第3章 组合逻辑电路

前面学习了门电路的相关知识，也学习了分析和设计数字电路的工具——逻辑代数，就如同一个建筑工程，有了基本原材料和工具，接下来就利用工具和材料来搭建具有一定逻辑功能的电路。组合逻辑电路是数字电路一个重要分支，也是数字集成电路中的一个重要组成部分。本章首先介绍组合逻辑电路的一般分析方法和设计方法；然后介绍编码器、译码器、数据选择器、数值比较器、加法器等常用组合逻辑集成器件，重点分析这些器件的逻辑功能、工作原理及使用方法；最后介绍组合逻辑电路的 Verilog HDL 描述。

为了能够将理论与实践联系起来，本章以工程实例为依托，边理论边实践。

3.1 工程任务：子弹分装系统设计

3.1.1 系统介绍

子弹分装系统框图如图 3.1.1 所示。按键输入每把枪安装子弹的数目并显示出来，系统自动按照输入的数目将每把枪需要的子弹分装到瓶子中，同时将分装子弹的总数存储在计算机上。整个系统分为以下四个模块：

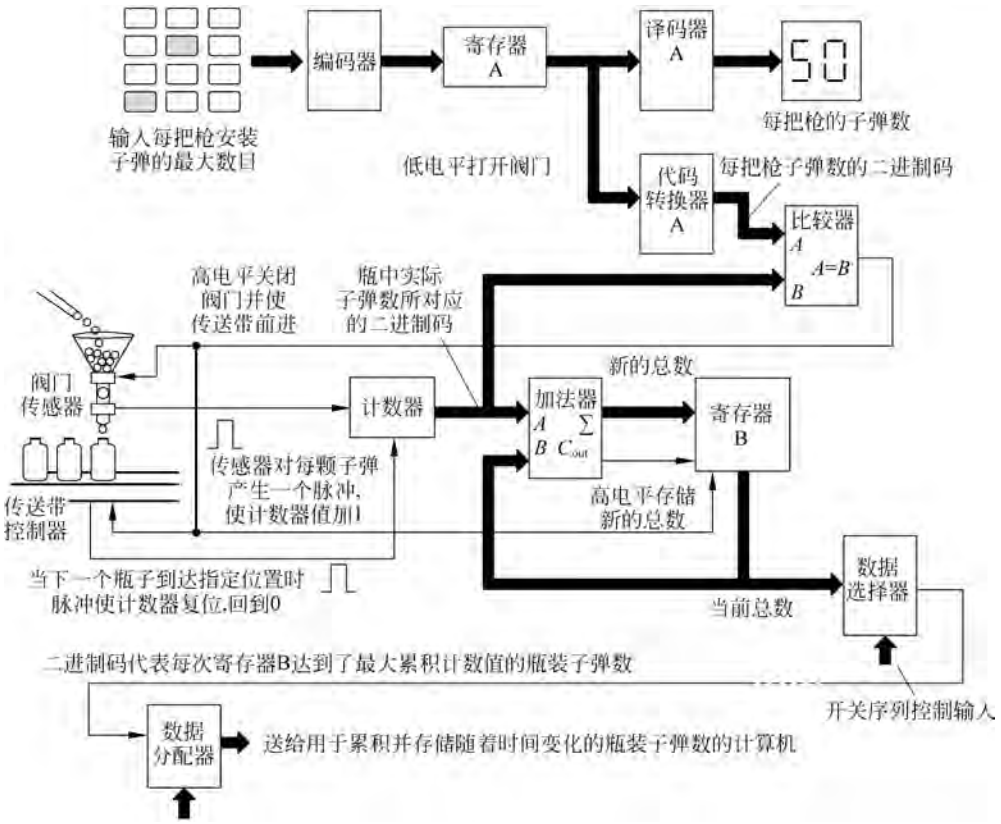


图 3.1.1 子弹分装系统框图

(1) 人机交互模块：键盘作为系统的输入，用来输入每把枪装入子弹的最大数目。数码管作为系统的输出，显示出输入的数字。

(2) 代码转换模块：将人机交互模块输出的 8421 码转换为二进制码。

(3) 自动分装模块：根据键盘输入的数字，控制阀门传感器将每把枪所需的子弹装入瓶中，当一瓶子弹装完后控制传送带将下一个瓶子自动传送到指定位置。

(4) 数据传输模块：将分装子弹总数通过数据线传送到计算机，以便跟踪特定时间的子弹分装总数。

3.1.2 任务需求

1. 人机交互

如图 3.1.1 所示，从键盘输入每把枪所需子弹的最大数目，由编码器转换为代码存储在寄存器 A 中，译码器 A 将存储在寄存器中的代码转换为数码管能够识别的七段代码，最终通过数码管将输入的数字显示出来。

需要设计的电路：编码器、寄存器、译码器、显示器。

2. 代码转换

如图 3.1.1 所示，比较器实现对于输入的每瓶子弹数目与瓶中装入的实际子弹数目进行比较。由于比较器输入的是二进制码，而寄存器存储的是 8421 码，因此需要一个能够把 8421 码转换为二进制码的转换电路。

需要设计的电路：代码转换器。

3. 自动分装

如图 3.1.1 所示，子弹被馈送到一个大的漏斗形料斗，在料斗的颈部安装一个光学传感器，该传感器一次只允许一颗子弹通过，通过后将产生一个脉冲信号，产生的脉冲送入计数器，使计数器的计数值加 1，因此子弹装入的过程中，计数器用来统计子弹的数目。将该数值与键盘输入每把手枪最大装入的子弹数目所对应的二进制数进行比较，假设当前每把枪装入子弹最大数是 50，当计数器中的二进制数达到 50 时，比较器的 $A=B$ 输出端变为高电平，表示子弹已装满。

利用比较器的高电平输出使得料斗颈部的阀门关闭和子弹流动停止。同时，比较器高电平输出激活传送机，将下一个瓶子移动到料斗下方，当到达相应的位置时，传送机的控制部分产生一个脉冲，将计数器复位，计数值回零，从而使比较器的输出回到低电平。打开料斗颈部的阀门，子弹再次流动。

需要设计的电路：计数器、比较器。

4. 数据传输

如图 3.1.1 所示，利用加法器可以计算出在一个给定的运行过程中子弹分发的总数。数据选择器将寄存器 B 中存储的二进制数总和从并行形式转换成串行形式，通过一条数据线传送到数据分配器，数据分配器将串行的数据再转回并行形式存储在计算机，以便跟踪在特定的时间子弹分装的总数。

需要设计的电路：加法器、寄存器、数据选择器、数据分配器。

中国芯片发展③——小规模集成电路(1962—1968)

在各类半导体研制单位的共同努力下，1963 年，中国科学院半导体研究所研制成功第一只半导体激光器，只比美国晚了一年。河北半导体所(现为中国电子科技集团公司第十三研究所)在 1965 年研制成功第一个硅基集成电路样品 GT31(美国在 1958 年)，同年在国内首次鉴定研制成功 DTL(Diode-Transistor-Logic)型逻辑电路。1966 年，上海元件五厂研制完成 TTL(Transistor-Transistor-Logic)型逻辑电路。这两种集成电路均属于小规模双极型数字集成电路，标志着中国在小规模集成电路方面实现自主可控。1966 年，我国研制成功第一台 65 型接触式光刻机，使得集成电路生产能力大幅提高。1968 年，我国第一台第三代计算机由华北计算技术研究所研制成功，采用的就是 DTL 型数字电路，其中与非门由北京电子管厂生产，与非驱动器由河北半导体研究所生产。

3.2 组合逻辑电路的分析与设计

(一) 课前篇

本节导学单

1. 学习目标

根据《布鲁姆教育目标分类学》，从知识维度和认知过程两方面进一步细化本节课的教学目标，明确学习本节课所必备的学习经验。

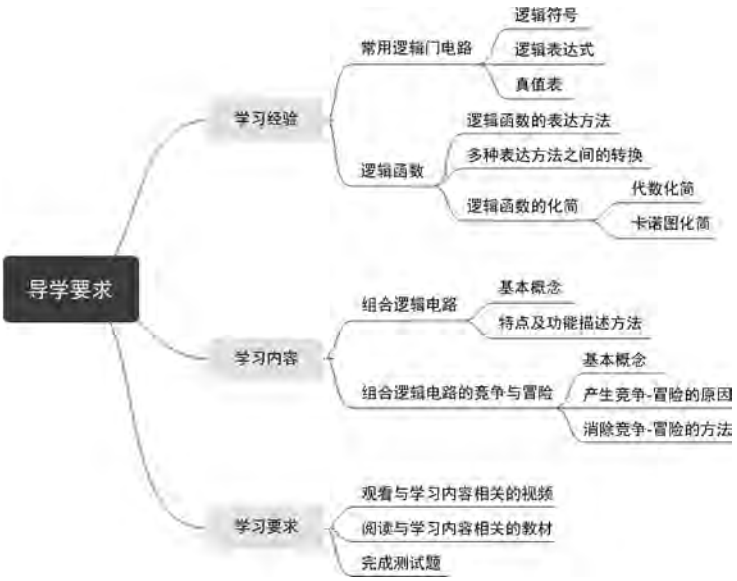
知识维度	认知过程					
	1. 回忆	2. 理解	3. 应用	4. 分析	5. 评价	6. 创造
A. 事实性知识	回忆真值表的基本概念	说出组合逻辑电路的基本概念；说出组合逻辑电路竞争与冒险的基本概念				

续表

知识维度	认知过程					
	1. 回忆	2. 理解	3. 应用	4. 分析	5. 评价	6. 创造
B. 概念性知识	回忆常用逻辑门电路的逻辑符号，逻辑表达式及真值表	阐释组合逻辑电路的特点及其功能描述方法	说出组合逻辑电路分析和设计的一般步骤，明晰每一个步骤所起的作用			
C. 程序性知识	回忆逻辑函数的多种表达式形式；回忆逻辑函数表示方法之间的转换；回忆逻辑函数的代数化简的步骤；回忆逻辑函数的卡诺图化简的步骤			根据组合逻辑电路分析的一般步骤，分析组合逻辑电路的逻辑功能；根据组合逻辑电路设计的一般步骤以及需求设计组合逻辑电路	借助于竞争与冒险，评价所设计的组合逻辑电路并进行优化	设计集成芯片的应用电路，会借助于仿真软件进行验证，会利用口袋实验包进行电路搭建与调试
D. 元认知知识	明晰学习经验是理解新知识的前提		根据概念及学习经验迁移得到新理论的能力	将知识分为若干任务，主动思考，举一反三，完成每个任务	在不断地发现问题、分析问题、解决问题的过程中体会精益求精的科学态度	通过电路的设计、仿真、搭建、调试，培养工程思维

2. 导学要求

根据本节的学习目标，将回忆、理解、应用层面学习目标所涉及的知识点课前自学完成，形成自学导学单。



3.2.1 课前自学——组合逻辑电路概述

1. 组合逻辑电路的定义

对于一个逻辑电路，其输出状态在任何时刻只取决于同一时刻的输入状态，而与电路原来的状态无关，这种电路称为**组合逻辑电路**。没有记忆功能是组合逻辑电路在逻辑功能上的共同特点。

图 3.2.1 是组合逻辑电路实例。它有三个输入变量 A、B、CI 和两个输出变量 S、CO。根据图 3.2.1 可以写出输出逻辑表达为

$$\begin{cases} S=(A\oplus B)\oplus \mathrm{CI} \\ \mathrm{CO}=(A\oplus B)\oplus \mathrm{CI}+AB \end{cases} \tag{3.2.1}$$

由上式可知，无论任何时刻，只要 A、B、CI 的取值确定，S 和 CO 的取值也随之确定，与电路过去的工作状态无关。

从组合逻辑电路功能特点不难想到，既然它的输出与电路之前的状况无关，那么电路中就不包含有存储单元。这就是组合逻辑电路在电路结构上的共同特点。

2. 组合逻辑电路的功能描述

对于任何一个多输入、多输出的组合逻辑电路，都可以用图 3.2.2 所示的框图来表示。

图中 $a_1,a_2,\cdots,a_n$ 表示输入变量，$y_1,y_2,\cdots,y_n$ 表示输出变量。输出与输入之间的逻辑关系可以用一组逻辑函数表示：

$$\begin{cases} y_1 = f_1(a_1, a_2, \cdots, a_n) \\ y_2 = f_2(a_1, a_2, \cdots, a_n) \\ \quad \vdots \\ y_m = f_m(a_1, a_2, \cdots, a_n) \end{cases} \tag{3.2.2}$$

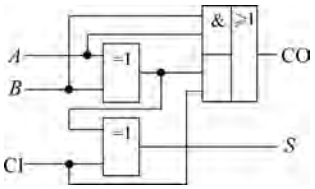


图 3.2.1　组合逻辑电路实例

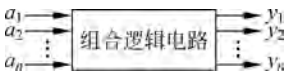


图 3.2.2　组合逻辑电路的一般框图

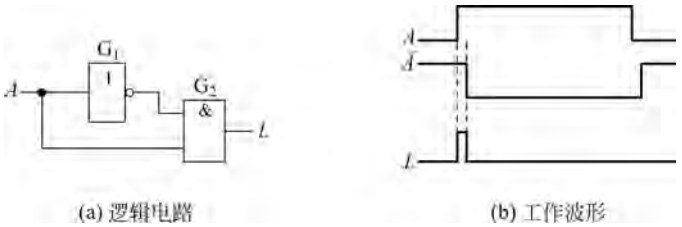
逻辑函数描述方法除了逻辑式以外，还有真值表、逻辑图、波形图等几种。因此，在分析或设计组合逻辑电路时，可以根据需要采用其中任何一种方式进行描述。

3.2.2　课前自学——组合逻辑电路的竞争与冒险

通过前面的学习知道，从信号加到逻辑门电路输入到得到稳定输出都需要一定时间，将这个时间称为门电路延迟时间。由于不同路径门的级数不同，信号经过不同路径传输时间不同，或者门的级数相同而各个门延迟时间差异，会造成信号传输时间不同。因此，电路在信号电平变化瞬间，可能与稳态下的逻辑功能不一致，产生错误输出，这种现象就是电路中的**竞争-冒险**。

1. 产生竞争-冒险的原因

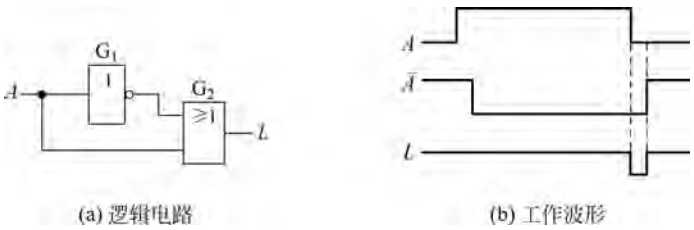
下面进一步分析组合逻辑电路产生的竞争-冒险。图 3.2.3(a)所示的逻辑电路中，它的输出逻辑表达式为 $L = A\overline{A}$。由图 3.2.3(b)所示的波形图可以看出，在 A 由 0 变 1 时，由于 G_1 门的输出有延迟，使得 $\overline{A}$ 由 1 变 0 有一延迟时间，而使输出 L 出现一正跳变的窄脉冲，该电路存在 1 冒险。



(a) 逻辑电路　　(b) 工作波形

图 3.2.3　组合逻辑电路的 1 冒险

图 3.2.4(a)所示的逻辑电路中，它的输出逻辑表达式为 $L = A + \overline{A}$。由图 3.2.4(b)所示的波形图可以看出，在 A 由 0 变 1 时，由于 G_1 门的输出有延迟，使得 $\overline{A}$ 由 1 变 0 有一延迟时间，而使输出 L 出现一负跳变的窄脉冲，该电路存在 0 冒险。



(a) 逻辑电路　　(b) 工作波形

图 3.2.4　组合逻辑电路的 0 冒险

综上所述，一个逻辑门的两个输入端的信号同时朝相反方向变化，而变化的时间有差异的现象称为竞争。由竞争而可能产生输出干扰脉冲的现象称为冒险。

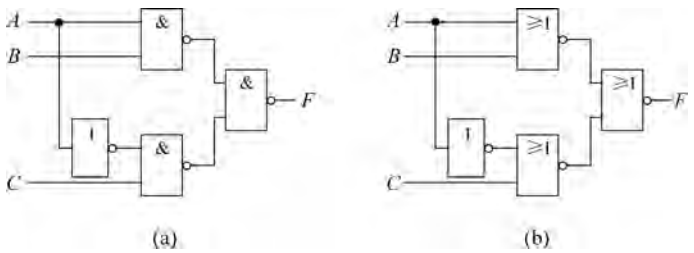
2. 检查竞争-冒险的方法

在输入变量每次只有一个改变状态的情况下，可以通过逻辑函数式判断组合逻辑电路中是否有竞争-冒险现象存在。如果输出端门电路的两个输入信号 A 和 $\overline{A}$ 是输入变量 A 经过两个不同的传输途径而来的(图 3.2.3 和图 3.2.4)，那么当输入变量 A 的状态发生突变时输出端便有可能产生尖峰脉冲。因此，只要输出端的逻辑函数在一定条件下能简化成

$$Y=A+\overline{A} \quad 或 \quad Y=A\cdot\overline{A} \tag{3.2.3}$$

则可判定存在竞争-冒险现象。

例 3.2.1　试判断图 3.2.5 中的两个电路中是否存在竞争-冒险现象。



(a)　　(b)

图 3.2.5　例 3.2.1 的电路

解：图 3.2.5(a)电路输出的逻辑表达式可写为

$$Y=AB+\overline{A}C \tag{3.2.4}$$

当 $B=C=1$ 时，上式将成为

$$Y=A+\overline{A} \tag{3.2.5}$$

故图 3.2.5(a)电路中存在竞争-冒险现象。

图 3.2.5(b)电路输出的逻辑表达式可写为

$$Y=(A+B)(\overline{A}+C) \tag{3.2.6}$$

当 $B=C=0$ 时，上式将成为

$$Y=A\cdot\overline{A} \tag{3.2.7}$$

故图 3.2.5(b)电路中存在竞争-冒险现象。

这种方法虽然简单，但是局限性太大，因为多数情况下输入变量都有两个以上同时改变状态的可能性。如果输入变量的数目有很多，就更难于从逻辑函数式上简单地找出所有产生竞争-冒险现象的情况。

另一种方法是将计算机辅助分析手段用于分析数字电路后，从原理上检查复杂数字电路竞争-冒险现象提供有效手段。通过在计算机上运行数字电路的程序，能够迅速查出电路是否会存在竞争-冒险现象。即使用计算机辅助分析手段检查过的电路，往往也还需要经过实验的方法检验，才能最后确定电路是否存在竞争-冒险现象。

3. 消除竞争-冒险的方法

针对上述分析，可以采取以下措施来消去竞争-冒险现象：

(1) 接入滤波电容。由于竞争-冒险而产生的尖峰脉冲一般都很窄(几十纳秒)，所以只要在输出端并接一个很小的滤波电容 C(图 3.2.6(a))，就足以把尖峰脉冲的幅度削弱至门电路的阈值电压以下。在 TTL 电路中，C 的数值通常为几十皮法至几百皮法。这种方法的优点是简单易行，缺点是增加了输出电压波形的上升时间和下降时间，使波形变坏。

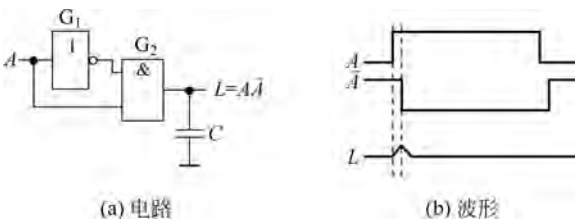


图 3.2.6　接入滤波电容消除竞争-冒险

(2) 引入选通脉冲。在电路中引入一个选通脉冲 EI，如图 3.2.7(a)所示。当电路输入端信号发生变化时不选通它，禁止电路工作；当所有输入信号变化完毕并达到稳态再让电路工作。即引入选通脉冲后输出端得到的永远是输入稳态后的输出值，不会出现尖峰脉冲。

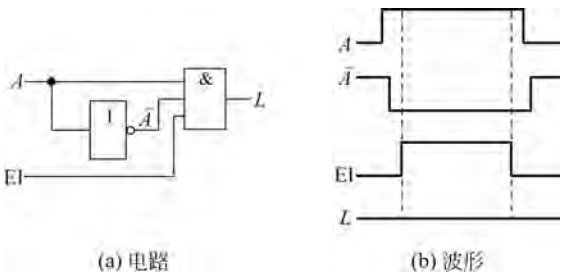


图 3.2.7　引入选通脉冲消除竞争-冒险

(3) 修改逻辑设计。以图 3.2.5(a)所示电路为例，已经得到了输出表达式 $Y=AB+\overline{A}C$，而且知道在 $B=C=1$ 的条件下，当 A 改变状态时存在竞争-冒险现象。

根据逻辑代数的常用公式可知

$$Y=AB+\overline{A}C=AB+\overline{A}C+BC \tag{3.2.8}$$

发现在增加了 BC 项后，在 $B=C=1$ 时无论 A 如何变化，输出始终保持 $Y=1$。因此，A 的状态变化不再会引起竞争-冒险现象。修改逻辑设计后的电路如图 3.2.8 所示。

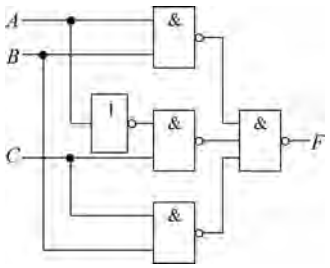


图 3.2.8　修改逻辑设计后的电路

3.2.3　课前自学——预习自测

3.2.3.1　下列对组合逻辑电路描述不正确的是(　　)。

A. 由常用门电路组合而成

B. 无输出到输入的反馈连接

C. 包含可以存储信号的记忆元件

D. 输出只由输入决定

3.2.3.2　下列表达式中不存在竞争冒险的是(　　)。

A. $Y=(A+B)(\overline{A}+C)$　　B. $Y=AB+\overline{B}C$

C. $Y=\overline{A}\overline{B}+AC$　　D. $Y=AD+\overline{A}B+BD$

3.2.3.3　画出逻辑函数 $L(A,B,C)=(A+C)(B+\overline{C})$ 的逻辑图，电路在什么条件下产生竞争-冒险？怎样修改电路能消除竞争-冒险？

(二) 课上篇

3.2.4　课中学习——组合逻辑电路的分析方法

组合逻辑电路分析目的是确定组合逻辑电路逻辑功能。分析组合逻辑电路步骤大致如下：

(1) 根据逻辑电路，从输入到输出逐级写出逻辑函数表达式，最终得到表述输出信号与输入信号关系的逻辑函数表达式。

(2) 利用公式化简法或卡诺图化简法将逻辑表达式化简或变换，以使逻辑关系更加简单明了。

(3) 为了使电路逻辑功能更加直观，根据简化后的逻辑表达式列写真值表。

(4) 根据真值表总结电路逻辑功能。

例 3.2.2 试分析图 3.2.9 所示电路的逻辑功能。

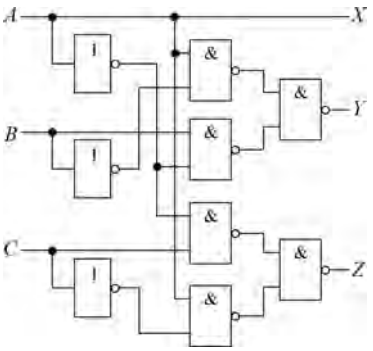


图 3.2.9 例 3.2.2 的电路

解：由图 3.2.9 写出 X、Y、Z 的逻辑表达式：

$$\begin{cases} X = A \\ Y = \overline{\overline{A\overline{B}} \cdot \overline{\overline{A}B}} \\ Z = \overline{\overline{A\overline{C}} \cdot \overline{\overline{A}C}} \end{cases} \tag{3.2.9}$$

逻辑表达式的化简：

$$\begin{cases} X = A \\ Y = A\overline{B} + \overline{A}B \\ Z = \overline{A}C + A\overline{C} \end{cases} \tag{3.2.10}$$

将式(3.2.10)转换成真值表的形式，得到表 3.2.1。

表 3.2.1 图 3.2.9 所示电路的逻辑真值表

输入			输出		
A	B	C	X	Y	Z
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	0	0

经过观察不难发现，真值表的前四行输出与输入相同，后四行输出 Y、Z 是对 B、C 进行求反。说明最高位为符号位，0 表示正数，1 表示负数，正数的反码与原码相同，负数的

数值部分是在原码的基础上逐位求反，所以该电路逻辑功能是对输入的二进制数求反码。

3.2.5 课中学习——组合逻辑电路的设计方法

根据给出的实际逻辑问题，完成实现这一逻辑功能的最简逻辑电路，是设计组合逻辑电路时要完成的工作。这里所说的最简，是指电路所用的器件数目最少，器件种类最少，而且器件之间的连线也最少。

设计组合逻辑电路的步骤大致如下：

(1) 逻辑抽象。在许多情况下，提出的设计要求都是用文字描述的具有一定因果关系的事件，这就需要通过逻辑抽象的方法，用一个逻辑函数来描述这一因果关系。

逻辑抽象的工作过程如下：

① 分析事件的因果关系确定输入和输出变量。一般总是把引起事件的原因定为输入，把事件的结果作为输出。

② 对输入变量和输出变量进行二进制编码，其编码规则和含义由设计者根据事件选定。

③ 对给定的因果关系列出真值表。在完成输入和输出变量的二进制编码后，根据给定的因果关系进行逻辑关系描述。

真值表是所有描述方法中最直接的描述方式，因此经常首先根据给定的因果关系列出真值表。至此，便将一个实际的逻辑问题抽象成一个逻辑函数，而且这个逻辑函数通常首先是以真值表形式给出的。

(2) 写出逻辑表达式并化简。为便于对逻辑函数进行化简和变换，需要把真值表转换为对应的逻辑函数式，转换的方法已在第 1 章中讲过。在使用小规模集成的逻辑门电路进行电路设计时，为获得最简单的设计结果，应将函数式化成最简形式，即函数式相加的乘积项最少，而且每个乘积项中的因子也最少。如果对所用器件的种类有附加限制(如只允许用单一类型的与非门)，则还应将函数式变换成与器件种类相适应的形式(如将函数式化为与非-与非形式)。如何使用中规模器件设计实现组合逻辑电路，将在本章后续章节进行介绍。

(3) 根据化简或转换后的逻辑表达式画出逻辑电路的连接图。至此，原理性设计(或称逻辑设计)已经完成。

(4) 设计验证。对已经得到的原理图进行分析，或借助计算机仿真软件进行功能和动态特性仿真，验证其是否符合设计要求。

例 3.2.3 设计一个监视交通信号灯工作状态的逻辑电路。每一组信号灯由红、黄、绿三盏灯组成，如图 3.2.10 所示。正常工作情况下，任何时刻必有一盏灯亮，而且只允许有一盏灯亮。其他情况出现，电路发生故障，这时要求发出故障信号，提醒维护人员修理。


仿真视频

解：(1) 逻辑抽象。

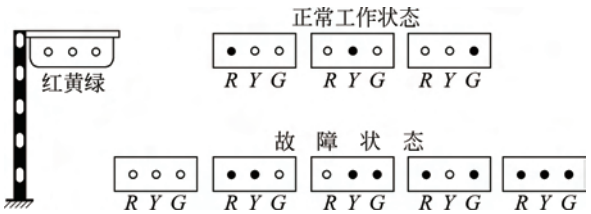


图 3.2.10　交通信号灯的正常工作状态和故障状态

取红、黄、绿三盏灯状态为输入变量，分别用 R、Y、G 表示，并规定灯亮时为 1，不亮时为 0。取故障信号为输出变量，用 Z 表示，并规定正常工作状态下 Z 为 0，发生故障时 Z 为 1。

根据题意可列出表 3.2.2 所示的逻辑真值表。

表 3.2.2　例 3.2.3 的逻辑真值表

输　入			输　出
R	Y	G	Z
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(2) 写出逻辑表达式。

由表 3.2.2 可知

$$Z=\bar{R}\bar{Y}\bar{G}+\bar{R}YG+R\bar{Y}G+RY\bar{G}+RYG \tag{3.2.11}$$

将该表达式进行化简(图 3.2.11)，可得

$$Z=\bar{R}\bar{Y}\bar{G}+YG+RY+RG \tag{3.2.12}$$

(3) 根据式(3.2.12)的化简结果画出逻辑电路图，如图 3.2.12 所示。

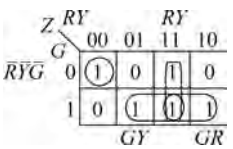


图 3.2.11　例 3.2.3 卡诺图

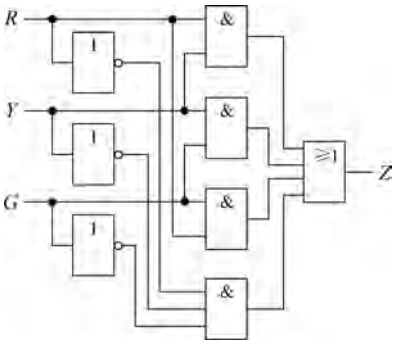


图 3.2.12　例 3.2.3 的逻辑图一

由于式(3.2.12)为最简与或表达式，所以只有在使用与门和或门组成电路时才得到最简单的电路。如果要求用其他类型的门电路来组成这个逻辑电路，则为了得到最简单的电路，化简的结果也需相应地改变。例如，在要求全部用与非门组成这个逻辑电路时，就应当将函数式化为最简与非-与非表达式。这种形式通常可以通过将与或表达式两次求反得到。将式(3.2.12)两次求反后可得

$$Z=\overline{\overline{\overline{R\bar{Y}\bar{G}}+YG+RY+RG}}=\overline{\overline{R\bar{Y}\bar{G}}\cdot\overline{YG}\cdot\overline{RY}\cdot\overline{RG}} \tag{3.2.13}$$

根据式(3.2.13)即可画出全部用与非门组成的逻辑电路，如图 3.2.13 所示。

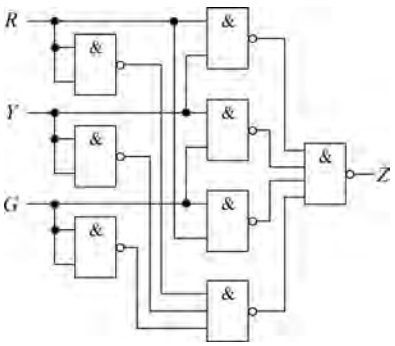


图 3.2.13　例 3.2.3 的逻辑图二

若要求用非门和与或非门实现这个逻辑电路，则必须将式(3.2.11)化为最简与或非表达式。在第 1 章曾经讲过，最简与或非表达式可以通过合并卡诺图上的 0，然后求反而得到。为此，将函数 Z 的卡诺图画出，如图 3.2.14 所示。

将图 3.2.14 中的 0 合并、求反得到

$$Z=\overline{R\bar{Y}\bar{G}+R\bar{Y}\bar{G}+\bar{R}\bar{Y}G} \tag{3.2.14}$$

对照式(3.2.14)画出的用与或非门组成的逻辑电路图，如图 3.2.15 所示。

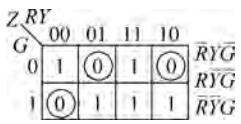


图 3.2.14　例 3.2.3 的卡诺图

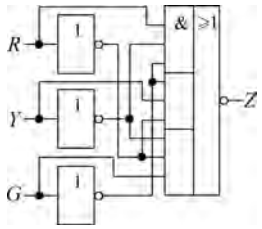


图 3.2.15　例 3.2.3 的逻辑图三

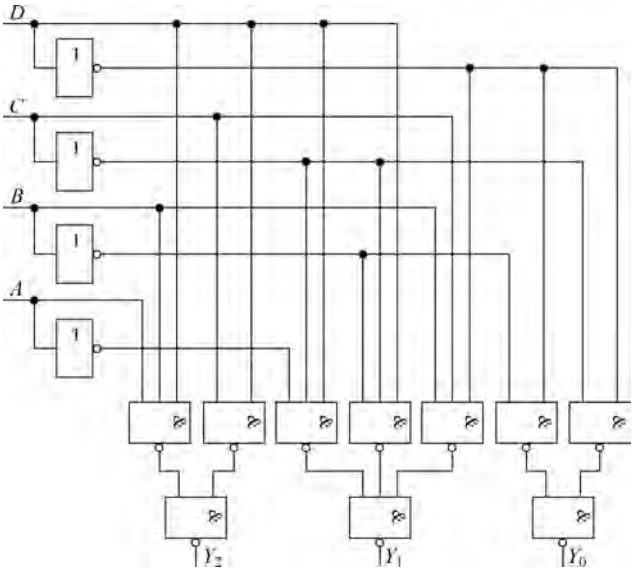
(三) 课后篇

3.2.6　课后巩固——练习实践

1. 知识巩固练习

3.2.6.1　试分析题图 3.2.6.1 所示电路的逻辑功能。

讲解视频



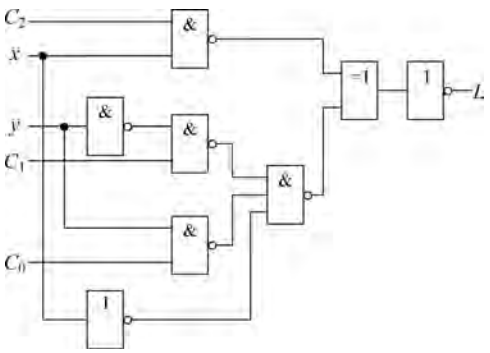
题图 3.2.6.1

讲解视频

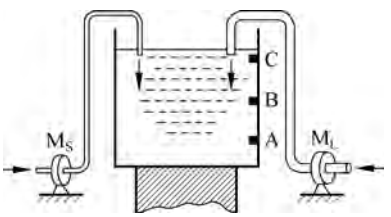
3.2.6.2　如题图 3.2.6.2 所示电路是一个多功能函数发生器，其中 C_2、C_1、C_0 为控制信号，x、y 为数据输入。(1)写出输出 L 的逻辑表达式并化简为最简与或式。(2)列表说明当 $C_2C_1C_0$ 为不同取值组合时，输出端 L 的逻辑功能。

讲解视频

3.2.6.3　一水箱由大、小两台水泵 M_L 和 M_S 供水，如题图 3.2.6.3 所示。水箱中设置了 3 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 点而高于 B 点时 M_S 单独工作；水位低于 B 点而高于 A 点时 M_L 单独工作；水位低于 A 点时 M_L 和 M_S 同时工作。试用门电路设计一个控制两台水泵的逻辑电路，要求电路尽量简单。



题图 3.2.6.2

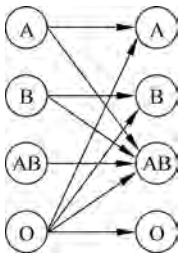


题图 3.2.6.3

3.2.6.4　设计一个电话机信号控制电路。电路有 I_0(火警)、I_1(盗警)和 I_2(日常业务)三种输入信号,通过排队电路分别从 L_0、L_1 和 L_2 输出,在同一时间只能有一个信号通过。如果同时有两个以上信号出现时,应首先接通火警信号,其次为盗警信号,最后是日常业务信号。试按照上述轻重缓急设计该信号控制电路。要求用集成门电路 7400(每片含 4 个二输入与非门)实现。

3.2.6.5　试设计一个 4 位的奇偶校验器,即当 4 位数中有奇数个 1 时输出为 0,否则输出为 1。(可以采用各种逻辑功能的门电路来实现。)

3.2.6.6　某产品有 A、B、C 三项指标,至少有两项达标(必须包含 A)时才算合格,试用与非门设计符合上述规定的逻辑电路。



2. 工程实践练习

3.2.6.7　人类有 A、B、O、AB 四种常见血型,输血时输血者血型与受血者血型必须符合题图 3.2.6.7 中用箭头指示的关系。试设计一个逻辑电路,判断输血者与受血者的血型是否符合上述规定。要求:(1)利用仿真软件完成电路的仿真;(2)利用口袋实验包完成电路的搭建与调试。

题图 3.2.6.7

3.2.7　本节思维导图

知识拓展——国内芯片设计企业：华为海思　集成电路国产化进程

2004年，在原有ASIC设计中心基础上成立了深圳市海思半导体有限公司，即华为海思（HI-SILICON，HI是HUAWEI，SILICON是硅）。海思公司主要从事手机芯片、移动通信系统设备芯片、传输网络设备芯片、家庭数字设备芯片等设计，海思公司的安防监控领域产品所占全球市场份额已达到90%。美国不允许台积电公司给华为公司代工，影响最大的是华为公司的麒麟系列手机芯片，该芯片是集成了手机基带、CPU、GPU和电源管理芯片等的SoC芯片，其生产工艺水平要求高、难度大，目前国内最好的中芯国际公司也处于工艺研发阶段，尚无法大规模量产，急需更多的科研人才付出努力。

3.3　编码器

（一）课前篇

本节导学单

1. 学习目标

根据《布鲁姆教育目标分类学》，从知识维度和认知过程两方面进一步细化本节课的教学目标，明确学习本节课所必备的学习经验。

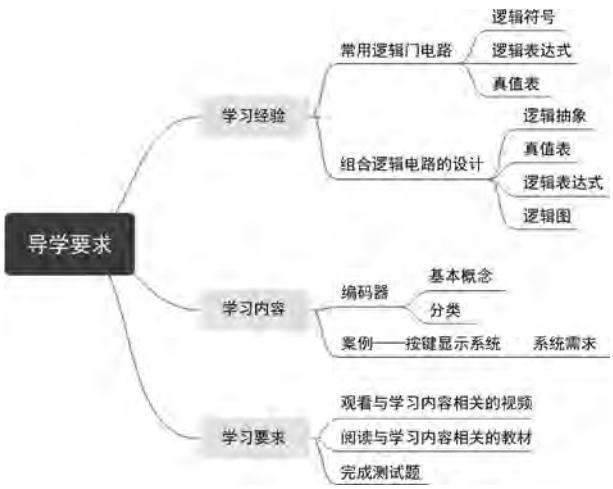
知识维度	认知过程					
	1. 回忆	2. 理解	3. 应用	4. 分析	5. 评价	6. 创造
A. 事实性知识		说出编码器的基本概念及分类				
B. 概念性知识	回忆常用逻辑门电路的逻辑符号，逻辑表达式及真值表					
C. 程序性知识	回忆组合逻辑电路设计的一般步骤	画出集成优先编码器引脚功能、真值表	应用门电路设计普通二进制编码器	分析普通二进制编码器不足，用门电路设计优先编码器		利用集成芯片设计按键显示系统的电路，会借助仿真软件进行验证，会利用口袋实验包进行电路搭建与调试

续表

知识维度	认知过程					
	1. 回忆	2. 理解	3. 应用	4. 分析	5. 评价	6. 创造
D. 元认知知识	明晰学习经验是理解新知识的前提		根据概念及学习经验迁移得到新理论的能力	在不断地发现问题、分析问题、解决问题的过程中体会精益求精的科学态度		通过电路的设计、仿真、搭建、调试，培养科学的工程思维

2. 导学要求

根据本节的学习目标，将回忆、理解、应用层面学习目标所涉及的知识点课前自学完成，形成自学导学单。



3.3.1 课前自学——编码器的定义及分类

如图 3.3.1 所示，这些图片都很熟悉，有运动员的号码簿、居民身份证号码，还有商品的条形码。编码就是用一组代码按一定规则表示某种事物或信息，是社会管理中的一种常用方法。在数字系统中，为了区分一系列不同的事物，将其中的每个事物用一个二进制码表示。能够实现这种编码功能的逻辑部件称为编码器。

编码器有若干输入，对每一个有效的输入信号，编码器产生一组唯一的二进制码输出。因为数字系统只有 0 和 1 两个数符，一位二进制数只能区分两个不同的信号。一般而言，N 个不同的信号至少需要 n 位二进制数编码。N 和 n 之间满足下列关系：

$$2^n \geqslant N \tag{3.3.1}$$

图 3.3.1　生活中的编码

例如，有 10 个按键 $S_0 \sim S_9$，将其完全区分开至少需要 4 位二进制码。

由于逻辑电路中信号都是以高、低电平的形式给出。因此编码器的逻辑功能就是将输入的每一个高、低电平信号编成一个对应的二进制码输出。图 3.3.2 为二进制编码器结构图，它有 2^n 个高、低电平输入，对应有 n 位二进制码输出。

编码器分为普通编码器和优先编码器。普通编码器任何时刻只允许一个输入信号有效，否则将产生错误输出；优先编码器允许多个输入信号同时有效，输出仅对优先级别最高的输入信号进行编码。

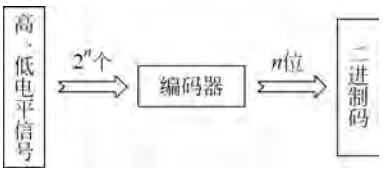


图 3.3.2　二进制编码器的结构框图

3.3.2　课前自学——预习自测

3.3.2.1　对全班 43 个学生以二进制码编码表示，最少需要二进制码的位数是（　　）。

3.3.2.2　计算机键盘上有 101 个按键，若用二进制码进行编码，至少需要（　　）位二进制数，第 99 个键的二进制码为（　　）。

（二）课上篇

3.3.3　课中学习——普通二进制编码器

以 4 线-2 线普通二进制编码器为例，分析普通编码器的工作原理。图 3.3.3 是 4 线-2 线普通二进制编码器的框图，它的输入是 4 个电平信号 $I_0 \sim I_3$，高电平有效，输出是 2 位二进制码 Y_1Y_0。

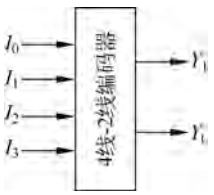


图 3.3.3　4 线-2 线普通二进制编码器框图

输出与输入对应关系如表 3.3.1 所示，根据真值表设计编码器电路。4 个输入 $I_0 \sim I_3$ 为高电平有效，任何时刻只能有一个取值为 1，并且有一组对应的二进制码输出。除表中列出 4 个输入变量的 4 种取值组合有效外，其余 12 种组合对应的输出均应为 0。

表 3.3.1　4 线-2 线编码器真值表

输　入				输　出	
I_0	I_1	I_2	I_3	Y_1	Y_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

由真值表可以得到如下逻辑表达式：

$$Y_1 = \bar{I}_0 \bar{I}_1 I_2 \bar{I}_3 + \bar{I}_0 \bar{I}_1 \bar{I}_2 I_3 \tag{3.3.2}$$

$$Y_0 = \bar{I}_0 I_1 \bar{I}_2 \bar{I}_3 + \bar{I}_0 \bar{I}_1 \bar{I}_2 I_3 \tag{3.3.3}$$

任何时刻只有一个输入为 1，如果将表 3.3.1 中没有列出的 12 种组合所对应的输出看作无关项，并化简得

$$Y_1 = I_2 + I_3 \tag{3.3.4}$$

$$Y_0 = I_1 + I_3 \tag{3.3.5}$$

式(3.3.4)和式(3.3.5)是考虑了无关项的简化结果，比式(3.3.2)和式(3.3.3)简单。

3.3.4　课中学习——优先编码器

在实际应用中经常会遇到两个以上的输入同时有效的情况，因此必须根据轻重缓急事先规定好这些输入编码的先后次序，即优先级别。识别这类请求信号的优先级别并进行编码的逻辑电路称为优先编码器。编码器应用广泛，为了方便使用将编码器电路模块化，制成了标准化的集成器件。下面介绍集成优先编码器 74148。

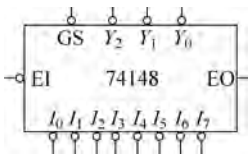


图 3.3.4　74148 逻辑符号

1. 集成优先编码器 74148

74148 逻辑符号如图 3.3.4 所示。$I_0 \sim I_7$ 是 8 个输入端；$Y_2 \sim Y_0$ 是 3 个输出端。此外还有三个控制端：GS 是优先编码工作状态标志；EI 是输入使能端；EO 是输出使能端。

74148 功能表如表 3.3.2 所示，在 EI=0 时，优先编码器正常工作，允许 $I_0 \sim I_7$ 中同时有几个输入端为低电平，即有编码输入信号，但是编码器仅对级别最高的编码器进行编码。从 74148 的功能表中不难看出，I_7 的级别最高，I_0 的级别最低。当 $I_7=0$ 时，无论其他输入端有无输入信号(表中以“×”表示)，输出端只对 I_7 进行编码，$Y_2Y_1Y_0=000$，

即反码输出。当 $I_7=1, I_6=0$ 时，无论其他输入端有无输入信号，只对 I_6 编码，输出为 $Y_2Y_1Y_0=001$。其余输入状态可自行分析。

表 3.3.2　74148 功能表

输　入									输　出				
EI	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Y_2	Y_1	Y_0	GS	EO
1	×	×	×	×	×	×	×	×	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	×	×	×	×	×	×	×	0	0	0	0	0	1
0	×	×	×	×	×	×	0	1	0	0	1	0	1
0	×	×	×	×	×	0	1	1	0	1	0	0	1
0	×	×	×	×	0	1	1	1	0	1	1	0	1
0	×	×	×	0	1	1	1	1	1	0	0	0	1
0	×	×	0	1	1	1	1	1	1	0	1	0	1
0	×	0	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

2. 8421 码编码器

利用集成优先编码器 74148 和门电路即可实现 8421 码编码器，如图 3.3.5 所示。输入仍为低电平有效，输出为 8421 码。

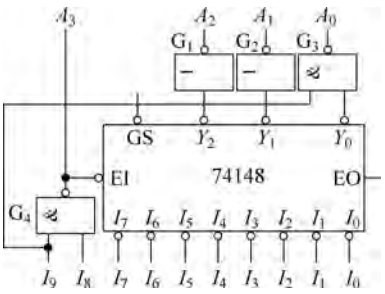


图 3.3.5　8421 码编码器

其工作原理：当 I_9、I_8 无输入（I_9、I_8 均为高电平）时，与非门 G_4 的输出 $A_3=0$，同时使 74148 的 EI=0，允许 74148 工作，74148 对输入 $I_0\sim I_7$ 进行编码。如 $I_5=0$，则 $Y_2Y_1Y_0=010$，经门 G_1、G_2、G_3 处理后，$A_2A_1A_0=101$，所以总输出 $A_3A_2A_1A_0=0101$，这正好是 5 的 8421 码。当 I_9 或 I_8 有输入（低电平）时，与非门 G_4 的输出 $Y_3=1$，同时使 74148 的 EI=1，禁止 74148 工作，使得 $Y_2Y_1Y_0=111$。如果此时 $I_9=0$，则输出 $A_3A_2A_1A_0=1001$；如果 $I_8=0$，则输出 $A_3A_2A_1A_0=1000$，这正好是 9 和 8 的 8421 码。

（三）课后篇

3.3.5 课后巩固——练习实践

1. 知识巩固练习

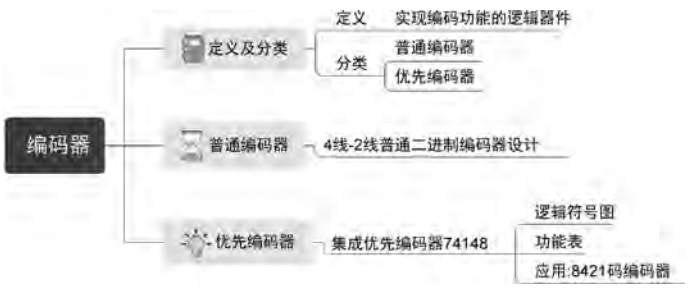
3.3.5.1 试用4片8线-3线优先编码器74148组成32线-5线优先编码器的逻辑图，允许附加必要的门电路。

3.3.5.2 试用74148设计键盘编码电路，10个按键分别对应十进制数0～9，编码器的输出为8421码。要求按键9的优先级别最高，并且有工作状态标志，以区分没有按键按下和按键0按下两种情况。

2. 工程实践练习

3.3.5.3 利用口袋实验包完成具有优先功能的10个按键编码电路的搭建与调试。

3.3.6 本节思维导图



知识拓展——国内芯片制造企业：中芯国际　　集成电路国产化进程

2018年，华为公司、中兴公司被美国制裁，最主要的“卡脖子”技术就是芯片制造技术。目前中国内地规模最大、技术最先进、首家能提供14nm先进制程技术的集成电路芯片制造企业是中芯国际集成电路制造有限公司（简称中芯国际）。作为全球排名第四的芯片代工厂商，其主要业务是根据客户本身或第三方的集成电路设计为客户制造集成电路芯片。中芯国际是内地高科技产业在美国制裁下唯一可以实现14nm FinFET量产的芯片代工企业，代表着内地自主研发集成电路制造技术的最先进水平，距离芯片制造龙头台积电公司3nm工艺相差2、3代的技术工艺，亟须集中国内优势人才设备，补足缺项，迎头赶上。

3.4 译码器/数据分配器

（一）课前篇

本节导学单

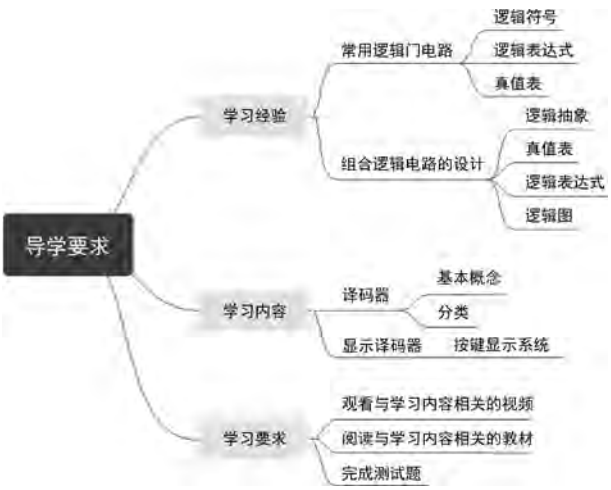
1. 学习目标

根据《布鲁姆教育目标分类学》，从知识维度和认知过程两方面进一步细化本节课的教学目标，明确学习本节课所必备的学习经验。

知识维度	认知过程					
	1. 回忆	2. 理解	3. 应用	4. 分析	5. 评价	6. 创造
A. 事实性知识		说出译码器的基本概念及分类				
B. 概念性知识	回忆常用逻辑门电路的逻辑符号，逻辑表达式及真值表					
C. 程序性知识	回忆组合逻辑电路设计的一般步骤	画出集成译码器、集成显示译码器引脚功能、真值表	应用门电路设计二进制译码器	应用集成译码器实现组合逻辑函数。应用集成译码器实现数据分配器	对比评价译码器实现组合逻辑函数和门电路实现组合逻辑函数的方法和步骤	利用集成芯片设计应用电路，会借助仿真软件进行验证，会利用口袋实验包进行电路搭建与调试
D. 元认知知识	明晰学习经验是理解新知识的前提		根据概念及学习经验迁移得到新理论的能力	将知识分为若干任务，主动思考，举一反三，完成每个任务	类比评价也是知识整合吸收的过程	通过电路的设计、仿真、搭建、调试，培养工程思维

2. 导学要求

根据本节的学习目标，将回忆、理解、应用层面学习目标所涉及的知识点课前自学完成，形成自学导学单。



3.4.1 课前自学——译码器的定义及分类

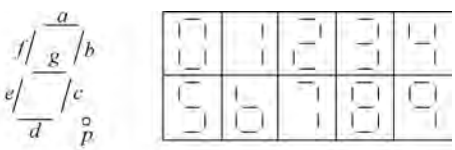
译码是编码的逆过程，它的功能将具有特定含义的二进制码转换成对应输出的高/低电平信号。具有译码功能的逻辑电路称为译码器。译码器常用在计算机和其他类型系统中实现 I/O 的选择。

常见的译码器分为二进制译码器、二-十进制译码器和显示译码器三类。二进制译码器输入是一组二进制码，输出是一组与输入代码一一对应的高、低电平信号；二-十进制译码器是将输入的 10 个 BCD 码译成对应的高/低电平输出信号；显示译码器将 BCD 码译成数码管所需要的驱动信号，使数码管用十进制数字显示出 BCD 码所表示的数值。

3.4.2 课前自学——显示译码器

1. 七段字符显示器

图 3.4.1 表示七段数字显示器发光段组合，显示 0～9 阿拉伯数字。这种数码管的每一段都是一个发光二极管(LED)，因而也称它为 LED 数码管或 LED 七段显示器。有些数码显示器增加了一段，作为小数点。



(a) 分段布置　　(b) 段组合

图 3.4.1　七段数字显示器发光段组合

发光二极管构成的七段显示器有两种，分别为共阴极电路和共阳极电路，如图 3.4.2 所示。共阴极电路中，8 个发光二极管的阴极连在一起接低电平，需要某一段发光，就将相应二极管的阳极接高电平。共阳极显示器的驱动则刚好相反。

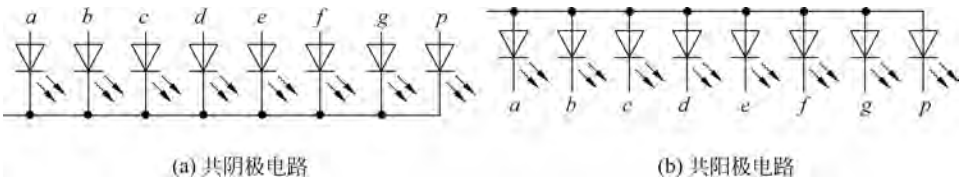


图 3.4.2　二极管显示器等效电路

2. 显示译码器的逻辑功能

显示译码器的功能是将 BCD 码转换为显示器所需要的七段代码，如图 3.4.3 所示。以共阴极显示器为例来分析显示译码器的工作过程。例如，要显示 9，共阴极显示器的 a、b、c、f、g 段发光二极管要点亮，因此 a、b、c、f、g 引脚为 1。所以显示译码器的功能就是将 9 对应的 BCD 码 1001 转换为共阴极显示器所需要的七段代码 1110011。

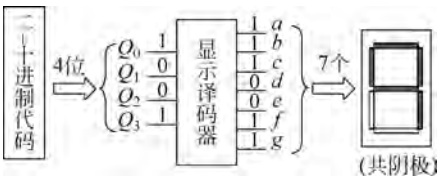


图 3.4.3　显示译码器功能框图

进而可以得到显示译码器的真值表，如表 3.4.1 所示。

表 3.4.1　显示译码器的真值表

输　入				输　出						
Q_3	Q_2	Q_1	Q_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1

根据真值表可以画出输出 $a \sim g$ 的卡诺图，如图 3.4.4 所示。

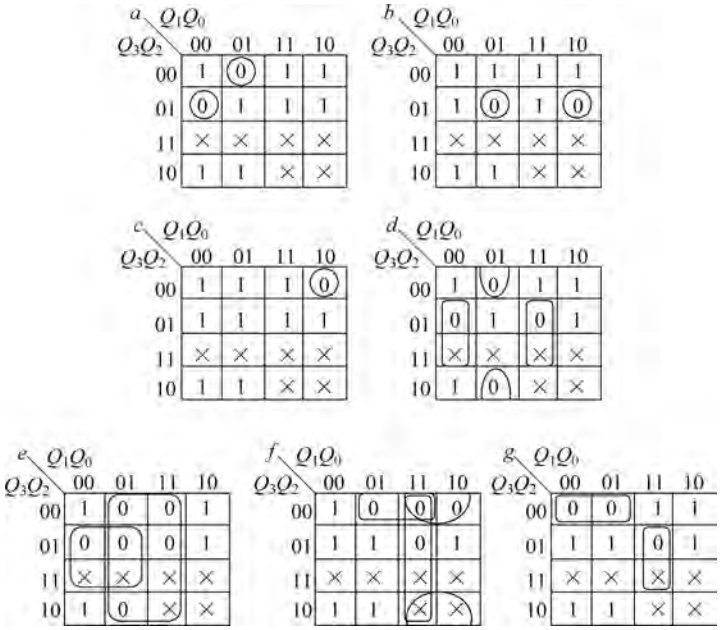


图 3.4.4　BCD-七段显示译码器的卡诺图

在卡诺图上采用“合并 0 然后求反”的化简方法将 $a\sim g$ 化简，得到

$$\begin{cases} a=\overline{\bar{Q}_3\bar{Q}_2\bar{Q}_1Q_0+\bar{Q}_3Q_2\bar{Q}_1\bar{Q}_0} \\ b=\overline{\bar{Q}_3\bar{Q}_2\bar{Q}_1Q_0+\bar{Q}_3Q_2Q_1\bar{Q}_0} \\ c=\overline{\bar{Q}_3\bar{Q}_2Q_1\bar{Q}_0} \\ d=\overline{Q_2\bar{Q}_1\bar{Q}_0+\bar{Q}_2\bar{Q}_1Q_0+Q_2Q_1Q_0} \\ e=\overline{Q_0+Q_2\bar{Q}_1} \\ f=\overline{Q_1Q_0+\bar{Q}_3\bar{Q}_2Q_0+\bar{Q}_2Q_1} \\ g=\overline{\bar{Q}_3\bar{Q}_2\bar{Q}_1+Q_2Q_1Q_0} \end{cases} \tag{3.4.1}$$

根据式(3.4.1)用门电路即可得到显示译码器电路。

3. 集成显示译码器

常用集成七段显示译码器有两类：一类译码器输出高电平有效信号，用来驱动共阴极显示器；另一类译码器输出低电平有效信号，用来驱动共阳极显示器。下面介绍常用七段显示译码器 7447。

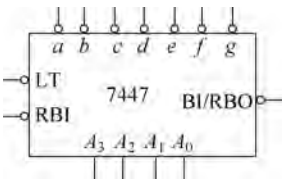


图 3.4.5　7447 七段显示译码器逻辑符号

7447 七段显示译码器逻辑符号如图 3.4.5 所示，功能表如表 3.4.2 所示。当输入 8421 码时，输出低电平有

表 3.4.2　7447 七段显示译码器功能表

十进制数字或功能	输入							输出							显示字型
	LT	RBI	A_3	A_2	A_1	A_0	BI/RBO	a	b	c	d	e	f	g	
0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0
1	1	×	0	0	0	1	1	1	0	0	1	1	1	1	1
2	1	×	0	0	1	0	1	0	0	1	0	0	1	0	2
3	1	×	0	0	1	1	1	0	0	0	0	1	1	0	3
4	1	×	0	1	0	0	1	1	0	0	1	1	0	0	4
5	1	×	0	1	0	1	1	0	1	0	0	1	0	0	5
6	1	×	0	1	1	0	1	1	1	0	0	0	0	0	6
7	1	×	0	1	1	1	1	0	0	0	1	1	1	1	7
8	1	×	1	0	0	0	1	0	0	0	0	0	0	0	8
9	1	×	1	0	0	1	1	0	0	0	1	1	0	0	9
10	1	×	1	0	1	0	1	1	1	1	0	0	1	0	c
11	1	×	1	0	1	1	1	1	1	0	0	1	1	0	3
12	1	×	1	1	0	0	1	1	0	1	1	1	0	0	L
13	1	×	1	1	0	1	1	0	1	1	0	1	0	0	5
14	1	×	1	1	1	0	1	1	1	1	0	0	0	0	E
15	1	×	1	1	1	1	1	1	1	1	1	1	1	1	熄灭
BI	×	×	×	×	×	×	0	1	1	1	1	1	1	1	熄灭
RBI	1	0	0	0	0	0	0	1	1	1	1	1	1	1	灭零
LT	0	×	×	×	×	×	1	0	0	0	0	0	0	0	试灯

效，用以驱动共阳极显示器。该集成显示译码器设有三个控制端 LT、RBI、BI/RBO，用以扩展电路的功能。

7447 七段显示译码器控制端的功能如下：

(1) 测灯输入 LT：当 LT＝0 时，不论 RBI 和 $A_3A_2A_1A_0$ 输入为何值，数码管的七段全亮，用以检查该数码管各段能否正常发光，正常工作时应将 LT 置为 1。

(2) 灭零输入 RBI：设置灭零输入信号的目的是把不希望显示的零熄灭。例如，有一个 8 位的数码显示电路，整数部分为 5 位，小数部分为 3 位，在显示 13.7 这个数时将呈现 00013.700 字样。如果将前、后多余的零熄灭，则显示的结果将更加醒目。当 RBI＝0 且 $A_3A_2A_1A_0$ 输入 0000 时，此时执行灭零操作。

(3) 灭灯输入/灭零输出 BI/RBO：作为输入端使用时，称灭灯输入控制端。只要 BI＝0，无论 $A_3A_2A_1A_0$ 是什么状态，数码管的各段同时熄灭。作为输出端使用时，称为灭零输出端。只有当输入 $A_3＝A_2＝A_1＝A_0＝0$，而且有灭零输入信号(RBI＝0)时，RBO 才会为 0，表示译码器已将本来应该显示的零熄灭。

将灭零输入端和灭零输出端配合使用即可实现多位数码显示系统的灭零控制，如图 3.4.6 所示。只需将整数部分高位 RBO 与低位 RBI 相连，小数部分低位 RBO 与高位

RBI 相连，就可以把前、后多余零熄灭。在这种连接方式下，整数部分只有高位是零且被熄灭的情况下，低位才有灭零输入信号。同理，小数部分只有在低位是零而且被熄灭时，高位才有灭零输入信号。

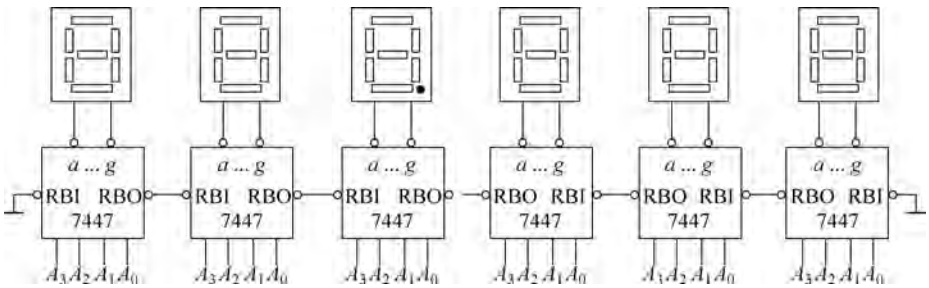


图 3.4.6　有灭零控制的 8 位数码显示系统

3.4.3　课前自学——预习自测

3.4.3.1　显示译码器 7447 可以驱动共(　　)数码管。

3.4.3.2　用显示译码器 7447 驱动七段数码管时，发现数码管只显示 1、3、5、7、9，试分析故障可能在哪里？

讲解视频

(二) 课上篇

3.4.4　课中学习——二进制译码器

二进制译码器输入是一组二进制码，输出是一组与输入代码一一对应的高/低电平信号。

1. 工作原理

二进制译码器结构如图 3.4.7 所示，输入的 n 位二进制码共有 2^n 种状态，译码器将每个输入代码译成对应的一根输出线上的高/低电平信号，因此也把这个译码器称为 n 线-2^n 线译码器。

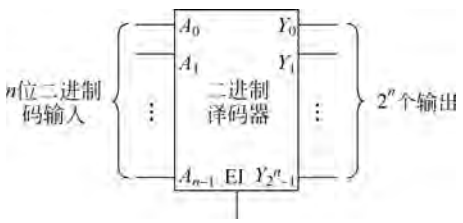


图 3.4.7　二进制译码器结构

下面以 2 线-4 线译码器为例，分析译码器的工作原理和电路结构，其中 2 线表示 2 个输入变量，4 线表示有 4 个输出端。两个输入变量 A_1、A_0 有 4 种不同状态组合，因而译码器有 4 个输出信号 $Y_3 \sim Y_0$，并且输出为低电平有效。2 线-4 线译码器真值表如表 3.4.3 所示。

表 3.4.3　2 线-4 线译码器真值表

输　入			输　出			
E	A_1	A_0	Y_3	Y_2	Y_1	Y_0
1	×	×	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

另外，设置了使能控制端 E，当 E 为 1 时，无论 A_1、A_0 为何种状态，输出全部为 1，译码器处于非工作状态。而当 E 为 0 时，根据 A_1、A_0 的状态组合，使得对应输出为 0，其余输出为 1。例如，$A_1A_0=01$ 时，输出 Y_1 为 0，其余输出 Y_3、Y_2、Y_0 均为 1。由此可见，译码器是通过输出端逻辑电平以识别不同代码。

根据真值表写出各输出端的逻辑表达式：

$$\begin{cases} Y_3=\overline{\overline{E}\cdot A_1A_0} \\ Y_2=\overline{\overline{E}\cdot A_1\overline{A_0}} \\ Y_1=\overline{\overline{E}\cdot\overline{A_1}A_0} \\ Y_0=\overline{\overline{E}\cdot\overline{A_1}\ \overline{A_0}} \end{cases} \tag{3.4.2}$$

根据逻辑表达式画出逻辑图，如图 3.4.8 所示。

2. 集成译码器

常用集成译码器有 2 线-4 线译码器 74139 和 3 线-8 线译码器 74138。

1）2 线-4 线集成译码器 74139

2 线-4 线译码器 74139 逻辑符号如图 3.4.9 所示，它有两个独立的译码器封装在一个集成芯片中。A_1、A_0 为输入端，E 为使能控制端，低电平有效，$Y_3\sim Y_0$ 为输出端，低电平有效。当 E 为 1 时，无论 A_1、A_0 为何种状态，输出全部为 1，译码器处于非工作状态。而当 E 为 0 时，根据 A_1A_0 的输入组合译成相应输出端的有效低电平。2 线-4 线译码器 74139 真值表如表 3.4.3 所示。

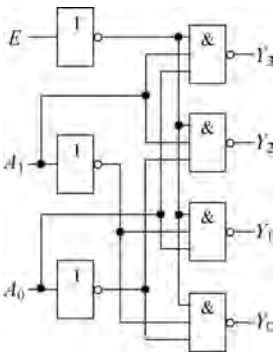


图 3.4.8　2 线-4 线译码器逻辑图

图 3.4.9　2 线-4 线译码器 74139 逻辑符号

2）3 线-8 线集成译码器 74138

3 线-8 线译码器 74138 的逻辑符号和引脚图如图 3.4.10 所示，该译码器有三个二进制输入 A_2、A_1、A_0，它们共有 8 种组合状态，即可译出 8 个输出信号 $Y_7 \sim Y_0$，低电平有效。

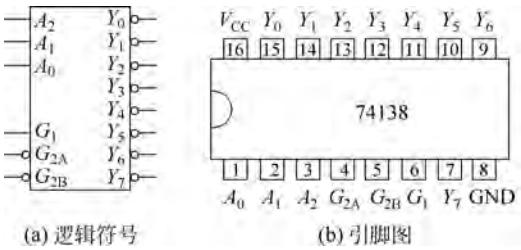


图 3.4.10　3 线-8 线译码器 74138 的逻辑符号和引脚图

此外，还设置了 G_1、G_{2A}、G_{2B} 三个控制端。当 $G_1=1$，$G_{2A}=G_{2B}=0$ 时，译码器处于工作状态，根据 $A_2A_1A_0$ 译成对应输出上的有效低电平信号；否则译码器不工作，所有的输出端被封锁在高电平。3 线-8 线译码器 74138 真值表如表 3.4.4 所示。

表 3.4.4　3 线-8 线译码器 74138 真值表

输入						输出							
G_1	G_{2A}	G_{2B}	A_2	A_1	A_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
×	1	×	×	×	×	1	1	1	1	1	1	1	1
×	×	1	×	×	×	1	1	1	1	1	1	1	1
0	×	×	×	×	×	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

由真值表可得出 3 线-8 线译码器 74138 各个输出的表达式：

$$\begin{cases} Y_0 = \overline{G_1\overline{G}_{2A}\overline{G}_{2B}\overline{A}_2\overline{A}_1\overline{A}_0} \\ Y_1 = \overline{G_1\overline{G}_{2A}\overline{G}_{2B}\overline{A}_2\overline{A}_1A_0} \\ Y_2 = \overline{G_1\overline{G}_{2A}\overline{G}_{2B}\overline{A}_2A_1\overline{A}_0} \\ Y_3 = \overline{G_1\overline{G}_{2A}\overline{G}_{2B}\overline{A}_2A_1A_0} \\ Y_4 = \overline{G_1\overline{G}_{2A}\overline{G}_{2B}A_2\overline{A}_1\overline{A}_0} \\ Y_5 = \overline{G_1\overline{G}_{2A}\overline{G}_{2B}A_2\overline{A}_1A_0} \\ Y_6 = \overline{G_1\overline{G}_{2A}\overline{G}_{2B}A_2A_1\overline{A}_0} \\ Y_7 = \overline{G_1\overline{G}_{2A}\overline{G}_{2B}A_2A_1A_0} \end{cases} \tag{3.4.3}$$

当 $G_1=1$，$G_{2A}=G_{2B}=0$ 时，式(3.4.3)可简化为

$$\begin{cases} Y_0=\overline{\overline{A}_2\overline{A}_1\overline{A}_0}=\overline{m}_0 \\ Y_1=\overline{\overline{A}_2\overline{A}_1A_0}=\overline{m}_1 \\ Y_2=\overline{\overline{A}_2A_1\overline{A}_0}=\overline{m}_2 \\ Y_3=\overline{\overline{A}_2A_1A_0}=\overline{m}_3 \\ Y_4=\overline{A_2\overline{A}_1\overline{A}_0}=\overline{m}_4 \\ Y_5=\overline{A_2\overline{A}_1A_0}=\overline{m}_5 \\ Y_6=\overline{A_2A_1\overline{A}_0}=\overline{m}_6 \\ Y_7=\overline{A_2A_1A_0}=\overline{m}_7 \end{cases} \tag{3.4.4}$$

由式(3.4.4)可知，$Y_0\sim Y_7$ 同时又是 A_2、A_1、A_0 这三个变量的全部最小项的译码输出，所以这种译码器也称为最小项译码器。

3. 应用

1) 扩展

利用译码器的控制端可以方便地扩展译码器容量。图 3.4.11 是将两片 74138 扩展为 4 线-16 线译码器。其中 74138(1)为低位片，74138(2)为高位片，将高位片的 G_1 与低位片的 G_{2A} 相连作为 A_3，将高位片的 G_{2A}、G_{2B} 与低位片的 G_{2B} 相连作为使能端 E。其工作原理：当 $E=1$ 时，两个译码器都禁止工作，输出全为 1；当 $E=0$ 时，译码器工作。这时，如果 $A_3=0$，高位片禁止，低位片工作，输出 $Y_0\sim Y_7$ 由输入二进制码 $A_2A_1A_0$ 决定；如果 $A_3=1$，低位片禁止，高位片工作，输出 $Y_8\sim Y_{15}$ 由输入二进制码 $A_2A_1A_0$ 决定。可见，这样可以实现 4 线-16 线译码器功能。

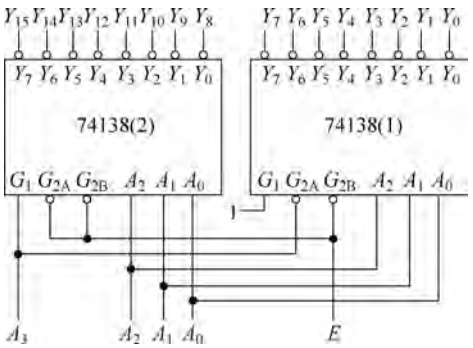


图 3.4.11　两片 74138 扩展为 4 线-16 线译码器

例 3.4.1　试用四片 74138 和一片 74139 构成 5 线-32 线译码器，输入为 5 位二进制码 $B_4B_3B_2B_1B_0$，对应输出 $L_0\sim L_{31}$ 为低电平有效信号。

解：列出 5 线-32 线译码器的真值表，如表 3.4.5 所示。

表 3.4.5　5 线-32 线译码器的真值表

输入					输出										
B_4	B_3	B_2	B_1	B_0	L_0	L_1	L_2	L_3	L_4	…	L_{27}	L_{28}	L_{29}	L_{30}	L_{31}
0	0	0	0	0	0	1	1	1	1	…	1	1	1	1	1
0	0	0	0	1	1	0	1	1	1	…	1	1	1	1	1
0	0	0	1	0	1	1	0	1	1	…	1	1	1	1	1
0	0	0	1	1	1	1	1	0	1	…	1	1	1	1	1
0	0	1	0	0	1	1	1	1	0	…	1	1	1	1	1
⋮					⋮										
1	1	0	1	1	1	1	1	1	1	…	0	1	1	1	1
1	1	1	0	0	1	1	1	1	1	…	1	0	1	1	1
1	1	1	0	1	1	1	1	1	1	…	1	1	0	1	1
1	1	1	1	0	1	1	1	1	1	…	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	…	1	1	1	1	0

从表 3.4.5 可以看出，当 $B_4B_3=00$，而 $B_2B_1B_0$ 从 000 变化到 111 时，对应 $L_0\sim L_7$ 中有一个输出为 0，其余输出全为 1，因此 4 片 74138 中，设置片(1)为译码状态，其余 3 片为禁止译码状态，对应的输出 $L_8\sim L_{31}$ 全为 1。

以此类推，当 $B_4B_3=01$，而 $B_2B_1B_0$ 从 000 变化到 111 时，对应 $L_8\sim L_{15}$ 中有一个输出为 0，其余输出全为 1，此时设置片(2)为译码状态。当 $B_4B_3=10$ 和 11 时，分别设置片(2)和片(3)为译码状态。因此，将 5 位二进制码的低三位 $B_2B_1B_0$ 分别与 4 片 74138 的三个地址输入端并接在一起。

高位 B_4B_3 有 4 种状态组合，因此接入 74139 的两个地址输入 A_1A_0，74139 的 4 个有效输出信号分别接入 4 片 74138 的低使能控制端，使 4 片 74138 在 B_4B_3 的控制下轮流工作在译码状态。这样就得到 5 线-32 线译码器，逻辑图如图 3.4.12 所示。

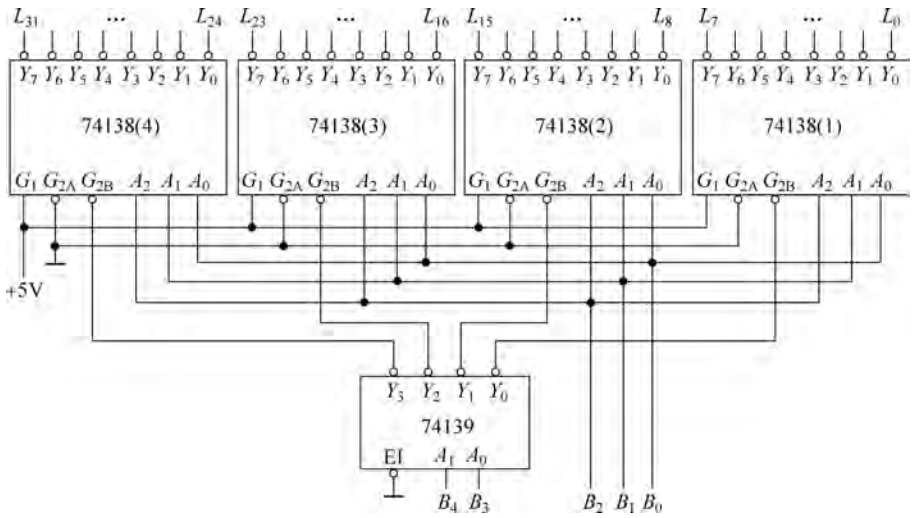


图 3.4.12　例 4.4.1 的逻辑图

思考：如果要构成 6 线-64 线译码器，需要几片 74138？

2）实现组合逻辑函数

由式(3.4.4)可以看出，当 $G_1=1$，$G_{2A}=G_{2B}=0$ 时，若将 A_2、A_1、A_0 作为 3 个输入逻辑变量，则 8 个输出端分别与 3 输入变量最小项相对应。利用附加的门电路将这些最小项适当地组合起来，就可以产生组合逻辑函数。

例 3.4.2 试用集成译码器 74138 和门电路实现组合逻辑函数 $L=AB+BC+AC$。

解：将逻辑函数转换成最小表达式，即

$$L=\bar{A}BC+A\bar{B}C+AB\bar{C}+ABC=m_3+m_5+m_6+m_7 \tag{3.4.5}$$

由图 3.4.9 和式(3.4.4)可知，只要令集成译码器 74138 的输入 $A_2=A$，$A_1=B$，$A_0=C$，则逻辑函数 L 的最小项表达式即可用编号的形式表示，如式(3.4.5)所示。由于译码器的输出以 $\bar{m}_i$ 形式给出，所以还需要把 L 变换为 $\bar{m}_i$ 的形式。

将最小项表达式转换成与非-与非形式，即

$$L=m_3+m_5+m_6+m_7=\overline{\bar{m}_3\cdot\bar{m}_5\cdot\bar{m}_6\cdot\bar{m}_7} \tag{3.4.6}$$

根据 74138 的功能，$Y_3=\bar{m}_3$，$Y_5=\bar{m}_5$，$Y_6=\bar{m}_6$，$Y_7=\bar{m}_7$，则有

$$L=\overline{Y_3\cdot Y_5\cdot Y_6\cdot Y_7} \tag{3.4.7}$$

用一片 74138 加一个与非门就可实现逻辑函数 L，逻辑图如图 3.4.13 所示。

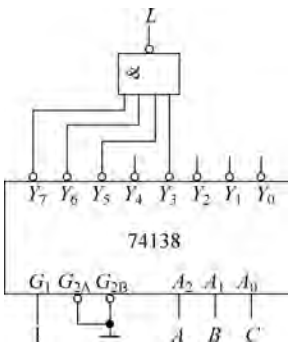


图 3.4.13 例 3.4.2 逻辑图

例 3.4.3 某组合逻辑电路的真值表如表 3.4.6 所示，试用译码器和门电路设计该逻辑电路。

表 3.4.6 例 3.4.3 的真值表

输入			输出		
A	B	C	L	F	G
0	0	0	0	0	1
0	0	1	1	0	0
0	1	0	1	0	1
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	0	1	0
1	1	0	0	1	1
1	1	1	1	0	0

解：根据真值表写出各输出的最小项表达式，即

$$\begin{cases} L=\overline{A}\overline{B}C+\overline{A}B\overline{C}+A\overline{B}\overline{C}+ABC=m_1+m_2+m_4+m_7 \\ F=\overline{A}BC+A\overline{B}C+AB\overline{C}=m_3+m_5+m_6 \\ G=\overline{A}\overline{B}\overline{C}+\overline{A}B\overline{C}+A\overline{B}\overline{C}+AB\overline{C}=m_0+m_2+m_4+m_6 \end{cases} \tag{3.4.8}$$

将最小项表达式转换成与非-与非形式，即

$$\begin{cases} L=\overline{\overline{m}_1\cdot\overline{m}_2\cdot\overline{m}_4\cdot\overline{m}_7} \\ F=\overline{\overline{m}_3\cdot\overline{m}_5\cdot\overline{m}_6} \\ G=\overline{\overline{m}_0\cdot\overline{m}_2\cdot\overline{m}_4\cdot\overline{m}_6} \end{cases} \tag{3.4.9}$$

设 $A=A_2$，$B=A_1$，$C=A_0$，将 L、F、G 的逻辑表达式与 74138 的输出表达式相比较，则有

$$\begin{cases} L=\overline{Y_1\cdot Y_2\cdot Y_4\cdot Y_7} \\ F=\overline{Y_3\cdot Y_5\cdot Y_6} \\ G=\overline{Y_0\cdot Y_2\cdot Y_4\cdot Y_6} \end{cases} \tag{3.4.10}$$

用一片 74138 加三个与非门就可实现该组合逻辑电路，逻辑图如图 3.4.14 所示。

3）数据分配器

数据分配是将公共数据线上的数据根据需要送到不同的通道，实现数据分配功能的逻辑电路称为数据分配器。它的作用相当于多个输出的单刀多掷开关，其示意图如图 3.4.15 所示。

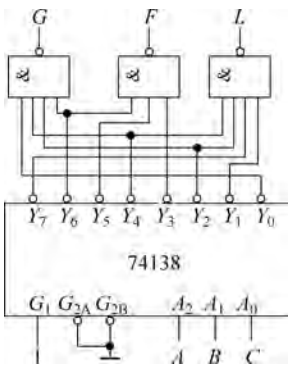


图 3.4.14　例 3.4.3 逻辑图

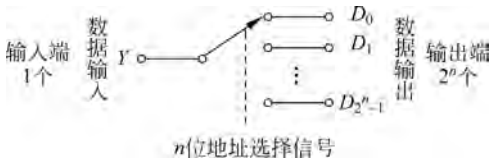


图 3.4.15　数据分配器示意图

数据分配器可以用译码器实现。例如，用 3 线-8 线译码器可以把 1 个数据信号分配到 8 个不同的通道。集成译码器 74138 作为数据分配器的真值表如表 3.4.7 所示。

如果将 G_{2A} 接低电平，G_1 作为使能端，A_2、A_1、A_0 作为选择通道地址输入，G_{2B} 作为数据输入，将会得到与输入相反的数据波形。

数据分配器的用途比较多，比如用它将一台个人计算机（PC）与多台外部设备连接，将计算机数据分送到外部设备中。它还可以与计数器结合组成脉冲分配器，与数据选择

器连接组成分时数据传送系统。

表 3.4.7　集成译码器 74138 作为数据分配器的真值表

输　入						输　出							
G_1	G_{2A}	G_{2B}	A_2	A_1	A_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	×	×	×	×	1	1	1	1	1	1	1	1
1	0	D	0	0	0	D	1	1	1	1	1	1	1
1	0	D	0	0	1	1	D	1	1	1	1	1	1
1	0	D	0	1	0	1	1	D	1	1	1	1	1
1	0	D	0	1	1	1	1	1	D	1	1	1	1
1	0	D	1	0	0	1	1	1	1	D	1	1	1
1	0	D	1	0	1	1	1	1	1	1	D	1	1
1	0	D	1	1	0	1	1	1	1	1	1	D	1
1	0	D	1	1	1	1	1	1	1	1	1	1	D

3.4.5　课中学习——二-十进制译码器 7442

第 1 章已经讨论过 8421 码，对应于 0～9 的十进制数，由 4 位二进制数 0000～1001 表示。由于人们不习惯直接识别二进制数，所以采用二-十进制译码器来解决。这种译码器应有 4 个输入端，10 个输出端。它的功能表如表 3.4.8 所示，其输出低电平有效。例如，当输入 8421 码 $A_3A_2A_1A_0=0010$ 时，输出 $Y_2=0$，它对应于十进制数 2，其余输出为高电平。当输入超过 8421 码的范围(1010～1111)时，输出均为高电平，即没有译码输出。

表 3.4.8　7442 二-十进制译码器功能表

数目	BCD 输入				输　出									
	A_3	A_2	A_1	A_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0
10	1	0	1	0	1	1	1	1	1	1	1	1	1	1
11	1	0	1	1	1	1	1	1	1	1	1	1	1	1
12	1	1	0	0	1	1	1	1	1	1	1	1	1	1
13	1	1	0	1	1	1	1	1	1	1	1	1	1	1
14	1	1	1	0	1	1	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1	1	1	1	1	1	1

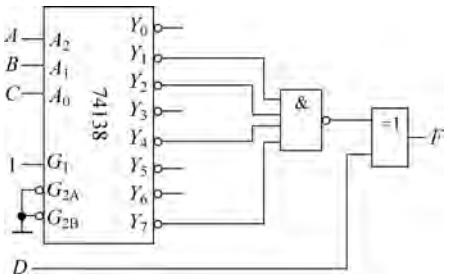
（三）课后篇

3.4.6 课后巩固——练习实践

1. 知识巩固练习

3.4.6.1 试用一片 3 线-8 线译码器 74138 和一个与非门设计一个 3 位数 $X_2X_1X_0$ 奇偶校验器。要求：当输入信号为偶数个 1(含 0 个 1)时，输出信号 F 为 1；否则，为 0。

3.4.6.2 题图 3.4.6.2 为 3 线-8 线译码器 74138 构成的电路，试分析电路的逻辑功能。要求：(1)列出输出表达式；(2)列出真值表；(3)说明逻辑功能。



题图 3.4.6.2

3.4.6.3 试用 3 线-8 线译码器 74138 和若干与非门设计一个 1 位全加器。

3.4.6.4 某雷达站有 A、B、C 三部雷达，其中雷达 A 和雷达 B 功率消耗相等，雷达 C 的功率是 A 的 2 倍。这些雷达由两台发电机 X 和 Y 供电，发电机 X 的最大输出功率等于雷达 A 的功率消耗，发电机 Y 的最大输出功率是 X 的 3 倍，要求用译码器 74138 和门电路设计一个逻辑电路，能够根据各雷达的启动和关闭信号，以最节约电能的方式启、停发电机。

3.4.6.5 已知函数 $F(D,C,B,A)=\sum m(2,5,7,8,10,12,15)$，试用 3 线 -8 线译码器 74138 和最少的门电路设计实现该逻辑函数的电路。

3.4.6.6 设计一个监视交通信号灯工作状态的逻辑电路。每一组信号灯由红、黄、绿三盏灯组成。正常工作情况下，任何时刻必有一盏灯点亮，而且只允许有一盏灯点亮。而当出现其他五种点亮状态时，电路发生故障，这时要求发出故障信号，以提醒维护人员前去修理。试用译码器 74138 及门电路设计满足上述控制要求的逻辑电路。要求：列出真值表，写出逻辑表达式，画出电路图。

3.4.6.7 某产品有 A、B、C、D 四项指标，至少有三项达标(必须包含 A)时才算合格，试用一片 3 线-8 线译码器 74138 和最少的门电路设计符合上述规定的逻辑电路。

2. 工程实践练习

3.4.6.8 利用口袋实验包完成数码管直观显示的 10 个按键显示系统电路的搭建与调试。要求：当某一个按键按下后，能够用数码管直观显示出该按键所对应的十进制数。

3.4.7 本节思维导图

知识拓展——OLED 屏国产先锋：京东方公司 **集成电路国产化进程**

OLED 因拥有 LCD 无法比拟的画质、柔性和设计优势而成为目前手机首选屏幕，而这一领域技术被韩国三星公司、LG 公司等掌握，国内只能拿到二流产品。直到京东方 OLED 屏被华为公司、VIVO 公司等采用，2020 年，京东方公司首入苹果公司供应商名单，实现我国 OLED 屏的国产替代和技术自主可控。京东方公司就是我国在 1956 年建成的北京电子管厂（代号国营 774 厂），曾是 20 世纪 60 年代亚洲最大的电子管厂。

3.5 数据选择器

（一）课前篇

本节导学单

1. 学习目标

根据《布鲁姆教育目标分类学》，从知识维度和认知过程两方面进一步细化本节课的教学目标，明确学习本节课所必备的学习经验。

知识维度	认知过程					
	1. 回忆	2. 理解	3. 应用	4. 分析	5. 评价	6. 创造
A. 事实性知识		说出数据选择器基本概念				
B. 概念性知识	回忆常用逻辑门电路的逻辑符号，逻辑表达式及真值表					
C. 程序性知识	回忆组合逻辑电路设计的一般步骤	画出集成数据选择器引脚、功能、真值表	应用门电路设计数据选择器	应用集成数据选择器实现组合逻辑函数，序列信号发生器以及多路数据分时传送	对比评价数据选择器实现组合逻辑函数和译码器实现组合逻辑函数的方法和步骤	利用集成芯片设计应用电路，会借助仿真软件进行验证，会利用口袋实验包进行电路搭建与调试
D. 元认知知识	明晰学习经验是理解新知识的前提		根据概念及学习经验迁移得到新理论的能力	将知识分为若干任务，主动思考，举一反三，完成每个任务	类比评价也是知识整合吸收的过程	通过电路的设计、仿真、搭建、调试，培养工程思维

2. 导学要求

根据本节的学习目标，将回忆、理解、应用层面学习目标所涉及的知识点课前自学完成，形成自学导学单。

3.5.1 课前自学——数据选择器的定义

在数字信号的传输过程中，有时需要从多路输入数据中选出某一路数据传送到公共数据线上，实现数据选择功能的逻辑电路称为数据选择器。它的作用相当于多个输入的单刀多掷开关，如图 3.5.1 所示。根据 n 位地址选择信号，从 2^n 个输入信号中选择一个输入信号送到输出。

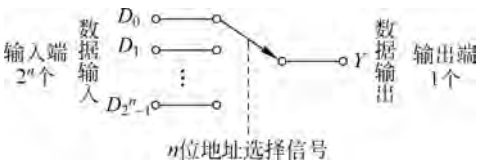


图 3.5.1 数据选择器示意图

下面以 4 选 1 数据选择器为例，说明数据选择器的工作原理及基本功能，其真值表如表 3.5.1 所示。

表 3.5.1 4 选 1 数据选择器真值表

输入							输出
G	A_1	A_0	D_3	D_2	D_1	D_0	Y
1	×	×	×	×	×	×	0
0	0	0	×	×	×	0	0
			×	×	×	1	1
	0	1	×	×	0	×	0
			×	×	1	×	1
	1	0	×	0	×	×	0
			×	1	×	×	1
	1	1	0	×	×	×	0
			1	×	×	×	1

为了对 4 个数据输入进行选择，使用 2 位地址码输入 A_1A_0，产生 4 个地址信号。任何时候 A_1A_0 只有一种可能的取值，使对应的那一路数据通过，送达 Y 端。使能输入 G 是低电平有效，当 $G=1$ 时，无论地址码是什么，输出 Y 总是等于 0；当 $G=0$ 时，根据地址码选择对应的数据输入送到 Y 端。

根据真值表可得出输出 Y 的表达式：

$$Y=(\overline{A}_1\overline{A}_0D_0+\overline{A}_1A_0D_1+A_1\overline{A}_0D_2+A_1A_0D_3)\cdot\overline{G} \tag{3.5.1}$$

根据式(3.5.1)可得到 4 选 1 数据选择器的逻辑图，如图 3.5.2 所示。

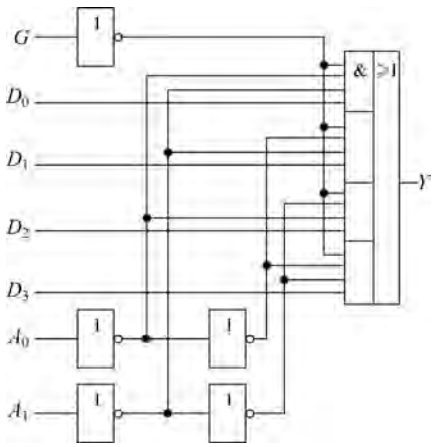


图 3.5.2　4 选 1 数据选择器的逻辑图

3.5.2　课前自学——预习自测

3.5.2.1　用十六进制数的方式写出 16 选 1 数据选择器的各地址码。

3.5.2.2　用 32 选 1 数据选择器选择数据，若选择的输入数据为 D_{20}、D_{17}、D_{18}、D_{27}、D_{31}，依次写出对应的地址码。

（二）课上篇

3.5.3　课中学习——集成数据选择器及其应用

1. 集成数据选择器

常见的集成数据选择器有双 4 选 1 数据选择器、8 选 1 数据选择器等。

1）双 4 选 1 数据选择器 74153、74253

74153、74253 的逻辑符号相同，如图 3.5.3 所示。

从图 3.5.3 中可以看出，一个芯片内部含有两个 4 选 1 数据选择器，所以称为双 4 选 1 数据选择器。两个数据选择器共用地址输入 A_1、A_0，使能信号 E、数据输入信号 D_0～D_3、数据输出 Y 相互独立，实现根据地址输入从 4 路数据输入中选择一路送到输出的功能。74153 的真值表如表 3.5.2 所示。

表 3.5.2　74153 的真值表

输　入			输　出
使能 E	地址选择		Y
	A_1	A_0	
1	×	×	0
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3

74253 与 74153 的区别：当 $E=1$ 时，74153 输出 0，而 74253 输出为高阻状态，因此 74253 具有三态输出功能。利用这一特点，可以将多个芯片的输出端连在一起，共用一根数据传输线。

2）8 选 1 数据选择器 74151、74251

74151、74251 的逻辑符号相同，如图 3.5.4 所示。

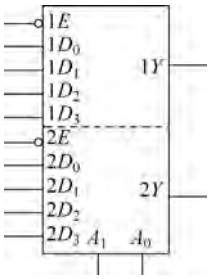


图 3.5.3　74153/74253 的逻辑符号

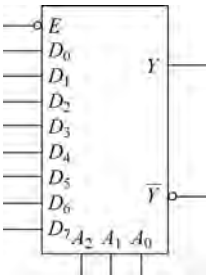


图 3.5.4　74151/74251 的逻辑符号

从图 3.5.4 中可以看出，它有 3 个地址输入端 A_2、A_1、A_0，可选择 $D_0 \sim D_7$ 共 8 路数据，具有两个互补输出端 Y 和 $\overline{Y}$，使能输入端 E 低电平有效。74151 的真值表如表 3.5.3 所示。

表 3.5.3　74151 的真值表

输　入				输　出	
使能 E	地址选择			Y	$\overline{Y}$
	A_2	A_1	A_0		
1	×	×	×	0	1
0	0	0	0	D_0	$\overline{D}_0$
0	0	0	1	D_1	$\overline{D}_1$
0	0	1	0	D_2	$\overline{D}_2$
0	0	1	1	D_3	$\overline{D}_3$
0	1	0	0	D_4	$\overline{D}_4$
0	1	0	1	D_5	$\overline{D}_5$
0	1	1	0	D_6	$\overline{D}_6$
0	1	1	1	D_7	$\overline{D}_7$

74151 输出 Y 的表达式为

$$Y=\sum_{i=0}^{7} m_i D_i \tag{3.5.2}$$

74251 与 74151 的区别：当 $E=1$ 时，74151 输出 0，而 74251 输出为高阻状态，因此 74251 具有三态输出功能。利用这一特点，可以将多个芯片输出端连在一起，共用一根数据传输线。

2. 应用

1) 扩展

利用数据选择器的控制端可以方便地实现数据选择器的通道扩展。图 3.5.5 是将两片 74151 和 3 个门电路扩展为 16 选 1 数据选择器。其中 74151(1)为低位片，74151(2)为高位片，将低位片 74151(1)的使能控制端 E 经一个非门门反相后与高位片 74151(2)的 E 相连，E 作为最高位的地址选择信号 A_3。若 $A_3=0$，则 74151(1)工作，根据 $A_3\sim A_0$ 从 $D_0\sim D_7$ 中选择一路输出；若 $A_3=1$，则 74151(2)工作，根据 $A_3\sim A_0$ 从 $D_8\sim D_{15}$ 中选择一路输出。因此，该电路实现了 16 选 1 功能。

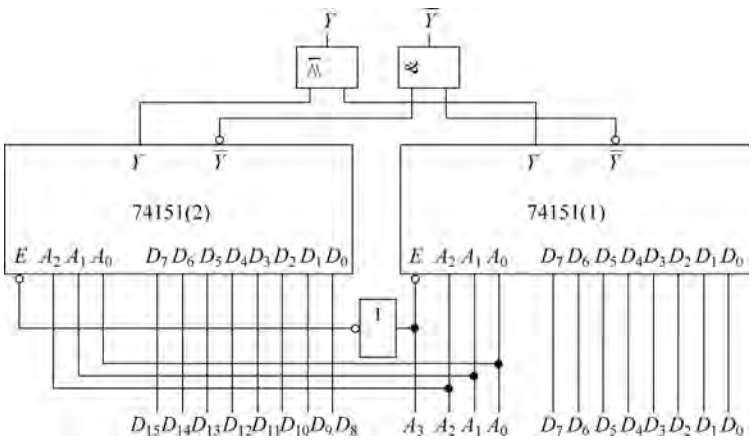


图 3.5.5　两片 74151 扩展为 16 选 1 数据选择器

2) 构成组合逻辑函数

根据 8 选 1 数据选择器输出与输入的关系式(3.5.2)，有

$$Y=\sum_{i=0}^{7}m_iD_i$$

式中：m_i 为地址选择输入端 A_2、A_1、A_0 构成的最小项。

数据输入作为控制信号，当 $D_i=1$ 时，其对应的最小项 m_i 在表达式中出现；当 $D_i=0$ 时，其对应的最小项就不出现。利用这一点将函数变换成最小项表达式，函数的变量接入地址选择输入端，就可以实现组合逻辑函数。

当逻辑函数的变量个数和数据选择器的地址输入变量个数相同时，可直接用数据选择器来实现逻辑函数。

例 3.5.1　试用 8 选 1 数据选择器 74151 实现逻辑函数 $L=AB+BC+AC$。

解：(1) 将逻辑函数转换成最小项表达式：

$$L=AB+BC+AC=\bar{A}BC+A\bar{B}C+AB\bar{C}+ABC=m_3+m_5+m_6+m_7$$

(2) 将输入变量接至数据选择器的地址输入端，即 $A=A_2$，$B=A_1$，$C=A_0$。输出变量接至数据选择器的输出端，即 $L=Y$。将逻辑函数 L 的最小项表达式与 74151 的功能表相比较，显然 L 式中出现的最小项对应的数据输入端接 1，L 式中没有出现的最小项

对应的数据输入端接 0，即 $D_3=D_5=D_6=D_7=1$，$D_0=D_1=D_2=D_4=0$。

（3）画出逻辑电路图，如图 3.5.6 所示。

当逻辑函数的变量个数大于数据选择器的地址输入变量个数时，不能用前述的简单办法，这时应分离出多余的变量，把它们加到适当的数据输入端。

例 3.5.2 试用 4 选 1 数据选择器实现逻辑函数 $L=AB+BC+AC$。

解：方法一。（1）由于函数 L 有三个输入信号 A、B、C，而 4 选 1 数据选择器仅有两个地址输入端 A_1 和 A_0，所以选 A、B 接到地址输入端，且 $A=A_1$，$B=A_0$，将逻辑函数转换成最小项表达式：

$$\begin{aligned}L&=AB+BC+AC=\overline{A}BC+A\overline{B}C+AB\overline{C}+ABC\\&=\overline{A}_1A_0C+A_1\overline{A}_0C+A_1A_0(C+\overline{C})\\&=m_0\cdot 0+m_1\cdot C+m_2\cdot C+m_3\cdot 1\end{aligned}$$

（2）将逻辑函数 L 的最小项表达式与 4 选 1 数据选择器的表达式进行对照，确定出数据输入端：$D_0=0$，$D_1=C$，$D_2=C$，$D_3=1$。

（3）画出逻辑电路图，如图 3.5.7 所示。

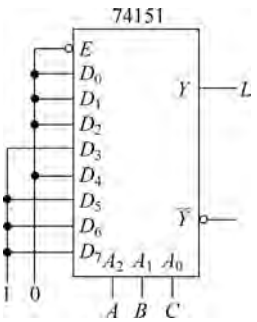


图 3.5.6 例 3.5.1 的逻辑电路图

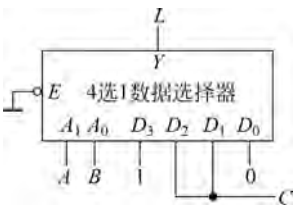


图 3.5.7 例 3.5.2 的逻辑电路图

方法二。（1）由于函数 L 有三个输入信号 A、B、C，而 4 选 1 数据选择器仅有两个地址输入端 A_1 和 A_0，所以选 A、B 接到地址输入端，且 $A=A_1$，$B=A_0$。

（2）将 C 加到适当的数据输入端。做出逻辑函数 L 的真值表，如表 3.5.4 所示。

表 3.5.4 例 3.5.2 真值表

A	B	C	L
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

当 $A=0,B=0$ 时(表中第一、二行),无论 C 取何值,L 都为 0,所以 $D_0=0$;当 $A=0,B=1$ 时(表中第三、四行),L 的取值与 C 相同,所以 $D_1=C$;当 $A=1,B=0$ 时(表中第五、六行),L 的取值与 C 相同,所以 $D_2=C$;当 $A=1,B=1$ 时(表中第七、八行),无论 C 取何值,L 都为 1,所以 $D_3=1$。

(3) 画出逻辑电路图,如图 3.5.7 所示。

3) 实现并行数据到串行数据的转换

图 3.5.8 为由 8 选 1 数据选择器构成的并/串行转换的电路图。数据选择器地址输入端 A_2、A_1、A_0 从 000 到 111 依次变化,则选择器的输出 Y 随之接通 $D_0,D_1,D_2,\cdots,D_7$。当选择器的数据输入端 $D_0\sim D_7$ 与一个并行 8 位数 11011001 相连时,输出端得到的数据依次为 1-1-0-1-1-0-0-1,即串行数据输出。

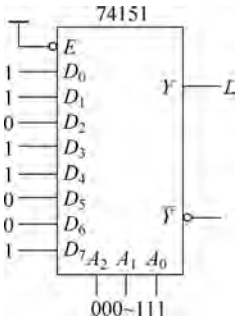


图 3.5.8 数据并行输入转换成串行输出

4) 多路数据的分时传送

数据选择器、数据分配器和传输总线连接构成的信号传输系统示意图如图 3.5.9 所示。该系统将多个数据信号中的一路信号连接到总线,经过远距离传送后,由数据分配器再将总线上的数据分配到被选中的多个目的地之一。这种信息传输的基本原理在通信系统、计算机网络系统以及计算机内部各功能部件之间的信息传送等都有广泛的应用。

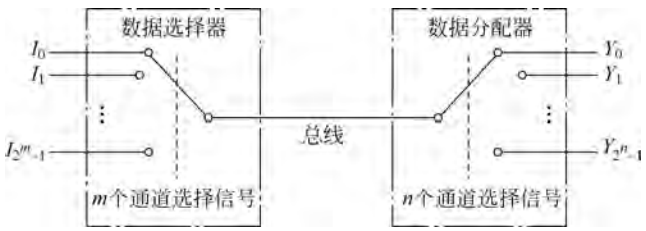


图 3.5.9 多路数据的分时传送

(三) 课后篇

3.5.4 课后巩固——练习实践

1. 知识巩固练习

3.5.4.1 试用 4 选 1 数据选择器分别实现下列逻辑函数:

(1) $L(A,B)=\sum m(0,1,3)$

(2) $L(A,B,C)=\sum m(0,1,5,7)$

(3) $L=A\overline{B}C+\overline{A}(\overline{B}+\overline{C})$

讲解视频

讲解视频

3.5.4.2　试用 8 选 1 数据选择器实现下列逻辑函数：

(1) $L(A,B,C)=\sum m(0,1,4,5,7)$　(2) $L(A,B,C,D)=\sum m(0,3,5,8,13,15)$

讲解视频

3.5.4.3　某保密锁的逻辑控制电路如题图 3.5.4.3 所示，图中的 A、B、C 为三个按钮，按下为 1；D 代表钥匙输入信号，插入为 1；F 代表开锁信号，Z 代表报警信号，均为 1 有效。试分析电路，列出真值表，写出 F、Z 的表达式并解出开锁信号的密码，指出何种情况下会报警。

讲解视频

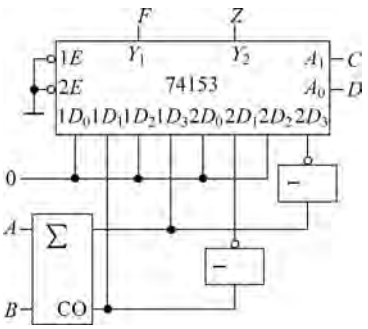
3.5.4.4　试用 2 片 8 选 1 数据选择器 74151 扩展成 16 选 1 数据选择器，在 4 位地址输入选通下，产生一序列信号 0100101110011011。

讲解视频

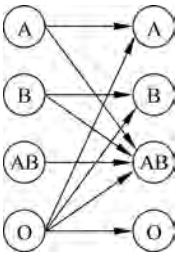
3.5.4.5　设计一个监视交通信号灯工作状态的逻辑电路。每组信号灯由红、黄、绿三盏灯组成。正常工作情况下，任何时刻必有一盏灯点亮，而且只允许有一盏灯点亮。而当出现其他五种点亮状态时，电路发生故障，这时要求发出故障信号，以提醒维护人员前去修理。试用 8 选 1 数据选择器 74151 及门电路设计满足上述控制要求的逻辑电路。要求：列出真值表，写出逻辑表达式，画出电路图。

讲解视频

3.5.4.6　人有 A、B、AB、O 四种血型，输血时输血者血型与受血者血型必须符合题图 3.5.4.6 中用箭头指示的授受关系。试用数据选择器设计一个逻辑电路，判断输血者与受血者的血型是否符合上述规定。



题图 3.5.4.3



题图 3.5.4.6

讲解视频

3.5.4.7　试用 74253 扩展为 8 选 1 数据选择器并实现逻辑函数 $F=AB+B\overline{C}+\overline{A}C$。

讲解视频

3.5.4.8　试用 8 选 1 数据选择器设计一个 4 位奇偶校验判断电路，当输入为奇数个 1 时输出为 1，否则输出为 0。

讲解视频

3.5.4.9　设计一个多功能组合逻辑电路，M_1、M_0 为功能选择输入信号，a、b 为逻辑变量，F 为电路的输出。当 M_1、M_0 取不同的数值时，电路具有题表 3.5.4.9 所示的逻辑功能。试用 8 选 1 数据选择器 74151 和最少的门实现此电路。

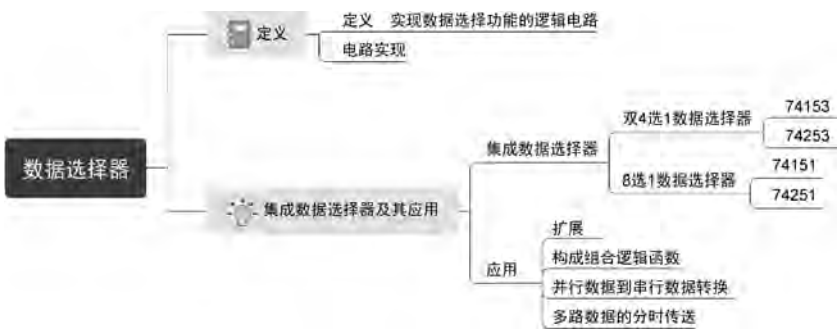
题表 3.5.4.9　题 3.5.4.9 电路的功能表

M_1	M_0	Y
0	0	a
0	1	$a\oplus b$
1	0	ab
1	1	$a+b$

2. 工程实践练习

3.5.4.10 利用口袋实验包完成多路复用显示器电路的搭建与调试，要求用一个显示译码器驱动两个数码管分时显示。

3.5.5 本节思维导图



知识拓展——国内 MCU 芯片企业：兆易创新公司 **集成电路国产化进程**

兆易创新公司成立于 2005 年，是一家立足于 IC 解决方案和存储技术的芯片设计公司，随着多年的产品线不断拓宽与技术并购，目前已成为“存储＋MCU＋传感器”三位一体的国内领先的半导体解决方案供应商。在 MCU 芯片技术领域，4 位、8 位、16 位等低端单片机芯片国内自给率较高，而 32 位以上中高端 MCU 市场则主要被 Microchip、ST、Renesas、NXP 等国外大厂垄断。兆易创新公司率先在 32 位 MCU 产品领域布局，在国内 32 位 MCU 的发展历史上创造了四个“第一颗”：2013 年 4 月，发布第一颗国产 Cortex-M3 32 位 MCU；2016 年 6 月，发布第一颗国产 Cortex-M4 32 位 MCU；2018 年 10 月，发布第一颗 Cortex-M23 32 位 MCU；2019 年 8 月，发布第一颗国产 RSIC-V 32 位 MCU。实现了中高端单片机产品的国产替代，主打产品已在汽车工业类、工控类、消费类的产品性能指标上达到国际大厂水平。

3.6 加法器

（一）课前篇

本节导学单

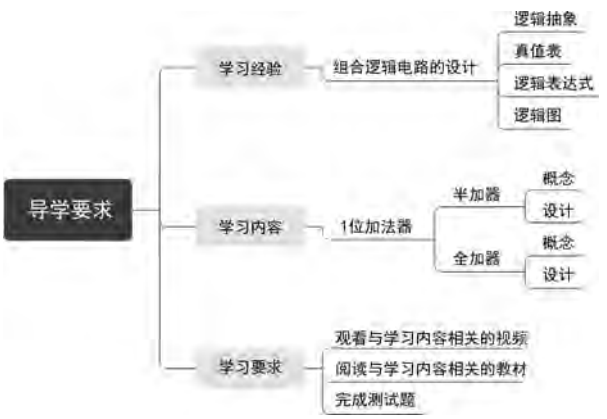
1. 学习目标

根据《布鲁姆教育目标分类学》，从知识维度和认知过程两方面进一步细化本节课的教学目标，明确学习本节课所必备的学习经验。

知识维度	认知过程					
	1. 回忆	2. 理解	3. 应用	4. 分析	5. 评价	6. 创造
A. 事实性知识		说出半加器、全加器的基本概念				
B. 概念性知识	回忆常用逻辑门电路的逻辑符号，逻辑表达式及真值表	阐释超前进位基本原理				
C. 程序性知识	回忆组合逻辑电路设计的一般步骤	画出集成加法器引脚功能、真值表	应用门电路设计半加器、全加器，串行进位加法器。应用集成加法器实现代码转换	分析串行进位加法器的不足，设计超前进位加法器	对比评价串行进位加法器和超前进位加法器	利用集成芯片设计应用电路，会借助仿真软件进行验证，会利用口袋实验包进行电路搭建与调试
D. 元认知知识	明晰学习经验是理解新知识的前提		根据概念及学习经验迁移得到新理论的能力	在不断地发现问题、分析问题、解决问题的过程中体会精益求精的科学态度	类比评价也是知识整合吸收的过程	通过电路的设计、仿真、搭建、调试，培养工程思维

2. 导学要求

根据本节的学习目标，将回忆、理解、应用层面学习目标所涉及的知识点课前自学完成，形成自学导学单。



3.6.1 课前自学——1位加法器

1. 半加器

如果不考虑有来自低位的进位将两个1位二进制数相加，称为半加。实现半加运算的电路称为半加器。

按照二进制加法运算规则可以列出如表3.6.1所示的半加器的真值表。表中，A、B是两个加数，S是相加的和，CO是向高位的进位。

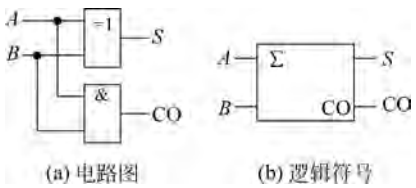
表3.6.1 半加器的真值表

输入		输出	
A	B	S	CO
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

将S、CO和A、B的关系写成逻辑表达式，即

$$\begin{cases} S=\overline{A}B+A\overline{B}=A\oplus B \\ \mathrm{CO}=AB \end{cases} \tag{3.6.1}$$

因此，半加器是由一个异或门和一个与门组成，如图3.6.1所示。



(a) 电路图　　(b) 逻辑符号

图3.6.1 半加器

2. 全加器

将两个多位二进制数相加时，除了最低位以外，每一位都应该考虑来自低位的进位，因此将加数、被加数和来自低位进位3个数相加，这种运算称为全加。实现全加运算的电路称为全加器。

根据二进制加法运算规则可列出1位全加器的真值表，如表3.6.2所示。

表 3.6.2　全加器的真值表

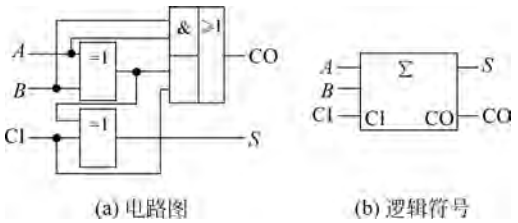
输　入			输　出	
A	B	CI	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

根据真值表直接写出 S 和 CO 的逻辑表达式，再经代数法化简得

$$\begin{aligned} S &= \overline{A}\,\overline{B}\mathrm{CI} + \overline{A}B\overline{\mathrm{CI}} + A\overline{B}\,\overline{\mathrm{CI}} + AB\mathrm{CI} \\ &= \overline{A}(B \oplus \mathrm{CI}) + A\overline{(B \oplus \mathrm{CI})} \\ &= A \oplus B \oplus \mathrm{CI} \end{aligned} \tag{3.6.2}$$

$$\begin{aligned} \mathrm{CI} &= \overline{A}B\mathrm{CI} + A\overline{B}\mathrm{CI} + AB\overline{\mathrm{CI}} + AB\mathrm{CI} \\ &= \mathrm{CI}(A \oplus B) + AB \end{aligned} \tag{3.6.3}$$

根据式(3.6.2)和式(3.6.3)画出全加器的逻辑电路图如图 3.6.2(a)所示，全加器的逻辑符号如图 3.6.2(b)所示。



(a) 电路图　　(b) 逻辑符号

图 3.6.2　全加器

3.6.2　课前自学——预习自测

3.6.2.1　试用半加器设计全加器电路。

3.6.2.2　试用 8 选 1 数据选择器 74151 实现 1 位二进制全减器*。

* 将两个多位二进制数相减时，除了最低位以外，每一位都应该考虑来自低位的借位，因此将对应位的减数、被减数和来自低位借位 3 个数相减，这种运算称为全减，所用的电路称为全减器。

（二）课上篇

3.6.3 课中学习——多位加法器

1. 串行进位加法器

半加器、全加器只能实现1位二进制数相加，若有多位数相加如何实现？比如，有2个4位二进制数 $A_3A_2A_1A_0$ 和 $B_3B_2B_1B_0$ 相加，可以采用4个全加器构成4位数加法器，如图3.6.3所示。将低位的进位输出作为高位的进位输入。因此，任何一位的加法运算都必须等到第一位的运算完成之后才能进行，将这种进位方式称为串行进位。这种加法器的逻辑电路简单，n 位二进制数相加只需要 n 个全加器即可。

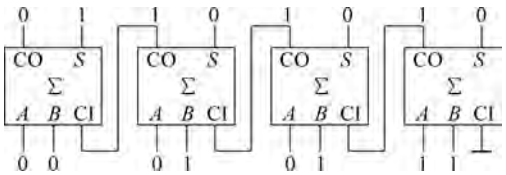


图3.6.3 4位串行进位加法器

串行进位加法器中每个全加器的进位输出连接到下一个高一级的全加器的输入（一级对应一个全加器），任何一级的输出及进位必须在上一级的进位到来后才能产生。这将引起加法过程中出现时间延迟，如图3.6.3所示。每个全加器进位传输延迟是指在假设输入 A 和 B 已经送入情况下从输入进位加到输出进位产生之间的时间。

2. 超前进位加法器

为了提高运算速度，必须设法减小或消除由于进位信号逐级传递所耗费的时间。那么高位的进位输入信号能否在相加运算开始时就知道？加到第 i 位的进位输入信号是这两个加数第 i 位以前各位状态的函数，所以第 i 位的进位输入信号 $(CI)_i$ 一定能由 $A_{i-1}A_{i-2}\cdots A_0$ 以及 $B_{i-1}B_{i-2}\cdots B_0$ 唯一确定。根据这个原理，就可以通过逻辑电路事先得出每一位全加器的进位输入信号，而无须再从最低位开始向高位逐位传递进位信号，这就有效地提高了运算速度。采用这种结构形式的加法器称为超前进位加法器。

下面具体分析超前进位信号的产生原理。从全加器的真值表3.6.2可以看出，以下两种情况下会产生进位输出信号：①A 和 B 都为1，也就是 $AB=1$ 时，此时 $CO=1$；②A 或 B 有一个为1且 $CI=1$，也就是 $A+B=1$ 且 $CI=1$，此时 $CO=1$。综合两种情况，两个多位数中第 i 位相加产生的进位输出 $(CO)_i$ 可表达为

$$(CO)_i = A_iB_i + (A_i + B_i)(CI)_i \tag{3.6.4}$$

若将 A_iB_i 定义为进位生成函数 G_i，同时将 A_i+B_i 定义为进位传递函数 P_i，则式(3.6.4)可改写为

$$(\mathrm{CO})_i = G_i + P_i(\mathrm{CI})_i \tag{3.6.5}$$

将上式展开后，得到

$$\begin{aligned}(\mathrm{CO})_i &= G_i + P_i(\mathrm{CI})_i \\ &= G_i + P_i[G_{i-1} + P_{i-1}(\mathrm{CI})_{i-1}] \\ &= G_i + P_iG_{i-1} + P_iP_{i-1}[G_{i-2} + P_{i-2}(\mathrm{CI})_{i-2}] \\ &\vdots \\ &= G_i + P_iG_{i-1} + P_iP_{i-1}G_{i-2} + \cdots + P_iP_{i-1}\cdots P_1G_0\end{aligned} \tag{3.6.6}$$

由式(3.6.6)可知，因为进位信号只与变量 G_i、P_i 和 C_0 有关，而 C_0 是向最低位的进位信号，其值是 0，所以各位的进位信号都只与两个加数有关，它们是可以并行产生的。

3.6.4 课中学习——集成加法器及其应用

1. 集成超前进位加法器 74283

集成超前进位加法器 74283 逻辑符号和芯片引脚图如图 3.6.4 所示。

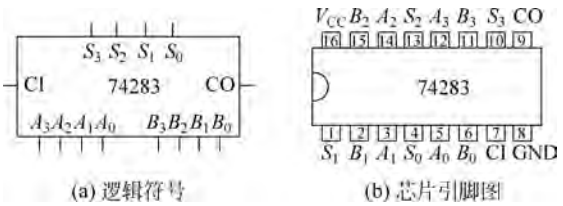


图 3.6.4　4 位超前进位加法器 74283

2. 应用

1) 扩展

用两片 74283 构成一个 8 位二进制数加法器，如图 3.6.5 所示。该电路的级联是串行进位方式，低位片 74283(1)的进位输出连到高位片 74283(2)的进位输入。

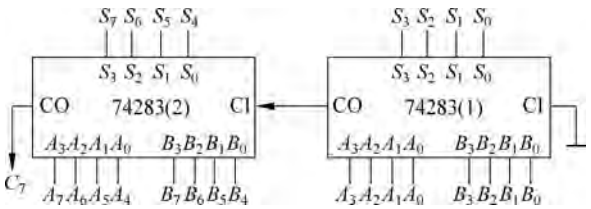


图 3.6.5　加法器串行进位扩展

2）代码变换

加法器除了构成加法运算电路外，还可以实现代码变换。

例 3.6.1 试设计一个将 8421 码转换为余三码的逻辑电路。

解：余三码作为输出，用 $F_3F_2F_1F_0$ 表示；8421 码作为输入，用 $X_3X_2X_1X_0$ 表示。由第 1 章有关内容可知，8421 码与余三码之间算术表达式可写为

$$F_3F_2F_1F_0 = X_3X_2X_1X_0 + 0011 \tag{3.6.7}$$

由于输出与输入仅差一个常数，用加法器实现设计最简单。将 8421 码连接到 74283 的一组输入端，另一组输入端接二进制数 0011，输出即为余三码，电路如图 3.6.6 所示。

例 3.6.2 由 4 位二进制数加法器 74283 构成的逻辑电路如图 3.6.7 所示，试分析电路逻辑功能。

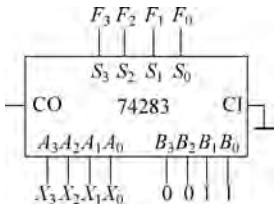


图 3.6.6 74283 实现代码变换电路

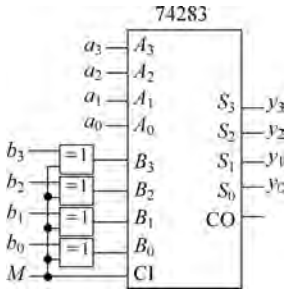


图 3.6.7 例 3.6.2 的逻辑电路图

解：该电路的突破口在 M，将 M 分两种情况分析：

（1）当 $M=0$ 时，有

$$B_i = b_i \oplus 0 = b_i \tag{3.6.8}$$

$$y_3y_2y_1y_0 = a_3a_2a_1a_0 + b_3b_2b_1b_0 \tag{3.6.9}$$

可以看出，当 $M=0$ 时，电路实现加法运算。

（2）当 $M=1$ 时，有

$$B_i = b_i \oplus 1 = \overline{b}_i \tag{3.6.10}$$

$$y_3y_2y_1y_0 = a_3a_2a_1a_0 + \overline{b}_3\overline{b}_2\overline{b}_1\overline{b}_0 + 1 \tag{3.6.11}$$

可以看出，当 $M=1$ 时，电路实现减法运算。

因此，该电路实现了 4 位二进制数的加/减法运算。

（三）课后篇

3.6.5 课后巩固——练习实践

1. 知识巩固练习

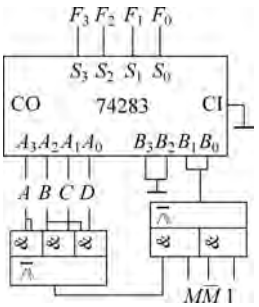
3.6.5.1 题图 3.6.5.1 为超前进位加法器 74283 构成的代码转换器，若该代码转

讲解视频

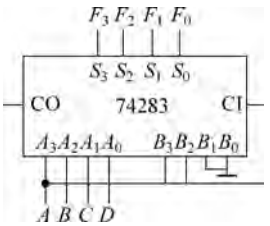
换器的输入 $ABCD$ 为 8421 码，求其在 M 控制下输出何种代码，要求列出真值表。

讲解视频

3.6.5.2　题图 3.6.5.2 为加法器 74283 构成的代码转换器，已知输入为 2421 码，试分析输出为何代码，并设计出输入与输出交换的电路。



题图 3.6.5.1



题图 3.6.5.2

3.6.5.3　用一片 74283 将余 3 码转换为 8421 码。

2. 工程实践练习

3.6.5.4　利用口袋实验包完成 6 个投票位的简易投票系统电路的搭建与调试。要求系统具有投票功能(可以做出二选一决定)、统计功能(统计选票数目)及显示(直观显示统计结果)功能。

3.6.6　本节思维导图

知识拓展——我国巨型计算机赶超史

中国集成电路发展历程

国防科技大学的慈云桂教授经历了我国计算机从电子管、晶体管到集成电路的更新换代。他带领的研究团队在 1958 年 9 月研制出代号为“901”的我国最早电子管专用计算机。当国产晶体管刚研制出来，性能很不稳定时，慈教授团队没有抱怨，而是利用巧妙的电路设计弥补了晶体管的不足，在 1964 年研制出全国产化晶体管计算机。20 世纪 70 年代国际计算机飞速发展之时，国内研制巨型机尚不具备科研条件，但慈教授团队没有停下步伐，继续埋头苦干，1977 年研制出我国第一台百万次集成电路计算机“151-3”，1983 年

研制出我国第一台一亿次“银河”巨型计算机，2009 年研制出我国首台千万亿次计算机“天河一号”，2013 年研制成功“天河二号”，首夺世界第一。实现我国巨型机从无到有，从弱到强的历史性转变。

3.7 数值比较器

（一）课前篇

本节导学单

1. 学习目标

根据《布鲁姆教育目标分类学》，从知识维度和认知过程两方面进一步细化本节课的教学目标，明确学习本节课所必备的学习经验。

知识维度	认知过程					
	1. 回忆	2. 理解	3. 应用	4. 分析	5. 评价	6. 创造
A. 事实性知识		说出数值比较器基本概念				
B. 概念性知识	回忆常用逻辑门电路的逻辑符号，逻辑表达式及真值表					
C. 程序性知识	回忆组合逻辑电路设计的一般步骤	画出集成数值比较器符号图、真值表	应用门电路设计 1 位、2 位数值比较器。应用集成数值比较器实现扩展	根据设计需求应用集成数值比较器设计电路		利用集成芯片设计应用电路，会借助仿真软件进行验证，会利用口袋实验包进行电路搭建与调试
D. 元认知知识	明晰学习经验是理解新知识的前提		根据概念及学习经验迁移得到新理论的能力	将知识分为若干任务，主动思考，举一反三，完成每个任务		通过电路的设计、仿真、搭建、调试，培养工程思维

2. 导学要求

根据本节的学习目标，将回忆、理解、应用层面学习目标所涉及的知识点课前自学完成，形成自学导学单。

3.7.1 课前自学——1 位数值比较器

在一些数字系统中(如数字计算机)经常要求比较两个数字的大小，为完成这一功能所设计的各种逻辑电路统称为数值比较器。数值比较器就是对两个二进制数 A、B 进行比较的逻辑电路，比较结果有 $A>B$、$A<B$ 以及 $A=B$ 三种情况。

1 位数值比较器是多位数值比较器的基础。当 A 和 B 都是 1 位数时，它们只能取 0 或 1 两种值，由此可写出 1 位数值比较器的真值表，如表 3.7.1 所示。

表 3.7.1 1 位数值比较器的真值表

输入		输出		
A	B	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

由真值表得到如下逻辑表达式：

$$\begin{cases} F_{A>B}=A\overline{B} \\ F_{A<B}=\overline{A}B \\ F_{A=B}=\overline{A}\overline{B}+AB \end{cases} \tag{3.7.1}$$

根据表达式可以画出 1 位数值比较器的逻辑电路，如图 3.7.1 所示。

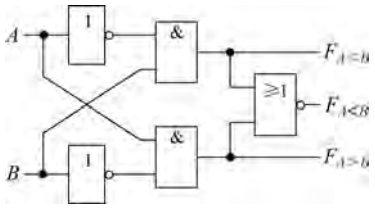


图 3.7.1　1 位数值比较器的逻辑电路

3.7.2　课前自学——2 位数值比较器

现在分析两位数字 A_1A_0 和 B_1B_0 的情况，用 $F_{A>B}$、$F_{A<B}$ 和 $F_{A=B}$ 表示比较结果。当高位(A_1、B_1)不相等时，无须比较低位(A_0、B_0)，两个数的比较结果就是高位比较的结果。当高位相等时，两数的比较结果由低位比较的结果决定。利用 1 位数值的比较结果，可以写出简化的真值表，如表 3.7.2 所示。

表 3.7.2　2 位数值比较器的真值表

输　入		输　出		
A_1　B_1	A_0　B_0	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
$A_1>B_1$	×	1	0	0
$A_1<B_1$	×	0	1	0
$A_1=B_1$	$A_0>B_0$	1	0	0
$A_1=B_1$	$A_0<B_0$	0	1	0
$A_1=B_1$	$A_0=B_0$	0	0	1

由表 3.7.2 可以写出如下逻辑表达式：

$$\begin{cases} F_{A>B}=F_{A_1>B_1}+F_{A_1=B_1}F_{A_0>B_0} \\ F_{A<B}=F_{A_1<B_1}+F_{A_1=B_1}F_{A_0<B_0} \\ F_{A=B}=F_{A_1=B_1}F_{A_0=B_0} \end{cases} \tag{3.7.2}$$

根据式(3.7.2)可画出 2 位数值比较器的逻辑电路，如图 3.7.2 所示。电路由 1 位数值比较器和门电路构成。根据的原则是：如果高位不相等，则高位的比较结果就是两数的比较结果，与低位无关。这时，高位输出 $F_{A_1=B_1}=0$，使与门 G_1、G_2、G_3 均封锁，而或门都打开，低位比较结果不能影响或门，高位比较结果则从或门直接输出。如果高位相等，即 $F_{A_1=B_1}=1$，使与门 G_1、G_2、G_3 均打开，同时由于 $F_{A_1>B_1}=0$ 和 $F_{A_1<B_1}=0$ 作用，或门也打开，低位的比较结果直接送达输出端，即低位的比较结果决定两数谁大、谁小或者相等。

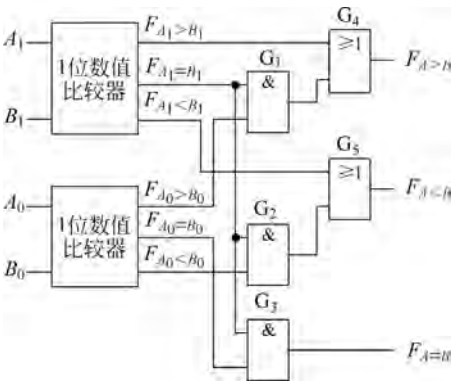


图 3.7.2　2 位数值比较器的逻辑电路

3.7.3　课前自学——预习自测

3.7.3.1　试用门电路设计 2 位二进制数的数值比较器。

（二）课上篇

3.7.4　课中学习——集成数值比较器及其位数扩展

1. 集成数值比较器 7485

7485 是 4 位集成数值比较器，其逻辑符号如图 3.7.3 所示。输入端包括 $A_3 \sim A_0$ 和 $B_3 \sim B_0$，输出端为 $F_{A>B}$、$F_{A<B}$、$F_{A=B}$，以及级联输入端 $I_{A>B}$、$I_{A<B}$、$I_{A=B}$，级联输入端便于与其他数值比较器输出相连，以便组成更多位数的数值比较器。

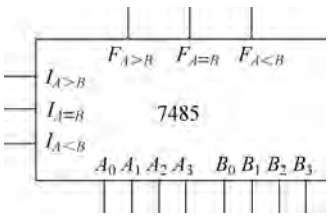


图 3.7.3　集成数值比较器 7485

4 位数值比较器 7485 的真值表如表 3.7.3 所示。4 位数值比较器的比较原理和 2 位数值比较器的比较原理相同，都是从最高位 A_3 和 B_3 开始比较，若不相等，则该位的比较结果就是两数的比较结果。若最高位 $A_3=B_3$，则再比较次高位 A_2 和 B_2。以此类推。显然，若两数相等，则必须将比较进行到最低位才能得到结果。若 4 位数相等，最终的结果由级联输入决定。从表 3.7.3 可以看出，7485 的三个级联输入中 $I_{A=B}$ 的级别最高。

表 3.7.3　4 位数值比较器 7485 的真值表

比较输入				级联输入			输出		
A_3,B_3	A_2,B_2	A_1,B_1	A_0,B_0	$I_{A<B}$	$I_{A=B}$	$I_{A>B}$	$F_{A<B}$	$F_{A=B}$	$F_{A>B}$
$A_3<B_3$	×	×	×	×	×	×	1	0	0
$A_3=B_3$	$A_2<B_2$	×	×	×	×	×	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	×	×	×	×	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	×	×	×	1	0	0
$A_3>B_3$	×	×	×	×	×	×	0	0	1
$A_3=B_3$	$A_2>B_2$	×	×	×	×	×	0	0	1
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	×	×	×	×	0	0	1
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	×	×	×	0	0	1
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	0	0	1	0	0	1
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	1	0	0	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	×	1	×	0	1	0

2. 位数扩展

数值比较器的扩展方式有串联扩展和并联扩展两种。图 3.7.4 为两个 4 位数值比较器串联而成为一个 8 位的数值比较器。对于两个 8 位数而言，如果高 4 位不相等，则比较的结果就是最终的结果，因此 8 位数值比较器的输出芯片 7485(1)输出；如果高 4 位相等，则比较的结果则由低 4 位决定，因此芯片 7485(2)的输出接到芯片 7485(1)的级联输入端。这种级联方式简单，但这种方式中比较的结果是逐级传递的。级联芯片数越多，传递时间越长，工作速度越慢。因此，当扩展位数较多时，常采用并联方式。

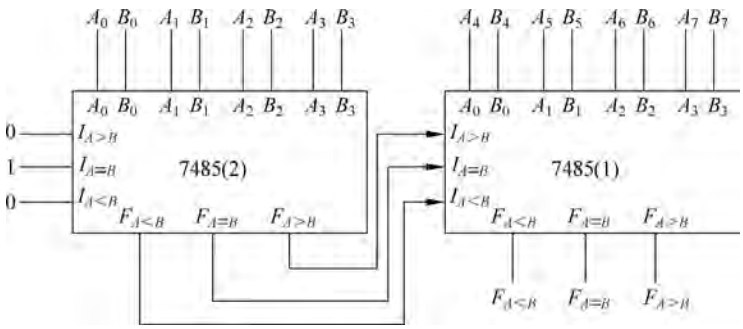


图 3.7.4　采用串联方式构成的 8 位数值比较器

图 3.7.5 是采用并联方式用 5 片 7485 组成的 16 位数值比较器。将 16 位二进制数按高低位次序分成 4 组，每组用 1 片 7485 进行比较，各组的比较是并行的。将每组的比较结果再经过 1 片 7485 芯片进行比较后得出比较结果。这样总的传递时间为 2 倍的 7485 的延迟时间。若用串联方式，则需要 4 倍的 7485 的延迟时间。

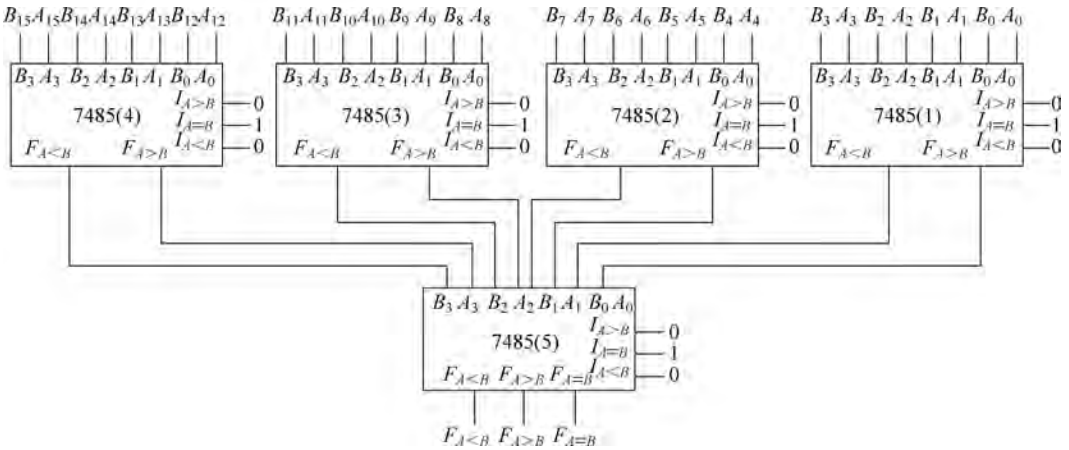


图 3.7.5　采用并联方式构成的 8 位数值比较器

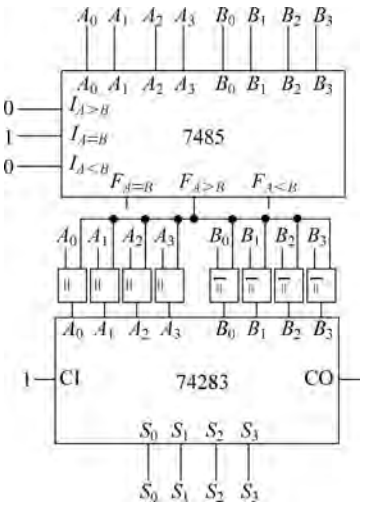
（三）课后篇

3.7.5　课后巩固——练习实践

1. 知识巩固练习

3.7.5.1　由 4 位数加法器 74283 和数值比较器 7485 构成的电路如题图 3.7.5.1 所示，试分析电路的逻辑功能。

讲解视频



题图 3.7.5.1

讲解视频

3.7.5.2　试用两片 7485 芯片实现 9 位二进制数的比较。

讲解视频

3.7.5.3　试用两个 4 位数值比较器组成三个数判别电路。要求能够判别三个 4 位

二进制数 $A(a_3a_2a_1a_0)$、$B(b_3b_2b_1b_0)$、$C(c_3c_2c_1c_0)$是否相等、A 是否最大、A 是否最小，并分别给出“三个数相等”“A 最大”“A 最小”的输出信号。（可以附加必要门电路。）

2. 工程实践练习

3.7.5.4 利用口袋实验包完成 1 位十进制数加/减法器电路的搭建与调试。

3.7.6 本节思维导图

知识拓展——数字电路设计理论奠基人　　世界集成电路发展历史

美国科学家克劳德·香农被誉为“信息论之父”，1938 年，他的硕士学位论文《继电器与开关电路的符号分析》(*A Symbolic Analysis of Relay and Switching Circuits*)将布尔代数的“真”与“假”和电路系统的“开”与“关”对应起来，用 1 和 0 表示，并用布尔代数分析和优化开关电路，奠定了数字电路的理论基础，论证了数字计算机及数字电路逻辑设计的可能性，哈佛大学的 Howard Gardner 教授说：“这可能是 20 世纪最重要、最著名的一篇硕士学位论文。”

3.8 用 Verilog HDL 描述组合逻辑电路

前面学习了各种组合逻辑电路功能模块，利用 Verilog HDL 可以实现硬件电路设计，本节将组合逻辑电路的三种建模技巧。

（一）课前篇

本节导学单

1. 学习目标

根据《布鲁姆教育目标分类学》，从知识维度和认知过程两方面进一步细化本节课的教学目标，明确学习本节课所必备的学习经验。

知识维度	认知过程					
	1. 回忆	2. 理解	3. 应用	4. 分析	5. 评价	6. 创造
A. 事实性知识						
B. 概念性知识	回忆C语言运算符	阐释数据流建模、行为级建模和结构化建模的基本思想				
C. 程序性知识	回忆条件语句、循环语句、多路分支语句的语法规则		应用三种建模方法描述组合逻辑电路		类比评价三种建模方法	设计组合逻辑电路，会借助仿真软件进行验证，会利用FPGA开发板进行电路搭建与调试
D. 元认知知识	明晰学习经验是理解新知识的前提		根据基本思想及学习经验迁移得到新理论的能力			通过电路的设计、仿真、搭建、调试，培养工程思维

2. 导学要求

根据本节的学习目标，将回忆、理解、应用层面学习目标所涉及的知识点课前自学完成，形成自学导学单。

3.8.1 课前自学——组合逻辑电路的数据流建模

对于基本单元逻辑电路，使用 Verilog HDL 提供的门级元件模型描述电路非常方便，但随着电路的复杂性增加，使用的逻辑门较多时，工作效率会降低。而数据流建模能在较高的抽象级别描述电路的逻辑功能，能够自动地将数据流描述转换成为门级电路。

在 Verilog HDL 中，数据流建模使用连续赋值语句对于 wire 型变量进行赋值，由关键词 assign 开始，后面跟着由操作数和运算符组成的逻辑表达式。

连续赋值语句的执行过程：只要逻辑表达式右边变量的逻辑值发生变化，等式右边表达式的值会立即被计算出来并赋给左边的变量。注意，在 assign 语句中，左边变量的数据类型必须是 wire 型。

例如，2 线-4 线译码器的连续赋值描述：

```
module decoder_df (A1,A0,E,Y)
 input  A1,A0,E;
 output  [3:0] Y;
 assign  Y[0] = ~ (~A1 & ~A0 & ~E);
 assign  Y[1] = ~ (~A1 & A0 & ~E);
 assign  Y[2] = ~ (A1 & ~A0 & ~E);
 assign  Y[3] = ~ (A1 & A0 & ~E);
endmodule
```

不难发现，数据流建模根据电路的逻辑功能进行描述，不必考虑电路的组成以及元件之间的连接，是描述组合逻辑电路常用的一种方法。

3.8.2 课前自学——预习自测

3.8.2.1 假设 m=4'b0101，按要求填写下列运算的结果：

&m=_____，|m=_____，^m=_____，~^m=_____。

3.8.2.2 试写出带有使能控制端的 3 线-8 线译码器的 Verilog HDL 数据流描述。

（二）课上篇

3.8.3 课中学习——组合逻辑电路的行为级建模

行为级建模一般使用 always 结构，通过条件语句、多路分支语句和 for 循环语句等描述电路。

1. 条件语句

Verilog 语言中有三种形式的 if 语句，一般用法如下：

```
if（条件表达式）  语句或语句块 ;
```

或

```
if（条件表达式）  语句或语句块 1 ;
else  语句或语句块 1  ;
```

或

```
if（表达式 1）  语句或语句块 1 ;
else if（表达式 2）语句或语句块 2 ;
else if(表达式 3)  语句或语句块 3 ;
……
else  ;
```

例 3.8.1　使用 if-else 语句对 8 选 1 数据选择器的行为进行描述。

解：假设这个程序模块的名称为 mux8to1_bh，输出端口为 F，输入端口使用一个 8 位的向量 D 来表示数据输入 D_7、D_6、D_5、D_4、D_3、D_2、D_1、D_0，选择输入端口使用一个 3 位的向量 S 来表示地址输入 S_2、S_1、S_0。于是，得到 4 选 1 数据选择器的行为描述代码如下：

```
module  mux8to1_bh(D,S,F);       //Verilog 1995 module port syntax
 input [7:0] D;                  //输入端口声明
 input [2:0] S;                  //输入端口声明
 output reg F;                   //输出端口及变量的数据类型声明
 always  @  (D,S)                //电路描述
      if( S = = 2'b000)         F = D[0];
      else if( S = = 3'b001)     F = D[1];
      else if( S = = 3'b010)     F = D[2];
      else if( S = = 3'b011)     F = D[3];
      else if( S = = 3'b100)     F = D[4];
      else if( S = = 3'b101)     F = D[5];
      else if( S = = 3'b110)     F = D[6];
      else             F = D[7];
endmodule
```

注意：过程赋值语句只能给寄存器型变量赋值，因此，程序中将输出变量 Y 的数据类型定义成 reg。

2. 多路分支语句

多路分支语句 case 语句的一般形式如下：

```
case（控制表达式）
  分支语句 1: 语句块 1;
  分支语句 2: 语句块 2;
  ……
  分支语句 n: 语句块 n;
  default:语句块 n + 1;  //default 语句可以省略
endcase
```

例 3.8.2　根据表 3.4.1 所示的功能，对共阴极的七段显示译码器的行为进行描述。

解：case 语句是一条多路决策语句，因此使用 case 语句描述七段显示译码器更方便。

使用位拼接运算符将输入端口 D3D2D1D0 拼接成一个 4 位的变量，输出端口 abcdefg 拼接成一个 7 位的变量。case 语句的分支项按顺序排列，最后一个分支项 default 把其他非 8421 码输入的情况设置成全零输出，省去所有无用输入码的显示。其代码如下：

```
module seg7_decoder(                                    //Verilog 2001,2005 module port syntax
 input LE,RBI,BI,D3,D2,D1,D0,                           //输入端口声明
 output reg a,b,c,d,e,f,g                               //输出端口及变量的数据类型声明
);
always @ ( * ) //电路描述
begin
  if (LT == 0) {a,b,c,d,e,f,g} = 7'b111_1111 ;          //让显示器的七段都发光
  else if ((RBI == 0)&(D3D2D1D0 = 0000)) {a,b,c,d,e,f,g} = 7'b000_0000 ; //灭零显示
  else if (BI == 0) {a,b,c,d,e,f,g} = 7'b000_0000 ; //灭灯显示
  else
   case({D3,D2,D1,D0})
     4'd0: {a,b,c,d,e,f,g} = 7'b111_1110;               //7e
     4'd1: {a,b,c,d,e,f,g} = 7'b011_0000;               //30
     4'd2: {a,b,c,d,e,f,g} = 7'b110_1101;               //6d
     4'd3: {a,b,c,d,e,f,g} = 7'b111_1001;               //79
     4'd4: {a,b,c,d,e,f,g} = 7'b011_0011;               //33
     4'd5: {a,b,c,d,e,f,g} = 7'b101_1011;               //5b
     4'd6: {a,b,c,d,e,f,g} = 7'b001_1111;               //1f
     4'd7: {a,b,c,d,e,f,g} = 7'b111_0000;               //70
     4'd8: {a,b,c,d,e,f,g} = 7'b111_1111;               //7f
     4'd9: {a,b,c,d,e,f,g} = 7'b111_1011;               //7b
     default: {a,b,c,d,e,f,g} = 7'b000_0000;            //非 8421 码不显示
   endcase
 end
endmodule
```

3. for 循环语句

for 循环语句的一般形式如下：

```
for(循环变量初始值;循环的条件;循环变量的步长)  语句或语句块;
```

例 3.8.3 利用 Verilog HDL 描述一个具有使能输入端的 3 线-8 线译码器的行为，要求输出为低电平有效。

解：该模块实现的功能与集成 3 线-8 线译码器 74138 类似，唯一区别是该模块使能信号只有一个高电平有效的使能信号，74138 有三个使能信号。当使能信号 EN＝1 时，针对循环变量 x 的变化，重复执行 if-else 语句 8 次。具体代码如下：

```
module decoder3to8 (
 input [2:0] A,                                         //输入端口声明
 input EN,                                              //输入端口声明
 output reg [7:0] Y                                     //输出端口及变量的数据类型声明
);
 integer x;                                             //声明一个整型变量 x
  always @ (A,EN)
  begin
```

```
  Y = 8'b1111_1111 ;                    //设置译码器输出的默认值
for(x = 0;x < = 7;x++)                  //循环 8 次
   if ((EN == 1)&&(A == x))
    Y[x] = 0;
   else
    Y[x] = 1;
end
  endmodule
```

（三）课后篇

3.8.4 课后巩固——练习实践

1. 知识巩固练习

3.8.4.1 试写出 8 线-3 线优先编码器的行为级描述，然后用 Quartus Ⅱ 软件进行逻辑功能仿真，并给出仿真波形。

3.8.4.2 试用 for 循环语句对 n 位串行加法器的行为进行描述。

2. 工程实践练习

3.8.4.3 利用 FPGA 开发板实现 3.3～3.7 节介绍的案例。

3.8.5 本节思维导图

知识拓展——FPGA 主要应用领域　　**集成电路先进技术介绍**

最初的 FPGA 就是一个可编程的数字电路，内含很多逻辑块和连接线，使得数字电路设计和验证通过编程就可完成，主要应用在数字逻辑电路、逻辑接口电路、ADC 采集电路、DAC 转换电路等。例如，用 FPGA 设计一个 24 位乘法器，SRAM、SRAM 或 Flash 存储芯片的接口控制器，计算机的 PCI 总线控制器等。近些年，FPGA 发展内含存储单元、总线和各种功能模块，FPGA 已可以完成配置成专用 CPU、专用控制器等复杂功能系统，各类 EDA 软件让开发者像点菜一样配置它的各类资源，完成设计。应用领域也扩展到通信系统、数字信号处理、视频图像处理、高速接口、人工智能(AI)、集成(IC)设计等领域，尤其在军事领域的安全通信、雷达和声呐、电子战和精确控制等方面得到了广泛应用。

3.9 子弹分装系统工程任务实现

3.9.1 基本要求分析

学习完了理论知识，再来看看本章开始布置的工程任务——子弹分装系统。该系统要求：按键输入每把枪安装子弹的数目并显示出来，系统自动按照输入数目将每把枪需要的子弹分装到瓶子中，同时将分装子弹总数存储在计算机上。虽然系统中计数器和寄存器在后续章中才会介绍，但是为了清楚阐述本章所介绍的逻辑模块在系统中的应用，这里仅介绍计数器和寄存器基本功能。与大多数系统一样，实现系统功能有很多种方法。通过学习系统结构和系统的工作原理，学会如何将多种逻辑功能模块相互连接起来形成一个完整的系统，以完成给定的任务。分析系统要求，可以将任务分为人机交互模块、代码转换模块、自动分装模块和数据传输模块，如图 3.9.1 所示。

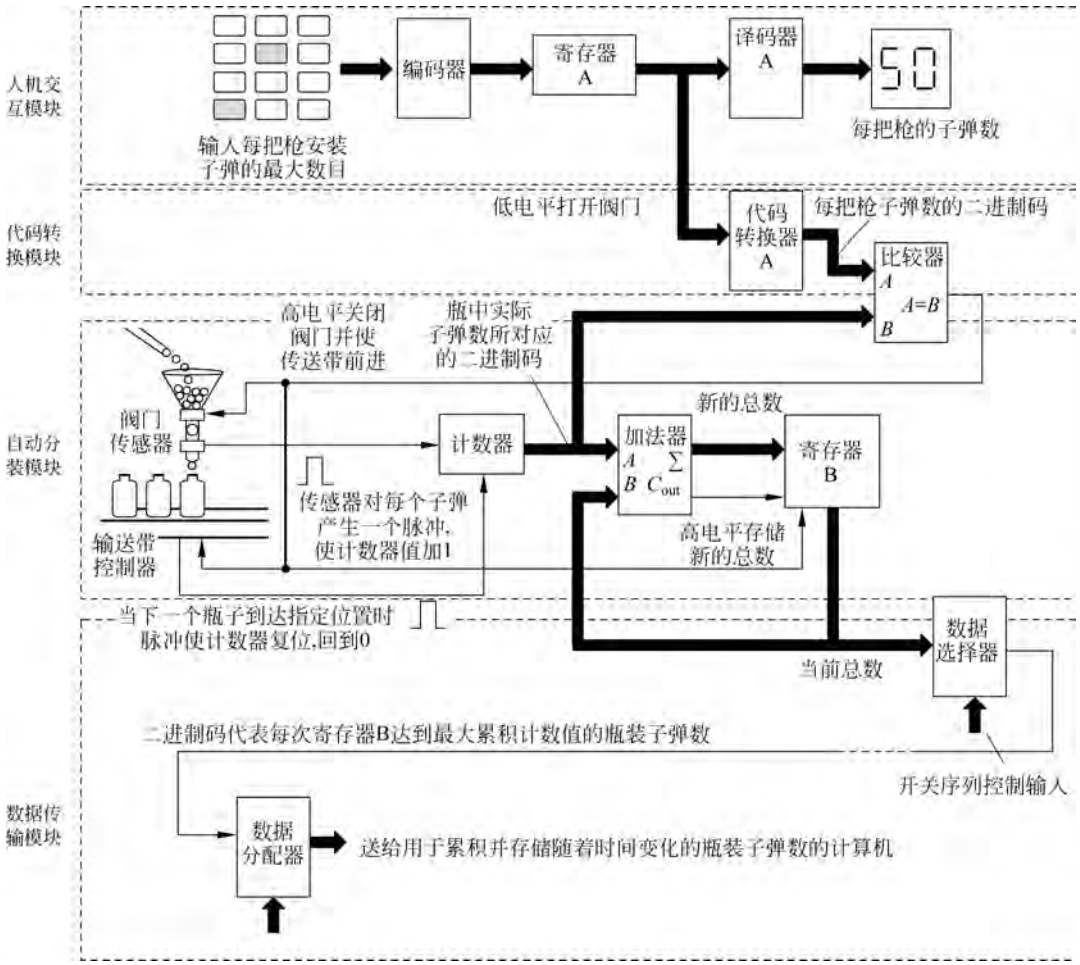


图 3.9.1 系统框图

3.9.2 模块电路设计

1. 人机交互模块

系统功能：每把枪安装子弹最大数目的输入并显示。

电路构成：编码器、寄存器、译码器、数码管。

如图 3.9.2 所示，键盘有 10 个按键，每个对应一个十进制数。当一个键按下时，编码器将有效电平转换为对应的 8421 码输出。寄存器 A 能够存储 8 位，对应于 2 个 8421 数。假定输入的子弹数目为 50 个，因此，首先通过键盘输入并转成对应的 8421 码 0101，存于寄存器中；接下来输入 0 并转成对应的 8421 码 0000 存入寄存器中。

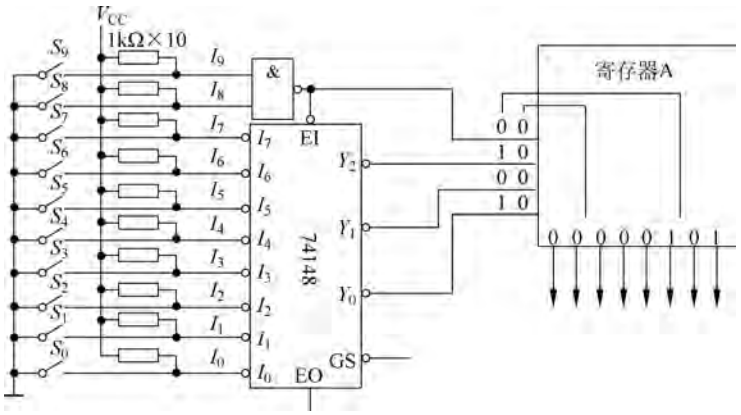


图 3.9.2 人机交互模块电路 1

如图 3.9.3 所示，译码器 A 将寄存器 A 送来的 8421 码转换成 2 个数字所对应的七段码，用于显示子弹数。实际上译码器 A 由 2 个显示译码器 7447 组成，以同时显示 2 个数字。

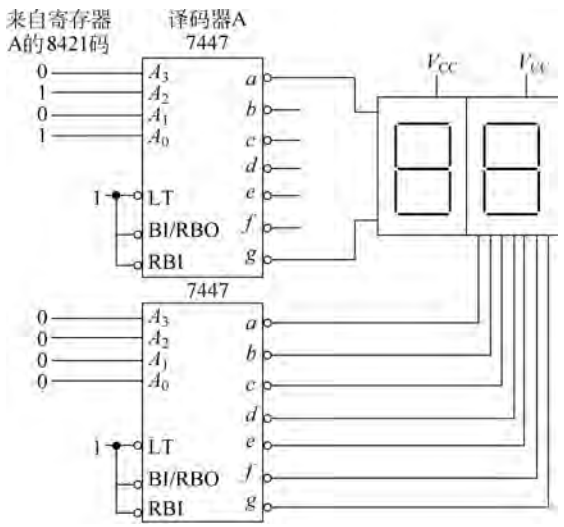


图 3.9.3 人机交互模块电路 2

2. 代码转换模块

系统功能：将 8421 码转换为二进制码。

电路构成：加法器。

自动分装模块根据键盘输入的子弹数目进行自动分装，首先需要比较器实现对于输入的每瓶子弹数目与瓶中装入的实际子弹数目进行比较。由于比较器输入的是二进制码，而寄存器存储的是 8421 码，因此需要一个能够把 8421 码转换为二进制码的代码转换电路。代码转换电路如图 3.9.4 所示。

二进制码左移一位等于未左移的二进制码×2。也就是说，二进制码左移 2 位加上左移 5 位加上左移 6 位就可以等效于二进制码乘以 100；二进制码左移 1 位加上左移 3 位就可以等效于二进制码乘以 10。由于 8421 码每一位的权重与二进制数前 4 位的权重相同，因此利用加法和移位即可实现 8421 码向二进制码的转换。代码转换电路如图 3.9.5 所示。

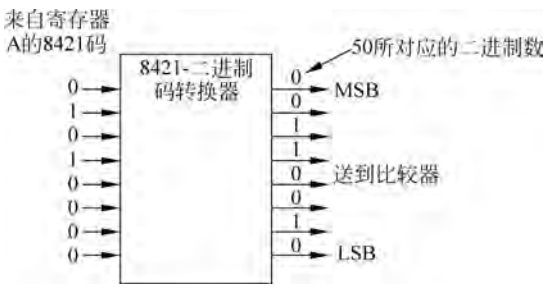


图 3.9.4　代码转换电路

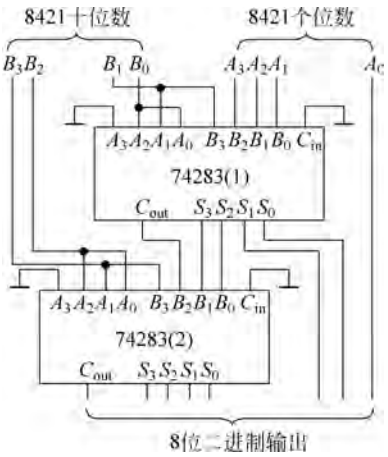


图 3.9.5　代码转换电路

3. 自动分装模块

系统功能：按照输入的子弹数目实现自动分装。

电路构成：编码器、寄存器、译码器、数码管。

需要设计的电路：计数器、加法器、寄存器、代码转换器、比较器。

将代码转换电路的输出送入比较器进行比较。计数器用于计数装入瓶中的子弹数，并不断将计数器的计数值与设置值进行比较，只要计数器中计数值小于设定值，阀门将一直保持打开的状态；当计数值达到设定值，比较器会产生一个高电平输出信号，关闭阀门，同时传送带送来另一个新的空瓶，计数器复位即计数值回到 0，当比较器的输出回到零时打开阀门，重新开始向新的瓶中装入子弹。

瓶中子弹数与寄存器 B 中先前存储的子弹总数进行相加，得到新的子弹总数存储在寄存器 B 中，电路如图 3.9.6 所示。图中给出了在瓶中子弹数满 50，先前在寄存器 B 中子弹总数为 100 时所对应的操作示意图。新的子弹总数值 150 代替原来的 100。图 3.9.6

中 8 位数值比较器用两片 4 位数值比较器 7485 扩展即可实现，8 位加法器用两片 4 位超前进位加法器 74283 串行扩展即可实现。

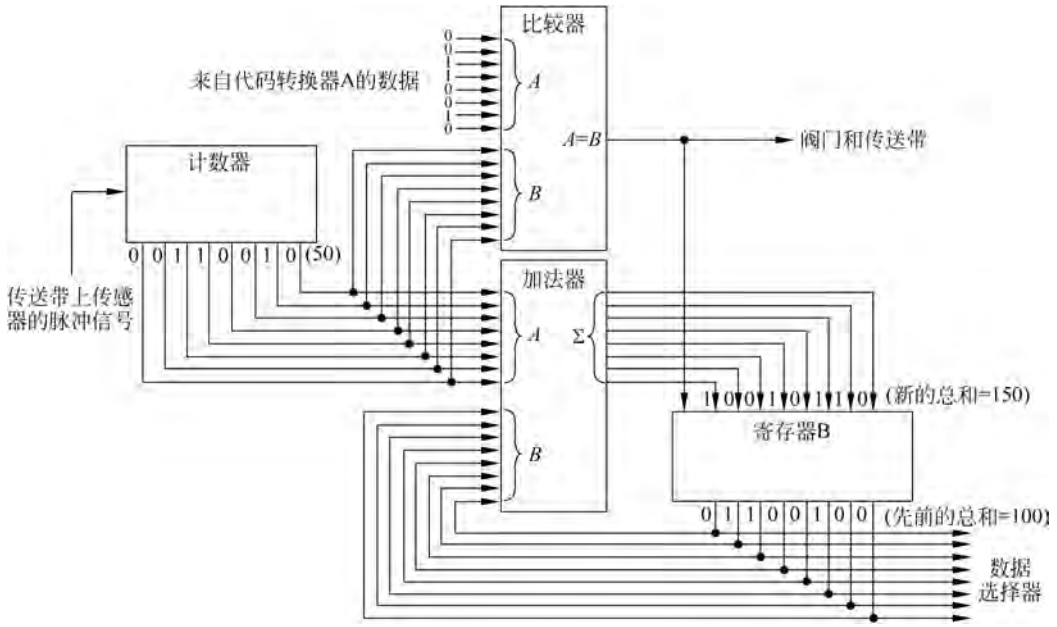


图 3.9.6　自动分装模块电路

4. 数据传输模块

系统功能：将子弹分装的总数传输到计算机上。

电路构成：译码器、数据选择器、数据分配器。

数据选择器寄存器 B 输出的 8 位并行数据转换成 8 位串行数据，传送到数据分配器。数据分配器将串行数据再转换成并行格式，送给计算机对子弹总数进行保存，电路如图 3.9.7 所示。

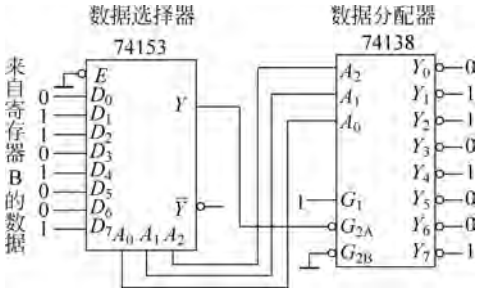


图 3.9.7　数据传输模块电路

3.9.3　系统搭建

将各个功能模块电路拼接起来就构建了子弹分装系统，如图 3.9.8 所示，读者可以自行仿真。

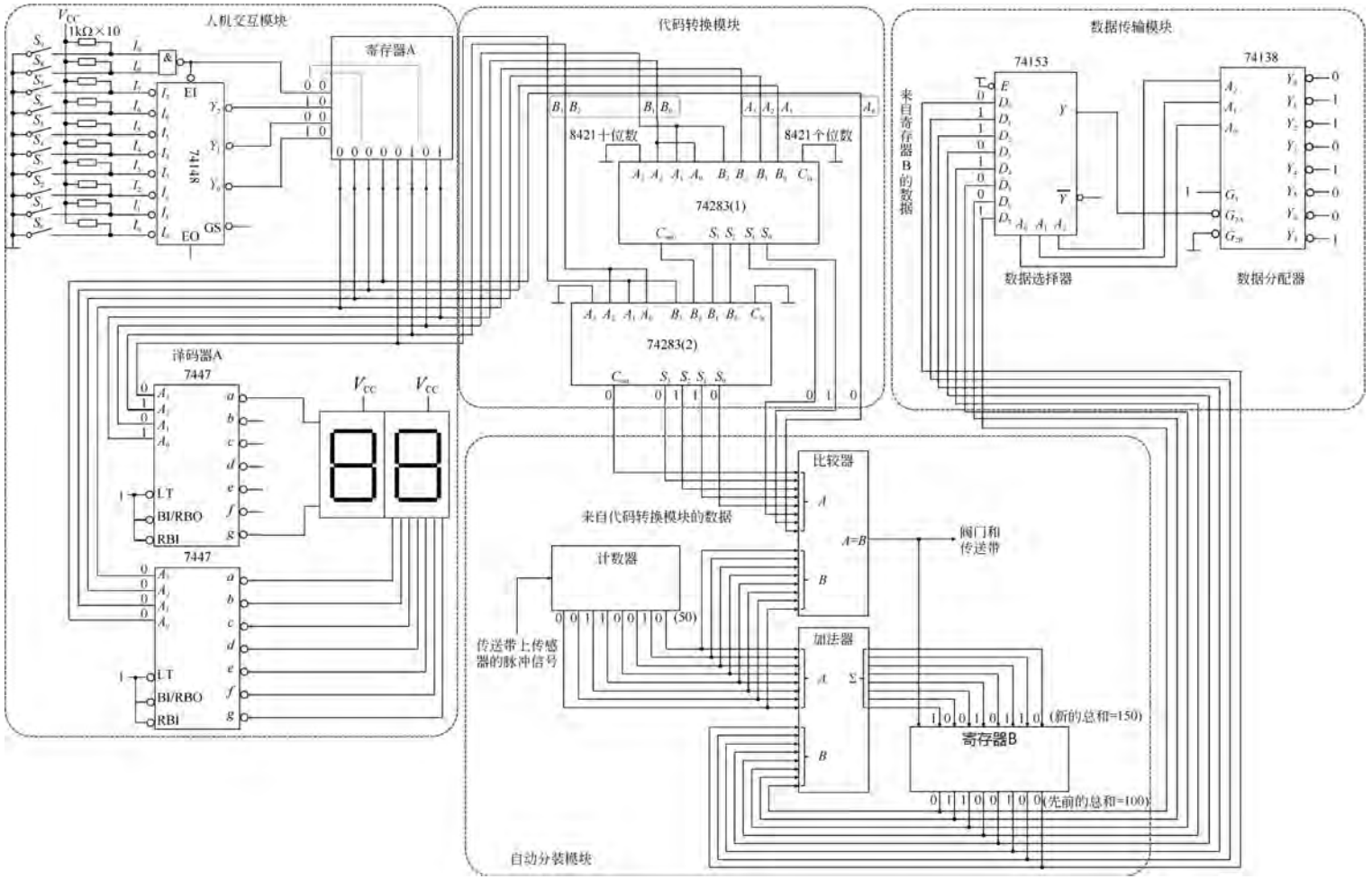


图 3.9.8　子弹分装系统电路

3.10 本章知识综合练习

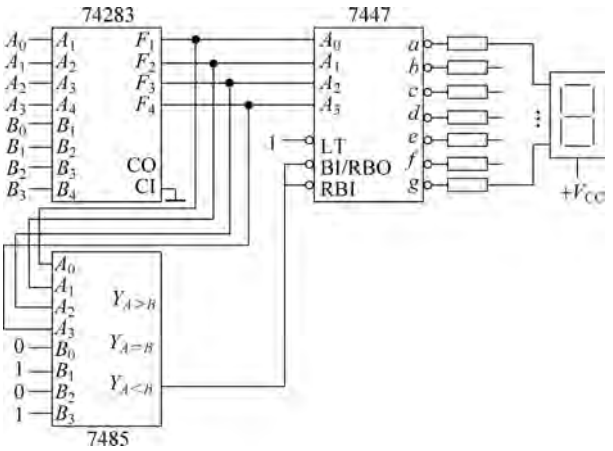
3.10.1 $A_3A_2A_1A_0$、$B_3B_2B_1B_0$、$C_3C_2C_1C_0$、$E_3E_2E_1E_0$ 是待传送的 4 路数据，试设计利用 $D_3D_2D_1D_0$ 数据总线分时传送各路数据的逻辑电路。

讲解视频

3.10.2 一把密码锁有 A、B、C 三个按键，当三个按键都不按下时，锁打不开，也不报警；当只有一个按键按下时，锁不打开，但发出报警信号；当有两个按键按下时，锁打开，但不报警；当三个按键都同时按下时，锁被打开同时报警。按下述要求设计逻辑电路：(1)门电路；(2)3 线-8 线译码器和与非门；(3)双四选一数据选择器和非门；(4)全加器。

讲解视频

3.10.3 题图 3.10.3 电路由一片 4 位超前进位加法器 74283、比较器 7485 与七段显示译码器 7447 及数码管组成的电路，试分析该电路的逻辑功能。



题图 3.10.3

讲解视频

3.10.4 试分别用 7485、74283、74138、74153、74151 芯片设计一个逻辑电路，当 $X_2X_1X_0>5$ 时，电路输出 L 为 1，否则 L 为 0。

3.11 本章课程设计拓展

随着人们生活水平的提高和安全意识的加强，对安全的要求也就越来越高。锁自古以来就是把守护门的“铁将军”，人们对它要求甚高，既要安全可靠地防盗，又要使用方便，这也是制锁者长期以来研制的主题。随着电子技术的发展，各类电子产品应运而生，电子密码锁就是其中之一。电子密码锁的研究从 20 世纪 30 年代就开始了，这种锁是通过键盘输入一组密码完成开锁过程。电子锁的种类繁多，如数码锁、指纹锁、磁卡锁等，但较实用的还是按键式电子密码锁。简单的数字电路可实现密码不限次数重写，可靠性高，价格便宜。

电子密码锁设计要求：

(1) 用按键输入 4 位十进制数字。

(2) 开锁信号。当输入正确密码时，给出开锁信号，用一个绿色指示灯表示输入密码正确；如果输入密码错误，用红灯表示。而且依次只能亮一盏灯，红灯亮、绿灯灭表示关锁，绿灯亮、红灯灭表示开锁。

(3) 设置倒计时电路和自锁电路。如果密码在 5s 内未能输入正确，则发出报警信号并且自锁电路。

(4) 设置密码设置开关，当开关闭合后，允许设置密码，设置好密码后，打开此开关。

(5) 需要在输入密码开始时识别输入，并由此触发计时电路。

3.12 本章思维导图


思维导图