第3章 组合逻辑电路

本章要点

- ◇ 熟悉组合电路的基本概念；
- ◇ 掌握组合电路的分析方法和设计方法，能完成实际逻辑问题的分析和设计；
- ◇ 熟悉中规模集成电路模块的功能；
- ◇ 掌握中规模集成电路芯片在电路设计中的灵活应用；
- ◇ 了解组合电路中的竞争-冒险现象。

在生产和生活实践中遇到的逻辑问题层出不穷，为之设计的逻辑电路也举不胜举。数字系统中的各种逻辑组件，从电路结构和逻辑功能上分为两大类：组合逻辑电路和时序逻辑电路。在数字系统中，由基本逻辑门组成的组合逻辑电路是研究数字电路的基础，常用的组合逻辑电路有编码器、译码器、数据分配器、数据选择器、数值比较器、全加器等。同时，为了使用方便，这些组合逻辑电路均有中规模集成电路器件(MSI)。

本章主要介绍采用小规模器件(SSI)及中规模器件(MSI)分析、设计组合电路的方法，并介绍常用 MSI 器件的功能及应用。这些器件具有特定的逻辑功能，通常用符号图、功能表、逻辑表达式或硬件描述语言描述其逻辑功能。在进行逻辑设计时，需要正确选择器件。从集成度角度讲，可选择中、大规模集成器件或高密度可编程逻辑器件；从工艺上讲，可用 TTL 或 CMOS 集成器件，即要综合考虑所用器件数量、功耗、速度、负载能力及价格等。

趣味知识

LED 照 明

在当前全球能源短缺的忧虑再度提升的背景下，LED 以其节能、高效、寿命长、无辐射、绿色环保等特点，广泛应用于指示灯、显示屏、景观照明等领域，在日常生活中处处可见。LED 的核心部分是一个 PN 结，在 PN 结中载流子复合时会把过剩的能量以光的形式释放出来，发出各种颜色的光。50 多年前，美国通用电气公司的一名普通研究人员 Nick Holonyakjr 研制出世界上第一个红色 LED，之后出现了黄色、蓝色、绿色 LED。直到 1996 年日本 Nichia(日亚)公司成功开发出白色 LED。

长期以来 LED 受到亮度差、价格昂贵等条件的限制，无法作为通用光源推广应用。近几年来，随着人们对半导体发光材料研究的不断深入和 LED 制造工艺的不断进步，各种颜色的超高亮度 LED 的研制取得了突破性进展，其发光效率提高了近千倍，其中最重要的是超高亮度白光 LED 的出现，使 LED 应用领域拓展至高效率照明光源市场，成为国际公认的

下一代固态照明光源、人类继爱迪生发明白炽灯泡后最伟大的发明之一。

目前 LED 照明有三种最新的设计理念：

① 情景照明。2008 年飞利浦公司提出情景照明，根据环境的需求来设计灯具。情景照明以场所为出发点，旨在营造一种漂亮、绚丽的光照环境，去烘托场景效果。

② 情调照明。2009 年由凯西欧公司提出情调照明，以人的需求来设计灯具。情调照明是以人的情感为出发点，从人的角度去创造一种意境光照环境。情调照明与情景照明不同，情调照明是动态的，是可以满足人们精神需求的照明方式；而情景照明是静态的，它只能强调场景光照的需求，不能表达人的情绪。

③ 人文照明。采用最新的科学技术对“光”进行创造，使照明技术参数与环境、氛围、活动、人物心情等相适应，成为一个从积极方面影响人的情绪、生活、工作的因素，在照明方式与照明效果方面，追求人文、科技、艺术的统一，是一种以人为本的照明方式。

那么到底这些 LED 显示设备是怎么工作的？其内部受到什么样的系统控制？应用到什么数字电路理论？大家可以根据本章中相关的知识，想一想最简单的电梯中看到的层 6 是如何显示出来的？层 28 又如何显示出来的？火车站内“北京站 14:12 开”等又是如何显示的？

3.1　组合电路的分析和设计

如果一个逻辑电路在任何时刻的输出状态只取决于这一时刻的输入状态，而与电路原来的状态无关，则称这样的电路为组合逻辑电路。

组合逻辑电路具有如下特点：

(1) 在电路结构上只由逻辑门组成，不含记忆性元件；

(2) 输出、输入之间没有反馈延迟电路。

通常组合电路可以有若干个输入变量：$I_0, I_1, I_2, \cdots, I_{n-1}$；也可以有若干个输出变量：$Y_0, Y_1, Y_2, \cdots, Y_{m-1}$。每一个输出函数是全部或部分输入变量的函数：$Y(t_n)=F[I(t_n)]$，即

$$\begin{aligned}
Y_0 &= F_0(I_0, I_1, I_2, \cdots, I_{n-1}) \\
Y_1 &= F_1(I_0, I_1, I_2, \cdots, I_{n-1}) \\
Y_2 &= F_2(I_0, I_1, I_2, \cdots, I_{n-1}) \\
&\vdots \\
Y_{m-1} &= F_{m-1}(I_0, I_1, I_2, \cdots, I_{n-1})
\end{aligned}$$

其框图如图 3.1 所示。

图 3.1　组合电路框图

组合逻辑电路逻辑功能的描述方法有逻辑函数表达式、真值表、逻辑图和卡诺图等，这些方法之间可以相互转换。

3.1.1　组合电路的分析

组合逻辑电路的分析，就是针对给定的组合电路，利用门电路和逻辑代数知识，找出电路输出和输入之间的逻辑关系，从而确定它的逻辑功能。

组合电路一般按照下列步骤进行分析(见图 3.2)。

图 3.2 组合电路分析步骤

(1) 根据给定的逻辑电路图,写出输出端的逻辑表达式。

已知逻辑图写出函数式,其方法是:从逻辑图输入端到输出端逐级写出每个逻辑门对应的逻辑表达式,就可以得到最后的函数表达式。

(2) 对输出逻辑函数进行化简,得到最简表达式。一般化为最简"与或"式。

(3) 列出真值表。

其方法是:将输入变量取值的所有状态组合逐一代入逻辑函数式求出函数值,列成表,即可得到真值表。

(4) 根据真值表中输入/输出的关系,说明电路的逻辑功能。

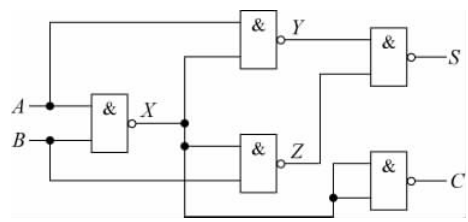


图 3.3 例 3.1 电路图

例 3.1 分析如图 3.3 所示电路的逻辑功能。

解:

(1) 由逻辑图逐级写出函数式。

由图 3.3 可以写出 Y 的逻辑函数式:

$$X = \overline{AB}$$

$$Y = \overline{XA}$$

$$Z = \overline{XB}$$

$$S = \overline{YZ} = \overline{\overline{\overline{AB}A} \cdot \overline{\overline{AB}B}}$$

$$C = \overline{X} = \overline{\overline{AB}}$$

(2) 化简和变换。

$$S = \overline{\overline{\overline{AB}A} \cdot \overline{\overline{AB}B}} = \overline{\overline{AB}A} + \overline{\overline{AB}B} = A\overline{B} + \overline{A}B$$

$$C = \overline{\overline{AB}} = AB$$

(3) 根据最简式列真值表。

由最简式可列出真值表,见表 3.1。

(4) 说明逻辑功能。

由真值表中可以看出,如图 3.3 所示组合逻辑电路完成对输入两个 1 位二进制变量 A、B 求和的功能,S 是两个二进制数的和,C 是向高位的进位。因为该电路不考虑来自低位的进位,故是一个半加器。

表 3.1 例 3.1 真值表

A	*B*	*S*	*C*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

例 3.2　分析如图 3.4 所示电路的逻辑功能，输入信号 A、B、C、D 是一组二进制代码。

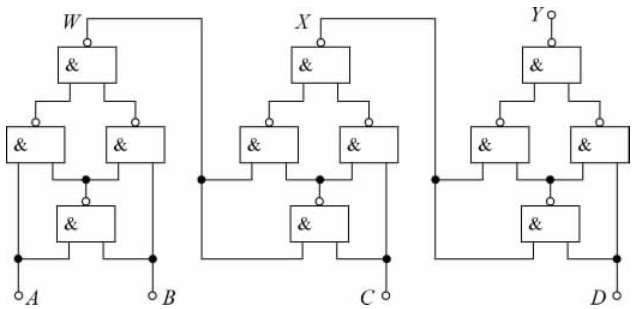


图 3.4　例 3.4 电路图

解：

(1) 由逻辑图逐级写出函数式。

设中间变量为 W、X，分别写出 W、X、Y 的函数式

$$W=\overline{\overline{A}\,\overline{AB}\,\overline{AB}B},\quad X=\overline{\overline{W}\,\overline{WC}\,\overline{WC}C},\quad Y=\overline{\overline{X}\,\overline{XD}\,\overline{XD}D}$$

(2) 将各级函数化简，最后得到 Y 的最简式。

$$W=\overline{\overline{A}\,\overline{AB}\,\overline{AB}B}=A\overline{B}+\overline{A}B$$

$$X=W\overline{C}+\overline{W}C=A\overline{B}\overline{C}+\overline{A}B\overline{C}+\overline{A}\,\overline{B}C+ABC$$

$$\begin{aligned}Y&=X\overline{D}+\overline{X}D\\&=A\overline{B}\overline{C}\overline{D}+\overline{A}B\overline{C}\overline{D}+\overline{A}\,\overline{B}C\overline{D}+ABC\overline{D}+\overline{A}\,\overline{B}\overline{C}D+\overline{A}BCD+A\overline{B}CD+AB\overline{C}D\end{aligned}$$

(3) 列真值表，见表 3.2。

表 3.2　例 3.2 真值表

A	*B*	*C*	*D*	*Y*	*A*	*B*	*C*	*D*	*Y*
0	0	0	0	0	1	0	0	0	1
0	0	0	1	1	1	0	0	1	0
0	0	1	0	1	1	0	1	0	0
0	0	1	1	0	1	0	1	1	1
0	1	0	0	1	1	1	0	0	0
0	1	0	1	0	1	1	0	1	1
0	1	1	0	0	1	1	1	0	1
0	1	1	1	1	1	1	1	1	0

(4) 说明逻辑功能。

由表 3.2 中可以看出，当输入变量 A、B、C、D 四位代码中 1 的个数为奇数时，输出变量 Y 的值为 1，为偶数时输出 Y 的值为 0，所以该电路为奇校检电路。

3.1.2　组合电路的设计

组合电路的设计，就是根据给定的设计要求，设计出能实现该要求的最简逻辑电路。所

谓“最简”,是指电路所用的器件数目最少,器件的种类最少,器件之间的连线最少。

设计过程与分析过程相反,一般按照下列步骤进行(见图 3.5)。

(1) 逻辑抽象。

逻辑抽象分为设定变量和状态赋值两个部分。

由实际问题出发,根据事件的因果关系确定输入/输出变量,并对输入/输出变量进行状态赋值,即用 0、1 表示两种不同的状态。

(2) 建立真值表。

根据输入/输出变量之间的逻辑关系列出真值表。即将一个实际问题抽象成由真值表描述的逻辑函数问题。有些实际问题,只出现输入变量的部分取值组合,未出现的取值组合可以不在真值表中列出,若列出则相应的输出用“×”表示,视为无关项。

(3) 由真值表写出逻辑函数,并进行化简或变换。

由真值表写逻辑函数的方法是:先找出真值表中使逻辑函数取值为 1 的那些输入变量取值组合,再将这些输入变量组合写成最小项的形式,最后将这些最小项相加即可。

由真值表直接得到的逻辑函数不一定是最简式,为使所设计的电路最简,必须用公式法或卡诺图法对逻辑函数进行化简。如果已选定了某种器件(如与非门、或非门等)来设计电路,则还要对函数表达式进行变换。例如要用与非门实现电路,则必须把表达式变换为“与非-与非”形式。

(4) 画出逻辑图。

根据化简或变换后的表达式画出相应的逻辑电路图。

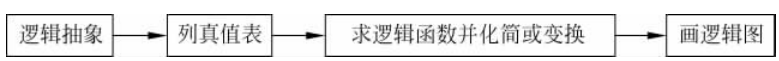


图 3.5 组合电路设计步骤

例 3.3 设计一个三人表决电路。三个人对某一事件进行表决,要求三人中两个或两个以上人同意,则事件通过。

解:

(1) 逻辑抽象。

设定变量:

输入变量三个:A、B、C;

输出变量一个:Y。

状态赋值:

A、B、C 等于 0:表示不同意;

A、B、C 等于 1:表示同意;

$Y=0$:表示未通过;

$Y=1$:表示通过。

(2) 列真值表。

将输入变量的所有取值组合及相应的输出值列入表中,见表 3.3。

表 3.3　例 3.3 真值表

A	*B*	*C*	*Y*
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

(3) 写出逻辑函数表达式并化简。

$$\begin{aligned} Y &= \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC \\ &= BC + A\bar{B}C + AB\bar{C} \\ &= BC + AC + AB \end{aligned}$$

该表达式为最简与或式，也可以将其转换为“与非-与非”式

$$\begin{aligned} Y &= \overline{\overline{BC + AC + AB}} \\ &= \overline{\overline{BC} \cdot \overline{AC} \cdot \overline{AB}} \end{aligned}$$

(4) 画出逻辑图。

若用与门和或门实现，见图 3.6；若用与非门实现，见图 3.7。

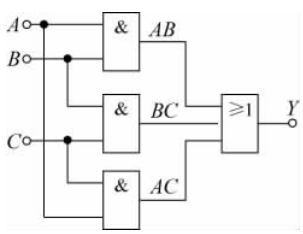


图 3.6　用与门和或门实现的逻辑图

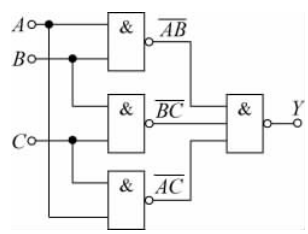


图 3.7　用与非门实现的逻辑图

例 3.4　设计一个监视交通信号灯工作状态的逻辑电路。正常情况下，红、黄、绿灯只有一个亮，否则视为故障状态，发出报警信号，提醒有关人员修理。

解：

(1) 逻辑抽象。

设定变量：

R、Y、G 分别表示红、黄、绿三个灯；Z 表示是否有故障。

状态赋值：

R、Y、G 等于 1 时表示灯亮，等于 0 时表示灯灭；$Z=0$ 表示正常，$Z=1$ 表示有故障。

(2) 列真值表。

根据题意，列出真值表见表 3.4。

表 3.4 例 3.4 真值表

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

(3) 写出输出表达式并化简。

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

用卡诺图化简,如图 3.8 所示。

所以 $Z=\bar{R}\bar{Y}\bar{G}+RY+RG+YG$。

(4) 画出逻辑图。

逻辑图如图 3.9 所示。

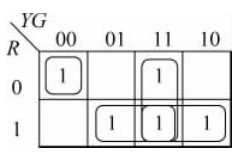


图 3.8 例 3.4 卡诺图

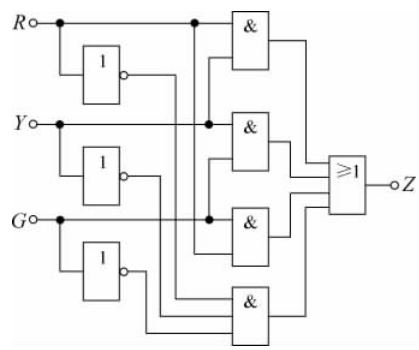


图 3.9 例 3.4 逻辑图

例 3.5 住宅供暖系统的控制逻辑操作如下:白天,温度低于 20℃时供暖;晚上,温度低于 18℃时供暖。设分别以逻辑符号 A、B、C 来表示:白天时 A 为 1,晚上 A 为 0;如果温度高于 20℃时 B 为 1,否则为 0;如果温度高于 18℃时 C 为 1,否则为 0。设计一个逻辑电路,它的输出信号用 Y 表示,需要供暖时为 1。

解:将上面描述的设计要求转换为真值表,如表 3.5 所示。表中列出了输入变量的所有组合,但是有两种组合不可能出现,因为温度不可能即高于 20℃($B=1$),又低于 18℃($C=0$)。这两种组合是无关项,以×表示。

表 3.5 例 3.5 真值表

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	×
0	1	1	0

续表

A	B	C	Y
1	0	0	1
1	0	1	1
1	1	0	×
1	1	1	0

根据真值表写逻辑函数时，也可以先由真值表画出卡诺图，再由卡诺图直接写出最简与或表达式。由图 3.10 所示的卡诺图可以得出输出信号的最简逻辑表达式：$Y=\overline{C}+A\overline{B}$。逻辑电路图如图 3.11 所示。

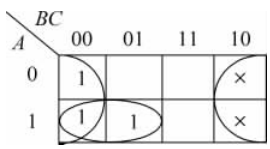


图 3.10　例 3.5 卡诺图

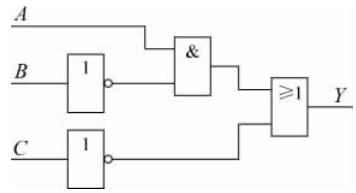


图 3.11　例 3.5 逻辑图

例 3.6　孩子们困惑的一个传统问题是：一个农夫带着一只狗、一只鹅和一袋黑麦旅行。农夫来到一条河边，他必须从河东岸到西岸。过河只使用一条船，并且船上只有两个位置，一个给农夫划船，一个给他所带的东西。如果农夫不在，那么鹅将吃掉黑麦或者狗将吃掉鹅。设计一个电路模拟解决这个问题。

解：用 A、B、C、D 分别表示农夫、狗、鹅、黑麦，如果相应对象在东岸，逻辑信号为 1，如果在西岸则为 0；用 Y 表示黑麦或鹅所处的状态，任何时候，处于安全状态为 1，处于危险状态为 0。由题中描述可得到真值表如表 3.6 所示。

表 3.6　例 3.6 真值表

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

画出卡诺图(如图 3.12 所示),由卡诺图可以得出输出信号的最简逻辑表达式:$Y=\overline{A}\overline{C}+AC+B\overline{C}D+\overline{B}C\overline{D}$。逻辑电路图如图 3.13 所示。

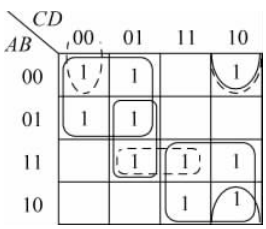


图 3.12 例 3.6 卡诺图

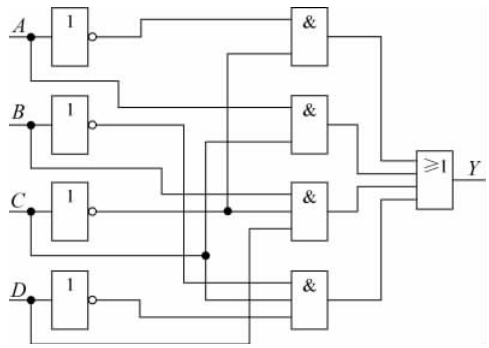


图 3.13 例 3.6 逻辑电路图

如果卡诺图 3.12 中包含两个最小项的包围圈,如图中虚线所示,则输出函数逻辑表达式为:$Y=\overline{A}\overline{C}+AC+ABD+\overline{A}\overline{B}\overline{D}$,其逻辑图读者可自行设计。两个逻辑函数虽然形式不同,但逻辑功能却是相同的。由此可见,同一逻辑函数可以有不同的逻辑表达式,同一逻辑功能也可以由不同的逻辑电路实现。这两种实现方法在实际操作中有什么不同呢?

思考题

1. “逻辑抽象”的概念是什么?包含哪些内容?
2. 对于同一个实际问题,不同的人经过逻辑抽象可能得到不同的逻辑函数,为什么?

3.2 编码器

在数字电路中,经常需要将有特定意义的信息(如文字、符号、图像等)转换成相应的二进制代码,这一过程称为编码,实现编码功能的电路称为编码器。

3.2.1 编码器的功能和分类

在数字系统中,数字信号不仅可以表示数字,还可以表示各种不同的指令、信息及事物等。在二值数字逻辑中,信号都是以高、低电平的形式出现的,所以编码器的功能就是把输入的高、低电平信号变成相应的二进制代码输出,如图 3.14 所示。通常编码器有 m 个信息输入端 $x_1,x_2,\cdots,x_m$,n 个二进制代码输出端 $z_1,z_2,\cdots,z_n$,m 和 n 应满足关系:$m\leqslant 2^n$。

目前,常用的编码器有二进制编码器和二-十进制编码器。

1. 二进制普通编码器

二进制编码器是用 n 位二进制代码对 2^n 个特定输入信息进行编码的电路。在二进制普通编码器中,2^n 个输入信号,任一时刻只允许一个编码输入有效,否则输出会发生混乱。

现在以 3 位二进制普通编码器为例,介绍一下二进制普通编码器的工作原理。图 3.15 为 3 位二进制编码器的框图。$I_0\sim I_7$ 是一组互斥的高电平输入变量,任一时刻只允许其中

一个取值为1。输出是3位二进制代码$Y_2Y_1Y_0$，因此又将此编码器称为8线-3线编码器。其输入/输出信号对应的编码关系见真值表3.7。

图3.14　编码器框图

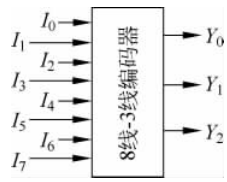


图3.15　3位二进制编码器框图

根据真值表，可以写出对应的输出表达式为

$$\begin{cases} Y_2 = I_4 + I_5 + I_6 + I_7 \\ Y_1 = I_2 + I_3 + I_6 + I_7 \\ Y_0 = I_1 + I_3 + I_5 + I_7 \end{cases}$$

表3.7　8线-3线二进制编码器真值表

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

由表达式可得出由或门构成的编码器电路，如图3.16(a)所示；也可以变换为与非表达式，得到由与非门构成的编码器电路，如图3.16(b)所示。

$$\begin{cases} Y_2 = \overline{\overline{I}_4 \cdot \overline{I}_5 \cdot \overline{I}_6 \cdot \overline{I}_7} \\ Y_1 = \overline{\overline{I}_2 \cdot \overline{I}_3 \cdot \overline{I}_6 \cdot \overline{I}_7} \\ Y_0 = \overline{\overline{I}_1 \cdot \overline{I}_3 \cdot \overline{I}_5 \cdot \overline{I}_7} \end{cases}$$

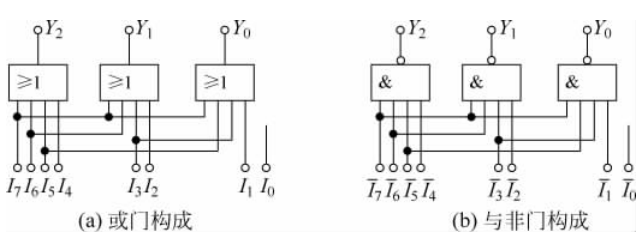


图3.16　3位二进制编码器逻辑图

2. 二进制优先编码器

在实际数字系统中,常常出现几个输入端同时加入输入信号的情况,因此需要编码器能够按照事先安排好的优先顺序,先对优先级高的信号进行编码。优先编码器就是根据优先顺序进行编码的电路。输入信号优先级的设定是设计者根据各个输入信号的轻重缓急而定的。优先编码器对输入信号没有严格要求,而且使用可靠、方便,所以应用最为广泛。

现在以 4 线-2 线二进制优先编码器为例,介绍一下二进制优先编码器的工作原理。其真值表见表 3.8。

根据真值表 3.8 可得如下逻辑表达式:

$$\begin{cases} Y_1 = I_3 + \overline{I_3} I_2 = I_3 + I_2 \\ Y_0 = I_3 + \overline{I_3}\ \overline{I_2} I_1 = I_3 + \overline{I_2} I_1 \end{cases}$$

表 3.8 4 线-2 线优先编码器真值表

输入				输出	
I_3	I_2	I_1	I_0	Y_1	Y_0
1	×	×	×	1	1
0	1	×	×	1	0
0	0	1	×	0	1
0	0	0	1	0	0

由逻辑表达式得 4 线-2 线优先编码器逻辑图,见图 3.17。

3. 二-十进制编码器

二-十进制编码器就是能将 10 个输入信号 $I_0 \sim I_9$ 分别转换成对应 BCD 码的电路,即 10 线-4 线编码器,其框图见图 3.18。它也可以分为普通编码器和优先编码器两类。

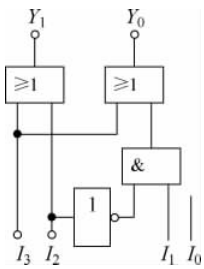


图 3.17 4 线-2 线优先编码器逻辑图

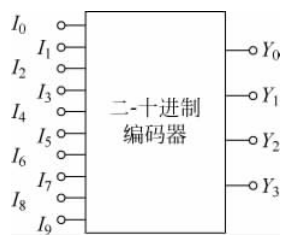


图 3.18 二-十进制编码器框图

二-十进制编码器的设计方法和二进制编码器类似,这里就不再重复。

3.2.2 集成电路编码器

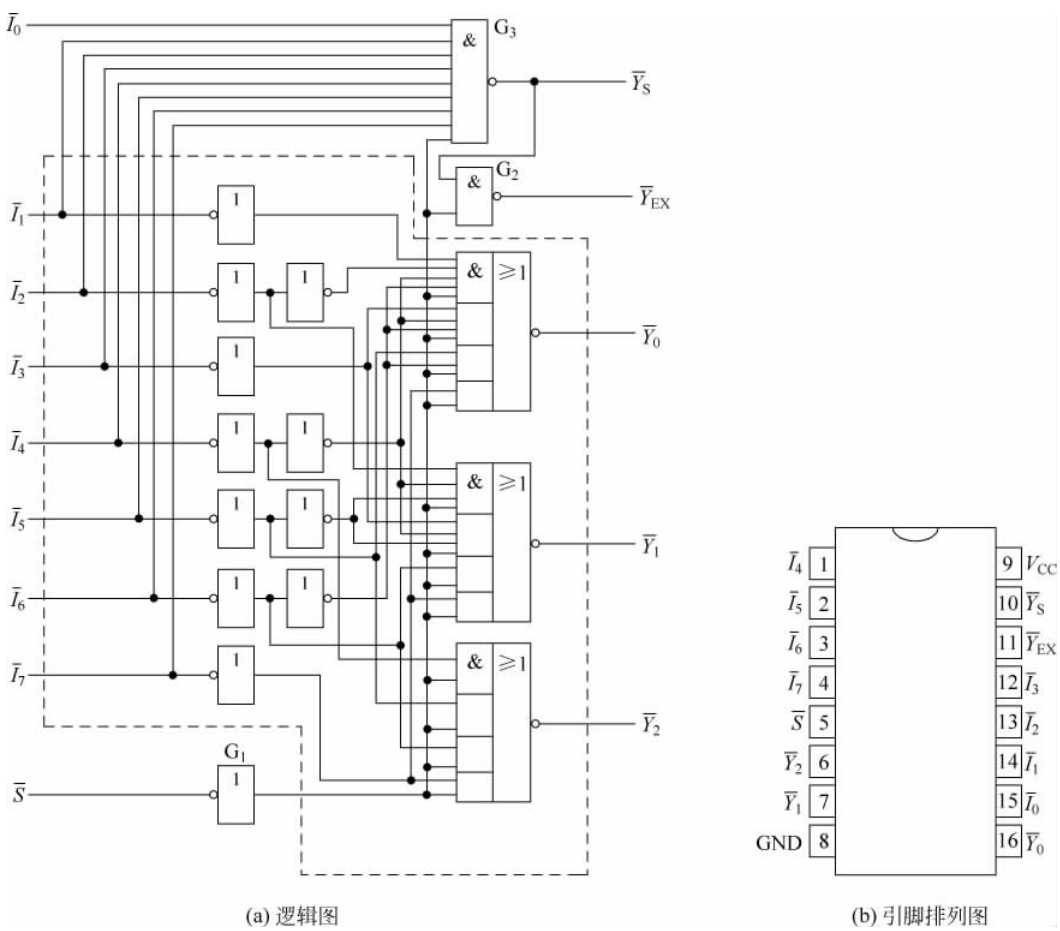
集成编码器芯片中,常用的有两种,即 8 线-3 线二进制优先编码器 74LS148 和 10 线-4 线二-十进制优先编码器 74LS147。

1. 8 线-3 线优先编码器 74LS148

表 3.9 列出了 8 线-3 线优先编码器 74LS148 的真值表，其逻辑图如图 3.19 所示。

表 3.9　74LS148 真值表

$\overline{S}$	输　入								输　出				
	$\overline{I}_0$	$\overline{I}_1$	$\overline{I}_2$	$\overline{I}_3$	$\overline{I}_4$	$\overline{I}_5$	$\overline{I}_6$	$\overline{I}_7$	$\overline{Y}_2$	$\overline{Y}_1$	$\overline{Y}_0$	$\overline{Y}_S$	$\overline{Y}_{EX}$
1	×	×	×	×	×	×	×	×	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	0	1
0	×	×	×	×	×	×	×	0	0	0	0	1	0
0	×	×	×	×	×	×	0	1	0	0	1	1	0
0	×	×	×	×	×	0	1	1	0	1	0	1	0
0	×	×	×	×	0	1	1	1	0	1	1	1	0
0	×	×	×	0	1	1	1	1	1	0	0	1	0
0	×	×	0	1	1	1	1	1	1	0	1	1	0
0	×	0	1	1	1	1	1	1	1	1	0	1	0
0	0	1	1	1	1	1	1	1	1	1	1	1	0



(a) 逻辑图　　(b) 引脚排列图

图 3.19　8 线-3 线优先编码器 74LS148 的逻辑图和引脚排列图

在图 3.19 中，$\overline{Y}_2$、$\overline{Y}_1$、$\overline{Y}_0$ 为三个输出端，低电平有效；$\overline{I}_7 \sim \overline{I}_0$ 为 8 个输入端，也是低电平有效。$\overline{I}_7$ 的优先级最高，$\overline{I}_0$ 的优先级最低。当$\overline{I_7}=0$ 时，不管其他输入端有无输入信号，都输出 $\overline{Y}_2\ \overline{Y}_1\ \overline{Y}_0=000$；当$\overline{I}_7$ 端输入无效，即$\overline{I_7}=1$，$\overline{I_6}=0$ 时，输出 $\overline{Y}_2\ \overline{Y}_1\ \overline{Y}_0=001$。依次类推，对其他输入信号按优先顺序编码，表 3.9 中×表示取任意值。由表 3.9 可得输出端的表达式如下

$$\begin{cases}\overline{Y}_2 = \overline{(I_7+I_6+I_5+I_4)\cdot S}\\ \overline{Y}_1 = \overline{(I_7+I_6+\overline{I}_5\overline{I}_4I_3+\overline{I}_5\overline{I}_4I_2)\cdot S}\\ \overline{Y}_0 = \overline{(I_7+\overline{I}_6I_5+\overline{I}_6\overline{I}_4I_3+\overline{I}_6\overline{I}_4\overline{I}_2I_1)\cdot S}\end{cases}$$

同时，为了增加电路的灵活性，扩展电路的功能，74LS148 电路中集成了三个控制端。$\overline{S}$ 为使能输入端，只有当 $\overline{S}=0$ 时，编码器才能正常工作；当 $\overline{S}=1$ 时，各输出门被封锁，输出均为高电平。$\overline{Y_S}$为使能输出端，当 $\overline{S}=0$ 时，只有$\overline{I}_7 \sim \overline{I}_0$ 都无信号输入(即$\overline{I}_7 \sim \overline{I}_0$ 都为 1)的情况下，$\overline{Y_S}=0$。所以在多片 74LS148 进行扩展连接时，通常将高位芯片的$\overline{Y_S}$端和低位芯片的 $\overline{S}$ 相连，当高位芯片无输入信号时，启动低位芯片工作。$\overline{Y}_{EX}$为优先编码器工作状态标志，当$\overline{I}_7 \sim \overline{I}_0$ 中无低电平输入信号或只有$\overline{I}_0$ 为低电平输入信号时，$\overline{Y}_2\ \overline{Y}_1\ \overline{Y}_0$ 均为 111，出现了输入条件不同而输出代码相同的情况，此时可以由 $\overline{Y}_{EX}$的状态来区别。$\overline{Y}_{EX}=0$ 表示有编码信号输入，此时$\overline{Y}_2\ \overline{Y}_1\ \overline{Y}_0=111$ 为编码输出，即响应$\overline{I}_0$ 的输入信号；而 $\overline{Y}_{EX}=1$ 表示无信号输入，此时$\overline{Y}_2\ \overline{Y}_1\ \overline{Y}_0=111$ 为非编码输出。$\overline{Y_S}$、$\overline{Y}_{EX}$的表达式如下

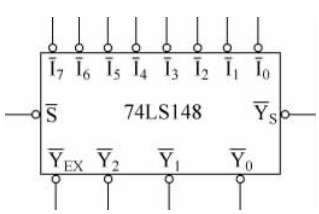


图 3.20　74LS148 的逻辑符号

$$\overline{Y_S} = \overline{\overline{I_0}\cdot\overline{I_1}\cdot\overline{I_2}\cdot\overline{I_3}\cdot\overline{I_4}\cdot\overline{I_5}\cdot\overline{I_6}\cdot\overline{I_7}\cdot S}$$

$$\overline{Y_{EX}} = \overline{(I_0+I_1+I_2+I_3+I_4+I_5+I_6+I_7)S}$$

画逻辑图时经常要使用 74LS148 的逻辑符号，如图 3.20 所示。

例 3.7　试用两片 8 线-3 线优先编码器 74LS148 扩展成 16 线-4 线优先编码器。

解：设$\overline{A}_{15} \sim \overline{A}_0$ 为 16 个低电平输入信号，$\overline{A}_{15}$的优先级最高，$\overline{A}_0$ 的优先级最低。编码为 16 个 4 位二进制代码 0000～1111。$\overline{Z}_3 \sim \overline{Z}_0$ 为 4 个低电平输出端。

因为每片的 74LS148 只有 8 个编码输入，所以需要将 16 个输入信号分别接在两个芯片上。先将$\overline{A}_{15} \sim \overline{A}_8$ 接到第一片的$\overline{I}_7 \sim \overline{I}_0$ 输入端，$\overline{A}_7 \sim \overline{A}_0$ 接到第二片的$\overline{I}_7 \sim \overline{I}_0$ 输入端。显然，根据优先顺序，只有第一片无编码输入时，第二片才能工作。这样，只要将一片的使能输出端$\overline{Y_S}$和第二片的使能输入端 $\overline{S}$ 相连就可以了。

另外，每片 74LS148 各有三个输出端$\overline{Y}_2$、$\overline{Y}_1$、$\overline{Y}_0$，不能组成 4 位输出二进制代码，所以输出端必须进行扩展。由于第一片芯片有编码输入信号时，其 $\overline{Y}_{EX}=0$，无编码输入时 $\overline{Y}_{EX}=1$，因此可见，整个编码器 4 位输出代码的最高位取值和第一片编码器有无信号有关，这样第一片的 $\overline{Y}_{EX}$作为代码输出的最高位$\overline{Z}_3$ 即可，低 3 位可由两片芯片的输出$\overline{Y}_2$、$\overline{Y}_1$、$\overline{Y}_0$ 通过“与”逻辑完成。

通过上述分析可得到该电路的逻辑图，如图 3.21 所示。

2. 10 线-4 线优先编码器 74LS147

常用集成编码器还有二-十进制优先编码器 74LS147，其逻辑图如图 3.22 所示。由

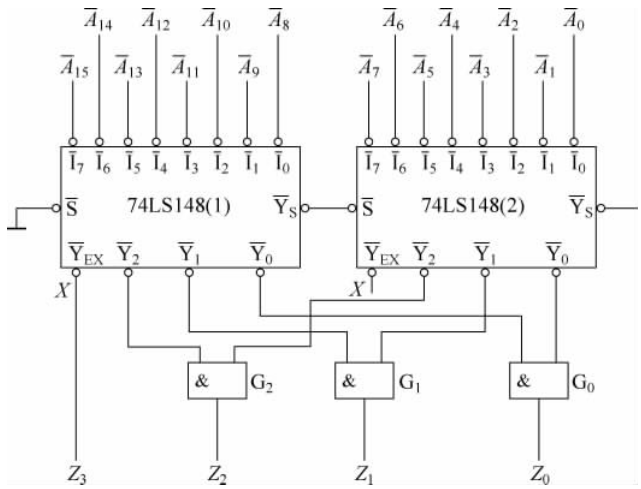


图 3.21　用两片 74LS148 组成的 16 线-4 线优先编码器逻辑图

图 3.22 可得其输出表达式：

$$\begin{cases} \overline{Y_3} = \overline{I_8 + I_9} \\ \overline{Y_2} = \overline{I_7\,\overline{I_8}\,\overline{I_9} + I_6\,\overline{I_8}\,\overline{I_9} + I_5\,\overline{I_8}\,\overline{I_9} + I_4\,\overline{I_8}\,\overline{I_9}} \\ \overline{Y_1} = \overline{I_7\,\overline{I_8}\,\overline{I_9} + I_6\,\overline{I_8}\,\overline{I_9} + I_3\,\overline{I_4}\,\overline{I_5}\,\overline{I_8}\,\overline{I_9} + I_2\,\overline{I_4}\,\overline{I_5}\,\overline{I_8}\,\overline{I_9}} \\ \overline{Y_0} = \overline{I_9 + I_7\,\overline{I_8}\,\overline{I_9} + I_5\,\overline{I_6}\,\overline{I_8}\,\overline{I_9} + I_3\,\overline{I_4}\,\overline{I_6}\,\overline{I_8}\,\overline{I_9} + I_1\,\overline{I_2}\,\overline{I_4}\,\overline{I_6}\,\overline{I_8}\,\overline{I_9}} \end{cases}$$

由图 3.22 可得 74LS147 的真值表，如表 3.10 所示。

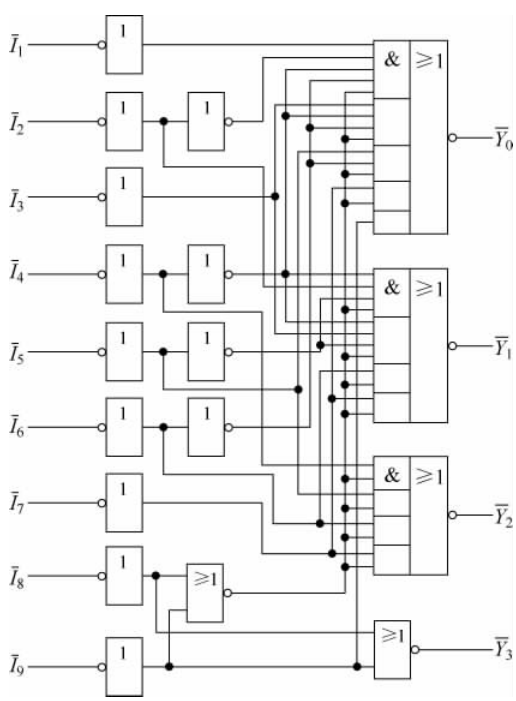


图 3.22　10 线-4 线优先编码器 74LS147 逻辑图

表 3.10 74LS147 的真值表

$\overline{I_0}$	$\overline{I_1}$	$\overline{I_2}$	$\overline{I_3}$	$\overline{I_4}$	$\overline{I_5}$	$\overline{I_6}$	$\overline{I_7}$	$\overline{I_8}$	$\overline{I_9}$	$\overline{Y_3}$	$\overline{Y_2}$	$\overline{Y_1}$	$\overline{Y_0}$
1	1	1	1	1	1	1	1	1	1	1	1	1	1
×	×	×	×	×	×	×	×	×	0	0	1	1	0
×	×	×	×	×	×	×	×	0	1	0	1	1	1
×	×	×	×	×	×	×	0	1	1	1	0	0	0
×	×	×	×	×	×	0	1	1	1	1	0	0	1
×	×	×	×	×	0	1	1	1	1	1	0	1	0
×	×	×	×	0	1	1	1	1	1	1	0	1	1
×	×	×	0	1	1	1	1	1	1	1	1	0	0
×	×	0	1	1	1	1	1	1	1	1	1	0	1
×	0	1	1	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1

思考题

在需要使用普通编码器的场合,能否使用优先编码器代替?在需要使用优先编码器的场合,能否使用普通编码器代替?

3.3 译码器/数据分配器

编码器是将有特定意义的信息(如文字、符号、图像等)转换成相应的二进制代码。译码器与之相反,是将二进制代码"翻译"出来,还原成有特定意义的输出信息。

3.3.1 译码器的结构

如图 3.23 所示为译码器的框图。输入为 n 位二进制代码 $A_{n-1}A_{n-2}\cdots A_1A_0$,输出为 m 个信号 $Y_0,Y_1,\cdots,Y_{m-1}$,n 与 m 之间应满足关系

$$m \leqslant 2^n \tag{3.1}$$

如果 $m=2^n$,则称该译码器为全译码器,如二进制译码器 2 线-4 线译码器、3 线-8 线译码器、4 线-16 线译码器等;如果 $m<2^n$,则称该译码器为部分译码器,如二-十进制译码器(也称为 4 线-10 线译码器)等。

下面以 2 线-4 线译码器为例来分析一下译码器的工作原理和电路结构。

图 3.24 为 2 线-4 线译码器的逻辑符号。

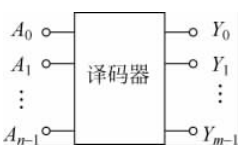


图 3.23 译码器框图

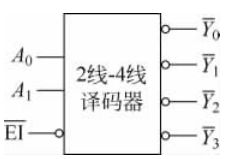


图 3.24 2 线-4 线译码器逻辑符号

表 3.11 为 2 线-4 线译码器的真值表,它有两个输入变量 A_1,A_0,4 个输入代码 00、01、10、11,可翻译出 4 个输出信号 $Y_0\sim Y_3$。输出可以是高电平有效,也可以是低电平有效,这

里是以低电平有效为例。表中$\overline{EI}$为使能输入端，低电平有效。

由表 3.11 可以很容易写出输出端的表达式，由此得到如图 3.25 所示的逻辑图。

$$\begin{cases}\overline{Y_3} = \overline{EI}A_1A_0 \\ \overline{Y_2} = \overline{EI}A_1\overline{A}_0 \\ \overline{Y_1} = \overline{EI}\overline{A}_1A_0 \\ \overline{Y_0} = \overline{EI}\overline{A}_1\overline{A}_0\end{cases}$$

表 3.11　2 线-4 线译码器真值表

输　入			输　出			
$\overline{EI}$	A_1	A_0	$\overline{Y_0}$	$\overline{Y_1}$	$\overline{Y_2}$	$\overline{Y_3}$
1	×	×	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

由于对每一组可能的输入代码，译码器仅有一个输出信号有效，也可以将译码器称作最小项译码器，即每个输出端对应于一个最小项。在数字系统中译码器有着广泛的应用，如各种显示译码器、用译码器实现的数据分配器、存储器中的地址译码器和控制器中的指令译码器等。

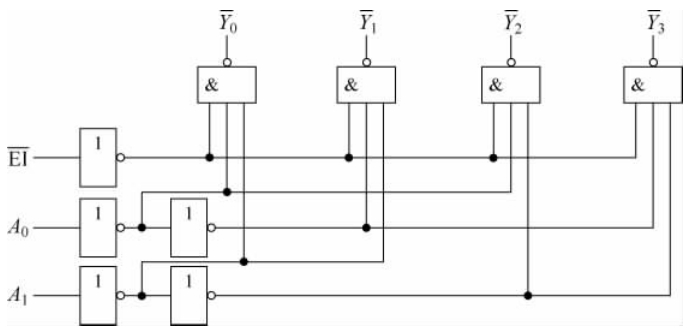


图 3.25　2 线-4 线译码器逻辑图

3.3.2　集成电路译码器

译码器的中规模集成电路有很多种，这里主要介绍三种通用的译码器：3 线-8 线译码器 74LS138、4 线-10 线译码器 74LS42 和显示译码器 74LS48。

1. 3 线-8 线译码器 74LS138

74LS138 是一种应用很广泛的二进制译码器，图 3.26 为其逻辑图及引脚图。它有 3 个代码输入端 A_2，A_1，A_0，8 个低电平信号输出端$\overline{Y_7}$～$\overline{Y_0}$，因此属于全译码器。

由图 3.26 可写出 74LS138 的输出表达式：

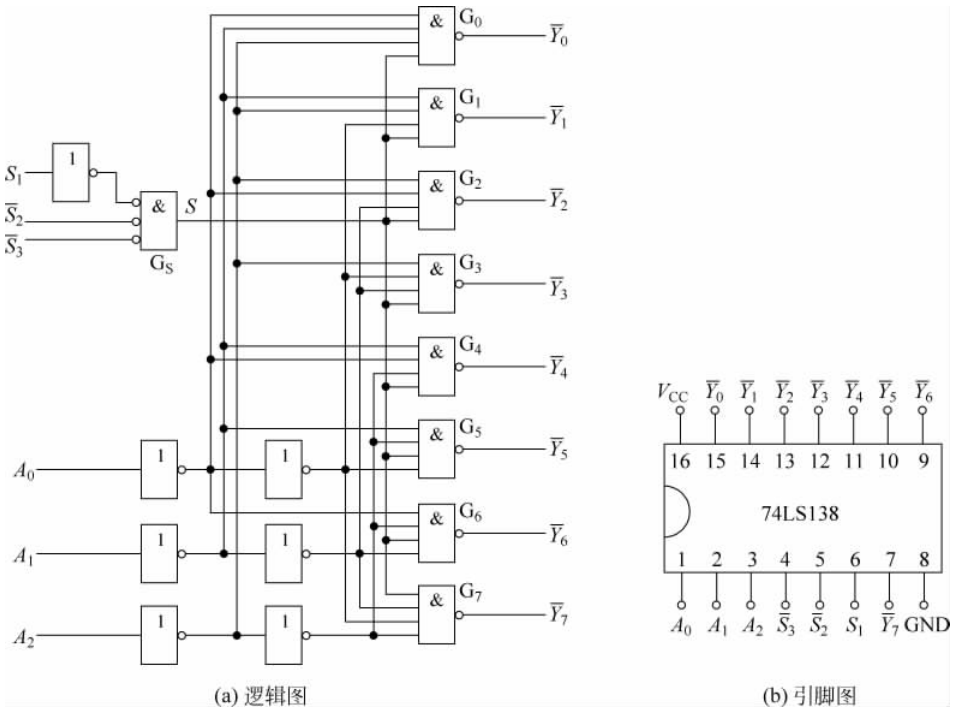


图 3.26　3 线-8 线译码器 74LS138

$$\begin{cases}\overline{Y_7}=\overline{A_2A_1A_0}=\overline{m_7}\\ \overline{Y_6}=\overline{A_2A_1\overline{A}_0}=\overline{m_6}\\ \overline{Y_5}=\overline{A_2\overline{A}_1A_0}=\overline{m_5}\\ \overline{Y_4}=\overline{A_2\overline{A}_1\overline{A}_0}=\overline{m_4}\\ \overline{Y_3}=\overline{\overline{A}_2A_1A_0}=\overline{m_3}\\ \overline{Y_2}=\overline{\overline{A}_2A_1\overline{A}_0}=\overline{m_2}\\ \overline{Y_1}=\overline{\overline{A}_2\overline{A}_1A_0}=\overline{m_1}\\ \overline{Y_0}=\overline{\overline{A}_2\overline{A}_1\overline{A}_0}=\overline{m_0}\end{cases}$$

此外，74LS138 还有三个附加控制端 S_1、$\overline{S_2}$、$\overline{S_3}$，为使能输入端。只有当三个使能端都有效，即 $S_1=1$ 且 $\overline{S}_2+\overline{S}_3=0$ 时，译码器才能正常工作；否则译码器被禁止，所有的输出都为高电平，如表 3.12 所示。利用这些附加控制端可以很方便地进行译码器的扩展。

表 3.12　74LS138 真值表

输　入					输　出							
S_1	$\overline{S}_2+\overline{S}_3$	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
0	×	×	×	×	1	1	1	1	1	1	1	1
×	1	×	×	×	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1

续表

输　入					输　出							
S_1	$\bar{S}_2+\bar{S}_3$	A_2	A_1	A_0	$\bar{Y}_0$	$\bar{Y}_1$	$\bar{Y}_2$	$\bar{Y}_3$	$\bar{Y}_4$	$\bar{Y}_5$	$\bar{Y}_6$	$\bar{Y}_7$
1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0

例 3.8　试用两片 3 线-8 线译码器 74LS138 组成 4 线-16 线译码器，将输入的 4 位二进制代码 $D_3D_2D_1D_0$ 译成 16 个独立的低电平信号 $\bar{Y}_0\sim\bar{Y}_{15}$。

解：因为 74LS138 只有 3 个代码输入端 A_0、A_1、A_2，所以想进行 4 位代码的译码，必须利用芯片附加控制端作为第四个代码输入端。

取第一片 74*LS*138 作为低位芯片，第二片 74LS138 作为高位芯片。将高位芯片的 S_1 和低位芯片的$\overline{S_2}$、$\overline{S_3}$相连，作为第四个代码输入端 D_3，使两片芯片的 $A_2=D_2$、$A_1=D_1$、$A_0=D_0$；同时使高位芯片的$\overline{S_2}$、$\overline{S_3}$都接地，低位芯片的 S_1 接＋5V 电源，如图 3.27 所示，便得到了所求的 4 线-16 线译码器。

由图 3.27 可以看出，当 $D_3=0$ 时，低位芯片工作而高位芯片禁止，这样就将输入代码 0000～0111 从低位芯片的 8 个输出端$\overline{Y_0}\sim\overline{Y_7}$输出；当 $D_3=1$ 时，高位芯片工作而低位芯片禁止，这样就将输入代码 1000～1111 从高位芯片的 8 个输出端$\overline{Y_8}\sim\overline{Y_{15}}$输出。综合两个芯片的译码过程，便构成 4 线-16 线译码器。

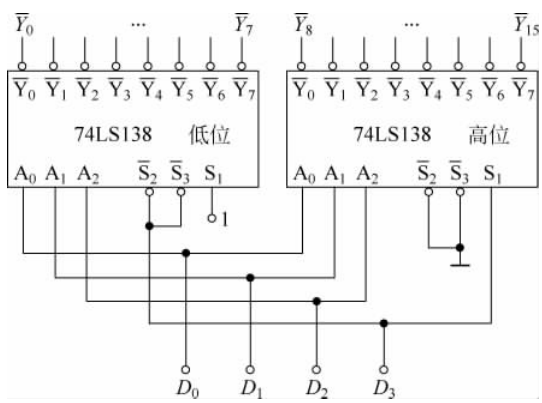


图 3.27　用两片 74LS138 组成的 4 线-16 线译码器

由于二进制译码器的每一个输出端都对应一个最小项，而任何一个逻辑函数都可以变换为最小项之和的形式，因此利用译码器和一些附加的门电路就可以实现组合逻辑函数。

例 3.9　用集成译码器实现逻辑函数 $Z=AB+BC+AC$。

解：

(1) 根据逻辑函数选择译码器。由于逻辑函数 Z 中含有 A、B、C 三个输入变量，所以应选用 3 线-8 线译码器 74LS138。

(2) 写出函数的标准与或式,再转换为与非-与非式。

$$\begin{aligned} Z &= AB + BC + AC \\ &= ABC + AB\overline{C} + \overline{A}BC + A\overline{B}C \\ &= m_3 + m_5 + m_6 + m_7 \\ &= \overline{\overline{m}_3 \cdot \overline{m}_5 \cdot \overline{m}_6 \cdot \overline{m}_7} \end{aligned}$$

(3) 确认函数变量和译码器输入的关系。

令

$$A_2 = A, \quad A_1 = B, \quad A_0 = C$$

则

$$Z = \overline{\overline{Y}_3 \cdot \overline{Y}_5 \cdot \overline{Y}_6 \cdot \overline{Y}_7}$$

(4) 画出连线图,如图 3.28 所示。

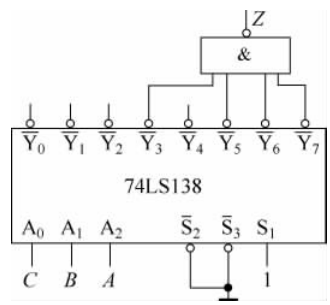


图 3.28 例 3.9 的逻辑图

例 3.10 用 3 线-8 线译码器 74LS138 设计一个 1 位二进制减法器。

解:

(1) 逻辑抽象。

1 位二进制减法器应能进行被减数 A_i 与减数 B_i 和来自低位的借位信号 C_i 相减,并给出差 D_i 和产生的向高位借位信号 C_{i+1}。

(2) 列真值表。

根据减法器的功能,列出真值表,如表 3.13。

(3) 写出逻辑函数表达式并化简。

根据设计要求用 74LS138 实现减法器,因此只需写出最小项表达式。

$$\begin{aligned} D_i &= \overline{A}_i\,\overline{B}_i C_i + \overline{A}_i B_i\,\overline{C}_i + A_i\,\overline{B}_i\,\overline{C}_i + A_i B_i C_i \\ &= \overline{\overline{\overline{A}_i\,\overline{B}_i C_i} \cdot \overline{\overline{A}_i B_i\,\overline{C}_i} \cdot \overline{A_i\,\overline{B}_i\,\overline{C}_i} \cdot \overline{A_i B_i C_i}} \\ &= \overline{\overline{m_1} \cdot \overline{m_2} \cdot \overline{m_4} \cdot \overline{m_7}} = \overline{\overline{Y_1} \cdot \overline{Y_2} \cdot \overline{Y_4} \cdot \overline{Y_7}} \\ C_{i+1} &= \overline{A}_i\,\overline{B}_i C_i + \overline{A}_i B_i\,\overline{C}_i + \overline{A}_i B_i C_i + A_i B_i C_i \\ &= \overline{\overline{\overline{A}_i\,\overline{B}_i C_i} \cdot \overline{\overline{A}_i B_i\,\overline{C}_i} \cdot \overline{\overline{A}_i B_i C_i} \cdot \overline{A_i B_i C_i}} \\ &= \overline{\overline{m_1} \cdot \overline{m_2} \cdot \overline{m_3} \cdot \overline{m_7}} = \overline{\overline{Y_1} \cdot \overline{Y_2} \cdot \overline{Y_3} \cdot \overline{Y_7}} \end{aligned}$$

表 3.13 例 3.10 真值表

输入变量			输出变量	
A_i	B_i	C_i	D_i	C_{i+1}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0

续表

输入变量			输出变量	
A_i	B_i	C_i	D_i	C_{i+1}
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

（4）画出逻辑图。

将 74LS138 的输入端接减法器的输入端，即 $A_i=A_2$、$B_i=A_1$、$C_i=A_0$，减法器的输出信号 D_i、C_{i+1}分别通过与非门与译码器的相应输出端相连，即可实现 1 位二进制减法器，如图 3.29 所示。

例 3.11　用 3 线-8 线集成译码器 74LS138 和 8 线-3 线集成编码器 74LS148 实现 3 位格雷码向二进制数转换的电路。

解：用 3 线-8 线集成译码器 74LS138 和 8 线-3 线集成编码器 74LS148 实现 3 位格雷码向二进制数转换，可以利用十进制数为媒介，先用译码器将输入的格雷码转换为十进制数，再通过编码器对十进制数进行编码，将其转换成二进制数。其真值表如表 3.14 所示。

表 3.14　例 3.11 真值表

输入变量			十进制数	输出变量		
G_2	G_1	G_0		B_2	B_1	B_0
0	0	0	0	0	0	0
0	0	1	1	0	0	1
0	1	1	2	0	1	0
0	1	0	3	0	1	1
1	1	0	4	1	0	0
1	1	1	5	1	0	1
1	0	1	6	1	1	0
1	0	0	7	1	1	1

从真值表可以看出，只要将译码器的输出端按图 3.30 所示与编码器的输入端对应相连，编码器的输出就是二进制数。

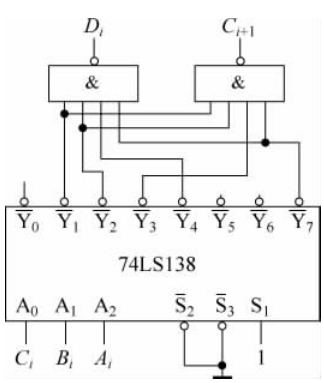


图 3.29　例 3.10 逻辑图

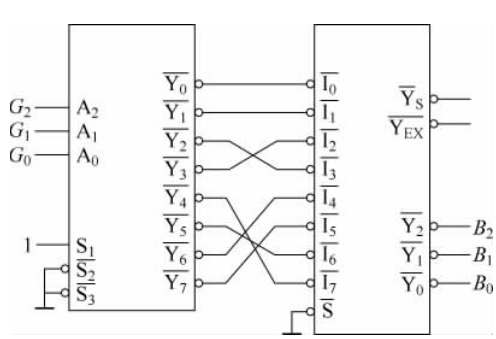


图 3.30　例 3.11 逻辑图

2. 4 线-10 线译码器 74LS42

74LS42 是一种二-十进制译码器,其输入是 4 位 BCD 码,输出是 10 个高、低电平信号,所以也称其为 4 线-10 线译码器。图 3.31 为 74LS42 的逻辑图。

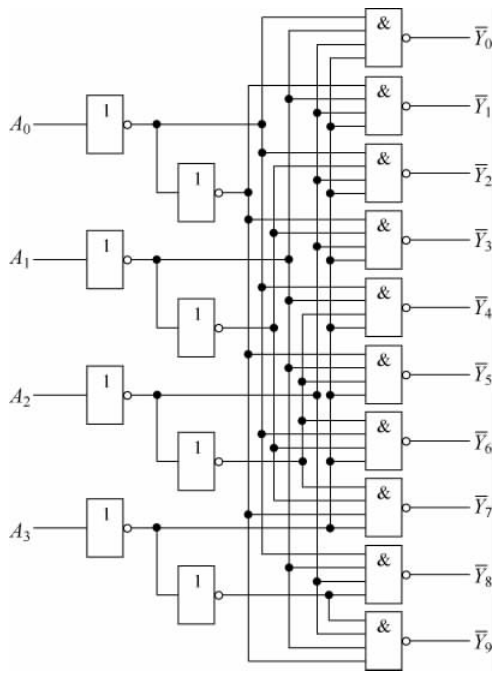


图 3.31 74LS42 逻辑图

根据图 3.31 可得到 74LS42 的输出表达式:

$$\begin{cases}
\overline{Y_0} = \overline{\overline{A_3}\ \overline{A_2}\ \overline{A_1}\ \overline{A_0}} \\
\overline{Y_1} = \overline{\overline{A_3}\ \overline{A_2}\ \overline{A_1}A_0} \\
\overline{Y_2} = \overline{\overline{A_3}\ \overline{A_2}A_1\ \overline{A_0}} \\
\overline{Y_3} = \overline{\overline{A_3}\ \overline{A_2}A_1A_0} \\
\overline{Y_4} = \overline{\overline{A_3}A_2\ \overline{A_1}\ \overline{A_0}} \\
\overline{Y_5} = \overline{\overline{A_3}A_2\ \overline{A_1}A_0} \\
\overline{Y_6} = \overline{\overline{A_3}A_2A_1\ \overline{A_0}} \\
\overline{Y_7} = \overline{\overline{A_3}A_2A_1A_0} \\
\overline{Y_8} = \overline{A_3\ \overline{A_2}\ \overline{A_1}\ \overline{A_0}} \\
\overline{Y_9} = \overline{A_3\ \overline{A_2}\ \overline{A_1}A_0}
\end{cases}$$

该电路的真值表如表 3.15 所示。由表可以看出,当输入信号为 1010～1111 时,译码器输出都为高电平,即译码器拒绝“翻译”。

表 3.15　4 线-10 线译码器 74LS42 的真值表

序号	输　入				输　出									
	A_3	A_2	A_2	A_0	$\bar{Y}_0$	$\bar{Y}_1$	$\bar{Y}_2$	$\bar{Y}_3$	$\bar{Y}_4$	$\bar{Y}_5$	$\bar{Y}_6$	$\bar{Y}_7$	$\bar{Y}_8$	$\bar{Y}_9$
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
伪码	1	0	1	0	1	1	1	1	1	1	1	1	1	1
	1	0	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	0	0	1	1	1	1	1	1	1	1	1	1
	1	1	0	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	0	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1

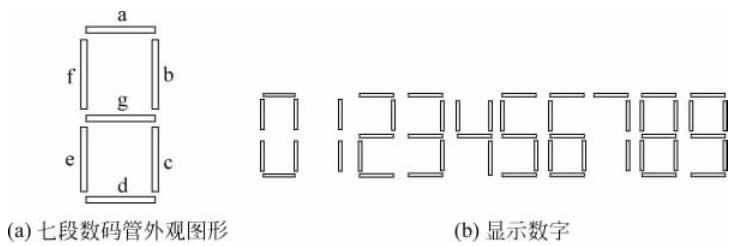
3. 显示译码器

在数字系统中，常需要将数字或运算结果显示出来，以便于人们观测、查看及监测数字系统的工作情况。显示译码器就是将二进制代码译成数字、文字、符号并加以显示的电路，由译码器、驱动器和显示器组成。显示译码器用来驱动各种显示器件，目前常用的显示器件有三种：七段数码显示器、点阵式显示器和液晶显示器。下面以七段数码管显示器为例，介绍一下七段数码显示器的工作原理及其驱动电路——BCD 七段译码器。

(1) 七段数码显示器

七段数码显示器又称七段数码管，是由七段可发光材料构成的，当给其中的某段加一定的驱动电压或电流时，相应材料段就会发光；利用不同发光段的组合，就可以显示出 0～9 十个数字。根据发光材料不同，可分为荧光、液晶(LCD)和发光二极管(LED)等多种七段数码管。目前应用最广泛的主要是发光二极管七段显示器。

如图 3.32 所示，a～g 每一段都是一个发光二极管，驱动不同的二极管发光就可以组成不同的数字。



(a) 七段数码管外观图形　　(b) 显示数字

图 3.32　七段数码管及其显示的数字图形

根据连接方式不同,七段数码管分为共阴极和共阳极两种。图 3.33(a)为共阳极形式,即将 7 个发光二极管的阳极连在一起作为公共端,使用时阴极接低电平的那些发光二极管发光,显示字形;如图 3.33(b)所示为共阴极形式,即将 7 个发光二极管的阴极连在一起作为公共端,使用时阳极接高电平的那些发光二极管发光,显示字形。使用时改变发光二极管两端的电压差就可以调整亮度。

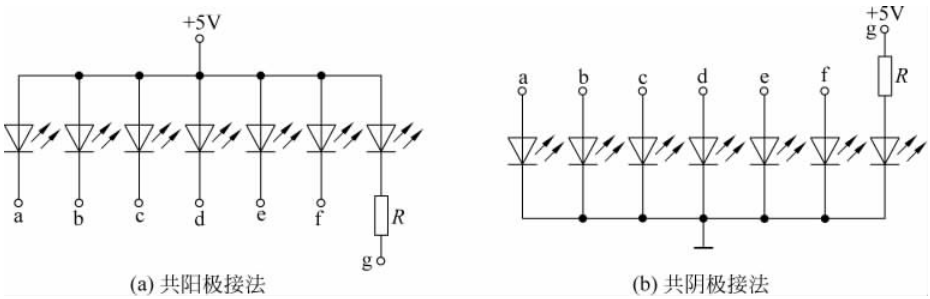


图 3.33 七段数码管的连接方式

LED 数码显示器的特点是工作电压较低(1.5～3V),体积小,寿命长(大于 1000h),响应速度快(1～100ns)、颜色丰富等。

(2) 七段显示译码器 74LS48

驱动七段数码管的译码器称为 BCD 七段译码器,它将 BCD 码变换成十进制数并在数码管上显示出来。图 3.34 为中规模集成 BCD 七段译码器 74LS48 的逻辑图,其真值表如表 3.16 所示。

表 3.16 BCD 七段译码器 74LS48 的真值表

数字	输入				输出						
	A_3	A_2	A_1	A_0	Y_a	Y_b	Y_c	Y_d	Y_e	Y_f	Y_g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
10	1	0	1	0	0	0	0	1	1	0	1
11	1	0	1	1	0	0	1	1	0	0	1
12	1	1	0	0	0	1	0	0	0	1	1
13	1	1	0	1	1	0	0	1	0	1	1
14	1	1	1	0	0	0	0	1	1	1	1
15	1	1	1	1	0	0	0	0	0	0	0

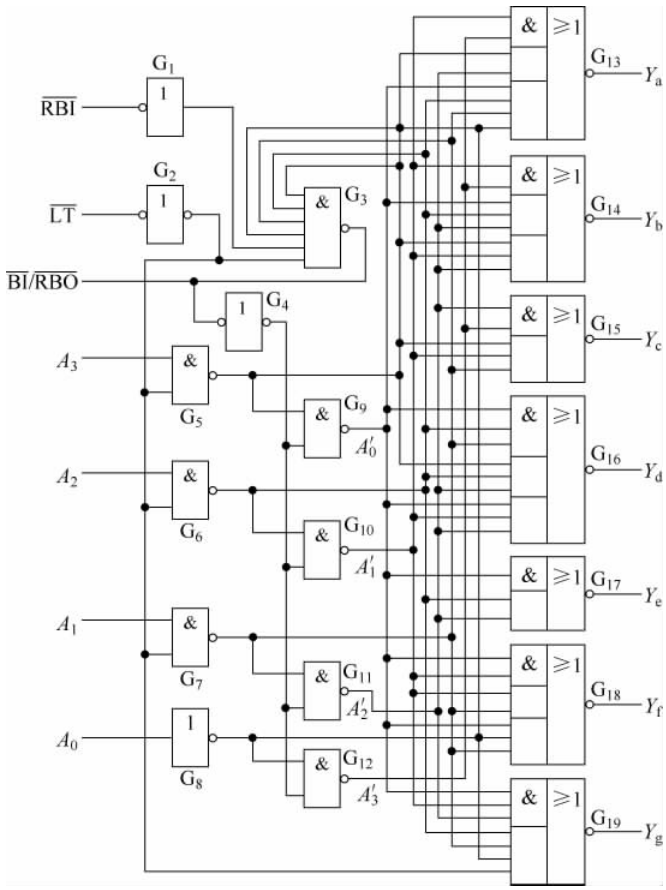


图 3.34　BCD 七段译码器 74LS48 的逻辑图

由图 3.34 可以看出，A_3，A_2，A_1，A_0 为 74LS48 译码器的输入端，Y_a～Y_g 为其输出端。此外，74LS48 还有灭灯输入/灭零输出（$\overline{BI}/\overline{RBO}$，该端既可作$\overline{BI}$输入又可作$\overline{RBO}$输出）、灭零输入$\overline{RBI}$和试灯输入$\overline{LT}$等输入/输出端。其作用如下。

试灯输入端$\overline{LT}$：低电平有效。当$\overline{LT}=0$时，数码管的七段全亮，与输入的译码信号无关。所以该输入端可以用来测试数码管的好坏。

灭零输入端$\overline{RBI}$：低电平有效。当$\overline{LT}=1$，$\overline{RBI}=0$且译码输入全为 0 时，该位输出不显示，即 0 字被熄灭；当译码输入不全为 0 时，该位正常显示。所以该输入端可以用来消隐无效的 0。

灭灯输入/灭零输出$\overline{BI}/\overline{RBO}$：当$\overline{BI}/\overline{RBO}$作为输入使用，且$\overline{BI}/\overline{RBO}=0$时，数码管七段全灭，与译码输入无关。当$\overline{BI}/\overline{RBO}$作为输出使用时，受控于$\overline{LT}$和$\overline{RBI}$，当$\overline{LT}=1$，$\overline{RBI}=0$时，$\overline{BI}/\overline{RBO}=0$，其他情况下$\overline{BI}/\overline{RBO}=1$。所以该控制端主要用于显示多位数字时多个译码器之间的连接。

图 3.35 为有灭零控制的 8 位数码显示系统。将$\overline{RBO}$和$\overline{RBI}$配合使用，可以实现多位十进制数码显示器整数前和小数点后的灭零控制。只要在整数部分把高位的$\overline{RBO}$与低位的$\overline{RBI}$相连，在小数部分将低位的$\overline{RBO}$与高位的$\overline{RBI}$相连，就可以把前、后多余的零熄灭（这里

用的数码管为增设了一个小数点的 BS201A 芯片)。

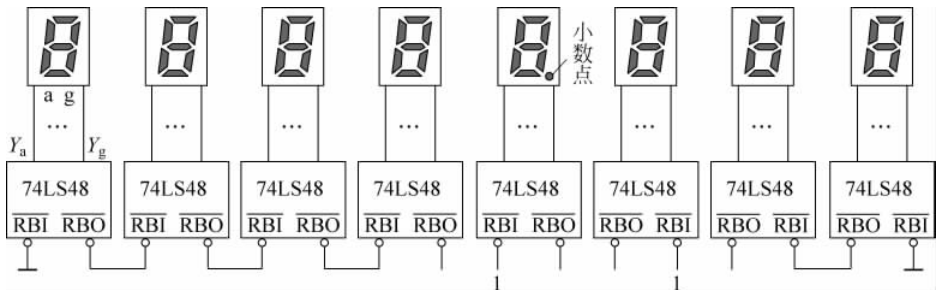


图 3.35 有灭零控制的 8 位数码显示系统

3.3.3 数据分配器

在数据传输的过程中,有时需要将数据分配到不同的数据通道上,能够完成这种功能的电路称为数据分配器,也称为多路分配器。

如图 3.36 所示为 1 路-4 路数据分配器框图,表 3.17 为其真值表。

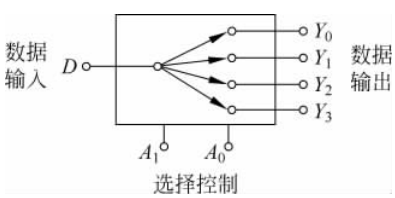


图 3.36 1 路-4 路数据分配器框图

表 3.17 1 路-4 路数据分配器真值表

A_1	A_0	Y_0	Y_1	Y_2	Y_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

由真值表 3.17 可以看出,当选择控制端 A_1A_0 为 00 时将数据 D 分配到 Y_0 端输出,为 01 时,数据 D 分配到 Y_1 端输出,依此类推。根据真值表可得数据分配器的输出表达式:

$$\begin{cases} Y_0 = D \cdot \overline{A}_1\overline{A}_0 \\ Y_1 = D \cdot \overline{A}_1 A_0 \\ Y_2 = D \cdot A_1\overline{A}_0 \\ Y_3 = D \cdot A_1 A_0 \end{cases}$$

根据输出表达式可以得到其逻辑图,如图 3.37 所示。

数据分配器实际上都是由译码器来实现的,具体做法是:将传送的数据接至译码器的使能端,就可以通过改变译码器的输入,把数据分配到不同的通道上。

图 3.38 为由 3 线-8 线译码器 74LS138 构成的 8 路数据分配器。译码器输出$\overline{Y_0}$~$\overline{Y_7}$作为 8 路数据输出,译码器输入 A_2,A_1,A_0 作为 3 个选择控制端,从使能控制端 S_1,$\overline{S_2}$,$\overline{S_3}$中选择一个作为数据输入端

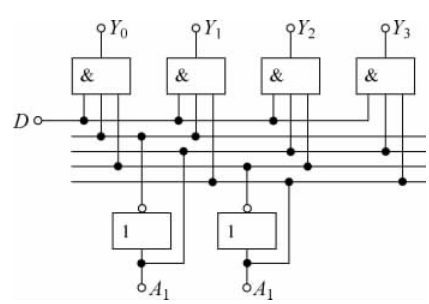


图 3.37 4 路数据分配器逻辑图

D。如果数据从 S_1 端输入时，则输出为反码形式；如果数据从$\overline{S_2}$或$\overline{S_3}$端输入时，则输出为原码形式。例如，如果数据从$\overline{S_2}$端输入，当 $S_1\ \overline{S_2}\ \overline{S_3}=100$ 时，译码器正常译码，此时选择控制端 A_2，A_1，A_0 选中的输出状态与$\overline{S_2}$相同（为 0）；当 $S_1\ \overline{S_2}\ \overline{S_3}=110$ 时，译码器不译码，此时选择控制端 A_2，A_1，A_0 选中的输出状态也与$\overline{S_2}$相同（为 1），满足了数据分配的功能。

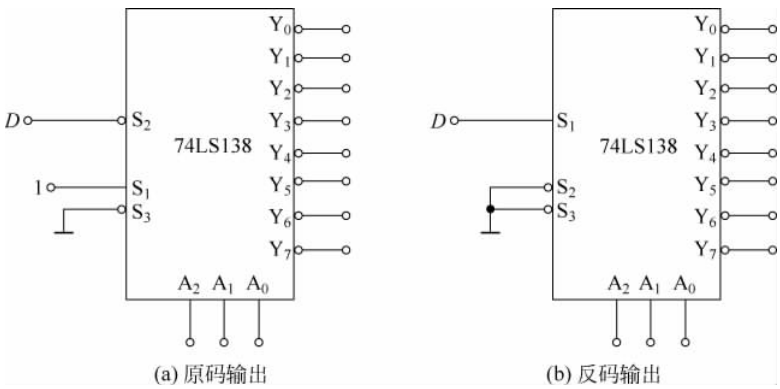


图 3.38　3 线-8 线译码器 74LS138 构成的 8 路数据分配器

思考题

1. 用二-十进制译码器附加门电路能否得到任意形式的四变量逻辑函数？为什么？

2. 用 4 线-16 线译码器（输入为 A_3、A_2、A_1、A_0，输出为$\overline{Y}_0\sim\overline{Y}_{15}$）能否取代图 3.27 所示的 3 线-8 线译码器？如果可以，应如何连线？

3.4　数据选择器

在数字信号的传输过程中，有时需要从一组输入数据中选出某一个，实现这种功能的器件称为数据选择器，又名多路选择器或多路开关。其功能是在 n 个选择控制信号的控制下，从 2^n 个输入信号中选择一个作为输出。

3.4.1　数据选择器的结构

现以 4 选 1 数据选择器为例介绍一下数据选择器的工作原理。其框图如图 3.39 所示，$D_0\sim D_3$ 为 4 路数据输入端，Y 为输出端，A_1，A_0 为选择控制信号，或称为地址信号。其真值表如表 3.18 所示。

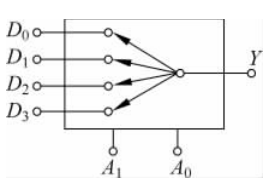


图 3.39　4 选 1 数据选择器框图

表 3.18　4 选 1 数据选择器真值表

D	A_1	A_0	Y
D_0	0	0	D_0
D_1	0	1	D_1
D_2	1	0	D_2
D_3	1	1	D_3

表 3.18 说明,当地址码 $A_1A_0=00$ 时选择 D_0 输出,当 $A_1A_0=01$ 时选择 D_1 输出,依此类推。4 选 1 数据选择器的输出表达式如下

$$Y = D_0\overline{A}_1\overline{A}_0 + D_1\overline{A}_1A_0 + D_2A_1\overline{A}_0 + D_3A_1A_0 \tag{3.2}$$

由输出表达式可得 4 选 1 数据选择器的逻辑图如图 3.40 所示。

3.4.2 集成电路数据选择器

作为一种集成器件,数据选择器的商品电路型号较多,这里主要介绍两类:双 4 选 1 数据选择器 74LS153 和 8 选 1 数据选择器 74LS151。

1. 双 4 选 1 数据选择器 74LS153

双 4 选 1 数据选择器 74LS153 的引脚图如图 3.41 所示,其逻辑图如图 3.42 所示,它是由两个完全相同的 4 选 1 数据选择器组成,图中用虚线框起。其中 $D_{13}\sim D_{10}$ 和 $D_{23}\sim D_{20}$ 为两个 4 选 1 数据选择器的输入端;$\overline{S}_1$、$\overline{S}_2$ 为两个使能输入端,低电平有效;A_1、A_0 为公共的地址信号输入端;Y_1、Y_2 为两个输出端。显然,上下两个 4 选 1 数据选择器是独立的,通过给定不同的地址码,从 4 路输入数据中选择 1 路输出。

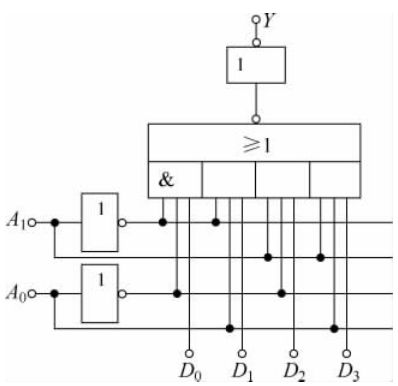


图 3.40 4 选 1 数据选择器的逻辑图

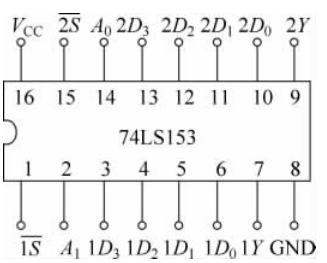


图 3.41 双 4 选 1 数据选择器 74LS153 的引脚图

用 1 片双 4 选 1 数据选择器 74LS153 可以构成一个 8 选 1 的数据选择器。因为 8 选 1 的数据选择器需要三位地址信号,而 74LS153 只有两个地址码 A_1,A_0,所以只能利用两个使能输入端$\overline{S}_1$,$\overline{S}_2$ 构成第三位地址码,如图 3.43 所示。

将高位地址码 A_2 直接与$\overline{S}_1$ 相连,将$\overline{A_2}$与$\overline{S}_2$ 相连,同时将两个数据选择器的输出端用或门连接,就得到了 8 选 1 的数据选择器。当 $A_2=0$ 时,上边的 4 选 1 数据选择器工作,通过给定的 A_1、A_0 的值,可从 $D_3\sim D_0$ 中选择某一路数据经过 G_2 门送到输出端 Y。反之,若 $A_2=1$,则下面的 4 选 1 数据选择器工作,通过给定的 A_1、A_0 的值,可从 $D_7\sim D_4$ 中选择某一路数据经过 G_2 门送到输出端 Y。

2. 8 选 1 数据选择器 74LS151

8 选 1 数据选择器 74LS151 的真值表如表 3.19 所示。

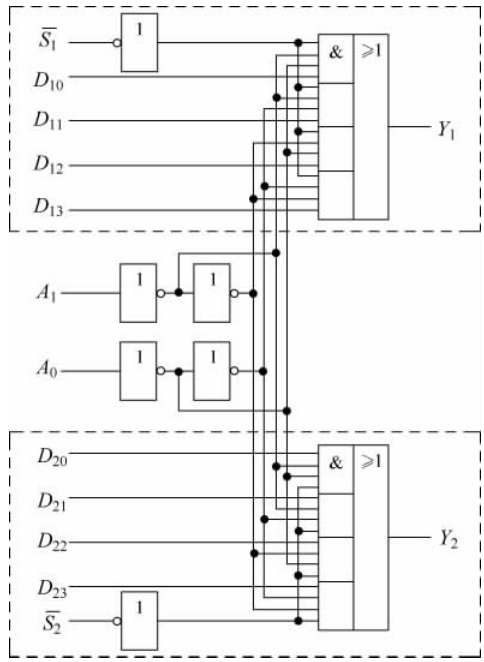


图 3.42　双 4 选 1 数据选择器 74LS153 的逻辑图

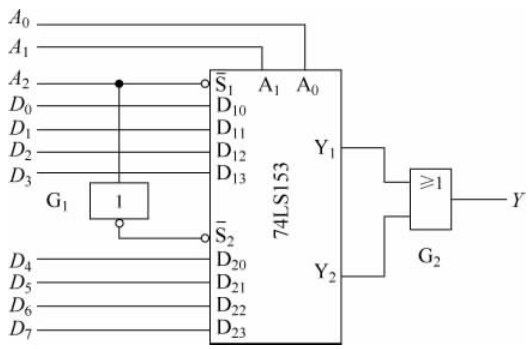


图 3.43　用双 4 选 1 数据选择器构成的 8 选 1 数据选择器

表 3.19　8 选 1 数据选择器 74LS151 的真值表

输　入					输出	
D	A_2	A_1	A_0	$\overline{S}$	Y	$\overline{Y}$
×	×	×	×	1	0	1
D_0	0	0	0	0	D_0	$\overline{D}_0$
D_1	0	0	1	0	D_1	$\overline{D}_1$
D_2	0	1	0	0	D_2	$\overline{D}_2$
D_3	0	1	1	0	D_3	$\overline{D}_3$
D_4	1	0	0	0	D_4	$\overline{D}_4$
D_5	1	0	1	0	D_5	$\overline{D}_5$
D_6	1	1	0	0	D_6	$\overline{D}_6$
D_7	1	1	1	0	D_7	$\overline{D}_7$

表 3.19 中 $D_7 \sim D_0$ 为 8 路数据输入端；$A_2 \sim A_0$ 为地址信号输入端；Y、$\overline{Y}$ 为互补输出端；$\overline{S}$ 为使能输入端。显然，当 $\overline{S}=1$ 时，芯片被禁止，无论地址码 $A_2 \sim A_0$ 取何值，输出总是 $Y=0$；只有当 $\overline{S}=0$ 时，芯片才被选中，由地址码 $A_2 \sim A_0$ 的不同取值选择某一路数据送到输出 Y 端。所以输出端的表达式为

$$\overline{S}=0,\quad \begin{cases} Y = D_0\overline{A}_2\overline{A}_1\overline{A}_0 + D_1\overline{A}_2\overline{A}_1A_0 + \cdots + D_7A_2A_1A_0 = \sum_{i=0}^{7} D_i m_i \\ \overline{Y} = \overline{D}_0\overline{A}_2\overline{A}_1\overline{A}_0 + \overline{D}_1\overline{A}_2\overline{A}_1A_0 + \cdots + \overline{D}_7A_2A_1A_0 = \sum_{i=0}^{7} \overline{D}_i m_i \end{cases} \tag{3.3}$$

图 3.44(a)所示是 74LS151 的内部逻辑图，图 3.44(b)为引脚图。

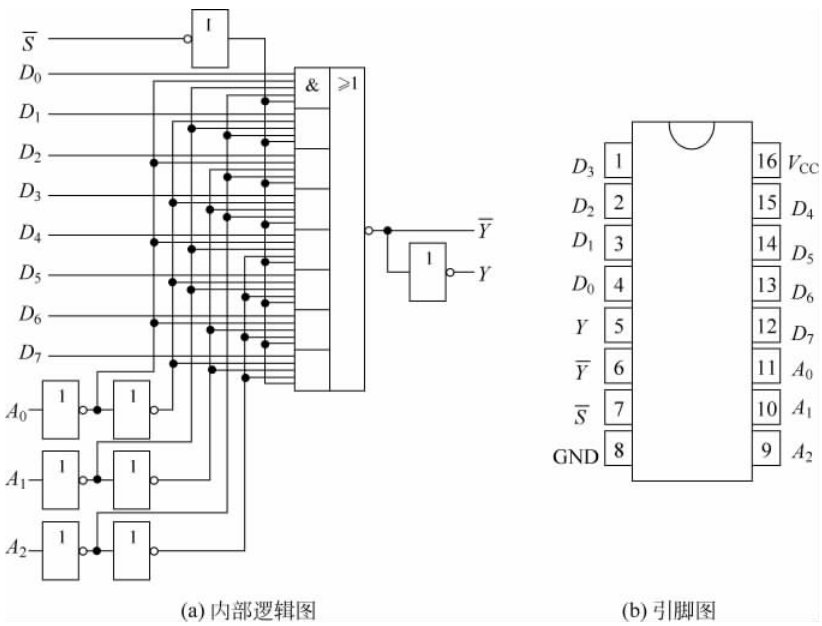


图 3.44　74LS151 的内部逻辑图及引脚图

数据选择器的另外一个重要应用，就是可以实现组合逻辑函数。从数据选择器的输出函数表达式可以看出，它是关于地址码的最小项和对应的各路输入端数据的“与-或”形式，而任何的组合逻辑函数都可以写成最小项之和的形式。因此，用数据选择器可以实现组合逻辑函数。其基本思想是：利用地址变量产生的最小项，通过数据输入信号 D_i 的不同取值，来选取组成逻辑函数所需的最小项。

例 3.12　用 8 选 1 数据选择器实现组合逻辑函数 $L=\overline{A}BC+A\overline{B}C+AB$。

解：首先写出函数的标准与或式

$$\begin{aligned} L &= \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC \\ &= m_3 + m_5 + m_6 + m_7 \\ &= m_3D_3 + m_5D_5 + m_6D_6 + m_7D_7 \end{aligned}$$

将上式与式(3.3)相比较，若令 $A_2=A$，$A_1=B$，$A_0=C$，则 $D_3=D_5=D_6=D_7=1$，而式中没有出现的最小项 m_0、m_1、m_2、m_4 对应的数据输入端 $D_0=D_1=D_2=D_4=0$，并将使能端接低电平。其逻辑图如图 3.45 所示。

例 3.13　用 8 选 1 数据选择器实现函数 $Z=\sum m(3,4,5,6,7,8,9,10,12,14)$。

解：8 选 1 数据选择器有 3 个地址控制端，与数据输入信号 D_i 配合使用可以实现任何形式的四变量以下函数。首先写出函数的标准与或式

$$Z=\overline{A}\,\overline{B}CD+\overline{A}B\overline{C}\,\overline{D}+\overline{A}B\overline{C}D+\overline{A}BC\overline{D}+\overline{A}BCD$$
$$+A\overline{B}\,\overline{C}\,\overline{D}+A\overline{B}\overline{C}D+A\overline{B}C\overline{D}+AB\overline{C}\,\overline{D}+ABC\overline{D}$$

将此式与数据选择器的输出表达式(3.3)，即 $Y=D_0\overline{A}_2\overline{A}_1\overline{A}_0+D_1\overline{A}_2\overline{A}_1A_0+\cdots+D_7A_2A_1A_0$ 相比较，确定输入变量和地址码的对应关系。若令 $A_2=A$，$A_1=B$，$A_0=C$，则有

$$Z=m_0\cdot 0+m_1\cdot D+m_2\cdot 1+m_3\cdot 1+m_4\cdot 1+m_5\cdot\overline{D}+m_6\cdot\overline{D}+m_7\cdot\overline{D}$$

只要令 $D_0=0$，$D_1=D$，$D_2=D_3=D_4=1$，$D_5=D_6=D_7=\overline{D}$，就可得到 3.46 所示为其逻辑图。

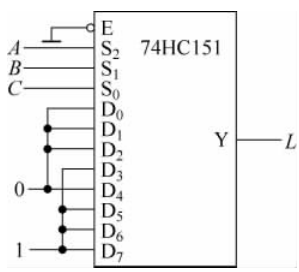


图 3.45　例 3.12 逻辑图

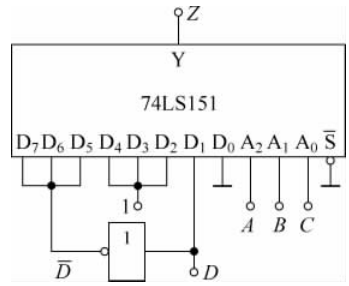


图 3.46　例 3.13 的逻辑图

思考题

1. 数据选择器输入数据的位数和地址信号的位数之间应满足怎样的关系？
2. 如果用一个 4 选 1 数据选择器产生一个三变量逻辑函数，电路的接法是否唯一？
3. 如何用 8 选 1 数据选择器实现 16 选 1 数据选择器？

3.5　数值比较器

在数字系统中经常需要比较两个数字的大小，数值比较器就是能对两个二进制数进行比较的逻辑电路。

3.5.1　数值比较器的结构

1. 1 位数值比较器

现在以比较 A、B 两个二进制数的第 i 位为例来分析 1 位数值比较器的工作原理，其框图如图 3.47 所示。因为两个 1 位数比较只能有三种结果：大于、等于或小于，分别用三个输出变量 L_i、G_i、M_i 来表示，可得表 3.20 所示的真值表。

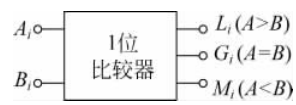


图 3.47　1 位数值比较器框图

表 3.20 1 位数值比较器真值表

A_i	B_i	L_i	G_i	M_i
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

由真值表写出输出函数表达式

$$\begin{cases} L_i = A_i\overline{B}_i \\ G_i = \overline{A}_i\overline{B}_i + A_iB_i \\ M_i = \overline{A}_iB_i \end{cases} \tag{3.4}$$

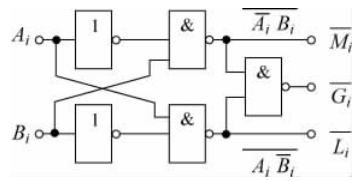


图 3.48 1 位数值比较器的逻辑图

由表达式画出 1 位数值比较器的逻辑图,如图 3.48 所示。

2. 多位数值比较器

当比较两个多位数时,应从高位开始比较,只有高位相等时才需要比较低位,这样逐次向低位进行比较。现以两个 4 位二进制数 $A=A_3A_2A_1A_0$、$B=B_3B_2B_1B_0$ 为例,分析一下多位数比较的原理。

如果 $A_3>B_3$(或 $A_3<B_3$),则无论两个数的低位大小如何,都有 $A>B$(或 $A<B$);如果 $A_3=B_3$,则需要比较次高位,依此类推。很明显,只有 4 位都对应相等时,才有 $A=B$。因此可得到 4 位数值比较器的真值表,如表 3.21 所示。

表 3.21 4 位数值比较器的真值表

比较输入				输出		
$A_3\ B_3$	$A_2\ B_2$	$A_1\ B_1$	$A_0\ B_0$	L	G	M
$>$	×	×	×	1	0	0
$=$	$>$	×	×	1	0	0
$=$	$=$	$>$	×	1	0	0
$=$	$=$	$=$	$>$	1	0	0
$=$	$=$	$=$	$=$	0	1	0
$<$	×	×	×	0	0	1
$=$	$<$	×	×	0	0	1
$=$	$=$	$<$	×	0	0	1
$=$	$=$	$=$	$<$	0	0	1

由真值表 3.21 可以得到四位数值比较器的输出表达式

$$\begin{cases} M = A_3B_3 + (A_3\odot B_3)A_2B_2 + (A_3\odot B_3)(A_2\odot B_2)A_1B_1 \\ \qquad + (A_3\odot B_3)(A_2\odot B_2)(A_1\odot B_1)A_0B_0 \\ G = (A_3\odot B_3)(A_2\odot B_2)(A_1\odot B_1)(A_0\odot B_0) \\ L = \overline{M+G} \end{cases} \tag{3.5}$$

两个 4 位数每一位进行比较,都可以由 1 位数值比较器完成,所以结合输出表达式,即

可得到 4 位数值比较器的逻辑图，如图 3.49 所示。

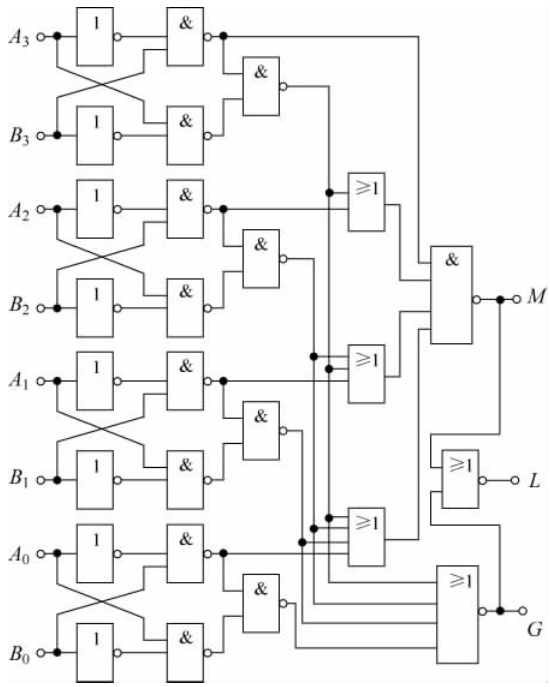


图 3.49　4 位数值比较器的逻辑图

3.5.2　集成数值比较器

74LS85 是典型的 4 位集成数值比较器，可以比较两个 4 位二进制数 $A=A_3A_2A_1A_0$ 和 $B=B_3B_2B_1B_0$ 的大小，三种输出结果由 $F_{(A>B)}$、$F_{(A<B)}$、$F_{(A=B)}$ 表示；$I_{(A>B)}$、$I_{(A<B)}$、$I_{(A=B)}$ 为级联输入端，当芯片扩展时与低位芯片的输出对应相连。其引脚图如图 3.50 所示。表 3.22 为 74LS85 的真值表。

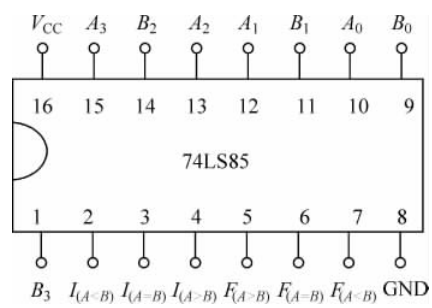


图 3.50　4 位数值比较器 74LS85 的引脚图

4 位集成数值比较器 74LS85 的逻辑图如图 3.51 所示，也是由 4 个 1 位数值比较器构成。

表 3.22　4 位数值比较器 74LS85 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
$A_3<B_3$	×	×	×	×	×	×	0	1	0
$A_3=B_3$	$A_2>B_2$	×	×	×	×	×	1	0	0
$A_3=B_3$	$A_2<B_2$	×	×	×	×	×	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	×	×	×	×	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	×	×	×	×	0	1	0

续表

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	$I_{(A>B)}$	$I_{(A<B)}$	$I_{(A=B)}$	$F_{(A>B)}$	$F_{(A<B)}$	$F_{(A=B)}$
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	×	×	×	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	×	×	×	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	1	0	0	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	0	1	0	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	0	0	1	0	0	1

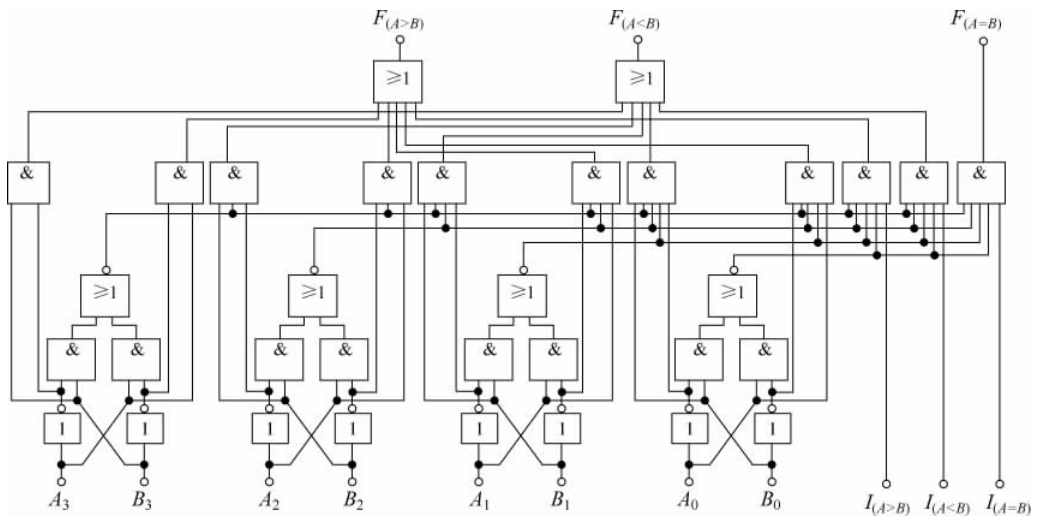


图 3.51 4 位集成数值比较器 74LS85 的逻辑图

一片 74LS85 可以比较两个 4 位二进制数的大小，此时级联输入端 $I_{(A>B)}$、$I_{(A<B)}$、$I_{(A=B)}$ 分别接 0、0、1；当要比较的两个数少于 4 位时，高位多余的输入端可同时接 0 或接 1。而当要比较的两个数大于 4 位时，一片 74LS85 不够用，就要利用级联输入端 $I_{(A>B)}$、$I_{(A<B)}$、$I_{(A=B)}$ 进行多片的级联扩展连接。

例 3.14 用两片 4 位集成数值比较器 74LS85 构成一个 8 位的数值比较器。

解：假设比较两个 8 位二进制数 $A(A_7A_6A_5A_4A_3A_2A_1A_0)$、$B(B_7B_6B_5B_4B_3B_2B_1B_0)$，需要两片 74LS85。低位芯片(1)的输入端接 A、B 的低四位($A_3A_2A_1A_0$ 和 $B_3B_2B_1B_0$)；其级联输入端 $I_{(A>B)}$、$I_{(A<B)}$、$I_{(A=B)}$ 分别接 0、0、1；其输出端 $F_{(A>B)}$、$F_{(A<B)}$、$F_{(A=B)}$ 分别对应接至高位芯片(2)的级联输入端 $I_{(A>B)}$、$I_{(A<B)}$、$I_{(A=B)}$。整个电路的比较结果由高位芯片(2)的输出端引出，如图 3.52 所示。

如果 A、B 两个数的高 4 位对应相等，即 $A_7A_6A_5A_4=B_7B_6B_5B_4$，则 A、B 两个数的大小由低位芯片的比较结果决定，此时整个电路的输出(即高位芯片的输出)与其级联输入端的状态一致；如果 A、B 两个数的高 4 位不相等 $A_7A_6A_5A_4\neq B_7B_6B_5B_4$，则无论低 4 位 $A_3A_2A_1A_0$ 和 $B_3B_2B_1B_0$ 大小关系如何，A、B 两个数的大小都由高 4 位决定，此时，高位芯片的输出与其级联输入端没有关系。

多片 74LS85 通过级联输入端进行上述串联扩展，可以对任意二进制数比较大小，但这

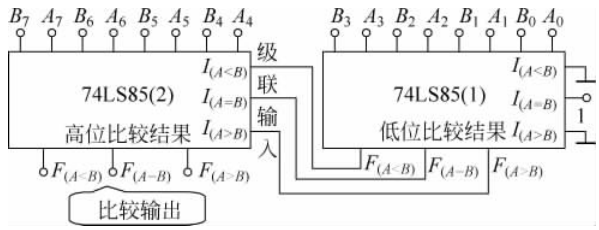


图 3.52　由两片 74LS85 构成的 8 位数值比较器

种串联比较是逐级进位的，当级联的芯片较多时，所需的进位传递时间就比较长，工作速度比较慢。因此，当扩展位数较多时，常采用并联扩展方式。

例 3.15　试用 5 片 4 位集成数值比较器 74LS85 构成一个 16 位的数值比较器。

解：将两个 16 位二进制数按高低位次序分成 4 组，每一组用一片 74LS85 进行比较，各组的比较是并行的。将每组比较的结果再经过一片 74LS85 进行比较后，最终得到两个 16 位数的比较结果，如图 3.53 所示。

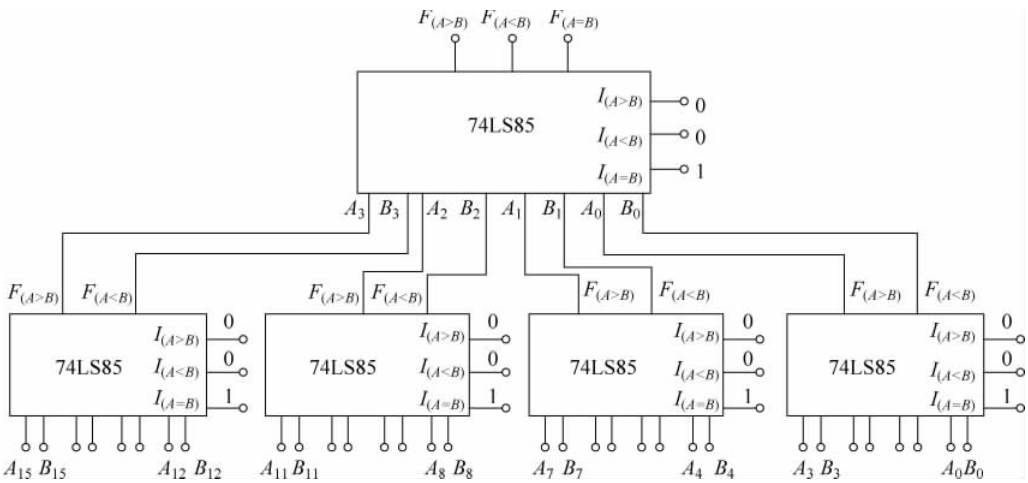


图 3.53　5 片 74LS85 构成的 16 位数值比较器

例 3.16　试用集成数值比较器 74LS85 设计电路实现真值表 3.23 的逻辑功能。

表 3.23　例 3.16 真值表

输入信号				输出信号			输入信号				输出信号		
A_3	A_2	A_1	A_0	F_2	F_1	F_0	A_3	A_2	A_1	A_0	F_2	F_1	F_0
0	0	0	0	0	1	0	1	0	0	0	1	0	0
0	0	0	1	0	1	0	1	0	0	1	1	0	0
0	0	1	0	0	1	0	1	0	1	0	1	0	0
0	0	1	1	0	1	0	1	0	1	1	1	0	0
0	1	0	0	0	1	0	1	1	0	0	1	0	0
0	1	0	1	0	1	0	1	1	0	1	1	0	0
0	1	1	0	0	0	1	1	1	1	0	1	0	0
0	1	1	1	1	0	0	1	1	1	1	1	0	0

解：从真值表可以看出，当输入信号 $A_3A_2A_1A_0>0110$ 时，输出 $F_2=1$；当输入信号 $A_3A_2A_1A_0<0110$ 时，输出 $F_1=0$。当输入信号 $A_3A_2A_1A_0=0110$ 时，输出 $F_0=1$。即完成的是输入信号与 0110 比较大小的功能，因此可以用一片 74LS85 实现上述功能。将输入信号 $A_3A_2A_1A_0$ 与对应输入端相连，将 0110 接在输入端 $B_3B_2B_1B_0$，级联输入 $I_{(A>B)}=0$，$I_{(A<B)}=0$，$I_{(A=B)}=1$，逻辑图如图 3.54 所示。

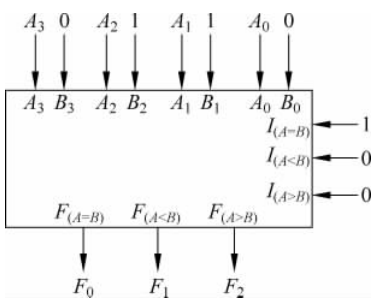


图 3.54　例 3.16 逻辑图

思考题

1. 如果用 4 位数值比较器比较两个 3 位二进制数的大小，可以有几种接法？

2. 用串联扩展方式组成的 16 位数值比较器，与用并联扩展方式组成的在进位传递时间上有什么关系？

3.6　加法器

数字电子系统对各种信息的处理，如加、减、乘、除，在计算机中都是转换为加法运算来实现的，因此，加法运算是最基本的算术逻辑运算。完成两个二进制数相加功能的电路称为加法器，显然，加法器是构成算术运算器的基本单元。

3.6.1　半加器与全加器

1. 半加器

能够完成两个 1 位二进制数相加，而不考虑低位进位的逻辑电路称为半加器。

假设两个加数为 A_i 和 B_i，本位和用 S_i 表示，本位向高位的进位用 C_i 表示，根据半加器的定义，可以得到半加器的真值表，如表 3.24 所示。

由表 3.24 可得半加器的输出表达式

$$\begin{cases} S_i=\overline{A}_iB_i+A_i\overline{B}_i=A_i\oplus B_i \\ C_i=A_iB_i \end{cases} \tag{3.6}$$

表 3.24　半加器的真值表

A_i	B_i	S_i	C_i
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

由输出表达式可得到半加器的逻辑图，如图 3.55(a)所示，其逻辑符号如图 3.55(b)所示。

2. 全加器

将两个多位二进制数相加时，除最低位外，还要考虑来自低位的进位。全加器可以将被加数、加数和低位进位信号相加，并根据求和结果，给出该位的进位信号，其框图和逻辑符号如图 3.56 所示。

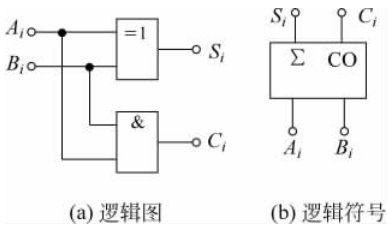


图 3.55　半加器的逻辑图及符号

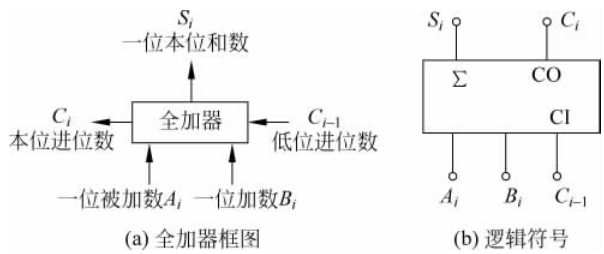


图 3.56　全加器

根据全加器的定义，可以得到其真值表，如表 3.25 所示。

表 3.25　全加器的真值表

A_i	B_i	C_{i-1}	S_i	C_i
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

由真值表 3.25 可得到 S_i 和 C_i 的卡诺图，如图 3.57 所示。

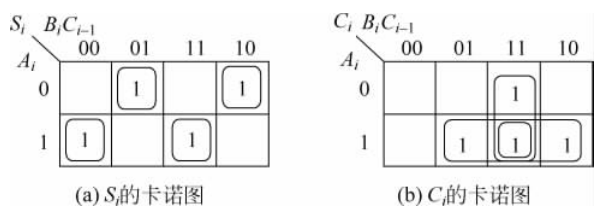


图 3.57　全加器的卡诺图

由卡诺图可以得到输出表达式

$$\begin{cases} S_i = \overline{A}_i\overline{B}_iC_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + A_i\overline{B}_i\overline{C}_{i-1} + A_iB_iC_{i-1} \\ C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1} \end{cases} \tag{3.7}$$

如图 3.58 所示是由与门、或门和非门实现的全加器电路。

例 3.17　设计一个 7 人表决电路。

解：根据要求，显然该电路有 7 个输入变量，分别用 A、B、C、D、E、F、G 表示，输入变量

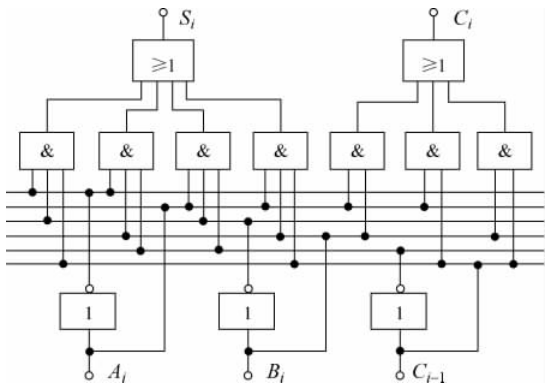


图 3.58 由与门、或门和非门实现的全加器

取值为 1 表示同意,反之表示不同意;有 1 个输出变量,用 Y 表示,Y 取值为 1 表示通过,反之表示没通过。

若按组合逻辑电路设计的一般方法,其输入变量的取值组合有 128 种,真值表有 128 行,用门电路设计将很复杂。这里考虑一下变量赋值不难发现,若使输出 $Y=1$,应使 7 个输入信号的算术和大于或等于 4,因此可以考虑用全加器实现。

每个全加器有 3 个输入端和 2 个输出端,进位输出信号 C_i 代表数字 2,输出和 S_i 代表数字 1。将输入信号 A、B、C 和 E、F、G 分别接到全加器 1 和全加器 2 的输入端,再将两个全加器的输出端 S_i 与输入信号 D 接到全加器 3 的输入端,将三个全加器的进位信号 C_i 接到全加器 4 的输入端,则全加器 4 的进位信号 $C_4=1$,就意味着 7 个输入信号之和大于或等于 4,表决通过,否则就是 7 个输入信号之和小于 4,表决没通过,逻辑图如图 3.59 所示。

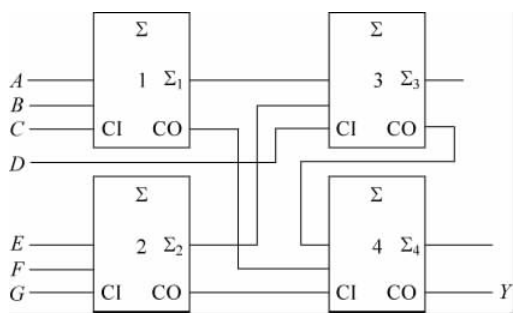


图 3.59 例 3.17 逻辑图

3.6.2 集成加法器

1. 双全加器 74LS183

双全加器 74LS183 集成了两个完全相同的全加器,其引脚图如图 3.60 所示。图 3.61 是 1/2 个 74LS183 的逻辑图。

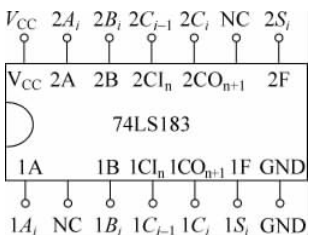


图 3.60　双全加器 74LS183 的引脚图

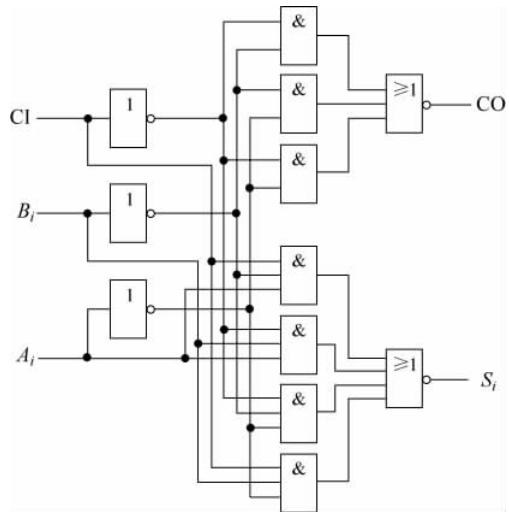


图 3.61　1/2 个 74LS183 的逻辑图

2. 4 位二进制加法器

实现多位数加法运算的电路称为加法器。加法器按照相加方式不同，分为串行加法器和并行加法器；按照参加运算的计数规则不同，分为二进制加法器和十进制加法器。

1）二进制串行进位加法器

两个多位数相加时，可采用多个 1 位全加器来实现。

如图 3.62 所示为 4 位加法器电路，由 4 个 1 位全加器构成，将每个全加器低位的进位输出 CO 接到相邻高位的进位输入端 CI 上，最低位进位输入端接地。$A(A_3A_2A_1A_0)$和 $B(B_3B_2B_1B_0)$为两个加数，S_3、S_2、S_1、S_0 表示和，C_3 是进位信号。显然，这种连接方式每一位输出的结果必须等到相邻低位产生进位信号后才能建立，因此把这种结构的电路称为串行进位加法器(或逐位进位加法器)。

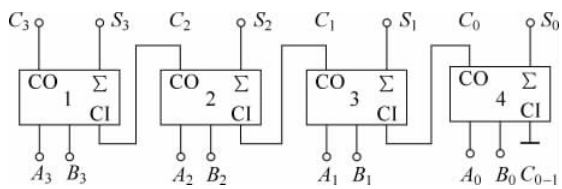


图 3.62　4 位串行加法器电路

串行进位加法器的特点是电路结构简单，连接方便，要构成 n 位串行加法器，只需将 n 个 1 位全加器串联就可以了，但这种连接方法运算速度慢。

2）二进制超前进位加法器 74LS283

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消耗的时间。因为第 i 位的进位输入信号是这两个加数第 i 位之前各位状态的函数，所以第 i 位的进位信号(C_{i-1})肯定能由 A_{i-1}，A_{i-2}，…，A_0 和 B_{i-1}，B_{i-2}，…，B_0 唯一确定。根据这个道理，就可以通过一定的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了，这样就能有效地提高运算速度。采用这种结构的加法器称为超前进

位加法器,也称为快速进位加法器。

根据全加器真值表 3.25,可得出两个多位数中第 i 位相加产生的进位输出 C_i 的表达式

$$C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1} = A_iB_i + (A_i + B_i)C_{i-1}$$

设进位生成函数 $G_i = A_iB_i$,进位传递函数 $P_i = A_i + B_i$,则

$$C_i = G_i + P_iC_{i-1}$$

将上式展开后可得到

$$\begin{aligned} C_i &= G_i + P_iC_{i-1} \\ &= G_i + P_i(G_{i-1} + P_{i-1}C_{i-2}) \\ &= G_i + P_i[G_{i-1} + P_{i-1}(G_{i-2} + P_{i-2}C_{i-3})] \\ &= \cdots \\ &= G_i + P_iG_{i-1} + P_iP_{i-1}G_{i-2} + \cdots + P_iP_{i-1}\cdots P_1G_0 + P_iP_{i-1}\cdots P_0C_0 \end{aligned}$$

仍然根据全加器真值表 3.25,可得出第 i 位相加产生的和 S_i 的表达式

$$\begin{aligned} S_i &= \overline{A}_i\overline{B}_iC_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + A_i\overline{B}_i\overline{C}_{i-1} + A_iB_iC_{i-1} \\ &= \overline{(A_i \oplus B_i)}C_{i-1} + (A_i \oplus B_i)\overline{C}_{i-1} \\ &= A_i \oplus B_i \oplus C_{i-1} \end{aligned}$$

根据 C_i 和 S_i 表达式构成的 4 位超前进位加法器 74LS283 如图 3.63 所示。

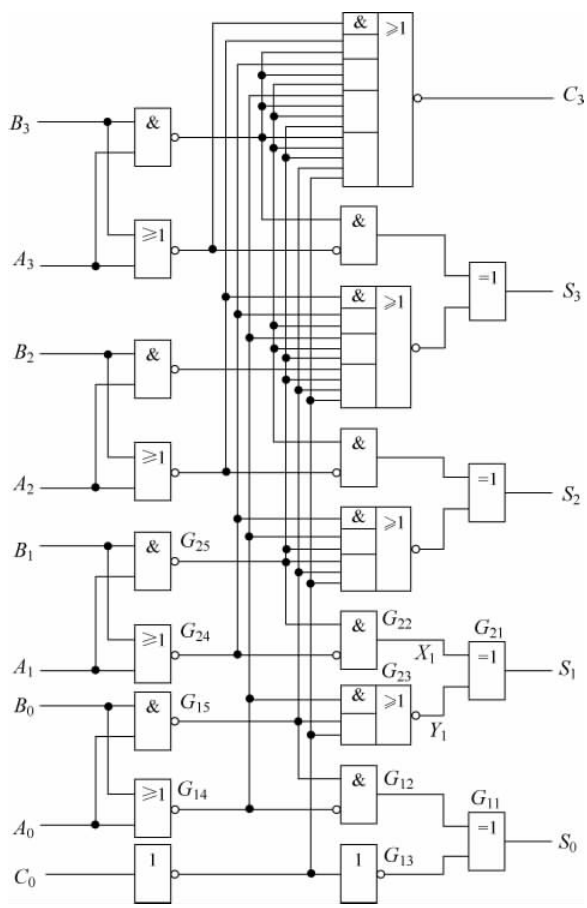


图 3.63 4 位超前进位加法器 74LS283

由图 3.63 可看出，两个 4 位数通过超前进位加法器进行运算时，完成运算只需三级门电路的传输延迟时间，而获得进位输出信号仅需一级反相器和一级与或非门的传输延迟时间，所以运算速度非常快，适合于高速数字计算机、数据处理及控制系统。但是，由图 3.63 明显可以看出电路较复杂，运算位数越多，电路越复杂。因此，运算速度的提高，是以增加电路复杂性为代价的。

思考题

1. 如表 3.25 所示的全加器真值表，若在卡诺图中圈 0，试写出其输出表达式并画出逻辑图。

2. 如何用两个半加器实现 1 位全加器？

3. 串行进位加法器和超前进位加法器有何区别？各有什么优缺点？

3.7　组合逻辑电路中的竞争-冒险现象

前面分析组合逻辑电路时，没有考虑门电路传输延迟时间的影响。实际上信号从输入到稳定输出需要一定时间。由于信号从输入到输出的过程中，不同通路上门电路的级数不同，不同门电路的传输延迟时间也不同，从而使信号由输入经不同通路传输到输出端的时间不同。这就可能使逻辑电路产生错误的输出，这种现象称为竞争-冒险。

3.7.1　竞争-冒险现象的成因

分析一下图 3.64(a)所示逻辑图的工作情况。输出端 L 的逻辑表达式为 $L=A\cdot\overline{A}$，理想情况下，输出应恒等于 0。但是由于 G_1 门的延迟，$\overline{A}$ 下降沿到达 G_2 门的时间滞后于 A 的上升沿，因此，使 G_2 输出端出现了一个正向窄脉冲，如图 3.64(b)所示。与门 G_2 的两个输入信号经不同路径在不同时刻到达的现象，称为竞争；由此而产生输出干扰脉冲的现象称为冒险。

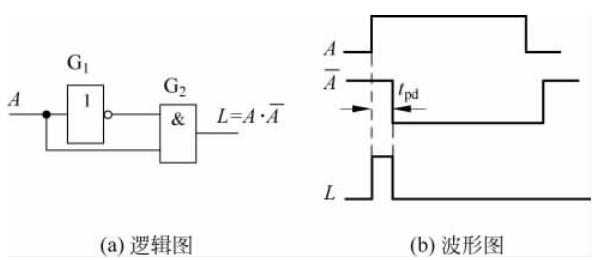


图 3.64　产生正跳变脉冲的竞争冒险现象

同理，如图 3.65(a)所示逻辑图中，理想情况下，$L=A+\overline{A}$ 应恒等于 1，但由于 G_1 门的延迟，会使 G_2 输出端产生一个负脉冲，如图 3.65(b)所示，也会产生竞争-冒险现象。

由以上分析可知，当电路中存在由反相器产生的互补信号，在互补信号的状态发生变化时，就可能出现竞争-冒险现象。

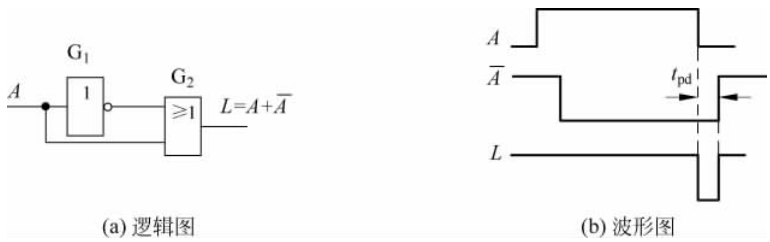


图 3.65 产生负跳变脉冲的竞争-冒险现象

3.7.2 消除竞争-冒险的措施

当电路中存在冒险现象时,通常采用以下几种方法进行消除。

1. 修改逻辑设计

在产生冒险现象的逻辑表达式上,即利用公式 $AB+\overline{A}C=AB+\overline{A}C+BC$,增加冗余项或乘上冗余因子,使之不出现 $A+\overline{A}$ 或 $A\overline{A}$ 的形式,就可以消除冒险现象。

如函数 $Y=AB+\overline{A}C$,在 $B=C=1$ 时,$Y=A+\overline{A}$ 产生冒险现象,那么如果在表达式 Y 中加入冗余项 BC,就可以消除冒险现象。又如函数 $Y=(A+C)(\overline{A}+B)$,在 $B=C=0$ 时,$Y=A\overline{A}$,产生冒险现象,那么如果在表达式 Y 中乘上冗余因子 $B+C$,也可消除冒险现象。

在卡诺图中,将"相切"的圈用一个多余的圈连接起来(相当于加入了一个冗余项或冗余因子),即可消除冒险现象。如图 3.66 所示卡诺图,增加了一个冗余项的圈(m_5 和 m_7),即可消除冒险现象。

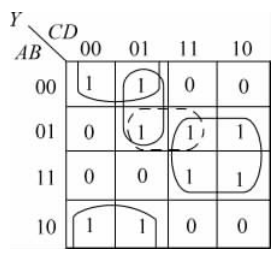


图 3.66 利用卡诺图加入冗余项

2. 加滤波电容,消除毛刺影响

由于竞争-冒险产生的干扰脉冲很窄,因此常在输出端并联滤波电容 C,如图 3.67 所示。但电容的引入会使输出波形上升沿和下降沿变斜,故电路参数选择要合适,一般电容量在 4～20pF 之间。

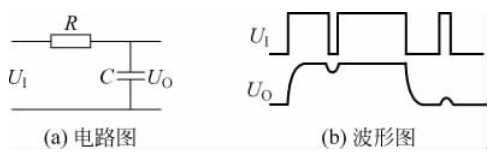


图 3.67 加入滤波电容消除冒险

3. 引入选通信号

经过上述分析可知,电路在稳定状态下没有冒险现象,毛刺的产生仅仅发生在输入信号变化的瞬间,因此可以引入选通脉冲,避开输入信号发生转换的瞬间,即可有效地避免冒险

发生。如图 3.68 所示，若在输入变量 A 变化的瞬间引入选通脉冲，则可有效地避免冒险发生。这种方法简单易行，但选通信号的作用时间和极性等一定要合适。

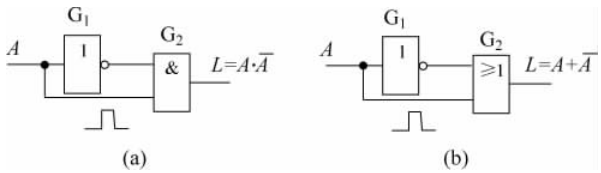


图 3.68　引入选通信号消除冒险

以上三种方法各有特点：增加冗余项的方法消除冒险的范围是有限的；加入滤波电容的方法简单易行，但增加了输出电压波形的上升时间和下降时间，使输出电压波形遭到破坏，所以只适用于对输出波形的前、后沿无严格要求的场合；引入选通信号的方法也比较简单，而且不需要增加电路元件，但这种方法对选通脉冲的宽度和作用时间有着严格的要求。因此在具体应用时，应根据具体电路结构及输出要求来选择使用哪种方法消除冒险现象。

思考题

1. 竞争-冒险现象产生的原因是什么？
2. 消除竞争-冒险的方法有哪些？这些方法各有何优缺点？

3.8　本章小结

本章讲述了组合逻辑电路的特点、组合逻辑电路的分析和设计方法，介绍了若干常用组合逻辑器件的工作原理及使用方法。最后简单介绍了组合逻辑电路中的竞争-冒险现象及其消除方法。

组合逻辑电路在逻辑功能上的特点是任意时刻的输出仅仅取决于该时刻的输入，而与电路原来的状态无关。它在电路结构上的特点是只包含门电路，而没有记忆性元件，输入/输出之间没有反馈通路。

在实际应用中，组合逻辑电路已制成了标准化的中规模集成电路，供用户直接使用。这些器件包括编码器、译码器、数据选择器、数值比较器、加法器等。为了增加使用的灵活性，便于功能扩展，在多数中规模集成电路上设置了附加的控制端（或称为使能端、片选端等）。这些控制端既可用于控制芯片的状态（工作或禁止），又可作为输出信号的选通端，还能扩展为输入端以增强电路功能。合理地运用这些控制端能最大限度地发挥电路的潜力，灵活地运用这些器件还可以设计出其他逻辑功能的组合电路。

组合电路的逻辑功能可以用逻辑图、真值表、逻辑表达式等多种方法表示，它们在本质上是相通的，可以相互转换。尽管各种组合电路在功能上千差万别，但是它们的分析方法和设计方法都是一致的。掌握了分析的一般方法，就可以识别任何一个给定电路的逻辑功能，就可以根据给定的要求设计出相应的逻辑电路。因此，学习本章内容时重在掌握分析和设计电路的方法，而不必去记忆各种具体电路。

竞争-冒险是组合逻辑电路工作状态转换过程中经常出现的一种现象。如果负载是一些对窄脉冲敏感的电路，则必须采取措施防止由于竞争而产生的干扰脉冲；如果负载电路

对窄脉冲不敏感(例如负载为光电显示器件),就不必考虑这个问题了。

习题 3

3.1 写出如图 3.69 所示电路的逻辑表达式,并列画出真值表,说明电路的逻辑功能。

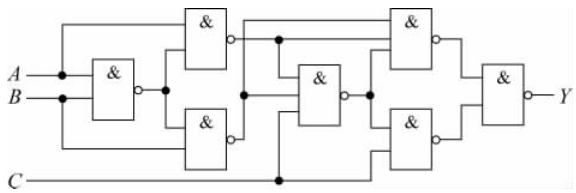


图 3.69 习题 3.1 图

3.2 写出如图 3.70 所示电路中 Y_1、Y_2 的逻辑表达式,并列出真值表,说明电路的逻辑功能。

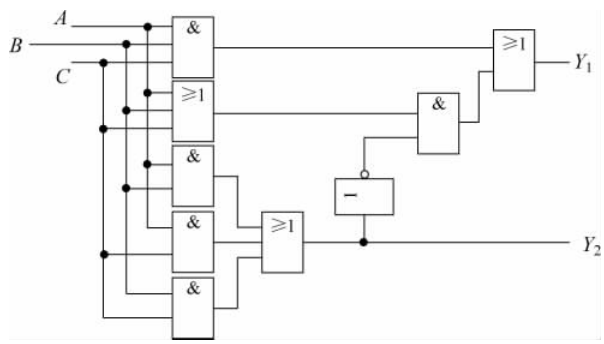


图 3.70 习题 3.2 图

3.3 用红、黄、绿三个指示灯表示 3 台设备的工作状况:绿灯亮表示 3 台设备全部正常;黄灯亮表示有 1 台设备不正常;红灯亮表示有 2 台设备不正常;红灯、黄灯都亮表示 3 台设备都不正常。试列出控制电路的真值表,并用合适的门电路实现该功能。

3.4 某雷达站有 3 部雷达 A、B、C,其中 A 和 B 的功率消耗相等,C 的功率是 A 的两倍。这些雷达由两台发电机 X、Y 供电,发电机 X 的最大输出功率等于雷达 A 的功率,发电机 Y 的最大输出功率是 X 的 3 倍。要求设计一个组合逻辑电路,能够根据各雷达启动或关闭的信号,以最节约电能的方式启、停发电机。

3.5 有一水箱由大、小两台水泵 M_L 和 M_S 供水,如图 3.71 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,检测元件输出高电平;水面高于检测元件时,检测元件输出低电平。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

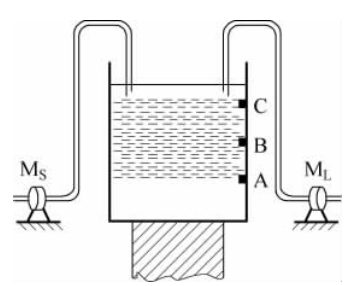


图 3.71 习题 3.5 图

泵的逻辑电路，要求电路尽量简单。

3.6　试用 4 片 8 线-3 线优先编码器 74LS148 组成 32 线-5 线优先编码器。允许附加必要的逻辑门电路。

3.7　某医院有 1、2、3、4 号病室共 4 间，每间病室设有呼叫按钮，同时在护士值班室对应装有 1、2、3、4 号 4 个指示灯。现要求当 1 号病室的按钮按下时，无论其他病室的按钮是否按下，只有 1 号灯亮。当 1 号病室的按钮没有按下而 2 号病室的按钮按下时，无论 3、4 号病室的按钮是否按下，只有 2 号灯亮。当 1、2 号病室的按钮都没有按下而 3 号病室的按钮按下时，无论 4 号病室的按钮是否按下，只有 3 号灯亮。只有在 1、2、3 号病室的按钮均未按下，而 4 号病室的按钮按下时，4 号灯才亮。试用优先编码器 74LS148 和门电路设计满足上述控制要求的逻辑电路。

3.8　写出图 3.72 中 Z_1、Z_2、Z_3 的逻辑表达式，并化为最简与-或表达式。

3.9　用 3 线-8 线译码器 74*LS*138 完成实现下列多输出函数，可附加必要的逻辑门。

(1) $Y_1=ABC+\overline{A}(B+C)$；

(2) $Y_2=A\overline{B}+\overline{A}B$；

(3) $Y_3=ABC+\overline{A}\overline{B}\overline{C}$。

3.10　用 4 线-16 线译码器 74LS154 和门电路产生如下多输出函数，并画出逻辑图。74LS154 的逻辑符号如图 3.73 所示。

(1) $Y_1=\sum m(0,2,6,8,10)$；

(2) $Y_2=\overline{A}\,\overline{B}\overline{C}D+\overline{A}B\overline{C}\overline{D}+\overline{A}B\overline{C}\overline{D}+A\overline{B}\overline{C}\overline{D}$；

(3) $Y_3=\overline{A+B}(A+B+C+D)$。

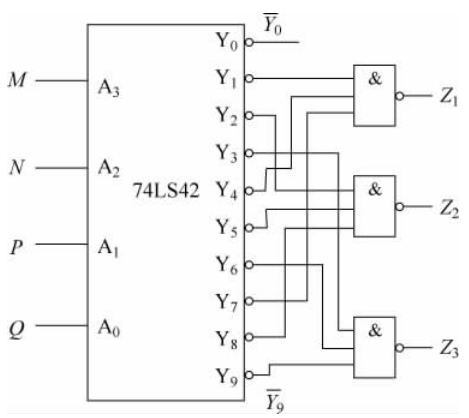


图 3.72　习题 3.8 图

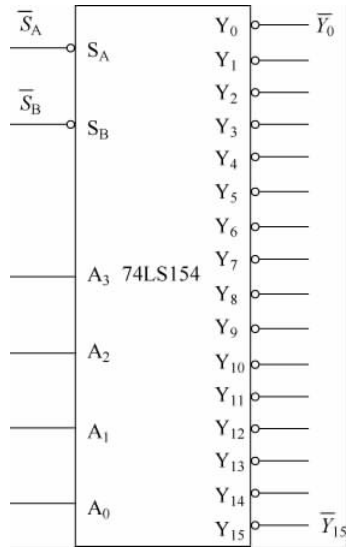


图 3.73　习题 3.10 图

3.11　一个密码锁有 3 个按键，分别为 A、B、C。当 3 个键都不按下时，锁打不开，也不报警；当只有一个键按下时，锁打不开，并发出报警信号；当有两个键同时按下时，锁打开，不报警；但当 3 个键同时按下时，锁被打开，且要报警。要求分别用①3 线-8 线译码器

74LS138 和逻辑门实现；②用双 4 选 1 数据选择器 74LS153 和逻辑门实现。

3.12 用 8 选 1 数据选择器 74LS151 实现下列多输出函数，可附加必要的逻辑门。

(1) $Y_1 = \sum m(0,2,5,7,8,10,13,15)$；

(2) $Y_2 = \sum m(2,3,4,5,8,10,11,14,15)$；

(3) $Y_3 = \sum m(0,5,10,15)$；

(4) $Y_4 = \sum m(2,3,5,7)$；

3.13 设计在一个走廊上用 3 个开关控制一盏灯的逻辑电路，当改变任何一个开关时，灯的状态都会改变。要求用数据选择器实现。

3.14 用 8 选 1 数据选择器 74LS151 设计一个函数发生器，它的功能表如表 3.26 所示。

表 3.26 习题 3.14 表

S_1	S_0	Y
0	0	$A \cdot B$
0	1	$A+B$
1	0	$A\overline{B}+\overline{A}B$
1	1	$\overline{A}$

3.15 图 3.74 是用两个 4 选 1 数据选择器组成的逻辑电路，试写出输出端 F 与输入端 A、B、C、D 之间的逻辑函数式。

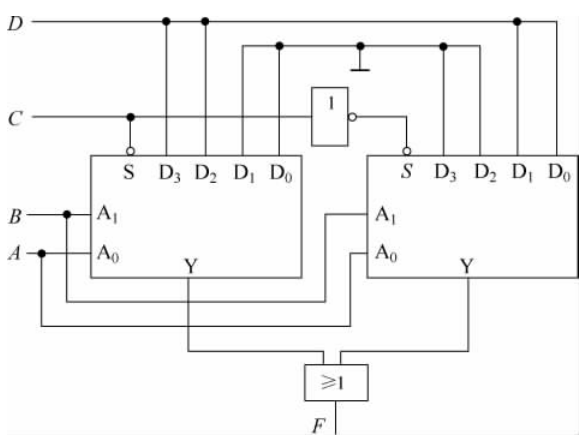


图 3.74 习题 3.15 图

3.16 试用加法器 74LS283 设计一个加/减运算电路。当控制信号 $M=0$ 时它将两个输入的两个 4 位二进制数相加；$M=1$ 时它将输入的两个 4 位二进制数相减。两数相加的绝对值不大于 15。允许附加必要的门电路。

3.17 若使用 4 位数值比较器 74LS85 组成 10 位数值比较器，需要几片芯片？芯片之间如何连接？

3.18 试用两个 4 位数值比较器组成 3 个 4 位二进制数的判断电路。要求能够判 3 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。可以附加必要的门电路。

3.19　设计一个组合逻辑电路，其输入为二进制数字0～15，当输入的数字为素数时输出为1，否则输出为0。

3.20　什么叫竞争-冒险现象？当门电路的两个输入端同时向相反的逻辑状态转换(即一个从0变成1，另一个从1变成0)时，输出是否一定有干扰脉冲产生？

3.21　判断下列表达式是否存在冒险-现象？

(1) $F(A,B,C)=A\overline{C}+BC$；

(2) $G(A,B,C)=(A+\overline{C})(B+C)$；