 CodeCode.net 平台上进行托管，然后使用浏览器就可以直接查看电路图。
- **元器件种类齐全。**Dream Logic 中提供了十几个型号库，其中包含各类数字器件和模拟器件，为不同功能、不同规模的电路和系统设计与仿真提供了强大的支撑。在 Dream Logic 的实际使用过程中，用户还可以根据需求自定义新的功能器件。
- **绘制原理图方便快捷。**绘制原理图时，用户可以快速地将元器件放置到原理图中，然后通过网络、标签、总线等多样化的完成电路连线。
- **可绘制模块化的、可复用的功能电路。**可将任意一个具有特定功能的电路封装成一个电路模块，从而可以重复使用。该功能极大地方便了复杂电路系统的设计，也符合复杂系统自顶向下功能模块划分、自底向上功能模块实现的设计原则。而且各个模块可以进行单独仿真，从而验证其功能的正确性。这就使诸如八位模型计算机、MIPS 微体系结构处理器等复杂系统的设计成为可能。
- **可视化程度高。**在仿真过程中，可以高亮显示具有高电平的网络，还可以使用指示灯、数码管等器件显示电路的输出，这就使得电路的工作状态一目了然。此外，还提供了逻辑分析仪、电压表等可视化的仪器仪表实时显示电路的状态。在仿真计算机系统时，可以显示寄存器、存储器、指令源代码等信息，还可以进入子模块电路的内部查看其工作状态。这些可视化功能大大方便了电路的功能测试与验证。
- **交互式仿真。**在仿真过程中，用户可以通过手动控制单周期时钟源、交互式数字常量等器件实时控制电路的输入，从而动态改变电路工作的状态，达到交互式仿真的目的。
- **提供多种体系结构微处理器。**提供了适合用于完成计算机组成原理实验的八位模型机 DM1000，适合用于完成微机原理与接口技术实验的 Intel 8086 微处理器，适合用于计算机体系结构实验的十六位单周期 MIPS 微处理器、十六位多周期 MIPS 微处理器以及十六位五级流水线 MIPS 微处理器，用户可以将它们用于学习或研究，或者以它们为基础设计出新的微处理器。
- **完全开源的指令集和汇编器。**为 8086、DM1000 和 MIPS 微处理器提供了开源的指令汇编器和微指令汇编器，这些软件完全使用 C 语言编写，用户可以通过修改源代码来添加自己设计的指令和微指令，或者以它们为基础编写出新的汇编器。

实验简介

数字电路实验是按照组合逻辑电路实验、时序逻辑电路实验、有限状态机实验、数字系统设计实验的顺序从易到难安排的。每个实验在任务的安排上遵循由易到难的原则，采用由功能验证到功能设计实现的实验模式，满足不同层次学生的实验需要，使每一个学生都能从实验中受益。

组合逻辑电路实验包含基本门电路实验、编码器和译码器实验以及加法器实验，通过这些实验，读者可以掌握基本逻辑门设计组合逻辑电路的方法，熟悉 74LS 系列常用组合逻辑芯片的功能和使用方法，并使用这些芯片设计一定规模的组合逻辑电路，实现相应的功能。

时序逻辑电路实验以触发器、计数器和寄存器为代表，包含 74LS 系列的主要时序逻辑芯片，通过这些实验，读者可以熟悉 74LS 系列时序逻辑芯片的功能和使用方法，掌握时序逻辑电路的特点和一般的设计方法。

有限状态机实验以交通灯设计为代表，通过交通灯的设计，读者可掌握有限状态机的概念，应用场景以及设计方法，学会使用有限状态机设计复杂数字系统的控制器。

数字系统设计实验属于综合设计实验，该部分实验需要读者使用前面实验中掌握的知识和方法，灵活运用组合逻辑电路，时序逻辑电路以及有限状态机，分模块实现数字系统的各个子功能，然后构建一个功能复杂的数字系统。通过这些实验，将加深读者对数字电路理论知识的理解，培养读者的知识应用和数字系统设计能力。

数字电路课程是计算机专业以及电子专业的基本理论课程，数字电路实验可为读者后续的计算机组成原理等课程打下坚实的基础。

实验 1 集成门电路的组合电路

实验性质：验证+设计
建议学时：2 学时

一、 实验目的

- 熟悉 Dream Logic 的基本使用方法。
- 熟悉在 Dream Logic 上绘制组合逻辑电路并仿真的过程。
- 学会设计具有特定功能的基本组合逻辑电路，并通过仿真测试其逻辑功能、修改其电路以完善最终的设计。

二、 实验内容

请读者按照下面的步骤完成实验内容，掌握 Dream Logic 软件和 CodeCode.net 平台的基本使用方法，为完成后续实验做好准备。

2.1 任务（一）实验软件的基本使用方法

从 CodeCode.net 平台领取任务

CodeCode.net 平台是用于教师在线布置实验任务，并统一管理学生提交的作业的平台。如果没有使用到 CodeCode.net 平台，可以跳过此部分。

1. 读者通过浏览器访问 https://www.codecode.net，可以打开 CodeCode.net 平台的登录页面。
2. 在登录页面中，读者输入帐号和密码，点击“登录”按钮，可以登录 CodeCode.net 平台。
3. 登录成功后，在“课程”列表页面中，可以找到数字电路相关的实验课程。点击此课程的链接，可以进入该课程的详细信息页面。
4. 在课程的详细信息页面中，可以查看“课程描述”，该信息对于完成实验十分重要，建议读者认真阅读。点击左侧导航中的“任务”链接，可以打开任务列表页面。
5. 在任务列表中找到本次实验对应的任务，点击右侧的“领取任务”按钮，可以进入“领取任务”页面。
6. 在“领取任务”页面中，“新建项目名称”、“新建项目路径”、“项目所在的群组”一般使用默认值即可。点击“领取任务”按钮后，可以创建一个个人项目用于完成本次实验，并自动跳转到该项目的详情页面。
7. 个人项目的详情页面主要包括左侧的导航栏、项目信息、文件列表、readme.md 文件内容等，如图 1-1 所示。
8. 在图 1-1 中，点击红色方框中的按钮，可以复制当前项目的 URL，在本实验后面的练习中会使用这个 URL 将此项目克隆到读者计算机的本地磁盘中，从而供读者进行修改。

图 1-1：个人项目的详情页面

启动 Dream Logic

在安装有 Dream Logic 的计算机上，可以使用两种方法来启动 Dream Logic：

- 在桌面上双击“Engintime Dream Logic”图标。
  或者
- 点击“开始”菜单，在“所有程序”中的“Engintime Dream Logic”中选择“Engintime Dream Logic”即可启动程序。

启动 Dream Logic 后会首先打开“登录”窗口，可以使用两种方法完成登录。

- 如果读者在CodeCode.net平台上注册了帐号，在连接互联网的情况下，可以使用CodeCode.net 平台中已注册的用户名和密码进行登录。
- 如果读者还没有 CodeCode.net 平台上的帐号，可以点击左下角“从加密锁获取授权”按钮，获取授权之后，完成登录。

主窗口布局

Dream Logic 的主窗口布局由下面的若干元素组成：

- 顶部的菜单栏、工具栏。
- 停靠在左侧的有项目管理器窗口、型号库窗口等。
- 停靠在右侧的通常是属性窗口。
- 打开原理图文档后，中间区域是电路图绘制区域。

提示：菜单栏、工具栏和各种工具窗口的位置可以随意拖动。如果想恢复窗口的默认布局，选择“窗口”菜单中的“重置窗口布局”即可。

将 CodeCode.net 平台上的项目克隆到本地

如果读者从 CodeCode.net 平台领取了任务，可以在领取任务后使用 Dream Logic 软件将读者的个人项目克隆到本地磁盘中，从而对其进行修改；如果读者没有 CodeCode.net 平台账号无法领取任务，那就只能直接将实验模板项目克隆到本地磁盘中了。

- **将领取任务后新建的个人项目克隆到本地**

1. 选择 Dream Logic 菜单“文件->新建->从 Git 远程库新建项目”，打开“从 Git 远程库新建项目”对话框。
2. 在“Git 远程库 URL(G)”中填入读者个人项目 Git 远程库的 URL 地址（在图 1-1 中，点击红色方框中的按钮，可以复制 URL）。
3. “项目文件夹名称”填入“test”。
4. “项目位置”选择新建项目需要保存到的磁盘目录。
5. 点击“确定”按钮后会弹出一个 Windows 控制台窗口，并开始运行 Git 克隆命令将 Git 远程库克隆到本地，一定要等克隆成功后再关闭该窗口。
6. 克隆项目成功后，在对话框中选择“克隆成功，打开新建项目”就可以打开新建的项目。

- **直接将实验模板项目克隆到本地**

由于 CodeCode.net 平台上提供的实验模板是开放的，所有的人都可以访问。所以，Dream Logic 也可以直接将实验模板克隆到本地磁盘。步骤如下：

1. 选择 Dream Logic 菜单“文件->新建->从 Git 远程库新建项目”，打开“从 Git 远程库新建项目”对话框。
2. 在“Git 远程库 URL(G)”中填入空白项目模板 Git 远程库的 URL 地址：
   https://www.codecode.net/engintime/Dream-Logic/Project-Template/blank.git
3. “项目文件夹名称”填入“test”。
4. “项目位置”选择新建项目需要保存到的磁盘目录。
5. 点击“确定”按钮后会弹出一个 Windows 控制台窗口，并开始运行 Git 克隆命令将 Git 远程库克隆到本地，一定要等克隆成功后再关闭该窗口。
6. 克隆项目成功后，在对话框中选择“克隆成功，打开新建项目”就可以打开新建的项目。

注意：

1.为了能正常使用从 Git 远程库新建项目功能，用户需要在本地安装 Git 客户端程序，并且在安装 Git 的过程中会有一个步骤询问是否设置 Windows 环境变量，此时一定要设置上 Windows 环境变量。

2. 文件路径中不允许包含中文，请读者在使用之初就养成使用英文命名项目或文件的习惯。

新建项目成功后，Dream Logic 会自动打开项目，并在左侧的“项目管理器”窗口中显示项目包含的所有文件。可通过菜单“视图->项目管理器”打开项目管理器窗口。刚刚新建的项目下已经包含了一个空白的原理图文件 top.dlsche，并且已经自动打开了此文件，读者可以在此原理图文件中绘制电路图，并对电路图进行仿真。

原理图的平移和缩放

读者可以用鼠标将原理图平移或者缩放到合适的位置。方法为：

1. 在绘图区按下鼠标中键并拖动鼠标，可移动绘图区域。
2. 在绘图区滚动鼠标滚轮可以缩放绘图区域。在软件底部的状态栏中显示了当前光标的位置坐标以及缩放比例。

绘制简单的原理图

接下来，读者可以按照下面的步骤在原理图文件 top.dlsche 中绘制一个验证“与门”功能的组合逻辑电路，进而掌握在原理图中绘制电路的基本方法。步骤如下：

1. 选择菜单栏中的“视图->型号库”，打开“型号库”窗口。Dream Logic 在该窗口中提供了大量的常用器件，并根据器件的类别将这些器件放到了多个型号库中，方便用户选用。
2. 在“型号库”窗口顶部的下拉列表中选择“逻辑门.dlcomp”，打开逻辑门型号库。
3. 在逻辑门型号库中，双击与门“AND”所在行，然后将鼠标移动到原理图中，就可以在原理图中放置器件了。
4. 将鼠标移动到原理图中的任意位置（同时可以通过鼠标中键平移和缩放原理图，以寻找一个合适的位置），然后单击左键，完成器件的放置。这样，就在原理图中绘制了一个与门。

小技巧：放置完器件后，按下 Tab 键可以重复放置同型号的器件，而按下 Esc 键可以退出器件的放置。将鼠标移到器件上并单击左键可选中该器件，此时会在右侧的“属性”窗口中显示选中对象的属性值。

5. 在型号库顶部的下拉列表中选择“数字信号源.dlcomp”，打开数字信号源型号库。
6. 在原理图中放置“时钟”和“一位交互式数字信号源”器件。它们将用于控制与门的两个输入信号。
7. 打开“数字信号显示.dlcomp”型号库，在原理图中放置一个“数字探针”，该探针用于检测与门的输出信号。当探针检测到高电平时灯亮，低电平时灯灭。
8. 所有器件放置完成后，如图 1-2 所示。在绘制原理图时，通常遵循左侧输入，右侧输出的原则。

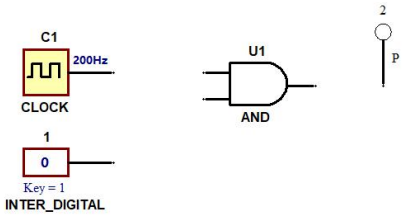


图 1-2：在原理图中绘制器件

器件绘制完成后，需要将各个器件的管脚连接起来，形成一个可工作的电路。步骤如下：

1. 选择菜单栏中的“绘制->网络”，开始绘制网络连线。
2. 将光标移动到器件管脚的末端，如图 1-3 所示。管脚末端会有一个白色的小点。

图 1-3：管脚末端的连接点

3. 当光标捕获到管脚末端的连接点时，会显示一个红色的交叉线。这就意味着，如果在此时单击鼠标左键，就会以该点为起点开始绘制一条网络，并且该网络与管脚是连接到一起的。
4. 移动光标到另一个器件管脚末端的连接点，出现红色交叉线后，单击左键，这样就在两个管脚之间绘制了一条网络。

注意，在绘制网络的过程中，网络节点之间的细实线或者虚线是绘制网络的预览，单击左键后细实线变为粗实线才表示一个网络线段绘制成功。此外，光标除了能捕获管脚末端的连接点以外，还能捕获网络上的任意点。

5. 按下键盘上的“Esc”键可结束一条网络的绘制。
6. 按下 Tab 键可以继续绘制新的网络连线。

7. 将各个器件的所有管脚使用网络完成连接后，如图 1-4 所示。

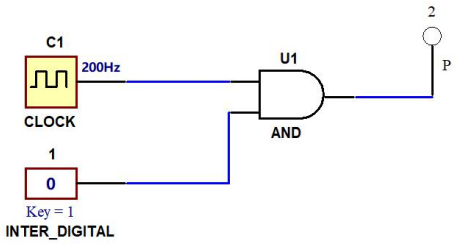


图 1-4：使用网络连接器件的管脚

交互式仿真

通过前面的实验内容，读者已经在原理图 top.dlsche 中绘制了一个“与门”的功能验证电路。接下来，读者按照下面的步骤，使用 Dream Logic 提供的交互式仿真功能，检验与门的逻辑功能：

1. 首先，为了仿真的效果比较便于观察，需要先将时钟频率设置的低一些。将鼠标移动到时钟上并单击选中，将其属性栏中的时钟频率修改为 10，也就是将时钟频率设置为 10Hz。
2. 选择菜单栏中的“仿真->启动仿真”，或按下键盘快捷键“F5”启动仿真。启动仿真前需要确保当前打开的原理图是 top.dlsche，因为任何时刻启动仿真都是对当前打开的原理图进行仿真。
3. 启动仿真后，观察电路中网络和器件管脚颜色的变化，以及指示灯的亮灭情况。若网络或器件管脚为红色，则表示数字信号 1，也就是高电平；若为浅灰色，则表示数字信号 0，即低电平。
4. 按下键盘上的按键“1”可以让原理图中的一位交互式数字信号在 0 和 1 之间切换，根据仿真结果检验与门的逻辑。
5. 选择菜单栏中的“仿真->结束仿真”可以结束仿真。

读者可以按照下面的步骤将时钟替换为一个一位交互式数字信号源再进行仿真。

1. 将鼠标移动到原理图中的时钟上，单击左键，选中这个时钟器件。
2. 按下键盘上的“delete”键，或选择菜单栏中的“编辑->删除”，删除选中的时钟器件。
3. 在原理图中添加一个一位交互式数字信号源器件，替换时钟作为与门的输入。如图 1-5 所示：

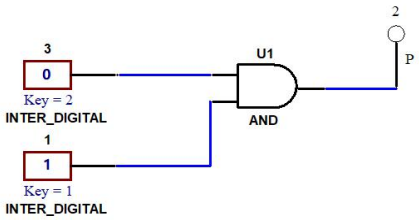


图 1-5：使用一位交互式数字信号源替换时钟

4. 启动仿真，通过键盘上的按键“1”和“2”控制两个数字信号源的输出，观察电路仿真情况。进一步验证与门的逻辑功能。
5. 结束仿真。

原理图常用操作

通过上面的练习，读者对原理图的绘制和仿真有了基本的掌握。接下来是绘制原理图过程中常用操作的详细介绍，读者在后续实验过程中可以参考该部分内容。

- **平移原理图。**在绘图区按下鼠标中键并拖动鼠标，可移动绘图区域。
- **缩放原理图。**在绘图区滚动鼠标滚轮可以缩放绘图区域。
- **新建原理图文件。**方法如下：
  1. 在左侧的“项目管理器”窗口中右键点击项目名称节点（根节点）。
  2. 在弹出的菜单中选择“添加->新建文件”，会弹出“添加新文件”对话框。

3. 在“添加新文件”对话框中，选择“原理图文件”模板，在“文件名称”中输入新建的原理文件名称。文件位置默认就是项目所在文件夹，所以一般不需要修改。
4. 点击“确定”按钮后，新建的原理图文件就会被添加到项目中，并自动打开。

- **添加器件。**绘制原理图时，首先需要在原理图中添加所需器件。所有器件都只能从 Dream Logic 提供的型号库中获取。打开型号库的方法为:
  1. 选择菜单栏中的“视图->型号库”，打开型号库窗口。
  2. 在型号库顶部的下拉列表中选择合适的型号库。被选中型号库中的所有型号都会显示在型号列表中，选中某个型号后会在下方的预览窗口中显示型号的图符。

  使用型号库中的型号在原理图中绘制器件的方法有以下三种：
  1. 在型号库中选中一个型号，点击“型号库”窗口工具栏中的“放置器件”，然后在原理图中的合适位置单击鼠标左键完成器件放置。
  2. 在型号库中的一个型号上单击右键，在弹出的菜单中选择“放置器件”，然后在原理图中的合适位置单击鼠标左键完成器件放置。
  3. 直接双击型号库中选中一个型号，然后在原理图中的合适位置单击鼠标左键完成器件放置。

小技巧：在器件随鼠标移动的过程中，用户可以通过按下键盘上的“Esc”按键取消绘制器件命令。在器件随鼠标移动的过程中，用户可以按下键盘上的空格键旋转器件到一个合适的方向后，再放置到原理图中。在器件放置完成后，可通过反复按下“Tab”重复放置相同型号的器件。

- **器件预选中、选中**。将鼠标移动到器件上，器件颜色会改变，说明器件被预选中了，单击鼠标左键就可以选中该器件。
- **全选**。按下键盘上的“Ctrl+A”可以选中原理图中的所有对象。
- **矩形选择**。选择菜单栏“编辑”中的“矩形选择”，然后在原理图中的合适位置点击鼠标左键，作为矩形的一个顶点，接着移动鼠标，可以画出一个矩形，再次单击左键，就可以选中矩形区域内的所有对象。矩形选择分为左右两种矩形，它们绘制的矩形选择区域颜色不同，选择方式也不同。其中右矩形只会选中那些完全包裹在矩形区域中的对象，而左矩形会选中那些与矩形区域有交集的对象。
- **多选**。单击预选中的器件的同时按下“Ctrl”键，可以逐个选中多个对象。
- **取消选中。**单击已经被选中的对象，就可以取消选中此对象。
- **旋转器件。**可使用旋转功能旋转器件，有以下三种旋转方法：
  1) 选中器件后，点击菜单栏中的“编辑”，选择“旋转”，即可将所选器件逆时针旋转 90°。
  2) 选中器件后，单击右键，选择菜单中的“旋转”，即可将所选器件逆时针旋转 90°。
  3) 选中器件后，按下空格键，可将所选器件逆时针旋转 90°。

  使用同样的方法，可以对任何一个或多个选中对象进行旋转。
- **平移器件。**当器件的位置需要调整时，可使用平移功能。首先选中器件，然后点击菜单栏“编辑”中的“平移”，或者在选中器件后，单击右键，选择菜单中的“平移”。然后，单击原理图中任意一点作为平移起始点，之后选中的器件就会随着光标移动。最后，在合适的位置单击鼠标左键即可将选中器件平移到新的位置。使用同样的方法，可以对任何一个或多个选中对象进行平移。
- **修改对象属性。**选中对象后，可以在右侧的“属性”窗口中查看或者修改对象的属性。选择菜单栏“视图”中的“属性”可以打开“属性”窗口。
- **单周期时钟**。单周期时钟是一个数字信号源器件，在“数字信号源”型号库中，如图 1-6 所示。其中的“key=A”是单周期时钟的控制键，对应于键盘上的按键“A”。在原理图中选中单周期时钟器件后，可在属性窗口中将其控制键修改为键盘上的其它按键。当启动仿真后，单周期时钟默认输出高电平信号，按下其控制键后，开始输出低电平信号，抬起按键后恢复输出高电平信号。这样，在按键按下然后抬起的过程中，产生了一个负脉冲，包含一个时钟下降沿和一个时钟上升沿。

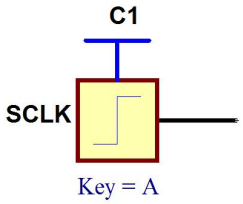


图 1-6：单周期时钟

- **一位交互式数字信号**。一位交互式数字信号是一个数字信号源器件，在“数字信号源”型号库中，如图 1-7 所示。其中的“key=4”是其控制键，对应于键盘上的按键“4”。在原理图中选中一位交互式数字信号器件后，可在属性窗口中将其控制键修改为键盘上的其它按键。当启动仿真后，一位交互式数字信号显示数字 0 时，输出低电平信号；显示数字 1 时，输出高电平信号。通过按下控制键可以使一位交互式数字信号的输出在高电平和低电平之间切换。

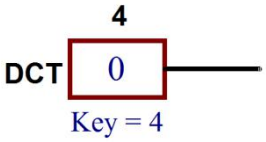


图 1-7：一位交互式数字信号

- **使用网络连接器件的管脚。**当器件放置完成后，可以使用“网络”连接器件的管脚，操作步骤为：
  1. 选择菜单栏中的“绘制->网络”，将光标移动到器件管脚的末端，当光标捕获到管脚末端的连接点时，会显示红色交叉线，此时单击鼠标左键，就会以该点为起点开始绘制一条网络。
  2. 移动光标到另一个器件管脚末端的连接点，出现红色交叉线后，单击左键，这样就在两个管脚之间绘制了一条网络。
  3. 按下键盘上的“Esc”可结束网络的绘制。

小技巧：如果需要连接的两个管脚距离较远，网络可能需要由多个线段组成，在绘制的过程中就需要多次点击鼠标左键，如果点击鼠标左键后发现位置不合适，可以选择“编辑”菜单中的“撤销”（快捷键 Ctrl+Z），取消刚刚点击左键绘制的网络线段。在绘制网络的过程中，按下空格键可以在网络的四种绘制模式之间进行切换。

- **使用网络标签连接器件的管脚。**当两个器件的管脚距离较远，不方便使用网络进行连接时，通常需要使用两个具有相同名称的网络标签为它们建立连接关系。步骤如下：
  1. 选择菜单栏中的“绘制->网络标签”。
  2. 将光标移动到器件管脚的末端，当光标捕获到管脚末端的连接点时，会显示红色交叉线，此时单击鼠标左键，就会在此管脚上放置一个网络标签。
  3. 使用同样的方法在另外一个管脚上也放置一个网络标签。
  4. 按下“Ctrl”键的同时，用鼠标依次点击刚刚放置的两个网络标签，这样可以同时选中两个网络标签对象。
  5. 在“属性”窗口中的“名称”栏中输入一个合适的网络名称，例如“NetLable1”。这样，两个被选中的网络标签就被修改为相同的名称了，它们标识的两个管脚之间就建立了连接关系。图 1-8 是一个一位交互式数字信号与数字探针使用网络标签完成连接后的原理图。

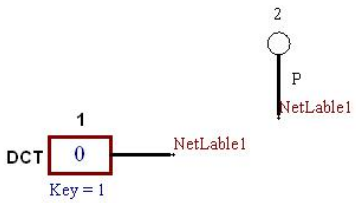


图 1-8：使用网络标签连接器件的管脚

注意:所有的网络和网络标签都具有一个网络名称属性。选中网络或者网络标签后，在“属性”窗口中可以查看他们的网络名称。通常情况下，网络标签是可以用来给网络命名的。而且本质上，只要网络或者网络标签具有相同的名称，就是表示它们之间建立了连接关系。所以，可以在多个管脚末端的连接点或者网络上放置具有相同名称的网络标签来为它们建立连接关系。

- **打印原理图。**选择“文件”菜单中的“打印”，可以打印当前原理图，打印纸张等属性可以自行设置。如果当前使用的计算机没有连接打印机，可以先将原理图打印到 XPS 文件，然后将 XPS 文件放入连接了打印机的计算机中打印即可。

注意，打印前，需将全部要打印的内容移动到坐标系的第一象限内（红蓝坐标原点右上方的区域）。因为只能将坐标系第一象限的内容打印出来。

提交作业

如果读者是通过从 CodeCode.net 平台领取任务创建的个人项目，并将个人项目克隆到本地进行实验，实验结束后可以将本地已更改的项目再推送到 CodeCode.net 平台的个人项目中，方便教师通过 CodeCode.net 平台查看读者提交的作业。步骤如下：

1. 在左侧的“项目管理器”窗口中，右键点击项目节点，在弹出的菜单中，选择“Git”中的“推送当前分支到 Git 远程库”菜单项，弹出“推送当前分支到 Git 远程库”的对话框。
2. 在“推送当前分支到 Git 远程库”对话框中，输入本次项目更改的提示信息。
3. 点击“推送”按钮，可以将当前项目推送到 CodeCode.net 平台的个人项目中。

可以按照下面的步骤查看推送后的项目：

1. 使用浏览器访问 https://www.codecode.net,使用已注册的帐号登录 CodeCode.net 平台。
2. 在“课程”列表中选择相应的课程，打开课程的详细信息页面，点击左侧导航栏的“任务”链接，打开任务列表页面。
3. 在任务列表页面打开本次实验对应的任务，在任务详情中点击“查看项目详情”按钮，可以跳转到读者的个人项目页面。
4. 在个人项目页面中，可以查看当前项目的全部内容。
5. 在项目左侧的导航栏中点击“仓库”中的“提交”链接，可以打开提交列表页面。
6. 在提交列表页面中，点击最后一次提交，可以查看此次提交发生变更的文件。

技巧：在 Dream Logic 的项目管理器窗口中，选中项目节点，点击鼠标右键，在弹出的右键菜单中，选择“Git”中的“使用浏览器访问 Git Origin”菜单项，可以自动打开本地项目在 CodeCode.net 平台上对应的个人项目。

2.2 任务（二）简单的组合逻辑电路

预备知识

数字电路可以分为**组合逻辑电路**和**时序逻辑电路**两大类。

组合逻辑电路的输出仅仅取决于输入的值。换句话说，它组合当前输入值来确定输出值。例如，逻辑门就是组合电路。时序电路的输出取决于当前输入值和之前的输入值。换句话说，它取决于输入的序列。组合电路是没有记忆的，但是时序电路是有记忆的。

组合逻辑电路在结构上不存在输出到输入的反馈通路，因此输出状态不影响输入状态。组合逻辑的特点是：任意时刻的输出状态取决于该时刻输入信号的状态，而与信号作用前电路状态无关。

组合逻辑分析，就是根据已知逻辑电路图，找出组合逻辑电路的输入与输出关系，确定在什么样的输入取值组合下对应的输出为 1。

组合逻辑电路是数字电路的最基本的一类，设计比较简单。一般根据设计要求，列出真值表，或用卡诺图，或用逻辑表达式进行化简，最后得到逻辑电路图并画出具体电路。有了逻辑图，在画出具体电路时，

先要根据实际条件选取器件，器件的数量和类型要尽可能少，而且要注意用摩根公式进行正负逻辑之间的转换。在实际应用中，可靠性高、成本低、维修方便是选择元器件的重要原则。

所谓的“正逻辑”，就是自变量和函数均是高电平有效，即把高电平定为逻辑“1”，把低电平定为逻辑“0”。“负逻辑”则正好相反，把高电平定为逻辑“0”，而低电平定为逻辑“1”。

- **奇偶校验电路**

奇偶检验电路是最简单的检验方法，广泛用于计算机和数字通信中。欲传送的二进制代码称为信息码，例如 $A_1A_2A_3$；在传输信道中，由于干扰等原因，接收到的码可能发生了变化，例如发送的是 101，而接收到的却是 100。一般一位错的可能性远大于两位错、三位错。假定我们只考虑一位错，那么如何能发现这种错误呢？利用奇偶校验是很简便的。所谓奇偶检验，就是在信息码之外再加一位校验码 F。校验码 F 是在发送端产生并和其他信息码 $A_1A_2A_3$ 一同发送到接收端。这样发送的代码就变成 $A_1A_2A_3F$ 了，这个码通常叫做码字。接收端根据所接收到的码字，就可以判断出传输是否出错。如果有错，就要求重发。码字中 1 的个数如果是奇数，就是奇校验；如果是偶数，就叫偶校验。

当信息码 $A_1A_2A_3$ 中 1 的个数为奇数时，F=1，则码字中 1 的个数就是偶数；当信息码 $A_1A_2A_3$ 中 1 的个数为偶数时，F=0，码字中 1 的个数依然是偶数。这就是偶校验。

当接收端码字 $A_1A_2A_3F$ 中 1 的个数为偶数时，检错码 E=0，说明传输无误；接收端码字 $A_1A_2A_3F$ 中 1 的个数为奇数时，E=1，说明传输中发生了错误，需要重发。

- **“菊花链”电路**

在计算机中，有一种 CPU 查询中断源的电路，叫“菊花链”。CPU 每执行完一条机器指令之后，都要查看一下有没有中断请求，如果有，就要找到中断源，然后转入相应的中断服务程序；如果没有，就继续执行下一条机器指令。“菊花链”电路的任务就是用来查询中断源是哪一个。

假定中断源有 3 个，分别是 I_1、I_2、I_3，优先级依次降低，低电平表示有中断；CPU 送来的查询信号为 B，也是低电平有效。图 1-9 给出了这种“菊花链 ”电路。当 B 为高电平时，无论中断源有无请求，$U_{2A}U_{2B}U_{2C}$ 的 3 个输出 I_1'、I_2'、I_3'均为 0。当 B 为低电平时，若 I_1、I_2、I_3都为高电平，即无中断请求，$U_{2A}U_{2B}U_{2C}$的 3 个输出为 0 保持不变；若 I_1为低电平，不论 I_2、I_3为何值，U_{2A}的输出 I_1'变为 1。而由于门 U_{1B}被封闭，所以，信号 B 的低电平就无法再向右传送，于是 I_2'、I_3'也就维持 0 不变。只有 I_1为高电平时，I_2的请求才能通过 U_{2B}送出去。故 I_1的优先级比 I_2高。同样，I_2的优先级比 I_3高。

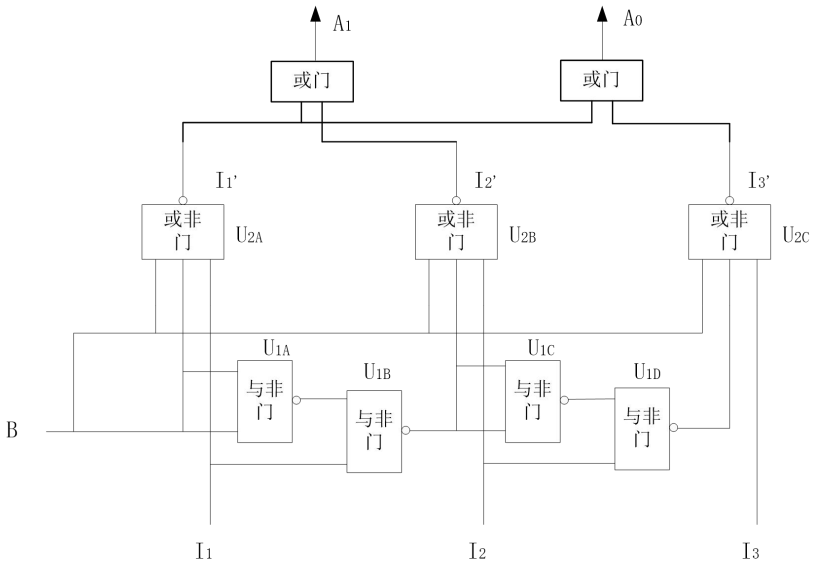


图 1-9：三中断源自动判优电路

CPU 要识别三个中断请求中优先级最高的是哪个源，还需要对以上的输出 I_1'、I_2'、I_3'进行编码。图 1-9 中使用 2 个或门编码 I_1'、I_2'、I_3'，输出为 A_1A_0，注意要留一个（如 A_1A_0=00）分配给没有任何源请求的状态。如果有更多的中断源，“菊花链”还可以继续扩展下去。“菊花链”也叫串行排队电路。

在掌握了本任务的预备知识后，读者可以按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/Digital-Circuit/Lab001.git

一位全加器

请读者在数字电路相关的教材中学习一位全加器的原理，然后按照下面的步骤查看一位全加器的原理图，并对其进行仿真验证：

1. 打开项目下的一位全加器原理图文件 Adder.dlsche。
2. 启动仿真后，根据表 1-1 给出的真值表验证全加器的功能。
3. 尝试写出输出端 S（和），以及输出端 Co（进位）的逻辑表达式，掌握一位全加器的工作原理。

表 1-1：一位全加器真值表(1 表示高电平，0 表示低电平)

输入端			输出端	
Ci	A	B	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

奇校验电路

请读者在数字电路相关的教材中学习奇校验电路的原理，然后按照下面的步骤查看奇校验电路的原理图，并对其进行仿真验证。步骤如下：

1. 在之前验证一位全加器时克隆的项目中找到原理图文件 oddcheck.dlsche，打开此文件就可以看到奇校验电路的原理图。
2. 启动仿真，并将奇校验电路真值表 1-2 填写完整。验证奇校验电路的功能时，假设接收端总是接收到正确的校验码 F。仿真结束后，尝试写出校验码 F 和检错码 E 的逻辑表达式。并说明奇校验电路的局限性。

表 1-2：奇校验真值表

发送端				接收端			检错码
A1	A2	A3	F	B1	B2	B3	E
0	0	0		0	0	0	
0	1	0		0	1	0	
1	1	0		1	1	0	
0	1	0		0	0	0	
1	0	0		1	1	0	
1	0	1		1	1	0	

3. 修改原理图文件 oddcheck.dlshche，将此奇校验电路扩展为可用于校验 4 位数据 A1~A4 的电路。并通过仿真确保电路能够正常工作。
4. 提交作业。

2.3 任务（三）设计三中断源自动判优电路

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/blank.git

请读者按照下面的步骤完成本次任务：

1. 参考图 1-9，在原理图文件 top.dlsche 中设计一个三中断源自动判优电路。需要用到的逻辑门可以在型号库“逻辑门”中找到。
2. 使用型号库“数字信号源”中的一位交互式数字信号控制输入信号 B、I_1、I_2、I_3。使用型号库“数字信号显示”中的数字探针连接输出信号 A_0和 A_1，用于表示输出结果。
3. 启动仿真后，根据表 1-3 给出的真值表测试电路(1 表示高电平，0 表示低电平，X 表示任意值)，确保电路准确无误。
4. 提交作业。

表 1-3：三中断源自动判优电路真值表

查询信号	中断源			输出	
B	I_1	I_2	I_3	A_0	A_1
1	X	X	X	0	0
0	0	0	0	1	1
0	0	0	1	1	1
0	0	1	0	1	1
0	0	1	1	1	1
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	0	1
0	1	1	1	0	0

2.4 任务（四）设计偶校验电路

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/blank.git

请读者按照下面的步骤完成本次任务：

1. 参考本实验中的奇校验电路，在原理图文件 top.dlsche 中设计一个偶校验电路。假设接收端总

是能接收到正确的校验码 F。当检错码 E=0 时，说明接收的数据无误。尝试写出校验码 F 和检错码 E 的逻辑表达式。

2. 参考表 1-2 设计偶校验真值表，要求至少包含五行数据，并且包含校验成功和校验失败的情况。
3. 提交作业。

2.5 任务（五）设计一位全减器

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/blank.git

请读者按照下面的步骤完成本次任务：

1. 参考本实验中的一位全加器，在原理图文件 top.dlsche 中设计一个一位全减器。当输入信号 Ci=1 时表示有借位输入，当输出信号 Co=1 时需要向高位借位。输入信号 A 是被减数，B 是减数，输出信号 S 是差。S 和 Co 可以接一个 16 进制 7 段数码管，该器件在型号库“数字信号显示”中。
2. 使用一位全减器真值表 1-4 测试电路，确保电路准确无误。
3. 提交作业

表 1-4：一位全减器真值表

输入端			输出端	
C_i	A	B	S	C_o
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

三、 思考与练习

1. 有一个水箱由大、小两台水泵 ML 和 MS 供水，如图 1-10 所示。水箱中设置了三个水位检测元件 A、B、C。水面低于检测元件时，检测元件给出高电平；水面高于检测元件时，检测元件给出低电平。现要求当水位超过 C 点时水泵停止工作；水位低于 C 点且高于 B 点时 MS 单独工作；水位低于 B 点而高于 A 点时 ML 单独工作；水位低于 A 点时 ML 和 MS 同时工作。试用门电路设计一个控制两台水泵的逻辑电路，要求尽量简单。

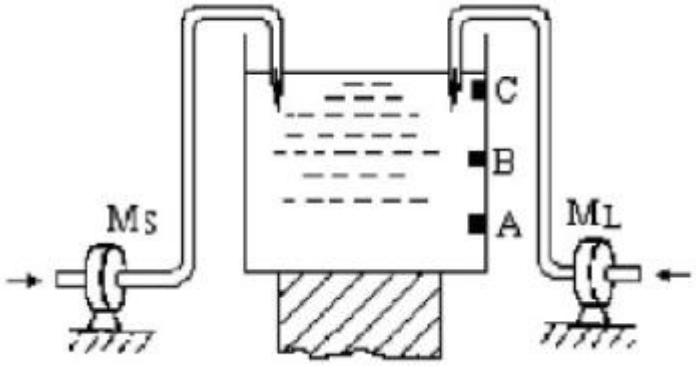


图 1-10：水箱供水

2. 设计一交通灯检测电路。红、绿、黄三只灯正常工作时只能一只灯亮，否则，将会发出交通灯故障信号。
3. 人类有四种基本血型 A、B、AB、O 型，输血者和受血者的血型必须符合一定的原则（如图 1-11），尝试设计一个检验输血者与受血者血型是否符合的电路。

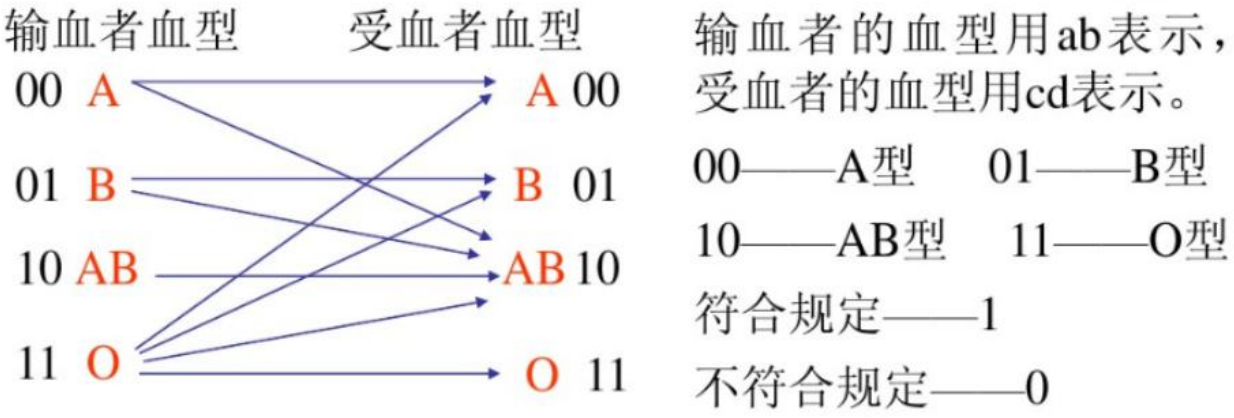


图 1-11：血型匹配规则

实验 2 编码器和译码器

实验性质：验证+设计
建议学时：2 学时

一、 实验目的

- 熟悉编码器的功能与基本应用。
- 熟悉译码器的功能与基本应用。
- 熟练使用编码器和译码器设计功能电路。

二、 预备知识

2.1 编码和译码

编码是把一组有特定含义（事件）的输入信号按一定的规律编成不同二进制代码输出的过程。一个事件对应唯一一组特定的编码，而一组特定的编码表示一个事件。用来完成编码工作的电路称为编码器。

译码是编码的反过程，就是把一组包含特定信息的二进制码翻译成一组高低电平信号，其中只有一个输出呈现有效状态。实现译码功能的组合逻辑电路称为译码器。它是数字系统中最常用的逻辑构件之一。常用的译码器有：2 线-4 线译码器，3 线-8 线译码器，由它们可以构成 4 线-16 线译码器、4 线-10 线译码器等。

2.2 多外设共享地址总线排队电路

计算机中的 CPU、各外设都可以访问存储器。但它们的优先级别不同，这应该事先规定好。可以将 CPU、各外设一律称作设备，假定设备 1 的优先级最高，依次是设备 2、设备 3……所谓“访问”，就是设备和存储器交换（读或写）数据的过程。该过程是这样进行的：某设备要访问存储器，首先要发出一个占用总线的请求信号，若没有比它优先级更高的设备也在请求的话，它的请求便得以响应。这时它发出的地址码就能送到地址总线上，存储器使用地址总线上地址码，确定要访问的单元（存放数据的地方），其中的数据就通过数据总线和该设备进行交换。现在我们只讨论从设备发出请求信号到把地址码送到地址总线这一段的工作原理。

为了和芯片的功能配合，假定设备要占用总线的请求为 0，不请求是 1；把各设备的请求信号加到优先编码器的输入端，进行优先编码，它输出的每一个有效码，既对应着一个一个的设备，又表示现在在申请占用总线的设备中，它的优先级最高（注意，一定要留一个无效码，表示没有任何设备申请）。然后，通过译码，让其输出打开一组传输门。所对应的设备通过该组门，把地址码送往地址总线（如图 2-1）。

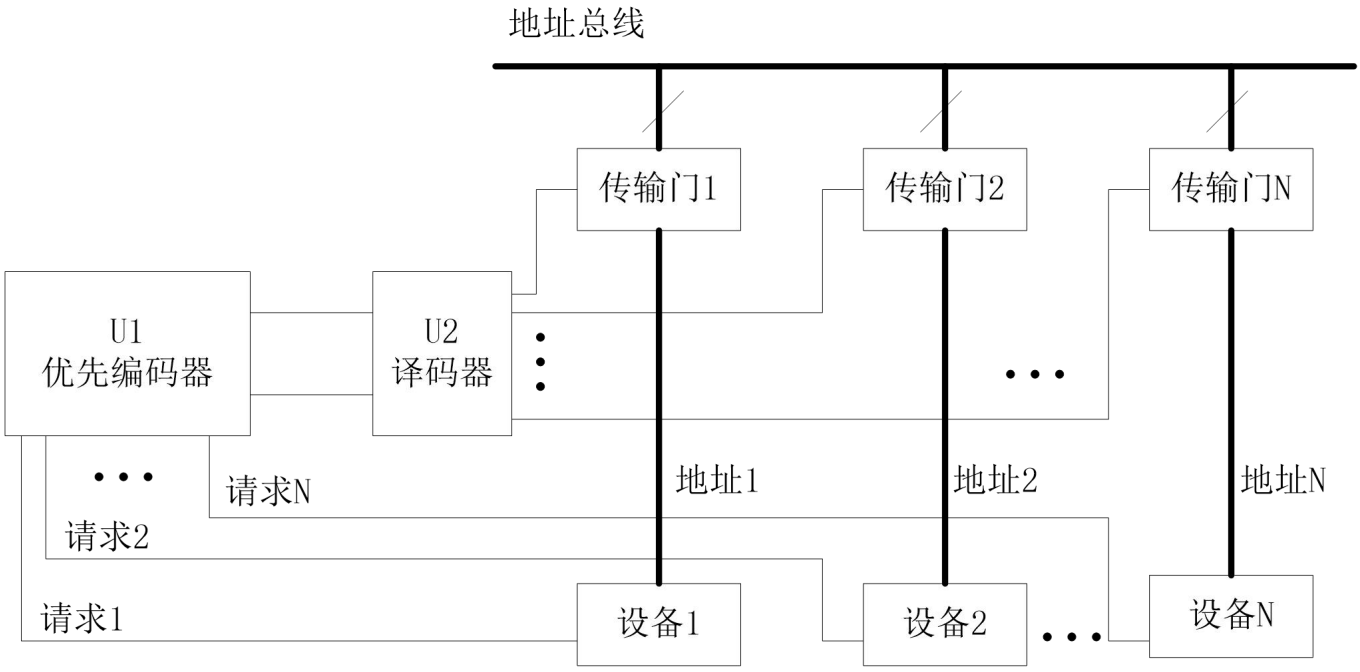


图 2-1：多外设共享地址总线排队电路原理图

三、 实验内容

3.1 任务（一）设计普通 BCD 编码器

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

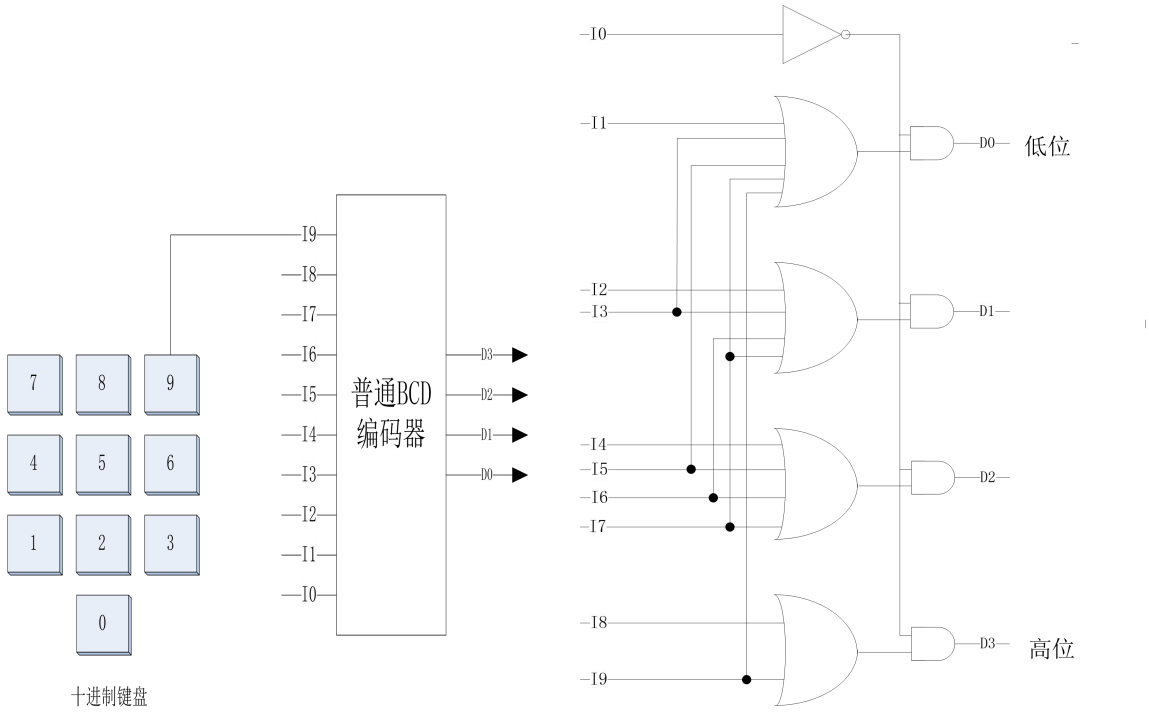
如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/blank.git

图 2-2 所示为一个普通 BCD 编码器的结构框图和逻辑电路图。普通 BCD 编码器有十个输入信号 I0~I9，每个输入信号连接一个代表十进制数（0 到 9）的信号，任意一个时刻只能有一个输入信号是高电平。其中输入信号 I0 为高电平时，表示输入 0；I9 为高电平时，表示输入 9。有四个输出信号 D0~D3，它们组成一个 BCD 码用于表示编码后的十进制数，其中 D3 为高位，D0 为低位。

请读者按照下面的步骤完成本次任务：

1. 参考图 2-2，在原理图文件 top.dlsche 中设计一个普通 BCD 编码器。
2. 对编码器进行仿真，填写普通 BCD 编码器的真值表 2-1。注意确认 BCD 码与十进制数之间的关系。



（a）结构框图　　（b）逻辑电路图

图 2-2：普通 BCD 编码器

表 2-1：普通 BCD 编码器真值表

输入	输出 BCD 码			
十进制数字信号	D3	D2	D1	D0
I0=1	0	0	0	0
I1=1	0	0	0	1
I2=1				
I3=1				
I4=1				
I5=1				
I6=1				
I7=1				
I8=1				
I9=1				

3.2 任务（二）常用编码器和译码器

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/Digital-Circuit/Lab002.git

8-3 优先编码器 74LS148

完成任务（一）后，读者可以了解到普通编码器对输入信号是有限制的，即在任意一个时刻所有输入信号中只允许一个输入信号有效，否则编码器将发生混乱。为了解决这一问题，可采用**优先编码器**，它允许多个输入信号同时有效，但是只对其中优先级最高的输入信号产生编码，而忽略掉优先级较低的输入信号。

请读者按照下面的步骤验证 8-3 优先编码器 74LS148 的功能：

1. 打开项目下的 74LS148.dlsche 文件。该原理图文件中绘制了 74LS148 的内部原理图。其输入信号包括级联入（使能）EI，低电平有效，还有 8 个输入信号 D0~D7，也是低电平有效，D0 优先级最低，D7 优先级最高。输出信号包括编码 A0~A2，以及级联出 EO 和 GS。
2. 启动仿真。根据 74LS148 的功能表 2-2，验证其功能。并将功能表补充完整。需要说明的是，在此原理图中使用了“八位交互式数字信号”控制 D0~D7 的输入，读者可以选中此器件，然后在右侧的属性窗口中编辑此器件的“数字信号”属性，在其中填入一个两位的十六进制数后按回车，就可以设定该器件输出的信号了。
3. 观察填写完成的功能表，当输入信号有效时（D7~D0 中至少有一位输入为 0），输出信号 A2~A0 的值与输入信号有何关系？根据输出信号的值能确定当前是哪一位最高优先级的输入信号有效吗？

表 2-2：74LS148 功能表

级联入（使能）	输入	输出	级联出
EI	D7 D6 D5 D4 D3 D2 D1 D0	A2A1A0	EO GS
1	X XXXXXXX	1 1 1	1 1
0	1 1 1 1 1110	1 1 1	1 0
0	1 1 1 1 110 X	1 1 0	1 0
0	1 1 1 1 1 0 X X	1 0 1	1 0
0	1 1 1 1 0 X XX	1 0 0	1 0
0	1 1 1 0 X XXX		
0	1 1 0 X XXXX		
0	1 0 X XXXXX		
0	0 X XXXXXX		
0	1 1 1 1 1 1 1 1		

3-8 译码器 74LS138

3-8 译码器 74LS138 有 3 个输入信号 A、B、C（C 表示最高位，A 表示最低位），8 个输出信号 Y0~Y7，另有 3 个使能输入信号 G1、G2A 和 G2B。由 3-8 译码器的功能表 2-3 可以看出，译码器根据 C，B，A 三个输入信号和 G1，G2A，G2B 三个使能输入信号的条件，决定哪个输出信号有效（低电平有效）。也就是说，在作为译码器使用时，当 Y0~Y7 中的某一位输出低电平时，对应的一组输入信号被译码。

请读者按照下面的步骤验证 3-8 译码器 74LS138 的功能：

1. 打开项目下的 74LS138.dlsche。该原理图文件中绘制了 74LS138 的内部原理图。
2. 启动仿真。根据 74LS138 的功能表 2-3，验证其功能。并将功能表补充完整。

表 2-3：74LS138 功能表（G2A+G2B 中的加号表示逻辑或）

门控		输入	输出
G1	G2A+G2B	C B A	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0
X	1	X XX	1 1 1 1 1 1 1 1
0	X	X XX	1 1 1 1 1 1 1 1
1	0	0 0 0	1 1 1 1 1 1 1 0
1	0	0 0 1	1 1 1 1 1 1 0 1
1	0	0 1 0	1 1 1 1 1 0 1 1

1	0	0 1 1	1 1 1 1 0 1 1 1
1	0		
1	0		
1	0		
1	0		

3. 观察填写完成的功能表，3-8 译码器的三个使能输入端输入何种信号时，译码器才能正常工作。尝试写出输出信号 Y0、Y3 和 Y7 的逻辑表达式。

七段数码管显示译码器 74LS48

在设计一个数字系统时，常常需要使用数码显示器件，将测量或运算结果以数字方式显示出来，以便监视系统的工作状况。请读者按照下面的步骤进行试验，测试 74LS48 七段数码管显示译码器的功能：

1. 打开项目下的原理图 74LS48.dlsche。该原理图文件中使用 74LS48 设计了一个用于将输入的 4 位二进制信号显示为十进制数字的电路。
2. 启动仿真。根据 74LS48 的功能表 2-4，验证其功能。并将功能表补充完整。

表 2-4：74LS48 功能表

输入				输出	显示数字
LT	RBI	D C B A	BI\RBO	a bc d e f g	
1	1	0000	1	1111110	0
1	X	0001	1	0110000	1
1	X	0010	1	1101101	2
1	X	0011	1	1111001	3
1	X	0100	1	0110011	4
1	X	0101	1		
1	X	0110	1		
1	X	0111	1		
1	X	1000	1		
1	X	1001	1		
X	X	X XXX	0	0000000	
0	X	X XXX	1	1111111	

3. 将原理图中的器件 74LS48 替换为 74LS47（在型号库窗口的 74LS 功能芯片型号库中可以找到 74LS47），并在各个输出端加上反相器（非门）。仿真测试，说明 74LS47 与 74LS48 的有何不同？

三态门

由于在后面的实验任务中会用到三态门，而且三态门也是一种非常重要的器件，所以需要读者按下列步骤熟悉三态门的工作原理和使用方法：

1. 打开项目下的原理图 tristate.dlsche。该原理图文件中将两个输入信号分别通过一个三态门连接到了同一条总线上，这样就可以通过控制三态门的状态来决定是将哪个信号输出到总线上，从而避免总线上的信号发生冲突。这里的三态门是低电平使能，也就是说当控制端信号为低电平时，三态门打开。
2. 启动仿真。根据表 2-5，验证三态门的工作原理，并将表格补充完整。开始仿真后，当总线为蓝色时，说明总线上的信号发生了冲突，是一个不确定的值，在一个实际可工作的数字系统中是绝对不允许出现这种情况的。
3. 修改原理图，将总线上连接的信号数量从两个扩展为四个。首先，在左侧型号库窗口的“逻辑门”型号库中找到低电平使能三态门的型号 LTri_Gate，使用此型号在原理图文件 tristate.dlsche 中再添加两个三态门，然后将原有的两个输入信号扩展为四个输入信号，最后完成仿真验证。
4. 提交作业。

通过上述实验不难得出，当多个三态门同时向总线输出数据时，为避免发生数据冲突，同一时刻必须

只能有一个三态门打开（通过控制端的低电平将其使能），而其他三态门关闭。这样，就能确保同一个时刻总线只与其中一个三态门导通。

表 2-5：三态门测试

输入信号		控制信号		输出信号
I1	I2	C1	C2	总线
1	0	0	0	冲突
0	0	1	0	0
0	1	1	0	1
1	0	1	0	0
1	1	1	0	1
0	0	0	1	
0	1	0	1	
1	0	0	1	
1	1	0	1	

3.3 任务（三）设计多外设共享地址总线排队电路

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/blank.git

根据本实验预备知识中介绍的多外设共享地址总线排队电路，并参考图 2-1，在原理图文件 top.dlsche 中设计一个多外设共享地址总线排队电路，并完成仿真验证。要求在电路中模拟 0、1、2、3 共 4 个设备，优先级逐次降低，0 号设备的请求优先级最高，4 号设备的请求优先级最低。设备地址码为 2 位，设备地址依次为 0,1,2,3，所以地址总线位宽为 2。

既然是模拟这些设备，就不需要真的设计出来这些设备，只需要让这些设备能够产生总线请求信号，可以产生地址信号并发送到地址总线即可。读者可以参考图 2-3 设计总线请求信号，然后参考图 2-4 设计设备地址信号与地址总线的连接关系，最后将编码器和译码器连接好就可以了。

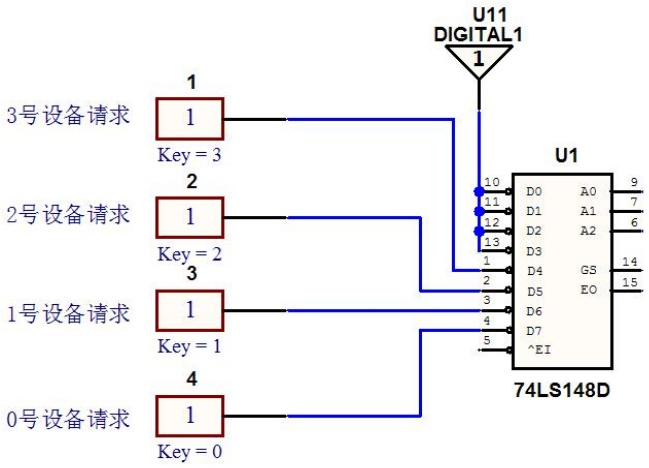


图 2-3：设备的总线请求信号与编码器的连接方法

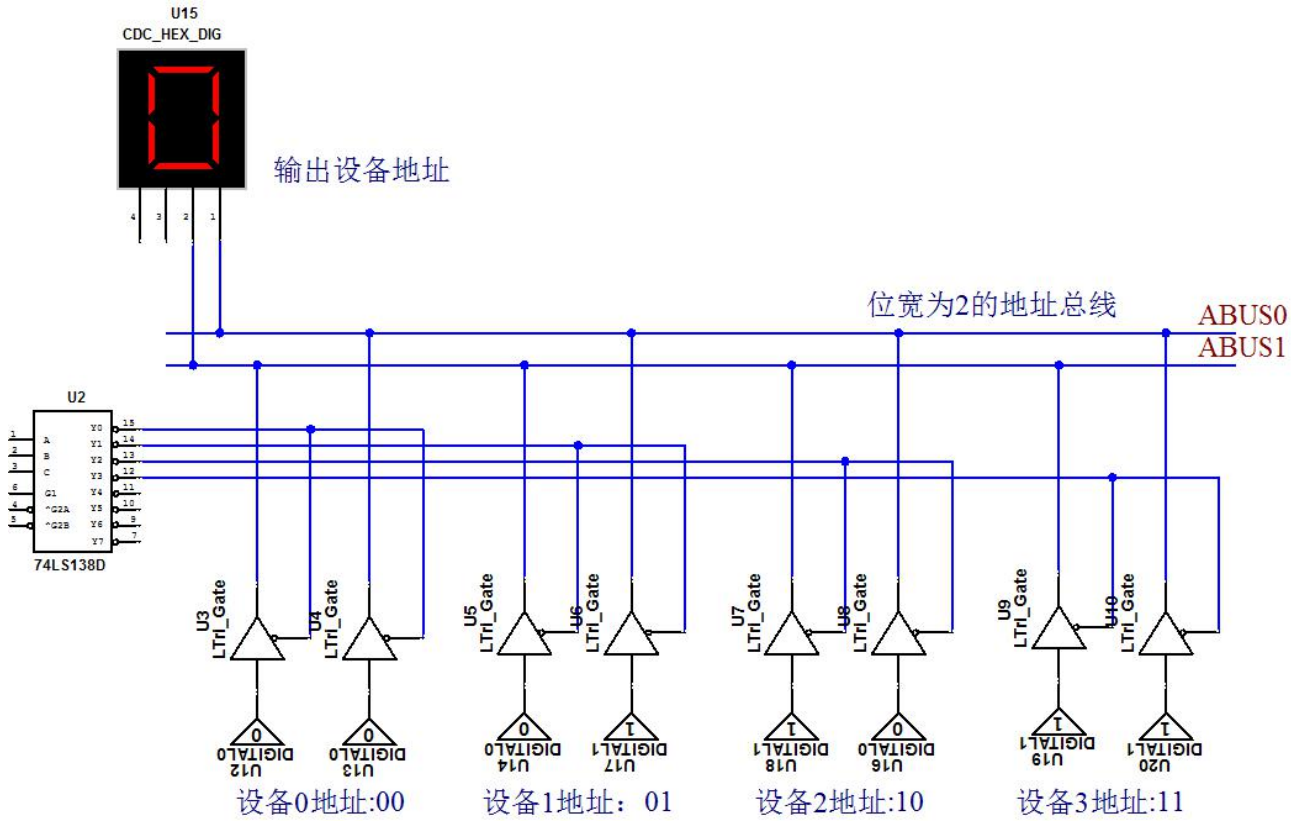


图 2-4：译码器控制三态门将指定的设备地址发送到地址总线

四、 思考与练习

1. 仿照 74LS148 的电路原理图，设计一个 4 位优先编码器并完成功能验证。
2. 仿照 74LS138 的电路原理图，设计一个 2-4 译码器并完成功能验证。
3. 使用两片 74LS138 设计一个 4-16 译码器。
4. 在什么条件下，编码器、译码器必须留一个无效码？什么时候可以不留？
5. 设计一个设备工作状态指示器电路，要求用红、黄、绿三个指示灯表示三台设备的工作情况：绿灯亮表示全部正常；红灯亮表示有一台不正常；黄灯亮表示两台不正常；红、黄灯全亮表示三台都不正常。
6. 用两片 74LS151 数据选择器设计实现一位全加器，实现电路并仿真验证其正确性。提示：可将全加器的三位输入 A，B，Ci 作为 8 选 1 数据选择器 74LS151 的选择信号，需要两个 8 选一数据选择器，分别输出和 S 以及高位进位信号 Co。
7. 用 74LS138 实现一位全减器并仿真验证其正确性。
8. 在计算机系统中通常有多个优先级不同的中断请求，当多个中断源都向 CPU 发出中断请求时，需要选出优先级最高的中断请求，进行优先服务，这时可通过一个优先编码器译出最高中断级别，根据级别就可以确定调用哪个中断服务程序。假设有 0~15 共 16 个中断源，0 号中断请求优先级为 0，优先级最高，15 号中断请求优先级为 15，优先级最低。请设计电路，根据当前中断请求，输出对应的优先级。**提示**：使用两片 74LS148 优先编码器和一片 74LS158 数据选择器设计一个扩展的 16-4 优先编码器，16 个输入端接各个中断请求，4 位输出二进制数就是当前最高优先级中断的级别。

实验 3 加法器

实验性质：验证+设计
建议学时：2 学时

一、 实验目的

- 了解常用加法器的设计方法。
- 掌握四位超前进位并行加法器 74LS283。
- 掌握算术逻辑运算器（ALU）74LS181。

二、 预备知识

加法器是计算机或其他数字系统中对二进制数进行运算处理的组合逻辑器件。按进位信号产生的方法不同，可分为串行进位加法器和并行进位加法器两种。

2.1 串行进位加法器

串行进位加法器逻辑结构如图 3-1 所示，它由多个一位全加器串行连接而成。每个一位全加器有三个输入（加数 A，被加数 B，由低位输入的进位信号 CI）和两个输出（和数 S，向高位的进位信号 CO），然后按照图 3-1 所示进行串行连接。

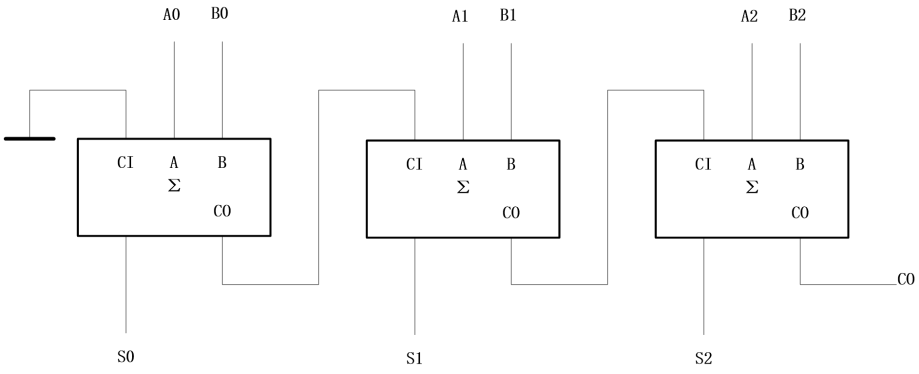


图 3-1：串行进位加法器

2.2 超前进位并行加法器

串行进位加法器须将低位全加器产生的进位信号逐位向高一位全加器上传递。因此，串行进位加法器求和的最高位的输出结果必须等到各位进位信号逐位传递后才能形成，工作速度很慢。为了提高工作速度，可采用四位超前进位并行加法器，其原理如下：

设有两个四位二进制加数 $A=A_4A_3A_2A_1$，$B=B_4B_3B_2B_1$。并行进位加法器的 S（和）以及 C（进位）的逻辑表达式如下：

$S_1=A_1 \oplus B_1 \oplus C_0$，　　$C_1=A_1B_1+(A_1 \oplus B_1)C_0$

$S_2=A_2 \oplus B_2 \oplus C_1$，　　$C_2=A_2B_2+(A_2 \oplus B_2)C_1$

$S_3=A_3 \oplus B_3 \oplus C_2$，　　$C_3=A_3B_3+(A_3 \oplus B_3)C_2$

$S_4=A_4 \oplus B_4 \oplus C_3$，　　$C_4=A_4B_4+(A_4 \oplus B_4)C_3$

设 Gi 为进位生成项，Pi 为进位传递项，即有：

$G_i=A_iB_i$……表达式 1

$P_i=A_i \oplus B_i$……表达式 2

则各位和的表达式变为

$S_i=P_i \oplus C_{i-1}$……表达式 3

各进位表达式变为

$C_i=G_i+P_iC_{i-1}$……表达式 4

采用递推公式，进位表达式变为

$C_1=G_1+P_1C_0$……表达式 5

$C_2=G_2+P_2C_1 =G_2+P_2G_1+P_2P_1C_0$……表达式 6

$C_3=G_3+P_3C_2 =G_3+P_3G_2+P_3P_2G_1+P_3P_2P_1C_0$……表达式 7

$C_4=G_4+P_4C_3 =G_4+P_4G_3+P_4P_3G_2+P_4P_3P_2G_1+P_4P_3P_2P_1C_0$……表达式 8

最终的进位表达式 5~8 表明，最低位的进位信号 C_0 可以超前传送到 C_2、C_3、C_4。将这些进位表达式 5~8 带入上面的各位和的表达式 3 中（请读者自行完成表达式的推导），则可以看到最低位的进位信号 C_0 也可以超前传送到 S_2、S_3、S_4 上，从而大大加快了进位信号的传递速度，进而加快运算速度。

并行进位运算虽然能显著加快运算速度，但同时也带来了运算电路器件数目增加的问题。在数字系统设计过程中，需要根据设计的目的来权衡运算速度和电路面积这一对矛盾。

三、 实验任务

3.1 任务（一）了解常用的加法器

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/Digital-Circuit/Lab003.git

四位超前进位并行加法器 74LS283

请读者按照下面的步骤验证四位超前进位并行加法器 74LS283 的功能：

1. 打开项目下的原理图文件 74LS283.dlsche。该原理图文件中绘制了 74LS283 的内部原理图。其输入信号包括两个加数 A1~A4、B1~B4,以及进位输入 C0。输出信号包括和 S1~S4，最高进位输出 C4。
2. 启动仿真。根据 74LS283 的功能表 3-1，验证其加法运算功能。并将表格补充完整。注意表格中使用的是十六进制数。
3. 在 74LS283 中使用一个红色矩形框将所有用于计算 S_1 的逻辑门包含在其中。并写出通过这些逻辑门计算 S_1 的表达式，然后证明这个表达式与本实验 2.2 节中推导的 S_1 的表达式（将表达式 2 带入表达式 3 可得到）等效。绘制红色矩形框的方法是，选择“绘制”菜单中的“矩形”，然后在原理图中将所有用于计算 S_1 的逻辑门包含在其中，绘制完毕后，选中矩形框，在右侧的“属性”视图中，将矩形框的“填充”属性修改为“否”，将轮廓的“颜色”修改为红色。
4. 请读者再尝试证明在 74LS283 中使用逻辑门计算 S_2 的表达式与本实验 2.2 节中推导的 S_2 的表达式是等效的。

表 3-1：74LS283 功能验证表

输入信号			输出信号	
四位加数 A	四位加数 B	进位输入 C0	四位和 S	进位输出 C4
3	5	0	8	0
3	5	1	9	0
7	9	0		
7	9	1		
F	F	0		
F	F	1		

算术逻辑运算单元 74LS181

算术逻辑运算单元（ALU）74LS181 是一种功能较强的组合逻辑电路，它能进行多种算术运算和逻辑运算。它的基本逻辑结构是超前进位加法器。

S_3~S_0：工作方式选择。

M：当 M=1 时，进行逻辑运算；M=0 时，进行算术运算。

A=A_3~A_0，B=B_3~B_0：参加运算的两个数（注脚 3 表示最高位）。

F_3~F_0：运算结果。

CN：当做加运算时，表示最低位进位输入，CN=1 时，无进位输入；CN=0 时，有进位输入。当做减法时，表示最低位借位，CN=1，有借位；CN=0，无借位。

CN4：最高位进位输出，低电平有效。

AEQB：当 F_3~F_0 全为高电平时为 1，否则为 0。

G 称为进位发生输出，P 称为进位传送输出。它们是为了便于实现多芯片 ALU 之间的超前进位的。

表 3-2：74LS181 真值表

方式				M=1 逻辑运算	M=0 算术运算	
S_3	S_2	S_1	S_0	逻辑运算	CN=1(无进位)	CN=0(有进位)
0	0	0	0	F=/A（非）	F=A（直传）	F=A 加 1
0	0	0	1	F=/(A+B)	F=A+B	F=(A+B)加 1
0	0	1	0	F=(/A)B	F=A+/B	F=(A+/B)加 1
0	0	1	1	F=0	F=负 1	F=0
0	1	0	0	F=/(AB)	F=A 加 A(/B)	F=A 加 A/B 加 1
0	1	0	1	F=/B	F=(A+B)加 A/B	F=(A+B)加 A/B 加 1
0	1	1	0	F=A⊕B（异或）	F=A 减 B 减 1	F=A 减 B
0	1	1	1	F=A/B	F=A(/B)减 1	F=A(/B)
1	0	0	0	F=/A+B	F=A 加 AB	F=A 加 AB 加 1
1	0	0	1	F=/(A⊕B)	F=A 加 B	F=A 加 B 加 1
1	0	1	0	F=B	F=(A+/B)加 AB	F=(A+/B)加 AB 加 1
1	0	1	1	F=AB（与）	F=AB 减 1	F=AB
1	1	0	0	F=1	F=A 加 A	F=A 加 A 加 1
1	1	0	1	F=A+/B	F=(A+B)加 A	F=(A+B)加 A 加 1
1	1	1	0	F=A+B（或）	F=(A+/B)加 A	F=(A+/B)加 A 加 1
1	1	1	1	F=A（直传）	F=A 减 1	F=A（直传）

提示：1—高电平，0—低电平。中文的“加”和“减”表示算数运算。其他的符号表示逻辑运算，例如“/”表示逻辑取反，“+”表示逻辑或。表中的 A 和 B 分别表示 4 位二进制数 $A_3A_2A_1A_0$、$B_3B_2B_1B_0$。

请读者按照下面的步骤验证算术逻辑运算单元 74LS181 的功能：

1. 打开项目下的原理图文件 74LS181.dlsche。该原理图文件中绘制了 74LS181 的内部原理图（该原理图有问题，可用芯片 74LS181 来替代进行）。
2. 启动仿真，根据表 3-2 对 74LS181 进行功能验证，由于 74LS181 的运算功能很多，可以选择一部分功能进行验证，请读者自行设计一个功能验证记录表，至少要验证表 3-2 中有阴影的功能项，并记录所有的输入输出信号。

3.2 任务（二）加法器与译码器综合设计

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/blank.git

本任务要求读者在原理图文件 top.dlsche 中使用一片 74LS181 设计一个简单的运算单元（ALU），使该 ALU 在 3 位输入信号 X_0~X_2控制下可以完成 8 种不同的运算，并通过仿真进行功能验证。具体要求如下：

通过 6 个输入信号 S3、S2、S1、S0、M、CN 可以控制 74LS181 完成 48 种不同的运算。假设一个使用 74LS181 做运算器的 4 位处理器，它的指令编码中只有 3 位可用于对运算方式进行编码（支持 2^3=8 种不同运算）。请读者设计一个译码电路，该译码电路可以接收 3 位运算方式编码，然后输出 6 个控制信号，使用这 6 个控制信号控制 74LS181 完成 8 种不同的运算。

假设这 3 位编码分别为 X_2、X_1、X_0，译码输出最小项的组合控制信号分别为 S3、S2、S1、S0、M、CN。$X_2X_1X_0$的值可以为 0~7，分别对应表 3-3 中的 8 种不同的运算。

提示：读者应首先根据表 3-3 设计一个真值表，该真值表的输入信号是 X_0~X_2，输出信号是 74LS181 的 6 个控制信号，然后根据真值表设计译码电路。译码电路可以使用一片 74LS138 将输入信号译码后，再单独设计一个编码电路，通过基本的逻辑门将 8 个信号编码为 6 个控制信号；也可以使用基本的逻辑门将 3 个输入信号直接译码为 6 个控制信号。

表 3-3：控制信号与 74LS181 运算方式对应表

X_2	X_1	X_0	74LS181 功能
0	0	0	F=A 加 B
0	0	1	F=A 加 B 加 1
0	1	0	F=A 减 B
0	1	1	F=A 减 B 减 1
1	0	0	F=A+B（或）
1	0	1	F=AB（与）
1	1	0	F=A⊕B（异或）
1	1	1	F=/A（非）

四、 思考与练习

1. 使用两片 74LS181 设计一个 8 位的 ALU 运算器。
2. 设计一个 3 位的串行进位加法器。
3. 设计一个 3 位的超前进位加法器。
4. 用 74LS181 设计一个 1 位十六进制数转换为 2 位十进制数 8421BCD 码的电路。**提示：**当十六进制数≤9 时，直接传送（即加 0）；当十六进制数≥10 时，加 6 便是十进制数，进位输出就是十位。这里可用直接转换法，例如十六进制数由 74LS181 的 A 端输入，同时对 A 进行判断，判断电路的输出接到 B2、B1 上，并令 B3B0=00；当 A≤9 时，判断电路输出 B2B1=00;当 A≥10 时，判断电路输出 B2B1=11。74SL181 的输出（连同进位输出）就是变换结果。

实验 4 触发器

实验性质：验证+设计
建议学时：2 学时

一、 实验目的

- 掌握各类锁存器的工作方式。
- 掌握各类触发器的触发方式、逻辑功能及原理。
- 学会运用触发器设计基本时序电路。

二、 预备知识

2.1 时序逻辑电路

时序逻辑电路的输出往往反馈到输入端，与输入变量一起决定电路的输出。时序逻辑电路的特点是：任意时刻输出不仅取决于该时刻的输入值，而且还与原来的状态有关。因此时序逻辑电路具有记忆功能，这是它与组合逻辑电路的本质区别。

2.2 锁存器的基本特性

锁存器在电路上具有两个稳定的物理状态，所以它们能记忆一位二进制数。它们具有以下特性：

1. 有两个互补的输出端 Q 和$\overline{Q}$。当 Q=1 时，$\overline{Q}$=0；而当 Q=0 时，$\overline{Q}$=1。
2. 有两个稳定状态。通常将 Q=1 和$\overline{Q}$=0 称为“1”状态，而把 Q=0 和$\overline{Q}$=1 称为“0”状态。若输入不发生变化，锁存器必定处于其中一个状态，并且长期保持下去。
3. 在输入信号的作用下，锁存器可以从一个稳定状态转换到另一个稳定状态。

我们把输入信号发生变化前的锁存器状态称为**现态**，用 Q^n 和 $\overline{Q}^n$ 来表示，而把输入信号发生变化后锁存器所进入的状态，称为**次态**，用 Q^{n+1} 和 $\overline{Q}^{n+1}$ 来表示。若用 X 来表示输入信号的集合，则锁存器的次态是它的现态和输入信号的函数，即

$$Q^{n+1}= f(Q^n, X)$$

该函数称为锁存器的次态方程，又称状态方程，它是描述时序电路的通用表达式。由于每一种具体的锁存器都有自己特定的状态方程，因此，也称为特征方程。常用的锁存器有基本 SR 锁存器、门控 SR 锁存器和门控 D 锁存器。

2.3 触发器

锁存器虽然能记忆一位二进制数，但接受的输入数据是在使能信号 EN 控制下进行的。EN 是电位信号，如果在 EN 有效期间，输入数据受到瞬时的干扰发生电平变化，那么锁存器就可能锁存错误的数据。为了提高锁存器工作的可靠性，人们又创新改进，推出边沿方式工作的触发器。

触发器是一种同步双稳态器件，同样用来记忆一位二进制数。所谓同步，是指触发器的记忆状态按时钟规定的启动指示点（脉冲边沿）来改变。触发器可在时钟脉冲的正沿（上升沿）改变状态，也可以在时钟脉冲的负沿（下降沿）改变状态。

2.4 触发器的应用

触发器是构成复杂时序逻辑电路最基本的组成单元。按照触发器在时序逻辑电路中的作用，它的应用主要有以下几个方面：

1. 用作并行数据寄存器。一个触发器只存储一位二进制数，若并行存储 n 位二进制数，则需要 n 个触发器。这 n 个触发器按并行方式连接就构成并行数据寄存器，简称寄存器。

2. 用作计数器。1 个触发器用作计数器时，可记忆两个状态；2 个触发器按串行方式连接成 2 位计数器时，可以记忆 2^2=4 个状态；n 个触发器按串行方式连接成 n 位计数器时，可记忆 2^n 个状态。这 2^n 个状态对应了 2^n 个时钟脉冲。换句话说，计数器记忆时钟脉冲的个数。
3. 用作分频器。一个触发器接成计数模式按时钟脉冲的固定频率工作，若时钟的频率为 f_n，则该触发器 Q 端输出信号的频率为 $f_n/2$。触发器的这种应用称为分频。

2.5 时序电路的设计与测试

同步时序电路的特点是：电路中时间的划分是以时钟脉冲为依据的。其设计的主要步骤是：根据设计要求写出动作说明，列出状态图或状态转换表，然后进行状态化简和状态分配，再根据所选触发器确定其驱动方程，然后画出电路图。当然，在设计中有时考虑自启动也是必不可少的。在进行设计时，不要拘泥于以上程式，应该融会贯通，灵活掌握。对于所设计的逻辑电路图，必须进行实验检测，只有实际电路符合设计要求时，才能证明设计是正确的。

时序电路的功能测试分静态和动态两种方法。静态测试就是直流稳态测试，就是测试电路的状态转换真值表。测试时，时钟脉冲由逻辑开关提供，电路输出用数字探针指示。动态测试是指，在时钟输入端输入矩形波或方波信号（自动时钟源），用逻辑分析仪观察电路各级的工作波形，它不仅可以看到电路的稳态情况，而且还可以观察到电路的过渡态（或叫瞬态）。

三、 实验任务

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

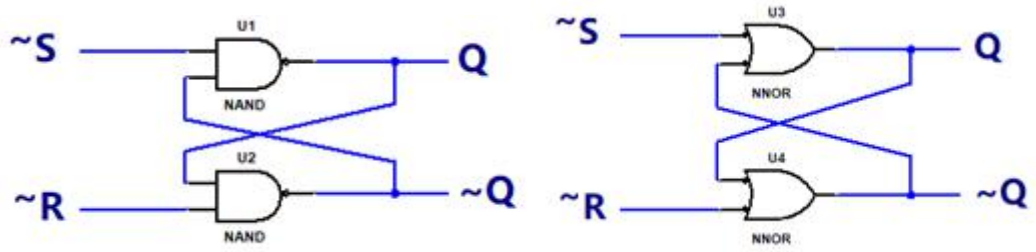
方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

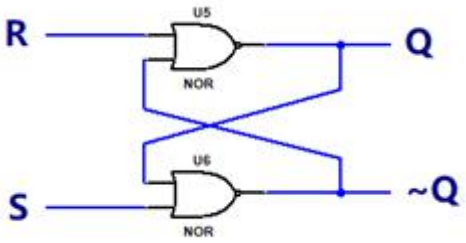
https://www.codecode.net/engintime/Dream-Logic/Project-Template/Digital-Circuit/Lab004.git

基本 SR 锁存器

基本 SR 锁存器有三种形式，如图 4-1 所示。~S 和~R 表示低电平有效，S 和 R 表示高电平有效。请读者按照下面的步骤仿真验证基本 SR 锁存器的功能：



第一种形式——与非门　　第二种形式——非或门



第三种形式——或非门（注意 R 和 S 的位置）

图 4-1：基本 SR 锁存器的三种形式

1. 打开项目下的原理图文件 SR.dlsche。
2. 启动仿真。根据第三种形式的仿真结果完成 SR 锁存器的特征表 4-1，并根据特征表，写出状态方程。在表 4-1 中的现态是指，当 S 或 R 发生变化后进入的状态，此时新的 Q 值作为次态。例如，读者通过按键盘上的 5、6 键，将 S 和 R 设置为 0，然后启动仿真，此时 Q 的值为 1，读者按下键盘上的 5，使 R 变为 1，则此瞬间 S、R、Q 的值分别为 0、1、1，这个就是表中的第 4 个现态，记录此时 Q 的值为 0，称为次态。

注意：在启动仿真之前，不要将第二种形式的~S 和~R 同时设置为 1，这样会导致无法启动仿真。

表 4-1：由或非门组成的基本 SR 锁存器特征表

现态			次态
S	R	Q	Q^{n+1}
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	状态不定
1	1	1	状态不定

3.1 门控 SR 锁存器

门控 SR 锁存器如图 4-2 所示。它是在基本 SR 锁存器的基础上加以改进，增加一级输入与非门，由使能控制信号 EN 进行控制。EN 有效时，锁存器才接收数据输入信号；EN 无效时，锁存器拒绝接收数据输入信号。门控 SR 锁存器也叫电平触发 SR 触发器，通常使用时钟 CLK 作为门控信号，也就是触发信号。

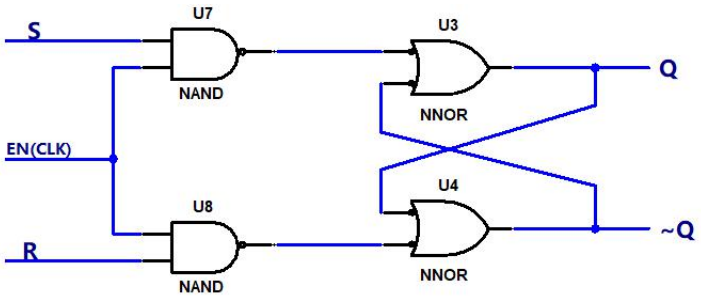


图 4-2：门控 SR 锁存器（电平触发 SR 触发器）

按照下面的步骤验证门控 SR 锁存器的功能：

1. 打开项目下的原理图文件 SR_gate.dlsche。
2. 参考表 4-1 为门控 SR 锁存器设计一个特征表，只考虑 EN=1 的情况即可。通过仿真验证门控 SR 锁存器的功能，并将其特征表填写完整。

3.2 门控 D 锁存器

图 4-3 是门控 D 锁存器。它与门控 SR 锁存器相同处在于第一级都是两个与非门，不同处在于只有一个数据输入端 D。D 输入经过一个非门加到原来门控 SR 锁存器的 R 输入端，变成互补输入，所以 D 锁存器是门控 SR 锁存器的一种改进形式。其工作原理是：当数据输入 D=1 且使能控制 EN=1，锁存器置 1；当 D=0 且 EN=1 时，锁存器置 0。门控 D 触发器也称为电平触发 D 触发器，通常使用时钟 CLK 作为门控信号，也就是触发信号。

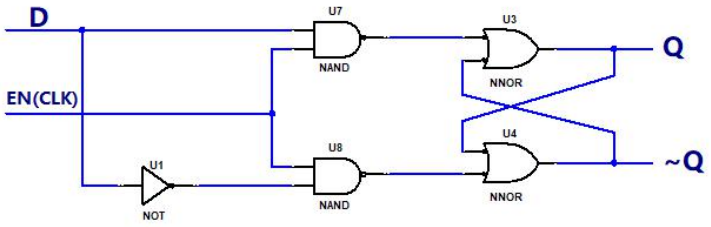


图 4-3：门控 D 锁存器（电平触发 D 触发器）

按照下面的步骤验证门控 D 锁存器的功能：

1. 打开项目下的原理图文件 D_gate.dlsche。
2. 参考表 4-1 为门控 D 锁存器设计一个特征表，只考虑 EN=1 的情况即可。通过仿真验证门控 D 锁存器的功能，并将其特征表填写完整。

3.3 边沿触发 D 触发器

电平触发 D 触发器在 EN（CLK）的有效电平期间输出 Q 始终跟随输入 D 变化，输出与输入的状态保持相同，所以又将其称为“透明的 D 型锁存器”。

为了提高触发器的可靠性，增强抗干扰能力，希望触发器的次态仅仅取决于 CLK 信号下降沿（或上升沿）到达时刻输入信号的状态。可以将其比作一个瞬时采样器，在时钟的边沿对输入 D 进行采样，而与时钟边沿前后的输入无关，这样就只需要保证时钟沿到来的那个瞬间，输入数据 D 稳定在一个有效值即可。为了实现这一设想，人们相继研制成了各种边沿触发器。边沿触发 D 触发器是由两个电平触发 D 触发器组成的。

请读者按下列步骤验证边沿触发 D 触发器的功能：

1. 打开项目下的原理图文件 D_edge.dlsche。
2. 启动仿真，根据表 4-2 验证边沿触发 D 触发器的功能。

表 4-2：边沿触发 D 触发器功能表

输入		输出	
CLK	D	Q	~Q
X	X	保持	保持
↑（上升沿）	1	1	0
↑（上升沿）	0	0	1

3.4 JK 触发器

在上述边沿触发 D 触发器实验中，使用 D 触发器的内部电路进行仿真测试其功能。在实际的应用中，需要将 D 触发器的内部电路封装起来，仅向外提供输入/输出管脚，以便使用。下面，就对另外一个常用的边沿触发器 JK 触发器进行功能验证。步骤如下：

1. 打开项目下的原理图文件 jk.dlsche。
2. 启动仿真。按 JK 触发器的功能表 4-3 进行验证。

表 4-3：JK 触发器的功能表（Q^n是触发转换前的状态）

输入					输出	
PR	CLR	CLK	J	K	Q	~Q
1	1	X	X	X	X	X
1	0	X	X	X	1(置 1)	0
0	1	X	X	X	0(清 0)	1
0	0	X	X	X	保持	保持
0	0	↑	1	0	1	0
0	0	↑	0	1	0	1
0	0	1	0	0	保持	保持
0	0	↑	1	1	$\sim Q^n$(翻转)	Q^n

3.5 二分频器

数字电路中的分频是指将源信号的频率降低为原来的 N 分之一，就称为 N 分频。所以这里说的二分频就是将源信号的频率降低为原来的二分之一。实现分频的电路称为分频器。

请读者按照下面的步骤观察一个二分频电路的实现方法，理解分频器的原理：

1. 打开项目下的原理图文件 freq_divider2.dlsch。该原理图文件中绘制了一个使用 D 触发器实现的二分频电路，并分别将源信号 CLK 和分频后的信号 CLK\2 接到了逻辑分析仪的管脚 1 和管脚 2，这样就可以很方便的使用逻辑分析仪比较两个信号的波形。
2. 启动仿真。
3. 按下键盘上的 R 键初始化 D 触发器。
4. 双击逻辑分析仪 XLA1，打开逻辑分析仪窗口，观察两个不同频率的波形。为了便于观察和比较波形，也可以先按 F6 暂停仿真再进行观察。在逻辑分析窗口中可以调整比例对波形进行缩放，还可以使用窗口底部的滚动条拖动波形。

注意：时钟频率和仿真步长有一定的关系，仿真步长要小于时钟周期，这样仿真引擎才能对一个周期内的电路状态完成采样。

3.6 设计四分频器

请读者参考二分频器的设计方法，使用 D 触发器设计一个四分频器，并仿真验证其功能。步骤如下：

1. 在项目下新建一个原理图文件 freq_divider4.dlsche。新建原理图文件的方法可以参考实验一中的相关内容。
2. 使用 D 触发器设计一个四分频电路，将源信号 CLK 和分频后的信号 CLK\4 接到逻辑分析仪的管脚 1 和管脚 2。在原理图中添加逻辑分析仪的方法是：选择菜单“仪器->逻辑分析仪”，将鼠标移动到原理图中，单击鼠标左键即可完成绘制。在型号库“触发器”中可以找到 D 触发器，在型号库“数字信号源”中可以找到单周期时钟和时钟。
3. 通过仿真检验四分频电路的功能，确保其可以产生正确的信号。

提交作业

在提交作业前，读者需要将四分频器电路 SVG 图形文件的链接添加到 README.md 文件中，这样，当使用浏览器查看提交后的线上项目时，就可以从 README.md 文件中看到四分频器的电路了。方法如下：

1. 双击“项目管理器”中的 README.md 文件，使用一种合适的文本编辑器或者 Markdown 编辑器打开此文件。
2. 在 README.md 文件的末尾添加如下的文本：

```
# 四分频器电路
![raw svg](freq_divider4.dlsche.svg)
```

3. 保存 README.md 文件。
4. 提交作业。

实验 5 寄存器和计数器

实验性质：验证+设计
建议学时：2 学时

一、 实验目的

- 掌握使用触发器设计寄存器和计数器的方法。
- 掌握计数器和移位寄存器的电路结构和工作原理。

二、 预备知识

2.1 寄存器

由锁存器或触发器组成、一次能够并行存储 N 位比特数据的逻辑部件称为寄存器。寄存器是计算机和数字系统中最常见的功能部件。锁存器与触发器构成的寄存器工作方式不同。

- **锁存器构成的寄存器：**寄存器中每一位对应一个锁存器，所有的锁存器共用一个门控信号 EN，当 EN 有效时，所有锁存器的输入传送到输出端，EN 无效后，寄存器的输出与输入隔离，若 EN 始终无效，那么寄存器就一直保存之前存入的数据不变。74LS373 就是由 8 个 D 锁存器构成的 8 位寄存器。
- **触发器构成的寄存器：**寄存器中每一位对应一个触发器，所有的触发器共用一个时钟 CLK，当时钟上升沿（或下降沿）到来时，所有触发器的输入端数据复制到输出端，寄存器的输出会一直保持不变，直到下个时钟上升沿（或下降沿）到来时，才会重新写入输入端的数据。74LS374 就是由 8 个 D 触发器构成的 8 位寄存器。

常用的寄存器大多由 D 触发器构成，这是因为 D 触发器采用时钟边沿触发方式，工作十分可靠，且只有一个数据输入端，使用也很方便。

2.2 移位寄存器

在时钟信号的控制下，将寄存器的数据向左或向右移位的寄存器称为移位寄存器。图 5-1 所示的是一个 4 位的右移寄存器逻辑图，它的结构很简单，只需要把左面一位触发器的输出端接到右面一位触发器的输入端，即连接关系满足 $D_i=Q_{i-1}$，同时把所有触发器的时钟端接在一起，用同步脉冲信号控制。

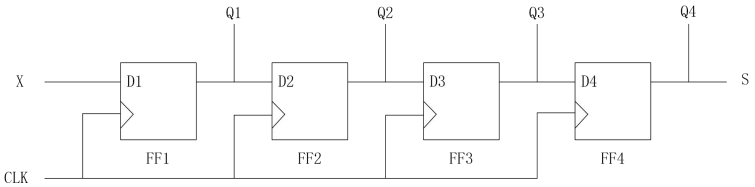


图 5-1：右移寄存器逻辑图

实际应用中常常采用中规模通用移位寄存器，它是将若干触发器和逻辑门集成在一块芯片上，从而为使用带来很大方便。在移位控制信号作用下，既能左移又能右移的，则称之为双向移位寄存器。移位寄存器可以临时寄存信息，也可以加工或传送数据，还可作为基本部件用于各种时序逻辑电路，如计数器、脉冲分配器、序列码发生器、序列码检测器等。

74LS299 就是一种 8 位的通用移位寄存器，它有多种工作模式，可并行置数、左移、右移、保持数据，也可以实现并入并出、并入串出、串入串出、串入并出操作。

通用移位寄存器用途十分广泛，在计算机系统中可用作累加寄存器、缓冲寄存器、串-并转换器等。

2.3 计数器

计数器的功能是记忆脉冲的个数，它是数字系统中应用最广泛的基本时序逻辑构件。计数器所能记忆

脉冲的最大数目称为计数器的**模**，用字母 M 表示。构成计数器的核心元件是触发器。

计数器的种类繁多，分类方法也不同。①按计数功能来分，可分为加法计数器、减法计数器、可逆计数器（加/减计数器）；②按进位基数来分，可分为二进制计数器、十进制计数器、任意进制计数器；③按进位方式来分，可分为同步计数器、异步计数器。

1. **同步计数器**

   同步计数器电路中，所有触发器的时钟都与同一个时钟源连在一起，每个触发器的状态变化都与时钟同步。

2. **异步计数器**

   异步计数器中各个触发器的时钟不是来自于同一个时钟源。第一个触发器的状态变化与时钟同步，其他则依次滞后。异步计数器的工作原理有点像“多米诺骨牌”。

3. **计数器的预置功能**

   计数器通常有一个预置控制端 LD，用来设置计数器开始计数时的初始值。预置分为两种方式：

   - 同步预置（同步置数）。预置控制信号 LD 有效后并不立即将计数器的输入值加载到输出端，而是要等待时钟有效边沿（上升沿或下降沿）到来时才能完成预置功能，即预置的实现与时钟有效边沿同步。
   - 异步预置（异步置数）。一旦预置信号 LD 有效，计数器立即置数，而与时钟无关。

4. **计数器的复位（清零）功能**

   计数器通常有一个复位控制端 CLR，用来清零计数器。复位功能也分为同步复位和异步复位，其含义与同步置数与异步置数的含义类似。

5. **时钟有效边沿的选择**

   若计数器对时钟的上升沿进行计数，即每来一个时钟上升沿，计数器加 1 或减 1，那么可称该计数器为上升沿计数器；反之则可称之为下降沿计数器。

6. **计数器级联**

   通过将计数器进行级联可以扩展计数器的位数，也就是计数器的模。

三、 实验任务

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/Digital-Circuit/Lab005.git

3.1 锁存器构成的寄存器和触发器构成的寄存器

比较锁存器和触发器构成的寄存器工作方式的不同之处。

1. 打开项目下的原理图 reg.dlsche，结合原理图中的相关注释，熟悉原理图中的器件及其功能。
2. 启动仿真。
3. 通过键盘按键 E，设置寄存器的使能端输入 EN=0（使能），观察电路的状态。
4. 选择八位交互式数字信号源，在属性栏中将其数字信号值改为 08。然后，观察两个寄存器的输出值，有何不同？
5. 按下并抬起键盘按键 C，向寄存器 74LS377D 输入一个时钟上升沿，再次观察它的输出值，有何变化？

通过前面的实验步骤可知，在两种类型的寄存器都使能的情况下，锁存寄存器的输出始终与输入端的数据保持一致，而触发寄存器的输入数据必须等待时钟上升沿到来时，才能传送到寄存器的输出端。下面检验使能端 EN 对寄存器的控制作用。

1. 通过键盘按键 E，设置寄存器使能输入端 EN=1（不使能）。
2. 将八位交互式数字信号源的值修改为 4b。
3. 观察两个寄存器的输出值是否为 4b？按下并抬起键盘按键 C，寄存器 74LS377D 的值有变化吗？
4. 设置 EN=0，再次观察寄存器输出值是否为 4b？
5. 按下键盘按键 C 不动，观察触发寄存器的输出值是否为 4b？抬起按键 C 后呢？

通过实验可知，当锁存寄存器不使能时，它的输出值保持不变，不会受输入值的影响，一旦使能，则立即将输入数据传送到输出端；当触发寄存器不使能时，即使时钟上升沿到来，也不能将输入端的数据传送到输出端。

3.2 移位寄存器

学习右移寄存器的工作原理。请读者按下列步骤进行实验：

1. 打开项目下的原理图文件 shift_reg.dlsche，查看 4 位右移寄存器的电路。
2. 启动仿真，观察各个 D 触发器的输出值，蓝色网络表示无效电平。
3. 通过按键 D 设置第一个触发器的数据输入 D1=1。
4. 通过按键 C 输入一个时钟脉冲，观察 Q1~Q4 的值，可以看到 Q1=1。
5. 通过按键 D 设置 D1=0。
6. 通过按键 C 逐个输入时钟脉冲，可以看到 Q1~Q4 指示灯依次点亮。表示输入数据在向右移位。

通过上述实验可知，右移寄存器的工作原理是，在同一个时钟驱动下，一序列二进制数依次向右传递。

设计循环移位寄存器

假设 Q1、Q2、Q3、Q4 的初始值为 1000，需要在时钟的驱动下进行循环右移，该如何改进电路？请读者在右移寄存器的基础上直接修改此电路图，实现一个 4 位循环右移寄存器。

提示：实现循环移位，需要将 Q4 端接到左侧第一个寄存器的数据输入端，形成一个数据环路。启动仿真后，需要使用触发器的异步置数端 PR 和异步复位端 CLR 将各个触发器初始化为对应的值，也就是 Q1 为 1，Q2、Q3、Q4 为 0，然后再触发时钟产生循环移位。

3.3 移位寄存器 74LS194

验证移位寄存器 74LS194（参见附录）的功能，注意 SR 触发器的使用方法：

1. 打开项目下的原理图 74LS194.dlsche。该原理图文件中绘制了 74LS194 的内部原理图。
2. 启动仿真，根据表 5-1 对 74LS194 进行功能验证。

表 5-1：74LS194 功能表

输入					输出	功能
CLR	Mode	CLK	Serial	Parallel	QA QB QC QD	
	S1 S0		SLSR	A B C D		
0	X X	X	X X	X XXX	0000	清除
1	X X	X	X X	X XXX	Q_{A0} Q_{B0} Q_{C0} Q_{D0}	保持
1	11	↑	X X	a b c d	a b c d	并行送数
1	01	↑	X 1	X XXX	$1Q_{An}Q_{Bn}Q_{Cn}$	右移
1	01	↑	X 0	X XXX	$0Q_{An}Q_{Bn}Q_{Cn}$	
1	10	↑	1 X	X XXX	$Q_{Bn}Q_{Cn}$ $Q_{Dn}1$	左移
1	10	↑	0 X	X XXX	$Q_{Bn}Q_{Cn}$ $Q_{Dn}0$	
1	00	X	X X	X XXX	Q_{A0} Q_{B0} Q_{C0} Q_{D0}	保持

3.4 使用 JK 触发器设计的同步计数器

测试一个由 3 个 JK 触发器构成的模 8 同步计数器。实验步骤如下：

1. 打开项目下的原理图文件 m8.dlsche。这是一个由 3 个 JK 触发器构成的 3 位模 8 同步计数器，结合电路注解，理解其工作原理。
2. 启动仿真，通过按键 C 依次输入时钟脉冲，观察计数显示器的计数值。

设计模 16 同步计数器

在电路中添加一个 JK 触发器以及必要的逻辑门，实现一个模 16 同步计数器，并通过仿真验证。

3.5 使用 D 触发器设计加法计数器

接下来，读者可以学习使用 D 触发器设计一个模 7 加法计数器的完整过程，使读者掌握计数器设计的一种基本方法。加法计数器是一种简单的有限状态机，它在时钟沿的驱动下，由当前稳定状态（当前计数值）进入下一个状态（计数器加 1 后的值）。请读者按照下列步骤，完成模 7 加法计数器的设计：

1. 首先在项目下新建一个原理图文件 m7.dlsche。
2. 模 7 加法计数器可以表示的值为 0~6，至少需要 3 位二进制数才能表示，所以需要 3 个 D 触发器。每一个 D 触发器存储一位二进制数。在原理图中依次放置 3 个 D 触发器，各个触发器之间留出一定的间隔，以便放置其他器件。如图 5-2 所示。

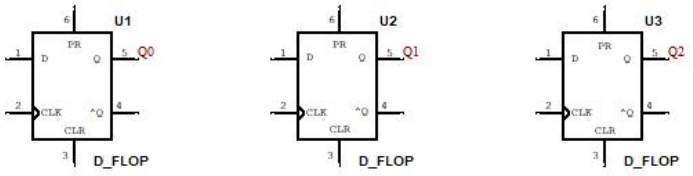


图 5-2：模 7 加法计数器的 3 个 D 触发器

3. 为了可以随时清零计数器，使其从 0 开始计数，就需要在原理图中添加一个手动的复位按键 R，如图 5-3 所示。

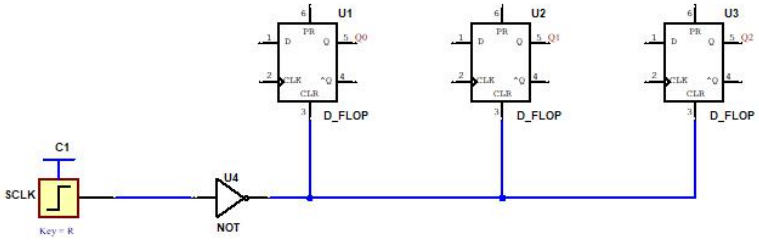


图 5-3：模 7 加法计数器的复位控制

4. 完成复位电路后，启动仿真，测试复位功能。按下复位按键 R 后抬起，使 Q2Q1Q0=000（计数器值为 0）。

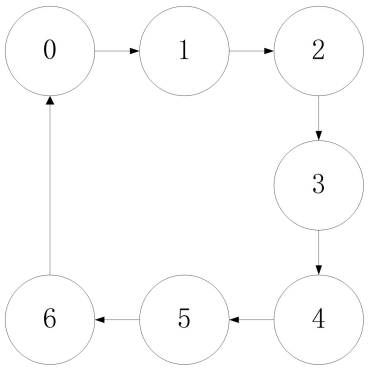


图 5-4：模 7 加法计数器的状态转换图

5. 模 7 加法计数器的状态转换如图 5-4 所示。当 $Q_2Q_1Q_0$ 稳定为某个状态时，表示一个唯一的数，例如 $Q_2Q_1Q_0$=011 时表示 3，那么下一个计数值一定是 $Q_2Q_1Q_0$=100，也就是当前状态决定了下一个状态。所以，只需得到次态方程 $Q^{n+1}=f(Q_{2n},Q_{1n},Q_{0n})$ 就能通过逻辑电路实现当前状态到下一个状态的转换。状态转换如表 5-2 所示。

表 5-2：模 7 加法计数器的状态转换表

时钟个数	PS（现态）			NS（次态）		
	$Q_2^nQ_1^nQ_0^n$			$Q_2^{n+1}Q_1^{n+1}Q_0^{n+1}$		
0	000			0	0	1
1	0	0	1	0	1	0
2	0	1	0	0	1	1
3	0	1	1	1	0	0
4	1	0	0	1	0	1
5	1	0	1	1	1	0
6（循环）	1	1	0	000		

提示：Q_2^n、Q_1^n、Q_0^n 分别表示当前各位触发器的值，Q_2^{n+1}、Q_1^{n+1}、Q_0^{n+1} 分别表示时钟上升沿到来后各位触发器的值。

6. 由模 7 加法计数器的状态转换表，可得到各位的次态方程。当前计数值为 3、4、5 时，下一个状态 Q_2^{n+1} 为 1。若用 m0~m6 表示计数值 0~6，那么

$$Q_2^{n+1}= m3 + m4 + m5$$（mx 为高电平时有效）

同理可得

$$Q_1^{n+1}= m1 + m2 + m5$$

$$Q_0^{n+1}= m0 + m2 + m4$$

7. 可以由 Q_2^n、Q_1^n、Q_0^n 通过 3-8 译码器得到 m0~m5 这 6 个值对应的信号。当译码输出任意一位有效时，就能确定当前计数值。例如，m5 有效（3-8 译码器输出低电平表示有效），那么当前计数值为 5。于是得到最终的电路如图 5-5 所示。注意在图 5-5 中，计数器输出的 $Q_2Q_1Q_0$ 通过网络标签与数码管和 3-8 译码器的输入端连接，这样既可显示计数器的值，又对计数器的现态完成了译码。
8. 读者仿照图 5-5 绘制完成电路后，可以仿真验证模 7 加法计数器的功能。

设计模 7 减法计数器

通过上述实验步骤，最终设计了一个模 7 加法计数器。请读者在项目下新建一个原理图文件 m7_minus.dlsche，并仿照上述步骤设计一个模 7 减法计数器。计数过程为 0->6->5->4->3->2->1->0->6->…。

提示：

① 先画出模 7 减法计数器的状态转换表，

② 根据状态转换表，写出次态方程；

③ 根据次态方程，完成电路设计。

提交作业

在提交作业前，读者需要将模 7 加法计数器和减法计数器 SVG 图形文件的链接添加到 README.md 文件中，这样，当使用浏览器查看提交后的线上项目时，就可以从 README.md 文件中看到电路了。方法如下：

1. 双击“项目管理器”中的 README.md 文件，使用一种合适的文本编辑器或者 Markdown 编辑器打开此文件。
2. 在 README.md 文件的末尾添加如下的文本：

```
# 模 7 加法计数器
![raw svg](m7.dlsche.svg)
# 模 7 减法计数器
![raw svg](m7_minus.dlsche.svg)
```

3. 保存 README.md 文件。

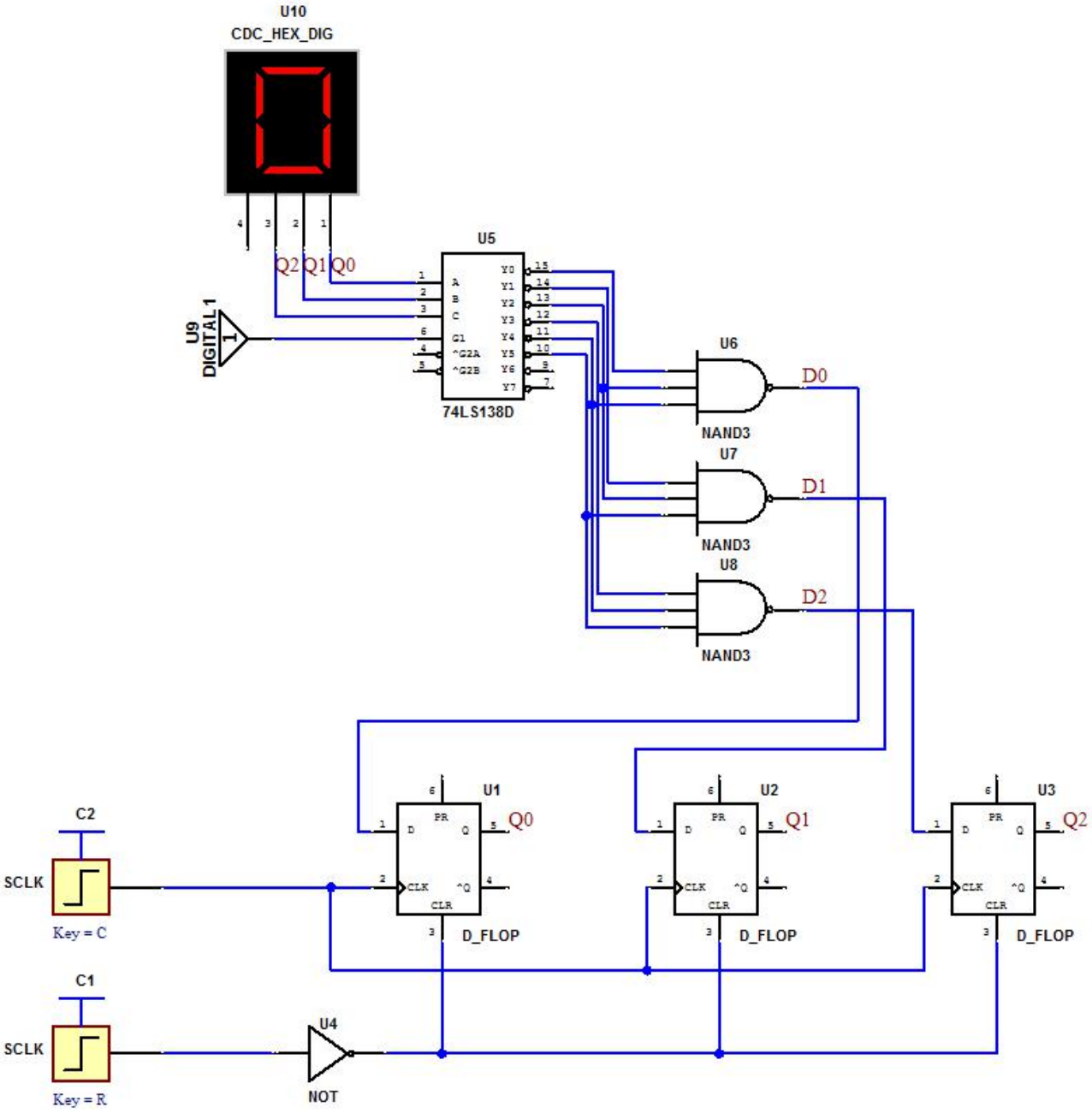


图 5-5：模 7 加法计数器

3.6 异步模 16 加法计数器

仿真验证异步模 16 加法计数器：

1. 打开项目下的原理图文件 m16.dlsche。
2. 启动仿真，验证其计数功能，并尝试描述其工作原理。

设计异步模 10 加法计数器

将异步模 16 加法计数器改造为异步模 10 加法计数器。通过改造电路，使计数器的计数到达 9 后又从 0 开始计数。提示，可以在计数值为 10 时，立即清零，这样就成了一个模 10 的计数器。

3.7 同步置数/异步清零十六进制加法计数器 74LS161

仿真验证 74LS161 的功能：

1. 打开项目下的原理图文件 74LS161.dlsche。该原理图文件中绘制了 74LS161 的内部原理图。
2. 启动仿真，根据表 5-3 的内容对 74LS161 进行功能验证。

表 5-3：74LS161 功能表

输入						输出
CLR	LOAD	ENT	ENP	CLK	ABC D	QA QB QC QD
0	X	X	X	X	X XXX	0000
1	0	X	X	↑	abc d	a b c d
1	1	1	1	↑	X XXX	计数(加 1)
1	1	0	X	X	X XXX	保持
1	1	X	0	X	X XXX	保持

四、 思考与练习

1. 在并行输入、串行输出的转换中，若二进制数码高位在前、低位在后，应采取何种移位方式？
2. 时序电路自启动的作用何在？它和人工预置的方法比较，对电路的作用有何异同？
3. 环形计数器的最大优点和最大缺点各是什么？为什么环形计数器的输出在译码时不存在竞争冒险？
4. 并使用两片 74LS161 芯片设计一个模 37 加法计数器。提示，可以使用两片 74LS48 显示两位十进制的计数值。

实验 6 顺序脉冲发生器

实验性质：设计
建议学时：2 学时

一、 实验目的

- 掌握顺序脉冲发生器的原理。
- 掌握顺序脉冲发生器的设计方法。

二、 预备知识

2.1 定时脉冲/节拍脉冲

按固定时间顺序再现的脉冲序列称为定时脉冲，也称为节拍脉冲。一个数字系统之所以有条不紊的工作，完全是受到定时脉冲的指挥。

2.2 顺序脉冲发生器

在数字系统和计算机中，往往需要机器按照人们事先规定的顺序进行运算或操作，这就要求机器的控制部分不仅能正确地发出各种控制信号，而且要求这些控制信号在时间上有一定的先后顺序。用顺序脉冲发生器可以实现这一功能。顺序脉冲也叫相位脉冲，或节拍脉冲。计算机之所以能一步一步地运行，就是要靠节拍脉冲一拍一拍地指挥。

顺序脉冲发生器用于产生时间上有先后顺序的脉冲，通常可以用移位寄存器产生，也可以由计数器和译码器组合而成，计数器状态提供译码器的地址码，译码器把该地址代码译成有一定顺序的点位脉冲。

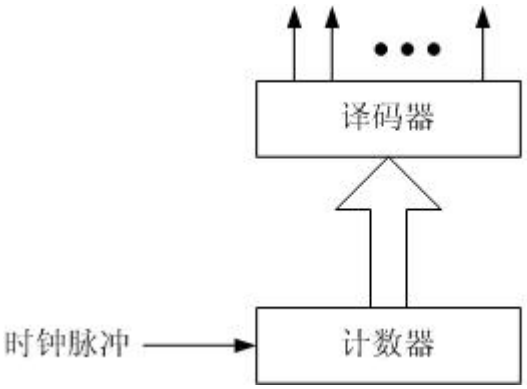


图 6-1：顺序脉冲实现原理图

三、 实验内容

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/Digital-Circuit/Lab006.git

3.1 自然二进制码同步八进制计数器顺序脉冲发生器

观察一个由自然二进制码同步八进制计数器构成的顺序脉冲发生器的波形，理解顺序脉冲发生器的工作原理。

1. 打开项目下的原理图文件 binary.dlsche。原理图中包含一个由 JK 触发器构成的同步模 8 计数器，一个 3-8 译码器和一个逻辑分析仪。3-8 译码器输出的 8 相顺序脉冲分别与逻辑分析仪的 8 个通道连接。通过逻辑分析仪可以观察顺序脉冲波形。
2. 启动仿真，观察计数器值的变化情况。双击逻辑分析仪，观察脉冲波形，理解顺序脉冲发生器的工作原理。

3.2 设计格雷码同步八进制计数器顺序脉冲发生器

在项目下新建一个原理图文件 gray.dlsche，在其中设计一个格雷码同步八进制计数器，并使用该计数器以及 3-8 译码器实现一个 8 相顺序脉冲发生器。格雷码如表 6-1 所示。产生的顺序脉冲信号应与图 6-2 中的波形一致。

表 6-1：常用 BCD 码

十进制数	8421 码	余 3 码	格雷码
0	0000	0011	0000
1	0001	0100	0001
2	0010	0101	0011
3	0011	0110	0010
4	0100	0111	0110
5	0101	1000	0111
6	0110	1001	0101
7	0111	1010	0100
8	1000	1011	1100
9	1001	1100	1101

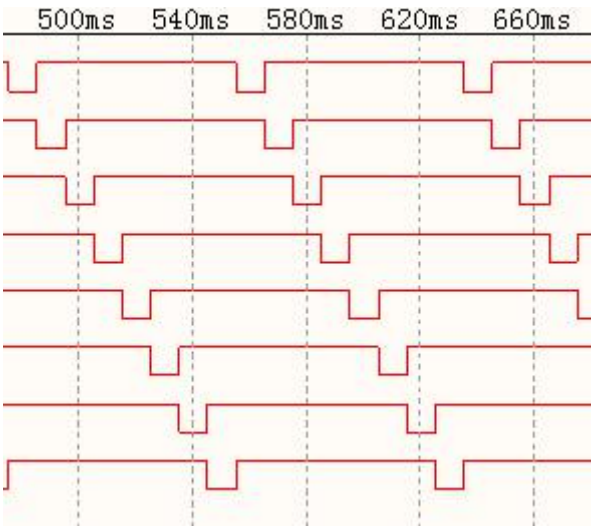


图 6-2：顺序脉冲信号

提交作业

在提交作业前，读者需要将顺序脉冲发生器 SVG 图形文件的链接添加到 README.md 文件中，这样，当使用浏览器查看提交后的线上项目时，就可以从 README.md 文件中看到电路了。方法如下：

1. 双击“项目管理器”中的 README.md 文件，使用一种合适的文本编辑器或者 Markdown 编辑器打开此文件。
2. 在 README.md 文件的末尾添加如下的文本：

```
# 格雷码同步八进制计数器顺序脉冲发生器
```

```
![raw svg](gray.dlsche.svg)
```

3. 保存 README.md 文件。
4. 提交作业。

四、 思考与练习

1. 计数器采用异步有何优缺点？
2. 计数器状态采用格雷码的目的是什么？
3. 若用移位寄存器获得节拍脉冲信号，其电路是怎样的？
4. 使用 74LS90 设计一个占空比为 50%的十分频器。

实验 7 序列信号的产生和检测

实验性质：设计
建议学时：2 学时

一、 实验目的

- 掌握序列信号发生器的设计方法。
- 掌握序列信号检测器的设计方法。

二、 预备知识

2.1 序列信号发生器原理

在数字信号的传输和数字系统的测试中，有时需要用到一组特定的串行数字信号，称之为序列信号。产生序列信号的电路称为序列信号发生器。

序列信号发生器可由计数器和数据选择器构成，其结构如图 7-1 所示。其中的锁存输出功能是为了消除序列信号产生时可能出现的毛刺现象。

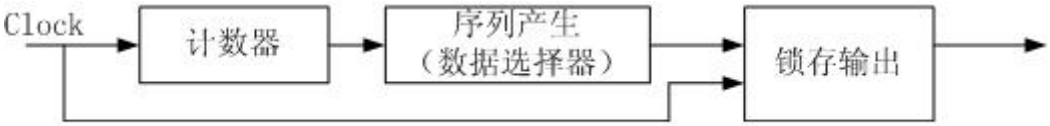


图 7-1：序列信号发生器结构图

2.2 序列信号检测器的基本工作过程

序列信号检测器用于检测一组或多组由二进制码组成的脉冲序列信号，在数字通信中有着广泛的应用。当序列信号检测器连续收到一组串行二进制码后，如果这组码与检测器中预先设定好的码相同，则输出 1，否则输出 0。由于这种检测的关键在于正确码的接收必须是连续的，这就要求检测器必须记住前一次的正确码及正确序列，直到在连续的检测中所收到的每一位码都与预置的对应码相同。在检测过程中，任何一位不相等都将回到初始状态重新开始检测。

三、 实验内容

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/Digital-Circuit/Lab007.git

3.1 8 选 1 数据选择器 74LS151

由于设计序列信号发生器的时候，需要用到 8 选 1 数据选择器 74LS151（参见附录），所以在这里先了解一下它的功能：

1. 打开项目下的原理图文件 74LS151.dlsche。该原理图文件中绘制了 74LS151 的内部原理图。
2. 启动仿真，根据表 7-1 验证 74LS151 的功能。

表 7-1：74LS151 功能表(“/”表示逻辑取反)

输入		输出
数据选择	选通	Y W
C B A	G	
X XX	1	01
000	0	D0 /D0
001	0	D1 /D1
010	0	D2 /D2
011	0	D3 /D3
100	0	D4 /D4
101	0	D5 /D5
110	0	D6 /D6
111	0	D7 /D7

3.2 序列信号发生器

在本实验中读者将看到一个序列信号发生器是如何工作的，并尝试产生不同的序列信号。

1. 打开项目下的原理图文件 happen.dlsche。在此原理图中提供了一个使用二-五进制计数器 74LS90（参见附录）和 8 选 1 数据选择器 74LS151 组成的序列信号发生器，通过为 74LS151 输入端提供的信号来产生“10110”序列信号。
2. 启动仿真，打开逻辑分析仪观察序列信号的波形，验证序列信号发生器的功能。
3. 修改为 74LS151 输入端提供的信号来产生“01110”序列信号。

3.3 设计序列信号检测器

接下来，读者需要在之前学习的序列信号发生器的基础上使用 D 触发器设计一个序列信号检测器：

1. 在项目下新建一个原理图文件 check.dlsche。
2. 在参考本实验 3.2 节的基础上，先设计一个可以产生序列信号“11110”的序列信号发生器，然后使用 D 触发器设计一个序列信号检测器，当检测到序列信号“11110”时，点亮指示灯。在设计序列信号检测电路时，可以参考图 7-2，其中提供了序列信号“101”的检测电路。注意，序列信号发生器与检测器要使用同一个手动时钟。检测结果信号还可以接入一个计数器，并使用十六进制数码管显示出检测到的序列信号数量。
3. 仿真验证序列信号检测器的功能。

提交作业

在提交作业前，读者需要将序列信号检测器 SVG 图形文件的链接添加到 README.md 文件中，这样，当使用浏览器查看提交后的线上项目时，就可以从 README.md 文件中看到电路了。方法如下：

1. 双击“项目管理器”中的 README.md 文件，使用一种合适的文本编辑器或者 Markdown 编辑器打开此文件。
2. 在 README.md 文件的末尾添加如下的文本：

```
# 序列信号检测器
![raw svg](check.dlsche.svg)
```

3. 保存 README.md 文件。
4. 提交作业。

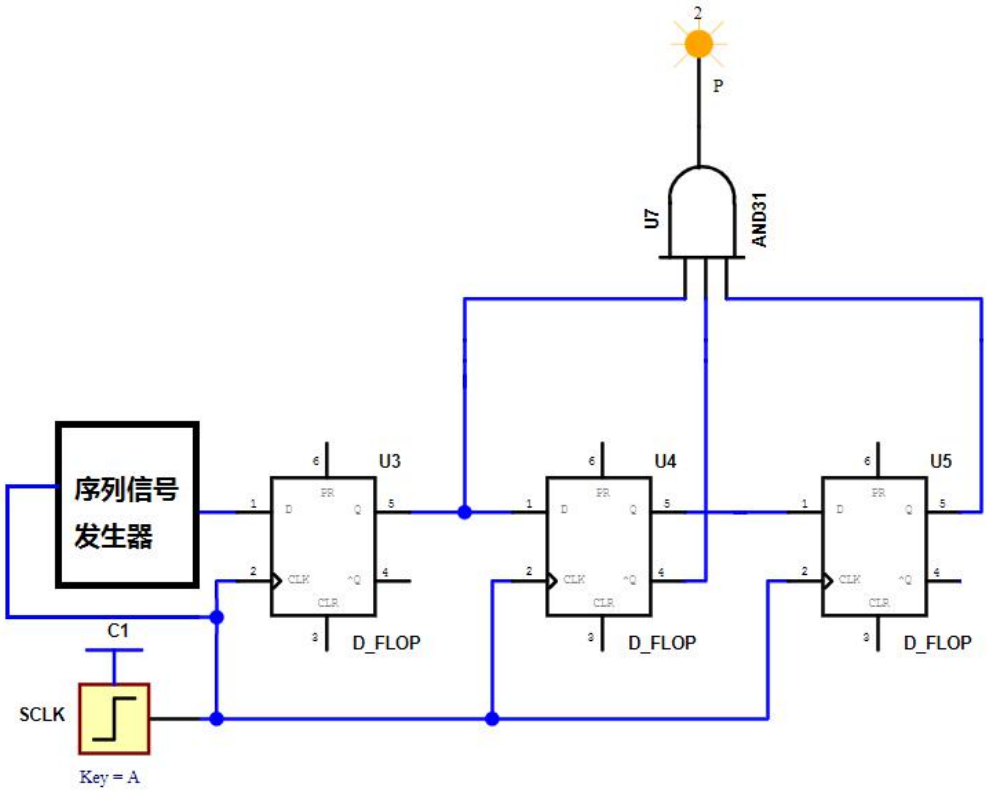


图 7-2：序列信号“101”的检测器

四、 思考与练习

1. 信号发生器根据结构不同,可分为反馈移位型和计数型两种。自行查阅资料，了解不同结构的信号发生器的特点。

实验 8 存储器

实验性质：验证+设计
建议学时：2 学时

一、 实验目的

- 了解大规模集成存储器的工作原理。
- 熟悉大规模集成存储器的工作特性、使用方法以及应用。
- 熟悉有限状态机的相关理论知识，掌握有限状态机的设计方法。

二、 预备知识

2.1 随机存取存储器

随机存取存储器可将数据随机地读出或写入存储器中任一位（或若干位），因而又称为读写存储器，简称 RAM。RAM 主要由地址译码器、存储矩阵、读/写控制器以及输入输出电路等组成，如图 8-1 所示。

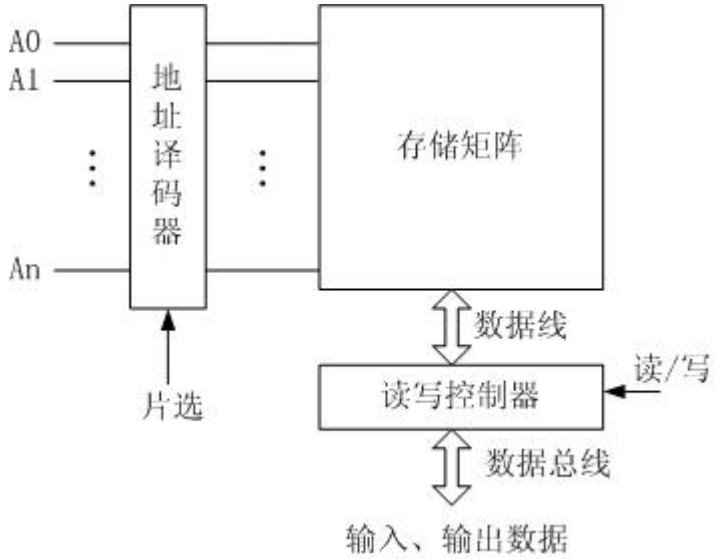


图 8-1：RAM 组成框图

其中，存储矩阵是 RAM 的核心部分，它由许多相同的存储元构成，并排列成矩阵结构形式。每一个存储元可以存放一位数据。若干个（如 8 个）存储元构成一个字，不同的字用不同的地址表示。该地址称为单元地址。

RAM 中的地址译码器用来对输入地址码进行译码，以确定需要访问的存储单元。读写控制器在读/写信号控制下控制数据的读出或写入。片选信号是在用若干个单片组成的存储体中，实现对各单片存储器工作的控制。输入输出（I/O）线是数据进出的通道，实际上是一线两用，它也由读/写、片选信号控制。I/O 线的根数，取决于一个地址单元中包含的位数（叫字长）。例如容量为 1024 字×4 位，则每个地址单位有 4 个存储元（即字长为 4），故有 4 根 I/O 线。I/O 线一般具有三态输出结构。

常用的存储器芯片 2114 是一种 1024×4 位的静态随机存取存储器，采用 NMOS 工艺制作，各引出端功能如表 8-1 所示。

表 8-1:2114 芯片引出端功能

端名	功能
A9~A0	地址输入端
I/O4~I/O1	数据输入输出端
WR	写控制
CS	芯片选择

2114 的读/写访问：

① CS=0，WR=1 时读出当前输入地址指定存储器单元的内容，读出不会破坏存储单元的内容。

② CS=0，WR=0 时将数据写入当前地址指定存储单元。

随机存取存储器是一种快速存取的存储器，广泛应用于计算机或其他数字系统作主存储器使用，通电后可以根据要求写入信息，并在工作过程中能不断更改其存储内容。但一旦断电，信息即全部消失。

2.2 RAM 的使用

在计算机中，RAM 是连接到总线上的，各地址输入端所接的总线叫地址总线，地址总线是单向传输的；各数据输入输出端所接的总线叫数据总线，数据总线是双向传输的，通常用三态门选通；各控制输入端所接的总线叫做控制总线。当某个设备要访问存储器时，首先要申请到总线占用权；然后把要访问单元的地址码，通过地址寄存器送到地址总线；再送片选信号和读/写信号，同时打开挂在数据总线上的相应的三态门，或向存储器存（写）数据，或向存储器取（读）数据。

三、 实验内容

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/Digital-Circuit/Lab008.git

3.1 2114 存储器的读写

通过仿真，对 2114 进行读写功能测试，掌握 2114 的管脚功能和使用方法。实验步骤如下：

1. 打开项目下的原理图文件 2114.dlsche。在此电路中可以实现对 2114 的读写。由于 2114 的 D0~D3 既作为输入，又作为输出，所以需要通过三态门芯片 74LS244 完成对 2114 的读写控制，避免发生冲突。具体的方法是在控制 2114 的 W 输入信号的同时，也控制 74LS244 的哪一组三态门可以导通。
2. 启动仿真前，将 2114 芯片的输入管脚 E 设为 0，W 设为 1，输入地址设为 0，准备读出 0 地址单元中存储的内容。
3. 启动仿真，由于 2114 的初始化内存均为无效值，所以从 0 地址单元读出的是一个无效值。

下面依次在 7~0 号存储单元中写入 7~0。步骤如下：

1. 将 74LS244 芯片的输入端 1A1~1A4 的值改为 07。
2. 将 2114 的地址端输入的值也改为 07。
3. 将 2114 的输入管脚 W 设为低电平。此时将十进制数 7 写入地址 7 对应的存储单元中。
4. 先将 2114 的地址减 1（地址设为 6），再将输入数据改为 6。此时将数据 6 写入地址为 6 的存储单元。
5. 仿照上面的步骤，将 7~0 依次写入地址为 7~0 的 8 个存储单元。
6. 数据写完后，将 W 设为高电平，依次读出地址为 7~0 的 8 个存储单元。为了方便显示从存储器中读取到的值，读者可以将 74LS244 的输出端 2Y1~2Y4 连接一个 16 进制 7 段数码管。

3.2 使用存储器实现码组变换

在项目下新建一个原理图文件 transform.dlsche，在其中设计一个 4 位二进制自然码到格雷码的码组变换电路。格雷码如表 8-2 所示。

表 8-2：二进制自然码与格雷码对照表

4 位二进制自然码	格雷码	十进制数
0000	0000	0
0001	0001	1
0010	0011	2
0011	0010	3
0100	0110	4
0101	1110	5
0110	1010	6
0111	1000	7
1000	1100	8
1001	1101	9

提示：

将 4 位二进制码作为存储器的地址，而将其对应的格雷码保存在该地址单元中。这样，4 位二进制码就与存储器中对应位置的格雷码建立了映射关系，通过地址值访问存储器得到的内容就是格雷码。在测试电路时，首先需要将 4 位二进制码对应的格雷码依次写入地址为 0~9 的存储器单元中，然后尝试读出存储器中的值，验证读出的值是否是 4 位地址值对应的格雷码。

提交作业

在提交作业前，读者需要将码组变换电路 SVG 图形文件的链接添加到 README.md 文件中，这样，当使用浏览器查看提交后的线上项目时，就可以从 README.md 文件中看到电路了。方法如下：

1. 双击“项目管理器”中的 README.md 文件，使用一种合适的文本编辑器或者 Markdown 编辑器打开此文件。
2. 在 README.md 文件的末尾添加如下的文本：

```
# 码组变换
![raw svg](transform.dlsche.svg)
```

3. 保存 README.md 文件。
4. 提交作业。

四、思考与练习

1. 自行查阅资料，了解计算机内部存储器的相关知识，掌握每种存储器的优缺点。
2. 对一块 2114，每单元地址的内容为一个字，字长是 4 位（bit）。若要进行位扩展，字长变为 8 位，应如何连接和操作（使用多块 2114）。要想把单元数增加一倍，这叫字扩展（字数扩展），应如何连接。
3. 使用 2114 和 D 触发器设计一个格雷码加法计数器。自行查阅关于有限状态机的书籍，了解有限状态机的概念以及设计方法。格雷码加法计数器的工作原理是：在时钟脉冲的驱动下，一种 4 位格雷码转换为另一种 4 位格雷码。实质上是一个有限状态机。格雷码计数器状态转换图如图 8-2 所示。将 4 位格雷码依次写入存储器地址 0~15 单元中，使用 4 个 D 触发器输出作为存储器地址。而这四个 D 触发器的输入（也就是加计数以后所得下一个格雷码的地址）由当前存储器输出（格雷码）决定。

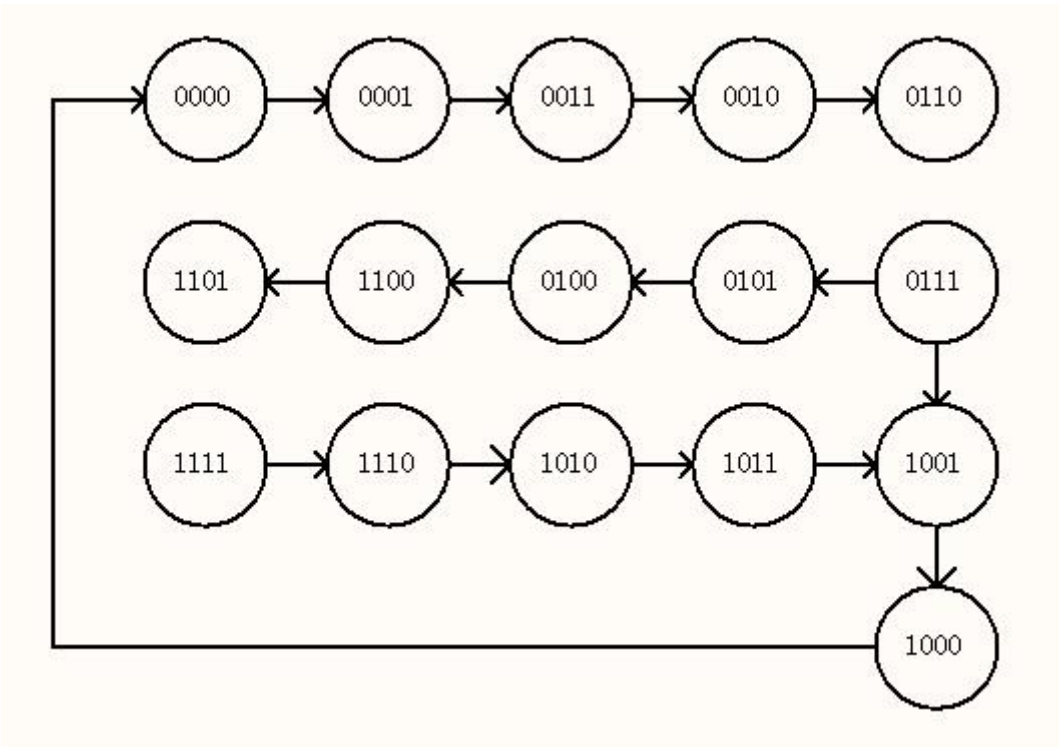


图 8-2:格雷码计数器状态转换图

实验 9 密码电子锁的设计

实验性质：综合设计
建议学时：2 学时
难易程度：简单

一、 实验目的

- 理解密码电子锁电路的基本原理。
- 进一步熟悉 D 触发器的基本功能。

二、 预备知识

密码电子锁一般预先设定密码，用每个码位去控制触发器翻转，若码位按错则码位触发器不能翻转。在设计密码电子锁时，可以用密码依次控制各位 D 触发器的翻转，达到密码开锁的目的。

例如，可以使用 4 个 D 触发器串行连接，构成四位密码电路。S0~S9 代表键盘上的按键 0~9。当密码为 1469 时，S1、S4、S6、S9 分别是 1、4、6、9 四位密码的按钮端（可以使用单周期时钟控制），并分别接 4 个 D 触发器的时钟端。平时 4 个 D 触发器的 CLK 皆保持高电平状态，触发器保持原状态不变。当按下并抬起 S1 后，Q1=D1=1;再按下并抬起 S4 后，Q2=D2=Q1=1;同理，按下并抬起 S6 后，Q3=D3=Q2=1;按下并抬起 S9 后，Q4=D4=Q3=1，用此 Q4=1 去控制开锁机构即可（可以接一个指示灯，灯亮表示开锁）。

三、 实验内容

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为
https://www.codecode.net/engintime/Dream-Logic/Project-Template/blank.git

参考预备知识中的内容，在原理图文件 top.dlsche 中设计一个 4 位密码电子锁。要求密码初始为“1469”，设计完成后仿真测试密码电子锁的功能。

改进设计

1. 成功开锁后，密码电子锁不应一直处于开锁状态，通常开锁信号触发开锁机构后，保持一段时间就会复位。设计一个自动复位电路，当密码电子锁成功开锁一段时间后，能够自动完成复位功能。
2. 使用四个 16 进制 7 段数码管依次显示用户输入的 4 个密码，复位后清零。
3. 想一想如何让用户可以修改预设的密码。

实验 10 七人表决器的设计

实验性质：综合设计
建议学时：2 学时
难易程度：简单

一、 实验目的

- 熟悉组合电路的设计方法。

二、 实验内容

多数表决器是对于一个行为由多个人投票，如果同意的票数多于反对的票数，则认为此行为有效通过。反之，则无效。要求读者使用四位二进制超前进位全加器 74LS283（参见附录 C），设计一个七人多数表决器（不允许弃权），并使用数码管将表决中赞成的票数显示出来。可以使用一位交互式数字信号作为表决按键，输入 0 表示反对，输入 1 表示赞成。按照下面的步骤完成电路设计：

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为
https://www.codecode.net/engintime/Dream-Logic/Project-Template/Digital-Circuit/Lab010.git

1. 打开项目下的原理图文件 74LS283.dlsche。该原理图文件中绘制了四位二进制超前进位全加器 74LS283 的内部原理图。
2. 启动仿真，根据表 10-1 验证 74LS283 的功能。
3. 在项目下新建一个原理图文件 7vote.dlsche，使用 3 个 74LS283 设计一个七人多数表决器。

表 10-1：74LS283 功能表

输入端			输出端	
加数	被加数	进位输入	和	进位输出
A	B	Ci-1	S	Ci
0000	0000	0	0000	0
0000	0000	1	0001	0
0000	1111	0	1111	0
0000	1111	1	0000	1
1111	0000	0	1111	0
1111	0000	1	0000	1
1111	1111	0	1110	1
1111	1111	1	1111	1

提交作业

在提交作业前，读者需要将七人多数表决器 SVG 图形文件的链接添加到 README.md 文件中，这样，当使用浏览器查看提交后的线上项目时，就可以从 README.md 文件中看到电路了。方法如下：

1. 双击“项目管理器”中的 README.md 文件，使用一种合适的文本编辑器或者 Markdown 编辑器打开此文件。
2. 在 README.md 文件的末尾添加如下的文本：

```
# 七人多数表决器
![raw svg](7vote.dlsche.svg)
```

3. 保存 README.md 文件。
4. 提交作业。

实验 11 智力竞赛抢答器的设计

实验性质：综合设计
建议学时：2 学时
难易程度：一般

一、 实验目的

- 了解一个数字系统的基本组成及它的控制电路的设计考虑。
- 熟悉数字系统模块化的设计方法。
- 熟悉集成芯片的综合应用。
- 学习如何用实验的方法来完善理论设计以及用实验的方法来验证电路模块的输入、输出逻辑。
- 熟悉数字系统逐级仿真调试的方法，从子模块到数字系统的功能验证过程，也是自底向上设计实现过程。

二、 预备知识

抢答器在各类竞赛性质的场合得到了广泛的应用，它的出现消除了原来由于人眼的误差而未能正确判断最先抢答的人的情况。

抢答器的原理比较简单，首先必须设置一个抢答允许标志位，目的就是为了允许或者禁止抢答者按下抢答按钮；如果抢答允许位有效，抢答器开始工作，4 个抢答者谁先按下抢答按钮，则谁抢答成功，同时记录按钮的序号，这样做的目的就是为了禁止后面再有人按下按钮。总的来说，抢答器的实现原理就是在抢答允许位有效后，第一个按下按钮的人将禁止再有按钮按下，同时显示抢答者的序号。

三、 实验内容

3.1 学习绘制模块化的原理图

为了使原理图更加简洁、明了，本实验将引入模块的概念，Dream Logic 支持自定义模块的功能，可以将绘制的部分原理图封装成一个模块供使用，大大提高了原理图的识图效率，还可以提供模块的复用功能。

请读者使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为 https://www.codecode.net/engintime/Dream-Logic/Project-Template/blank.git

1. 新建一个原理图文件 child.dlsche。此原理图将作为子模块的内部原理图。
2. 打开原理图文件 child.dlsche，放置一个非门。
3. 点击菜单栏中的“绘制->端口”，将两个端口分别和非门的输入端和输出端连接，放置时注意连接正确。如图 11-1 所示。注意，这里使用了端口锚点与器件管脚锚点直接连接的方法，读者也可以将端口放置在距离管脚一定距离的地方，然后使用网络完成连接。

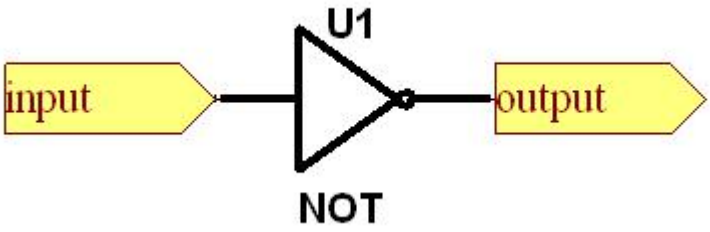


图 11-1：添加端口

4. 修改端口名称。选中端口，在属性窗口中将非门输入端口的名称修改为 input，输出端口的名称修改为 output。保存文件。
5. 打开原理图文件 top.dlsche，点击菜单栏中的“绘制->模块”，在原理图中拖放至所需大小。
6. 添加模块接口。点击菜单栏中的“绘制->模块接口”，根据 child.dlsche 中的输入端口和输出端口的数量在模块上放置相同数量的模块接口。通常输入接口放置在左侧，输出接口放置在右侧。
7. 修改模块接口名称。选中模块接口，将其名称修改为与原理图 child.dlsche 中的端口相对应的名称，这样就表示 top.dlsche 中的模块接口与 child.dlsche 中的端口相连接。也就是说相同名称的模块接口和子模块中的端口会连接在一起。
8. 为模块匹配原理图文件。在原理图文件 top.dlsche 中选中模块，在属性窗口中的“模块原理图文件”属性中选择 child.dlsche，这样就完成了模块与其封装电路的匹配。完成后的原理图文件 top.dlsche 如图 11-2 所示。

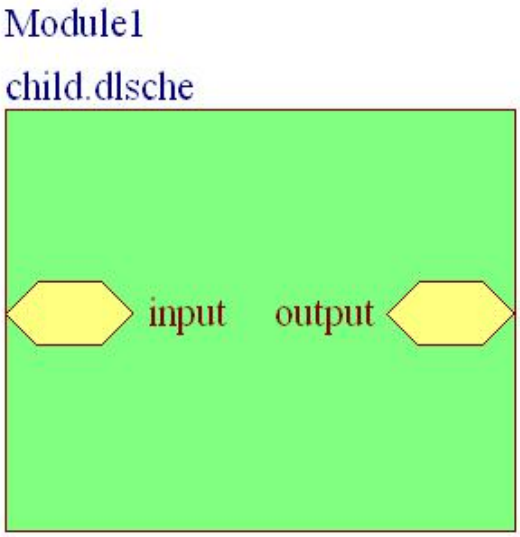


图 11-2：完成封装的模块

9. 完成封装后的模块即可添加元器件测试模块的功能了。由于模块内部只有一个非门，因此本模块的功能是完成高低电平的转换。测试电路如图 11-3 所示。

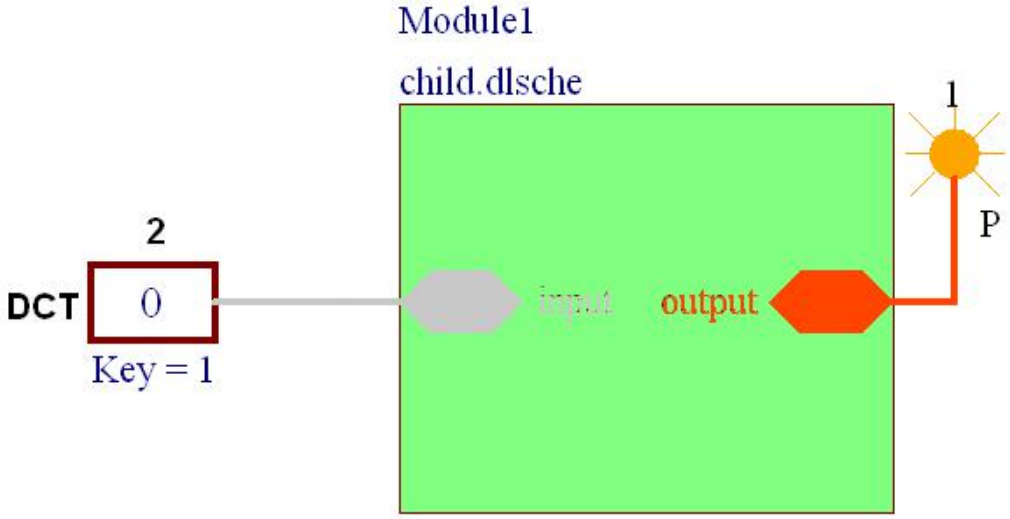


图 11-3：测试电路

10. 在仿真过程中如果想查看模块内部原理图 child.dlsche 中的状态，双击模块即可。若想从 child.dlsche 中返回到上层原理图 top.dlsche，在原理图 child.dlsche 的空白处双击即可。
11. 停止仿真，修改模块接口名称，将 input 修改为 input-err 后保存。

12. 重新启动仿真。此时仿真失败，在错误列表窗口中会显示出错误信息以及错误所在的原理图文件，双击错误信息，可直接定位到错误所在位置。这样就可以帮助用户找到那些由于模块接口和端口名称不对应而产生的错误。
13. 根据提示信息修改错误，将模块接口名称修改回 input，再次仿真后结果正确。

在后面的实验中为了使原理图更加简洁、清晰，读者可以尝试使用模块功能。以上内容仅是演示如何封装模块，若要添加更多的模块端口，应根据所需的数量灵活设置模块大小与模块接口的布局。

注意，内部网络名称可以与外部网络名称重名，但是并没有电气连接关系。内部网络只能通过端口与模块接口连接，从而与外部电路连接。

3.2 设计智力竞赛抢答器

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/blank.git

设计要求

（1）按下“启动”按钮后才允许抢答，30s 倒计时计时器开始工作，并在数码管上显示出倒计时时间（设计简单计时器的方法可以参考本书数字电路实验 12 中的相关内容）。

（2）抢答按钮分为 8 组，即序号 1、2、3、4、5、6、7、8，抢答者按本组序号的抢答按钮进行抢答，抢中的组号立即锁存，并显示到数码管上，同时封锁其它组号的抢答功能，30s 倒计时计时器停止计时。

（3）30 秒定时到，如果没有抢答者，本次抢答无效，系统自动复位。

（4）提供手动复位功能，复位后抢答号码、抢答计时等清零。

提交作业

在提交作业前，读者需要将所有原理图 SVG 图形文件的链接添加到 README.md 文件中，这样，当使用浏览器查看提交后的线上项目时，就可以从 README.md 文件中看到电路了。

实验 12 路口交通灯管理系统的设计

实验性质：综合设计
建议学时：2 学时
难易程度：较难

一、 实验目的

- 熟悉有限状态机的设计和调试。
- 熟练运用集成芯片设计电路。
- 进一步训练数字系统的设计和调试。

二、 预备知识

2.1 同步时序电路

组合逻辑电路中是没有环路和竞争的。在组合逻辑电路中，输出总是根据输入，在传播延迟内稳定为一个正确的值。但是，将组合电路的输出直接反馈到输入，形成环路，就成为了时序电路，而不再是组合电路。包含环路的时序电路存在不良的竞争和不稳定的行为。分析这样的电路十分耗时，而且很容易犯错。

为了避免这些问题，设计师在环路中插入寄存器来断开环路。这样，就将环路转变为组合逻辑电路和寄存器的集合。寄存器包含系统的状态，这些寄存器的内容仅仅在时钟上升沿或下降沿到来时才发生改变，所以说状态同步于时钟信号。如果时钟足够慢，使得在下一个时钟沿到达前输入到寄存器的信号都可以稳定下来，那么所有的竞争将被消除。根据反馈环路上总是使用寄存器的原则，可以得到同步时序电路的一个形式化定义。非同步的时序电路称为异步电路。

通过电路的输入、输出、功能规范和时序规范可以定义一个电路。一个时序电路有一组有限的离散状态{S0，S1，…，S_{k-1}}。同步时序电路有一个时钟输入，它的上升沿表示时序电路状态转变发生的时间。我们经常使用术语当前状态和下一个状态来区分目前系统状态和下一个时钟沿系统进入的状态。

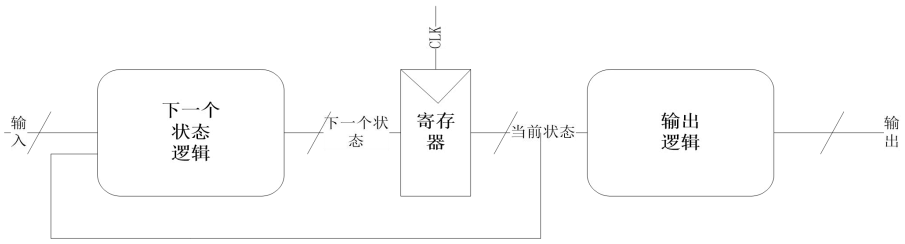
同步时序电路要满足的条件：

- 每一个电路元件都是寄存器或者组合电路。
- 至少有一个电路元件是寄存器。
- 所有寄存器都接收同一个时钟信号。
- 每个环路至少包含一个寄存器。

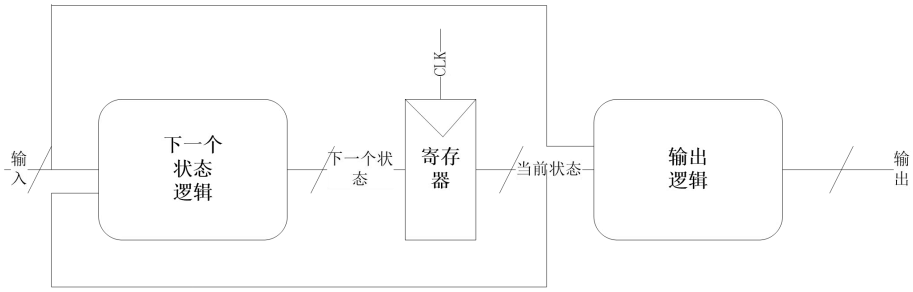
两种常见的同步时序电路是有限状态机和流水线。

2.2 有限状态机

同步时序电路可以用图 12-1 中的形式来描述。这些形式称为有限状态机（FiniteStateMachine，FSM）。这个名字源于具有 k 位寄存器的电路可以处于 2^k 种状态（每个寄存器值表示一种状态）中的某一种唯一状态。一个有限状态机有 M 个输入、N 个输出和 k 位状态。它还接收一个时钟信号和一个可选择的复位信号。有限状态机包含两个组合逻辑块，下一个状态逻辑和输出逻辑，以及一个存储这个状态的寄存器。在时钟沿到来时，有限状态机进入下一个状态，这个下一个状态是根据当前状态和输入计算出来的。根据有限状态机功能规范的描述，FSM 通常分为两类。在 Moore 型有限状态机中，输出仅仅取决于机器的当前状态。在 Mealy 型有限状态机中，输出取决于当前状态和当前输入。



a)Moore 型有限状态机



b）Mealy 型有限状态机

图 12-1：有限状态机

2.3 状态编码

a） 二进制编码

一个二进制数代表一种状态，因为 $\log_2 K$ 位可以表示 K 个不同的二进制数，所以有 K 个状态的系统只需要 $\log_2 K$ 位编码。

b） 独热编码

在独热编码中，状态的每一位表示一种状态。例如，一个有 3 个状态的有限状态机的独热编码为 001、010 和 100。状态的每一位存储在一个触发器中，所以独热编码比二进制编码需要更多的触发器。然而，使用独热编码，下一个状态和输出逻辑通常更简单，需要的门电路也更少。最佳编码方式取决于具体的有限状态机。

2.4 设计有限状态机的步骤

1. 确定输入和输出。
2. 画出状态转换图。
3. 写出状态转换表。
4. 选择状态编码。
5. 为下一个状态和输出逻辑写出布尔表达式。
6. 根据布尔表达式画出有限状态机的电路图。

三、 实验内容

在设计复杂的数字系统之前，读者应该掌握使用 Dream Logic 绘制模块化电路的方法，这部分内容请参考本书数字电路实验 11 中的相关内容。

3.1 设计简单的计时器

请读者使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/Digital-Circuit/Lab012.git

1. 打开项目下的原理图文件 74LS190.dlsche。该原理图文件中绘制了异步置数十进制加/减计数器

74LS190 的内部原理图。

2. 启动仿真，根据表 12-1 完成 74LS190 的功能验证。

表 12-1：74LS190 功能表

CLK	S	LD	U/D	工作状态
X	1	1	X	保持
X	X	0	X	预置数
↑	0	1	0	加法计数
↑	0	1	1	减法计数

3. 在项目下新建一个原理图文件 clock.dlsche。在其中使用 74SL190 和触发器等器件设计一个倒计时计时器。倒计时计数器从 35 开始减计数，当计数到 0 时，下个时钟沿又使计时器从 35 开始新一轮的倒计时。需要为计时器提供一个复位信号，任何时刻，复位信号变为 0 都会使计时器重新从 35 开始倒计时。仿真验证时，可以先使用手动的单周期时钟驱动计时器，仿真成功后，可以尝试使用自动时钟驱动计时器，设置自动时钟频率的方法可以参考下面的注意事项。

注意，需要灵活设置自动时钟的频率以及原理图的仿真步长与仿真时间。

- 选中用于驱动计时器的自动时钟器件，在属性窗口中将时钟频率设置为 1，再启动仿真后，可以观察到计时器上的时间与仿真时间保持一致。仿真时间显示在 Dream Logic 软件底部的状态栏中。
- 点击将要进行仿真的原理图的空白处，在属性窗口中会显示原理图的仿真参数。设置仿真步长为 100ms，可以使仿真进行的速度比较适中。设置仿真时间为 5000s，可以保证仿真在一段足够长的时间内进行，方便观察仿真结果。

3.2 设计路口交通灯管理系统

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/blank.git

1. 要求如下：
   1) 实现东西、南北两个方向上的交通灯控制。使用型号库“数字信号显示”中提供的交通灯显示灯光变化。
   2) 灯光变化的规律为：红灯亮→绿灯亮→黄灯亮→红灯亮……，如此依次循环。
   3) 使用两个数码管显示一个方向上的倒计时，红灯持续时间为 7 秒、绿灯持续时间为 5 秒、黄灯持续时间为 2 秒。
   4) 当任何一个方向出现警车、消防车、救护车等优先通行类车辆时，则东西、南北两个方向上的红灯全亮，其他灯灭，时钟停止计时，其他车辆禁止通行。当这种特殊状态结束后，恢复原来的状态，继续运行。
   5) 使用单周期时钟器件产生的负脉冲（低电平）表示有紧急事件发生，优先类车辆通行；负脉冲结束表示紧急事件结束。
   6) 使用有限状态机的方法设计交通灯控制器。
2. 交通灯设计提示。
   （1） 确定输入和输出。
   输入：

- 时钟 CLK，倒计时时钟源。
- 复位按钮 RESET，按下复位按钮后，南北方向为红灯（从 7 秒开始倒计时），东西方向为绿灯（从 5 秒开始倒计时）。
- 紧急事件按钮 Emergency。按下变为低电平，两个方向均为红灯。松开变为高电平，红绿灯恢复为初始状态继续工作。

输出：

- 南北方向的红、绿、黄灯控制信号 LRsn、LGsn、LYsn，高电平对应灯亮，以及倒计时 sn0、sn1、sn2，显示倒计时 0~7 只需 3 位二进制码。
- 东西方向的红、绿、黄灯控制信号 LRew、LGew、LYew，高电平对应灯亮，以及倒计时 ew0、ew1、ew2。

（2） 画出状态转换图，如图 12-2 所示，从图中可以看出，交通灯共有 14 个不同的状态。

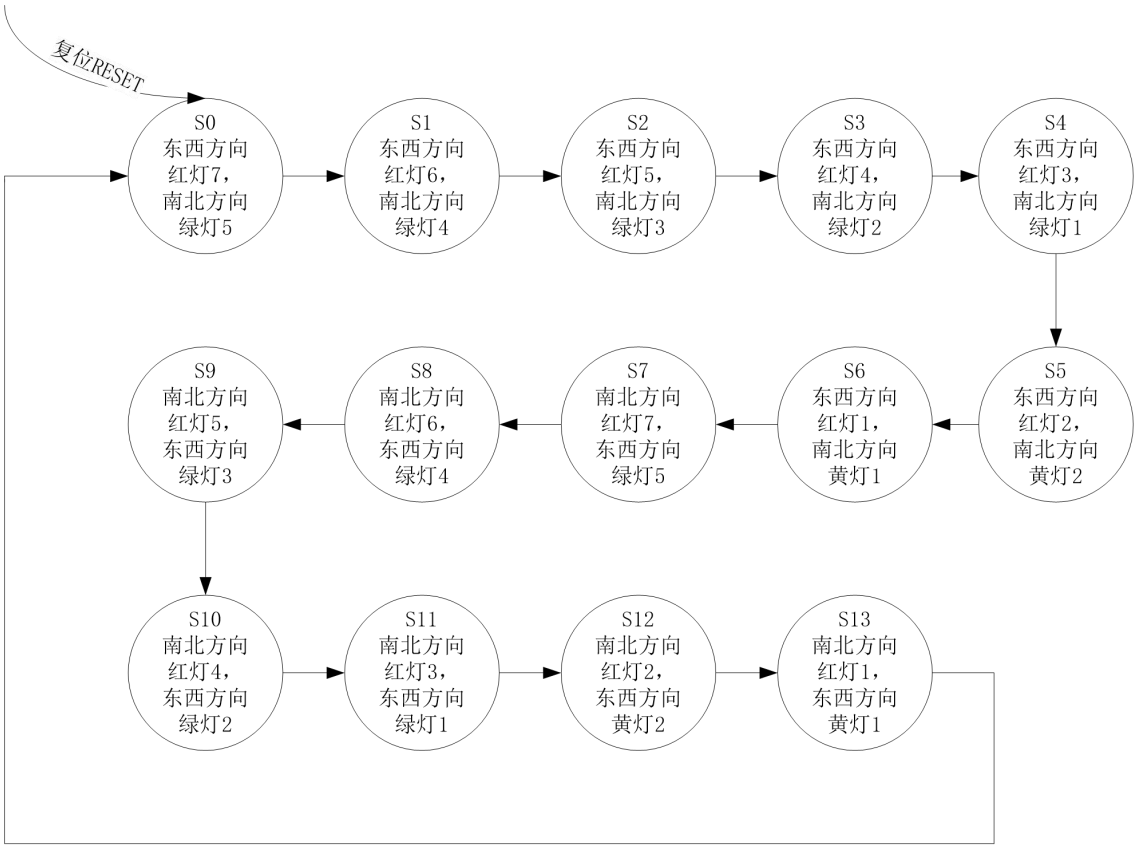


图 12-2：交通灯状态转换图

（3） 状态编码。

使用二进制编码，14 个状态需要至少 4 位二进制编码。使用 T_3、T_2、T_1、T_0四位编码状态。

（4） 写出状态转换表，如表 12-2 所示。

表 12-2：交通灯状态转换表

当前状态	当前状态编码				下一个状态	下一个状态编码			
	$T_3T_2T_1T_0$					$T_3'T_2'T_1'T_0'$			
S0	0	0	0	0	S1	0	0	0	1
S1	0	0	0	1	S2	0	0	1	0
S2	0	0	1	0	S3	0	0	1	1
S3	0	0	1	1	S4	0	1	0	0
S4	0	1	0	0	S5	0	1	0	1
S5	0	1	0	1	S6	0	1	1	0
S6	0	1	1	0	S7	0	1	1	1
S7	0	1	1	1	S8	1	0	0	0
S8	1	0	0	0	S9	1	0	0	1
S9	1	0	0	1	S10	1	0	1	0
S10	1	0	1	0	S11	1	0	1	1
S11	1	0	1	1	S12	1	1	0	0
S12	1	1	0	0	S13	1	1	0	1
S13	1	1	0	1	S0	0	0	0	0

（5） 为下一个状态逻辑写出布尔表达式

写下一个状态逻辑的布尔表达式时，不需要考虑时钟，只需要考虑当前状态和输入对下一个状态的影响。在交通灯控制器中，对下一个状态有影响的外部输入是复位信号 RESET 和紧急事件 Emergency，无论当前处于何种状态，一旦有复位信号 RESET，下一个状态一定是 S0（初始状态）。然后分析紧急事件信号的影响，紧急事件结束后，仍然恢复为原来的状态，也就说明，紧急事件不会影响下一个状态，它是通过使交通灯控制器的当前状态保持不变，停止状态转换，而仅仅使交通灯暂时维持红灯而已。所以，下一个状态的布尔表达式为以下样式：

T_3'=RESET（T_3,T_2,T_1,T_0的逻辑表达式）

T_2'=RESET（T_3,T_2,T_1,T_0的逻辑表达式）

T_1'=RESET（T_3,T_2,T_1,T_0的逻辑表达式）

T_0'=RESET（T_3,T_2,T_1,T_0的逻辑表达式）

请读者根据状态转化表完成下一个状态的布尔表达式。

提示：从交通灯控制器的状态转换图可以看出，交通灯的状态从 S0 开始，依次转换为 S1，S2，S3，…，S13，然后又转变为 S0。状态每经过一秒转换一次，正好是一个带异步置数端的 14 进制计数器。计数器的 4 位输出就是当前状态码 T_3,T_2,T_1,T_0，根据当前状态码，就可以得到当前输出。

（6） 画出电路图

交通灯的逻辑设计过程已经完成，请读者自行完成交通灯电路，通过仿真验证。

提交作业

在提交作业前，读者需要将所有原理图 SVG 图形文件的链接添加到 README.md 文件中，这样，当使用浏览器查看提交后的线上项目时，就可以从 README.md 文件中看到电路了。

实验 13 电梯控制器的设计

实验性质：综合设计
建议学时：2 学时
难易程度：较难

一、 实验目的

- 进一步熟悉有限状态机的设计方法。
- 进一步熟悉一个数字系统的设计和仿真调试方法。

二、 实验内容

在设计复杂的数字系统之前，读者应该掌握使用 Dream Logic 绘制模块化电路的方法，这部分内容请参考本书数字电路实验 11 中的相关内容。此外本书数字电路实验 12 中介绍了同步时序电路、有限状态机的概念，还提供了计时器的设计方法。

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为
https://www.codecode.net/engintime/Dream-Logic/Project-Template/blank.git

设计要求

设计一个三层楼房全自动电梯控制电路，该电路具有如下功能：

1. 每层电梯入口设有上下请求开关各一个（最底层只有向上请求开关，最高层只有向下请求开关），电梯内设有乘客到达层数的停站要求开关。
2. 电梯所处层数指示装置和电梯上下行状态指示装置。
3. 电梯每 3 秒升（降）一层，到达某一层时，数码管显示该层层数，并一直保存到电梯到达新一层为止。
4. 电梯到达有停站要求的层后，经过 1 秒，电梯门自动打开（开门指示灯亮），经过 5 秒后，电梯门自动关闭（开门指示灯灭）。
5. 能保证响应电梯内外的所有请求信号，并按照电梯运行规则依次响应，每个请求信号保留至执行后撤除。
6. 电梯运行规则：电梯处于上升模式时，只响应比所在位置高的楼层的上楼请求信息，由上而下逐个执行直到最后一个请求执行完毕。如更高层有下楼请求，则直接升到有下楼请求的最高层接客，然后转入下降模式。电梯处于下降模式时与之相反，仅响应比电梯所在位置低的下楼请求，由上到下逐个解决，直到最后一个请求被处理完毕。如果最底层有上楼请求时，则降至该层楼，并转入上升模式。电梯执行完所有的请求后，应停在最后所在位置不变，等待新的请求。
7. 开机时，电梯应停在一楼，而各种上下请求均被清除。

系统设计方案

1. 三层电梯控制面板如图 13-1 所示。在设计中，可以灵活使用单周期时钟、数码管、指示灯等器件代

替控制面板中的按钮。

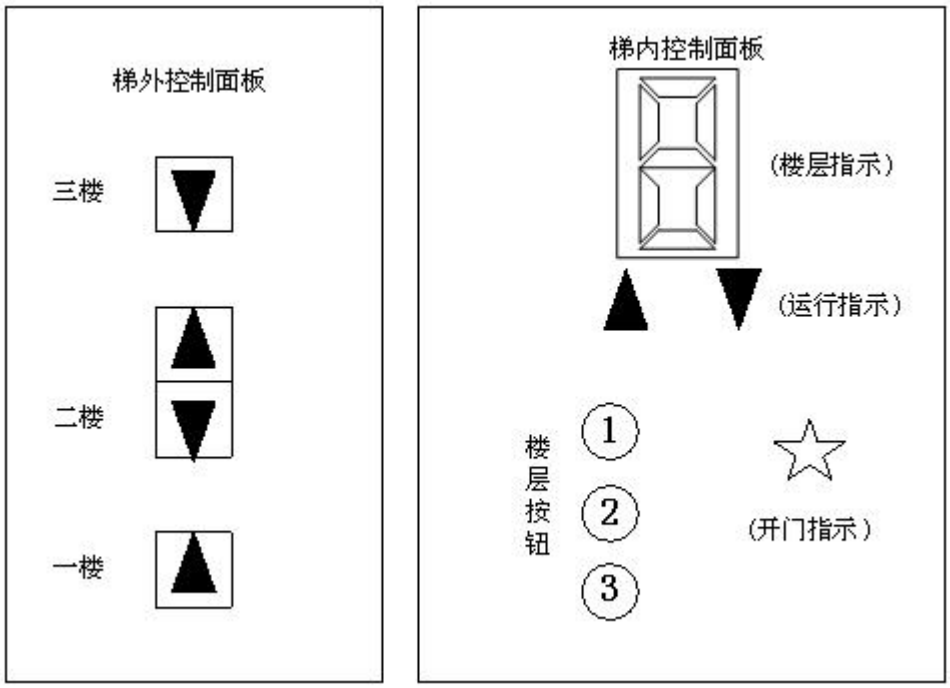


图 13-1：电梯控制面板

2. 设计框图如图 13-2 所示。

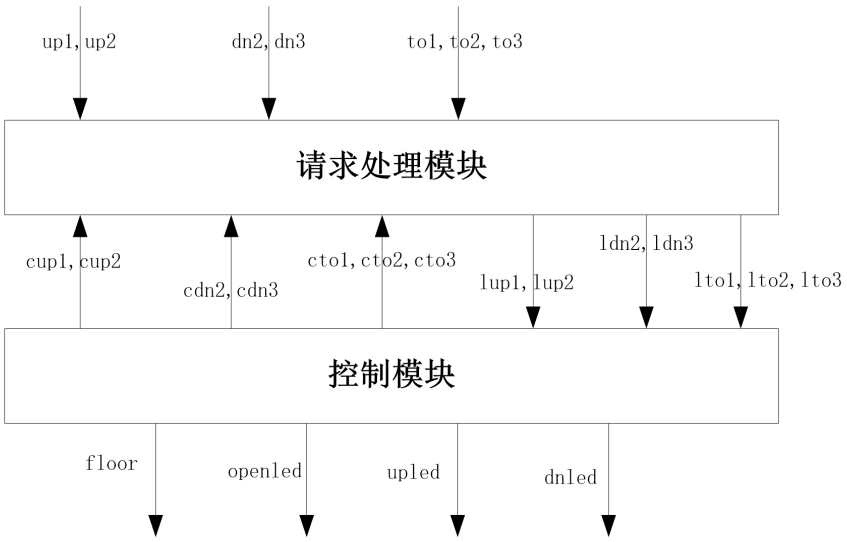


图 13-2：设计框图

3. 主要模块设计

根据上面的设计方案，设计中应具有一些信号和模块。

1) 信号说明

up1~up2：分别为电梯外 1、2 楼用户上楼请求信号。

dn2~dn3：分别为电梯外 2、3 楼用户下楼请求信号。

to1~to3：分别为电梯内用户到 1、2、3 楼的请求信号。

lup1~lup2：分别为电梯外 1、2 楼用户上楼请求指示。

ldn2~ldn3：分别为电梯外 2、3 楼用户下楼请求指示。

lto1~lto3：分别为电梯内用户到 1、2、3 楼的请求指示。

cup1~cup2：分别用于清除 1、2 楼用户的上楼请求。

cdn2~cdn3：分别用于清除 2、3 楼用户的下楼请求。

cto1~cto3：分别用于清除电梯内用户到 1、2、3 楼的请求。

floor：楼层显示。

Openled：开门指示。

Upled：上升指示。

Dnled：下降指示。

注意：当请求信号被满足时，请求指示清除，否则，请求指示一直有效。

2) 模块说明

请求处理模块：存储用户的请求以及当请求被处理后请求指示的清除。

控制模块：电梯到达有停站要求的楼层后，经过 1 秒，电梯门自动打开（开门指示灯亮），经过 5 秒后，电梯门自动关闭（开门指示灯灭），电梯继续运行；能保证响应电梯内外的所有请求信号，并按照电梯运行规则依次响应，每个请求信号保留至执行后撤除；开机时，电梯应停在一楼，而各种上下请求均被清除。

4. 电梯有限状态机设计参考

以 1 秒为电梯状态转换的时间单位，也就是每经过 1 秒，电梯由当前状态进入下一个状态。以此为依据，将电梯状态划分如下：

1) 电梯起始状态 stop：关停，电梯门关闭，电梯停止升降。
2) 电梯开门等待 1 秒 open1。
3) 电梯开门等待 2 秒 open2。
4) 电梯开门等待 3 秒 open3。
5) 电梯开门等待 4 秒 open4。
6) 电梯开门等待 5 秒 open5。
7) 电梯上升 1 秒 up1：电梯每上升一层需要 3 秒，将上升过程中的每一秒作为一个状态。
8) 电梯上升 2 秒 up2。
9) 电梯上升 3 秒 up3。
10) 电梯下降 1 秒 dn1，与电梯上升同理，划分为 3 个不同状态。
11) 电梯下降 2 秒 dn2。
12) 电梯下降 3 秒 dn3。

注意：以上列出是电梯的主要状态，每一个主要状态依据当前所在楼层可以细分为多个状态，比如 stop 状态，可以分为电梯停在 1 楼 stop1，电梯停在 2 楼 stop2，电梯停在 3 楼 stop3 共 3 个不同的状态。

5. 参考状态转换图 13-3。

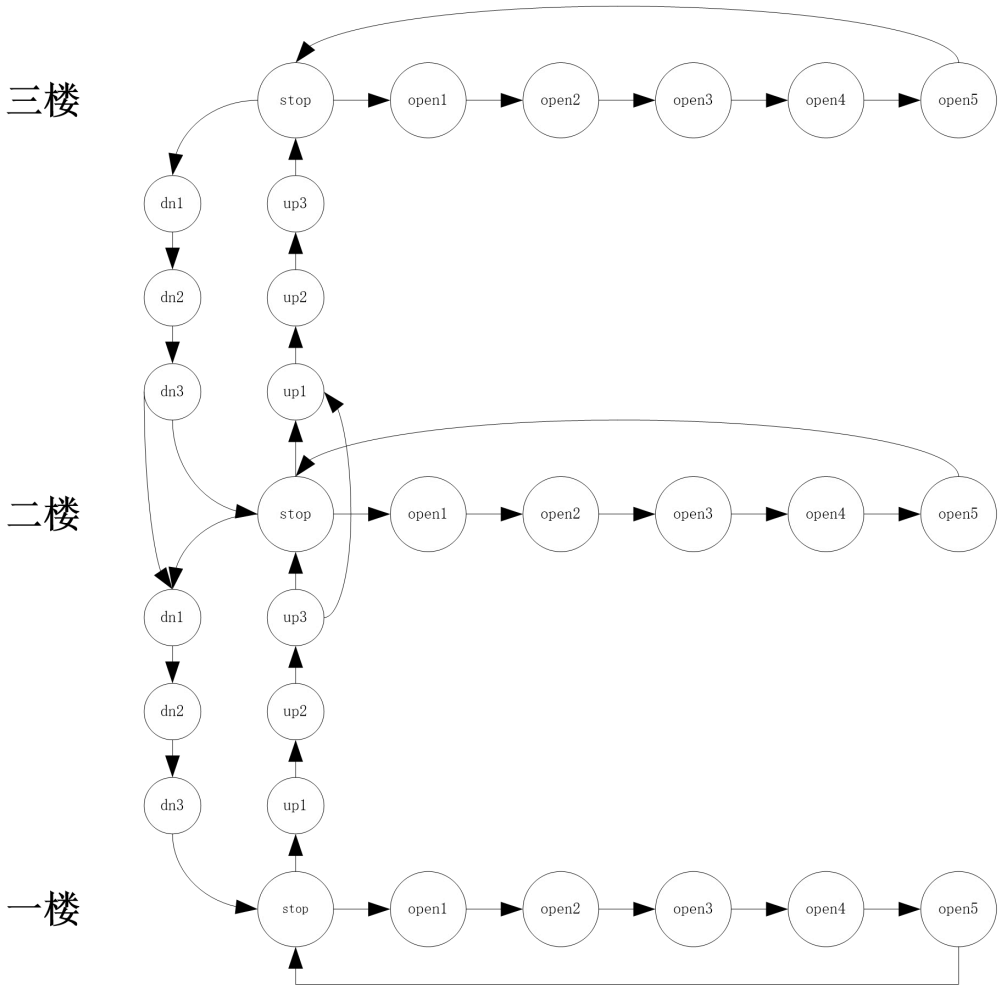


图 13-3：电梯状态转换图

注意：图中的每一个圆圈表示一个独立的状态，例如一楼的 stop 与二楼的 stop 表示两个不同的状态。此外，电梯控制器是一个 mealy 型有限状态机，下一个状态由当前状态以及当前的输入决定，请读者结合电梯运行规则，自行标出状态转换条件，也就是各个楼层的电梯请求。

6. 状态转换表（使用 5 位二进制编码所有状态）如表 13-1 所示。说明如下

- Lup1、lup2、ldn2、ldn3、lto1、lto2、lto3：1 或/1 均表示有请求，0 表示无请求，空表示任意值。
- 当前状态与所有有效输入均为“与”的关系，当一行中有多个“/1”时，表示只要其中一个请求有效即可，例如第一行的状态转换逻辑可以描述为：s1open1 = stop1(lup1 | lto1)，含义是，当电梯处于停在 1 楼的状态时，若 1 楼有上楼请求或者有到 1 楼的请求，则电梯开门并进入开门第一秒的状态。而

S1up3		0		/1		0	/1		0	1	1	1	1	S2up1

表示由当前状态 s1up3 进入下一个状态 s2up1。s2up1=s1up3(~lup2)(~lto2)(ldn3 | lto3)，含义是，当电梯处于 1 楼升 2 楼的第 3 秒状态时，若 2 楼没有上楼请求、没有到 2 楼的请求、3 楼有下楼请求或有到 3 楼的请求，则电梯到达 2 楼后不停止，继续上升，进入 2 楼上 3 楼的第一秒状态。

表 13-1：电梯状态转换表

当前状态	Lup1	Lup2	Ldn2	Ldn3	Lto1	Lto2	Lto3	Up/dn	S4	S3	S2	S1	S0	下一状态
Stop1	/1				/1				0	0	0	0	1	S10pen1
S10pen1									0	0	0	1	0	S10pen2
S10pen2									0	0	0	1	1	S10pen3
S1open3									0	0	1	0	0	S1open4
S1open4									0	0	1	0	1	S1open5
S1open5									0	0	0	0	0	Stop1
Stop1	0	/1	/1	/1	0	/1	/1		0	0	1	1	0	S1up1
S1up1									0	0	1	1	1	S1up2
S1up2									0	1	0	0	0	S1up3
S1up3		/1				/1			0	1	0	0	1	Stop2
S1up3		0		/1		0	/1		0	1	1	1	1	S2up1
Stop2						1			0	1	0	1	0	S2open1
Stop2		1						1	0	1	0	1	0	S2open1
Stop2			1					0	0	1	0	1	0	S2open1
S2open1									0	1	0	1	1	S2open2
S2open2									0	1	1	0	0	S2open3
S2open3									0	1	1	0	1	S2open4
S2open4									0	1	1	1	0	S2open5
S2open5									0	1	0	0	1	Stop2
Stop2		0		/1		0	/1	1	0	1	1	1	1	S2up1
S2up1									1	0	0	0	0	S2up2
S2up2									1	0	0	0	1	S2up3
S2up3									1	0	0	1	0	Stop3
Stop3				/1			/1		1	0	0	1	1	S3open1
S3open1									1	0	1	0	0	S3open2
S3open2									1	0	1	0	1	S3open3
S3open3									1	0	1	1	0	S3open4
S3open4									1	0	1	1	1	S3open5
S3open5									1	0	0	1	0	Stop3
Stop3	/1	/1	/1	0	/1	/1	0		1	1	0	0	0	S3dn1
S3dn1									1	1	0	0	1	S3dn2
S3dn2									1	1	0	1	0	S3dn3
S3dn3			/1			/1			0	1	0	0	1	Stop2
S3dn3	/1		0		/1	0			1	1	0	1	1	S2dn1
Stop2	/1		0		/1	0		0	1	1	0	1	1	S2dn1
Stop2	/1		0	0	/1	0	0	1	1	1	0	1	1	S2dn1
S2dn1									1	1	1	0	0	S2dn2
S2dn2									1	1	1	0	1	S2dn3
S2dn3									0	0	0	0	0	Stop1

提交作业

在提交作业前，读者需要将所有原理图 SVG 图形文件的链接添加到 README.md 文件中，这样，当使用浏览器查看提交后的线上项目时，就可以从 README.md 文件中看到电路了。

实验 14 数字钟的设计

实验性质：综合设计
建议学时：2 学时
难易程度：一般

一、 实验目的

- 学习数字电路系统的设计方法、及数字钟功能扩展电路的设计。
- 从电路角度出发，建立系统的概念，培养综合运用数字电路的能力。

二、 预备知识

2.1 数字钟电路的组成

数字钟的原理如图 14-1 所示。

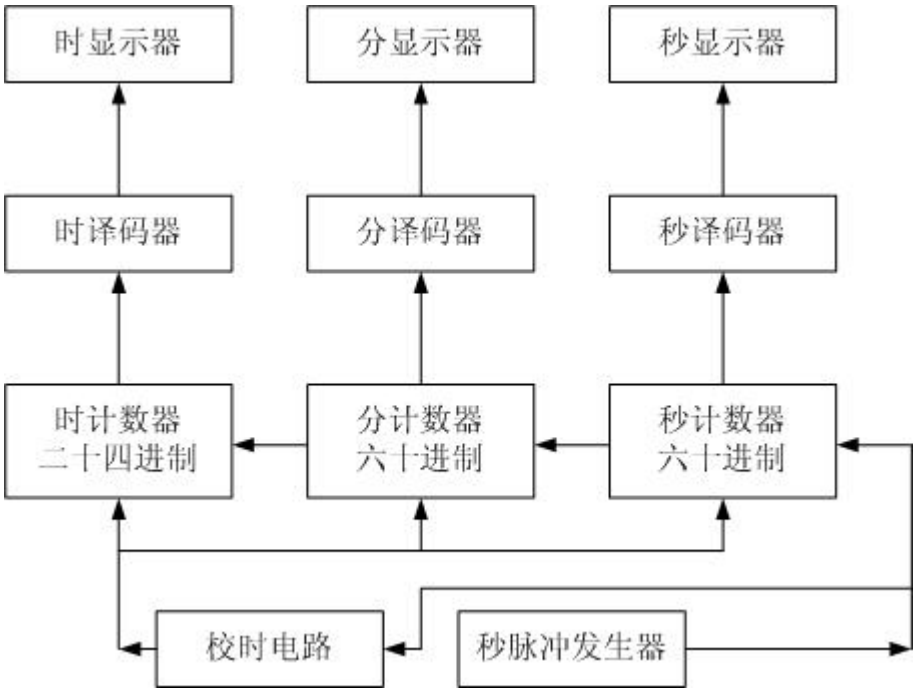


图 14-1：数字钟原理框图

在图 14-1 中，秒脉冲发生器是数字钟的核心，其工作频率应该为 1Hz，它产生的秒信号输入秒计数器进行计数，当达到 59 时产生进位信号给分计数器，秒计数器复位到 0 的同时，分计数器开始计时。当分计数器达到 59 时产生进位信号给时计数器，分计数器复位到 0 的同时，时计数器开始计数。

2.2 数字钟主体电路的设计

（1）秒脉冲发生器。

使用型号库“数字信号源”中的自动时钟作为秒脉冲发生器，通过将时钟的频率设置为 1Hz，使其每隔一秒输出一个时钟上升沿，从而触发秒计数器计数。

（2）计数器电路。

可用于计数的芯片很多，比如 74LS161，74LS90 等。

（3）译码显示电路。

可以使用型号库“数字信号显示”中的 16 进制 7 段数码管或者七段数码显示器来显示时、分、秒的数值。

（4）校时电路。

手动校时，可以手动设置时、分、秒的值。

三、 实验内容

在设计复杂的数字系统之前，读者应该掌握使用 Dream Logic 绘制模块化电路的方法，这部分内容请参考本书数字电路实验 11 中的相关内容。此外本书数字电路实验 12 中介绍了同步时序电路、有限状态机的概念，还提供了计时器的设计方法。

请读者按照下面方法之一在本地创建一个项目，用于完成本次任务：

方法一：从 CodeCode.net 平台领取任务

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 Dream Logic 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

方法二：不从 CodeCode.net 平台领取任务

如果读者不使用 CodeCode.net 平台，就需要使用 Dream Logic 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/Dream-Logic/Project-Template/blank.git

设计要求

根据数字钟原理框图 14-1，设计一个数字钟。要求如下：

1） 小时按 24 进制计数，分和秒按 60 进制计数。

2） 可以通过校时电路手动设置时、分、秒。

改进一

改进数字钟电路，实现倒计时功能。要求通过按键设置数字钟计时模式：

a） 当数字钟处于正常计时（加计数）模式时，按下此按键，数字钟转为倒计时（减计数）模式。

b） 当数字钟处于倒计时（减计数）模式时，按下此按键，数字钟转为正常计时（加计数）模式。

改进二

实现秒表功能。要求通过按键控制秒表复位（清零），并提供启动/暂停按键功能：

a） 复位按键按下后，设置数字钟为 00:00:00（0 时 0 分 0 秒）。

b） 当秒表正在计时，按下启动/暂停按键，秒表停止计时，再次按下此按键，秒表继续计时。

改进三

实现定时闹钟功能。要求可以设置闹钟响铃时刻，当数字钟的计时达到所设定的时刻后，点亮指示灯。

附录 数字电路实验芯片功能说明

在 DreamLogic 中，每一个型号的管脚样式都代表一定的含义，熟悉这些代表性的符号，便于直观识别器件的控制信号输入管脚的功能。下面以同步置数/异步清零十六进制加法计数器 74LS161 为例加以说明。

十六进制加法计数器 74LS161

74LS161 如下图 1 所示：

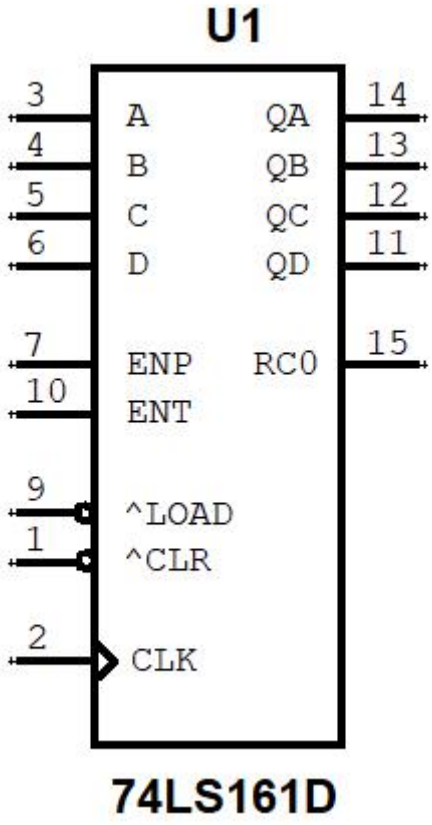


图 1:74LS161 器件型号

如图 1 所示 74LS161 器件图，器件的管脚编号与实际芯片的管脚编号相同，但是管脚的位置不同。例如，CLK 管脚编号为 2，实际芯片的时钟输入管脚 CLK 的编号也为 2，只是它们在芯片中的位置不同。

在实际芯片中，管脚是从芯片开口处逆时针进行编号的。

在图 1 中，74LS161 的控制管脚有 ENP、ENT、LOAD、CLR 以及 CLK，其中 ENP 和 ENT 管脚表示输入信号为高电平时有效；LOAD 和 CLR 管脚末端有一个圆圈，表示低电平有效；CLK 管脚末端有一个三角形，表示时钟上升沿有效，若在 CLK 管脚末端增加一个圆圈，则表示下降沿有效。管脚末端的圆圈通常表示输入信号经过一个非门后加载到输入端。

表 1:74LS161 管脚说明

管脚编号	管脚名称	有效电平	功能描述
1	CLR	低电平	异步清零端，信号优先级最高，一旦 CLR=0，立即将输出 QA~QD 清零。
9	LOAD	低电平	异步置数端，优先级次于 CLR，当清零信号 CLR 无效时（高电平），若 LOAD=0，则立即将输入数据 A~D 传送到输出端 QA~QD。

7	ENP	高电平	ENP 和 ENT 是计数使能端，当它们均为有效电平时，计数器才能做加计数。若其中一个为无效电平（低电平），那么计数器输出 QA~QD 保持不变。
10	ENT		
2	CLK	↑	计数器只对输入时钟的上升沿进行计数，在计数时，CLK 端每输入一个上升沿，计数器输出 QA~QD 的数字值加 1。
2~6	A~D	-	计数器置数数据输入端，在 LOAD 有效时，A~D 并行传送到输出端 QA~QD。
11~14	QA~QD	-	计数值输出端，在计数状态下，随着 CLK 上升沿的输入，输出计数值逐次加 1
15	RCO	高电平有效	RCO 是计数器发生进位输出端，也就是当计数值 QA~QD=1111=15 时，RCO=1，表示计数已达最大值，下个时钟上升沿将发生进位

优先编码器 74LS148

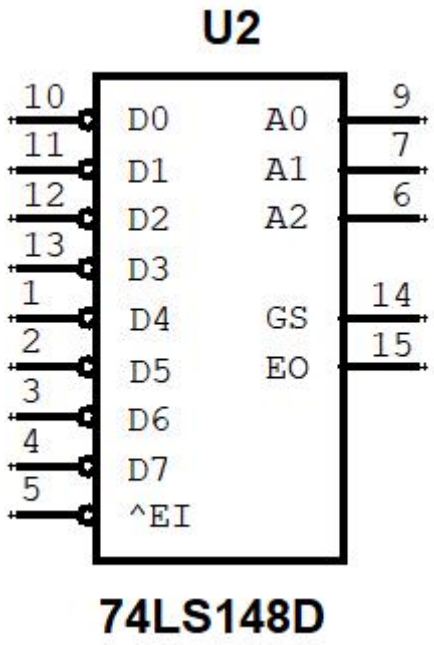


图 2:优先编码器 74LS148

表 2：74LS148 优先编码器管脚说明

管脚名称	有效电平	功能描述
EI	0	编码输入使能端，只有当 EI=0 使能时，才能对输入的 D0~D7 最高优先级位进行编码。否则，编码输出 A0~A2=111 且 GS=1，表示编码无效
D0~D7	0	编码输入端，D0 优先级最低，D7 优先级最高
A0~A2	-	编码输出端，A0 是最低位，A2 是最高位，A2A1A0 表示的十进制数对应于输入编码 D0~D7 中最高有效编码位的标号。例如，A2A1A0=4，则表示 D7~D0=1110 xxxx，也就是最高有

		效位 D4=0
GS	-	片选优先编码输出端，当 GS=0 时，表示有效编码，且 D0~D7 中至少有一位有效（为低电平）
EO	-	使能输出端，当 EO=1 时，唯一表示编码输入 EI=0 使能，且输入编码 D0~D7 均为高电平，即没有有效编码位的情况

3-8 译码器 74LS138

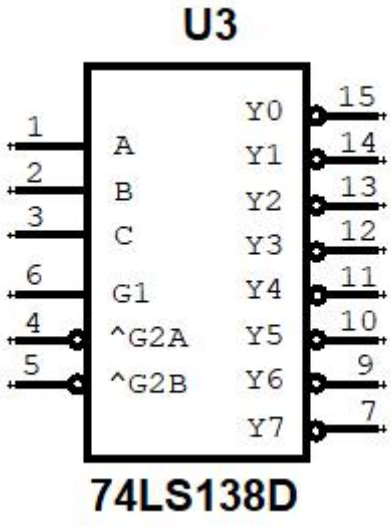


图 3: 3-8 译码器

表 3：74LS138 译码器管脚说明

管脚名称	有效电平	功能描述
A~C	-	地址输入端，A 为最低位，C 为最高位。当译码器工作时，可以看成一个分配器，将 G1=1 分配给由 CBA 地址唯一确定的输出端。例如，当 CBA=101 时，Y5 端接收 G1，经过一个非门输出低电平，而其余输出均为高电平。
G1	高电平	片选输入端，当 G1=1，G2A+G2B=0 时，译码器处于工作状态。否则，译码器被禁止，所有的输出端被封锁在高电平。利用片选的作用可以将多片连接起来以扩展译码器的功能。
G2A	低电平	
G2B	低电平	
Y0~Y7	低电平	译码器工作时，有且只有一位输出低电平。哪一位为低电平是由当前输入地址 CBA 决定的。

7 段数码显示器 74LS48

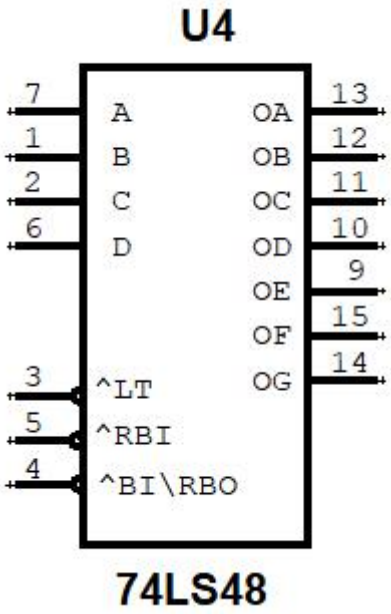


图 4：共阴极七段显示译码器

表 4：共阴极七段显示译码器 74LS48 管脚说明

管脚名称	有效电平	功能描述
LT	低电平	LT 是灯测试输入信号。 当 LT=0 的信号输入时，被驱动数码管的七段同时点亮，以检查该数码管各段能否正常发光。工作时应使 LT 为高电平。 译码时，LT 输入高电平。
RBI	低电平	RBI 是灭零输入。 设置灭零输入的目的是为了能把不希望显示的零熄灭。当被驱动的数码管显示的不是 0 时，灭零输入 RBI 不起作用。 译码时，RBI 输入高电平。
BI/RBO	低电平	灭灯输入/灭零输出信号。 当 BI/RBO 灭灯输入低电平时，被驱动数码管的七段同时熄灭。 译码时，BI/RBO 输入高电平。
A~D	-	数字码输入端，A 为最低位，D 为最高位。 译码时，被驱动的七段数码管显示 DCBA 表示的十六进制数值。
OA~OG	高电平	译码时，OA~OG 中的所有高电平驱动七段数码管显示输入数字。 若是共阳极七段显示译码器 74LS47，则是使用 OA~OG 中的所有低电平驱动七段数码管显示数字。这就是共阴极和共阳极的区别。

4 位超前进位加法器 74LS283

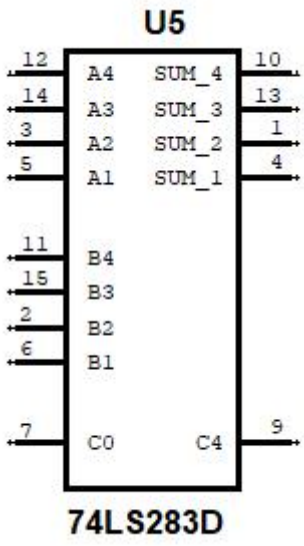


图 5:四位超前进位加法器 74LS283

表 5：超前进位加法器 74LS283 的管脚描述

管脚名称	有效电平	功能描述
A1~A4	-	四位二进制加数，A1 为最低位，A4 为最高位。
B1~B4	-	四位二进制加数，B1 为最低位，B4 为最高位。
SUM_1~SUM_4	-	加运算结果。 Sum=A+B+CO 的低四位
CO	高电平	来自低位的进位
C4	高电平	进位输出。 当加运算结果超出 SUM_4~SUM_1 所能表示的最大数 15 时，输出进位 C4=1。

4 位算术逻辑运算器 74LS181

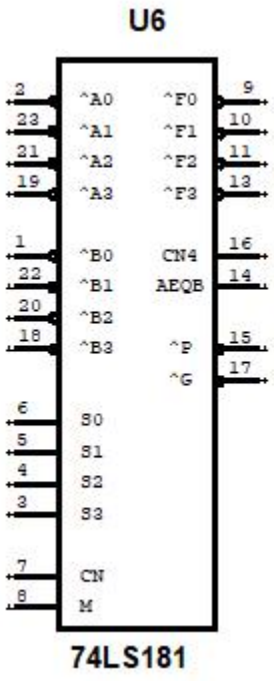


图 6：算术逻辑运算器 74LS181

表 6：算术运算器 74LS181 的管脚说明

管脚名称	功能描述
A0~A3	四位二进制运算操作数，A3 为最高位
B0~B3	四位二进制运算操作数，B3 为最高位
F0~F3	四位二进制运算结果，F3 为最高位
S0~S3	运算方式，4 位二进制可编码 2^4=16 种不同的运算方式
M	运算类型。 当 M=1 时，A 与 B 进行逻辑运算。逻辑运算包含与、或、非等。 当 M=0 时，A 与 B 进行算术运算。算术运算包含加、减以及逻辑与算术的混合运算
CN	CN 表示最低位进位输入。 CN=0，属于有进位型运算，共十六种； CN=1，属于无进位型运算，也包含十六种。
CN4	最高位进位输出，低电平有效。
AEQB	当 F3~F0=1111 时，AEQB=1，否则，AEQB=0
G	进位发生输出。
P	进位传送输出。

触发器的异步置数 PR 和异步清零 CLR 优先级相同，均为最高优先级。在触发器的其它信号起作用时，需要将 PR 和 CLR 置为 0，使其无效。

D 触发器

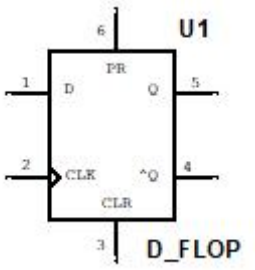


图 7：D 触发器

表 7：D 触发器管脚描述

管脚名称	有效电平	功能描述
D	-	1 位输入数据
PR	高电平	异步置数端。 当 PR=1，CLR=0 时，将输出 Q 置位为 1
CLR	高电平	异步清零端。 当 CLR=1，PR=0 时，将输出 Q 清零
CLK	↑	时钟上升沿有效。 当 PR=0 且 CLR=0 时，时钟上升沿到来时，输入数据 D 传送到输出端 Q，Q 取反传送到输出端^Q

JK 触发器

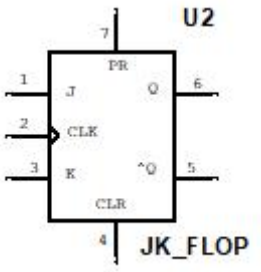


图 8：JK 触发器

表 8：JK 触发器管脚描述

管脚名称	有效电平	功能描述
PR	高电平	异步置数端。 当 PR=1，CLR=0 时，将输出 Q 置位为 1
CLR	高电平	异步清零端。 当 CLR=1，PR=0 时，将输出 Q 清零
J	高电平	同步置 1 端。 当 J=1，K=0，时钟上升沿到来时，输出端 Q 置 1

K	高电平	同步置 0 端。 当 k=1，J=0，时钟上升沿到来时，输出端 Q 置 0
CLK	↑	时钟上升沿有效。 当 J=K=0 时，Q 保持不变； 当 J=K=1，时钟上升沿到来时，输出端 Q 翻转，即 Q=^Q。利用这一特性，将 JK 连起来就构成了 T 触发器。

SR 触发器

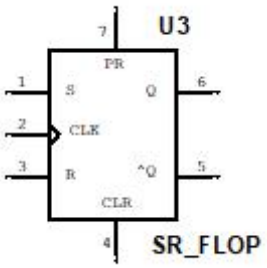


图 9：SR 触发器

表 9：SR 触发器管脚描述

管脚名称	有效电平	功能描述
PR	高电平	异步置数端。 当 PR=1，CLR=0 时，将输出 Q 置位为 1
CLR	高电平	异步清零端。 当 CLR=1，PR=0 时，将输出 Q 清零
S	高电平	同步置位端。 当 S=1，R=0，时钟上升沿到来时，输出端 Q 被置位为 1
R	高电平	同步清零端。 当 R=1，S=0，时钟上升沿到来时，输出端 Q 被清零
CLK	↑	时钟上升沿触发同步置位或清零

T 触发器

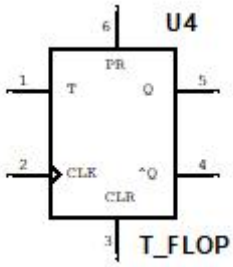


图 10：T 触发器

表 10：T 触发器管脚描述

管脚名称	有效电平	功能描述
PR	高电平	异步置数端。 当 PR=1，CLR=0 时，将输出 Q 置位为 1
CLR	高电平	异步清零端。 当 CLR=1，PR=0 时，将输出 Q 清零
T	高电平	当 T=1，时钟上升沿到来时，输出端 Q 发生翻转，即 Q=^Q。 当 T=0 时，输出 Q 保持不变
CLK	↑	时钟上升沿有效。

四位双向移位寄存器 74LS194

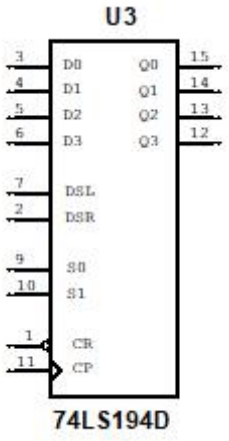


图 11：四位双向移位寄存器

表 11：四位双向移位寄存器管脚说明

管脚名称	有效电平	功能描述
D0~D3	-	移位数据输入端
Q0~Q3	-	移位数据输出端
DSL		左移串行数据输入端
DSR		右移串行数据输入端
S0~S1		移位方式控制端。 S1S0=00，输出保持； S1S0=01，右移； S1S0=10，左移； S1S0=11，并行传送，输出 Q0~Q3=D0~D3
CR	低电平	异步清零端，优先级最高，与时钟无关。 当 CR=0 时，输出 Q0~Q3 均为低电平
CP	↑	时钟输入端，上升沿有效。 触发移位或并行输出。

8 选 1 数据选择器 74LS151D

74LS151D 为互补输出的 8 选 1 数据选择器，引脚排列如图 12 所示，功能见表 12。选择控制端（地址端）为 C～A，按二进制译码，从 8 个输入数据 D0～D7 中，选择一个需要的数据送到输出端 Y，G 为使能端，低电平有效。

（1）使能端 G＝1 时，不论 C～A 状态如何，均无输出（Y＝0，W＝1），多路开关被禁止。

（2）使能端 G＝0 时，多路开关正常工作，根据地址码 C、B、A 的状态选择 D0～D7 中某一个通道的数据输送到输出端 Y。如：CBA＝000，则选择 D0 数据到输出端，即 Y＝D0。CBA＝001，则选择 D1 数据到输出端，即 Y＝D1，依此类推。

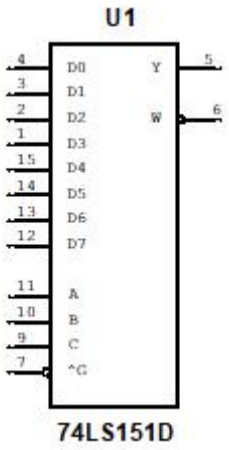


图 12:8 选 1 数据选择器

表 12：8 选 1 数据选择器管脚说明

管脚名称	功能描述
D0~D7	8 位数据输入端
A~C	数据选择端，CBA 表示的二进制数决定将哪一位输入数据作为输出。例如：CBA=101，那么选择 D5 输出到 Y 端
G	当 G=1 时，无论输入数据以及数据选择信号如何变化，输出端 Y 始终为低电平，W 始终为高电平。当 G=0 时，多路开关正常工作，根据地址码 C、B、A 的状态选择 D0～D7 中某一个通道的数据输送到输出端 Y。输出端 W 的值为 Y 取反。
W 和 Y	互补的数据输出端

二-五进制计数器 74LS90

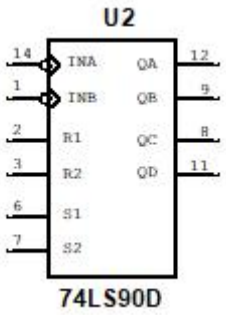


图 13：二-五进制计数器

表 13：二-五进制计数器功能表

复位输入				输出			
R1	R2	S1	S2	Q3	Q2	Q1	Q0
1	1	0	x	0	0	0	0
1	1	x	0	0	0	0	0
x	x	1	1	1	0	0	1
x	0	x	0	计数			
0	x	0	x	计数			
0	x	x	0	计数			
x	0	0	x	计数			

74LS90 是二-五十进制异步计数器，该芯片包含 4 个主从触发器和附加门，其中一级 Q0 组成二分频计数器，计数器输入端是 INA；另外三级（Q3、Q2、Q1）组成五分频计数器，计数输入端是 INB。四级的共同清零输入是 R1、R2，而 S1、S2 则为预置 9 输入端。利用该异步计数器，根据“计数到 N 时置 0”的原则（也叫反馈复位法），可构成 N 进制计数器。