组合逻辑电路

第3章

CHAPTER 3

兴趣阅读——计算机的诞生及我国计算机的发展历程

1946年2月，第一台电子计算机ENIAC(Electronic Numerical Intergrator and Computer)在美国研制成功。ENIAC不是一台机器，而是一屋子机器，它有8英尺(1英尺=30.48厘米)高，3英尺宽，100英尺长，装有16种型号的18000个真空管，1500个电磁继电器，70000个电阻器，18000个电容器，总重量有30吨。

ENIAC除了军事上的弹道计算外，还涉及天气预报、原子核能、宇宙结、热能点火、风洞试验设计等诸多领域，它把圆周率π精密无误地推算到小数点后面2037位，这是人类第一次用自己的创造物计算出的最周密的值。

1956年，周恩来总理主持制定的《十二年科学技术发展规划》中，把计算机列为发展科学技术的重点之一，并在1957年筹建中国第一个计算技术研究所，开始研制通用数字电子计算机。1958年8月1日，该机可以表演短程序运行，这标志着我国第一台电子数字计算机诞生。后来经过了晶体管第二代计算机、中小规模集成电路第三代计算机，到基于微处理器的第四代计算机发展过程，中国自主研发的计算机为国防和科研事业做出了重要贡献，并且推动了计算机产业的发展。

目前中国计算机在很多方向的研究已达到了世界前沿水平，部分方向已达到国际领先水平。与此同时，中国计算机事业的发展呈现出多元化的趋势，与国外发达国家同步形成了一系列新的学科，并获得了快速发展，很多领域在技术研发或产业化上达到甚至超越了同期国外水平。2019年11月，TOP500组织发布的最新一期世界超级计算机500强榜单中，中国占据了227个，“神威·太湖之光”超级计算机位居榜单第三位，“天河二号”超级计算机位居第四位。图3-1是ENIAC和我国的“神威·太湖之光”超级计算机。

(a) ENIAC　　(b) 神威·太湖之光

图3-1　世界上第一台电子计算机和我国研制的超级计算机

本章介绍数字系统中的组合逻辑电路。首先介绍组合逻辑电路的特点和功能描述方法，重点介绍组合逻辑电路的分析方法、组合逻辑电路的设计方法以及利用无关项的组合逻辑电路的设计方法，然后介绍组合逻辑电路中的竞争和冒险等内容，最后着重介绍数字系统中常用的组合逻辑电路。此外，还给出了用 Multisim 和 VHDL 分析设计组合逻辑电路的实例。本章学习要求如下：

(1) 了解组合逻辑电路的特点和功能描述方法；

(2) 掌握组合逻辑电路的分析方法；

(3) 掌握组合逻辑电路的设计方法；

(4) 掌握利用无关项的组合逻辑电路的设计方法；

(5) 熟悉编码器等常用组合逻辑电路的功能与应用；

(6) 了解组合逻辑电路中的竞争和冒险现象及判断方法；

(7) 了解用 Multisim 和 VHDL 分析设计组合逻辑电路的方法。

如何由给定组合电路找出其实现的逻辑功能，如何根据逻辑命题来设计组合电路，以及数字系统中经常用到的组合电路的原理及应用是本章学习的重点。

3.1 概述

3.1.1 组合逻辑电路的特点

1. 功能特点

由门电路构成的逻辑部件称为组合逻辑电路(简称组合电路)。组合电路是计算机等数字系统中的逻辑部件之一。数字系统中的另一类逻辑部件叫作时序逻辑电路(简称时序电路)，这部分内容将在后续章节中介绍。

组合电路任一时刻的输出仅仅取决于该时刻输入信号的状态，而与该时刻之前电路的状态无关，即组合电路无“记忆性”功能。

2. 结构特点

组合电路之所以具有“无记忆”功能特点，归根结底是由于结构上不含记忆(存储)元件，不存在输出到输入的反馈回路。

3.1.2 组合逻辑电路的功能描述方法

组合电路的功能描述方法主要有逻辑表达式、真值表、卡诺图和逻辑图等。

逻辑图本身是逻辑功能的一种表达方式，然而逻辑图所表示的逻辑功能不够直观，通常情况下还要把逻辑图转化为逻辑表达式或真值表的形式，以使电路的逻辑功能更加直观、明显。

图 3-2 组合逻辑电路的框图

对于任何一个多输入、多输出的组合电路，都可以用图 3-2 所示的框图表示。图中 $a_1, a_2, \cdots, a_n$ 表示输入变量，$y_1, y_2, \cdots, y_m$ 表示输出变量。输出与输入间的逻辑关系可以用一组逻辑函数表示为

$$\left.\begin{aligned} y_1 &= f_1(a_1, a_2, \cdots, a_n) \\ y_2 &= f_2(a_1, a_2, \cdots, a_n) \\ &\vdots \\ y_m &= f_m(a_1, a_2, \cdots, a_n) \end{aligned}\right\} \tag{3-1}$$

3.2　组合逻辑电路的分析

组合电路的分析主要是根据给定的逻辑图找出输出与输入的逻辑关系，从而确定其逻辑功能。

3.2.1　组合逻辑电路的分析方法

组合电路的分析过程如图 3-3 所示。

图 3-3　组合电路的分析过程

1. 由逻辑电路写出逻辑表达式

一般是从输入到输出逐级写出各个门电路的输出逻辑表达式，从而写出整个逻辑电路的输出对输入变量的逻辑表达式。必要时可简化，以求出最简逻辑表达式。较简单的逻辑功能从逻辑表达式上即可分析出来。

2. 列出逻辑函数的真值表

将输入变量的状态以自然二进制数顺序的各种取值组合代入输出逻辑表达式，求出相应的输出状态，并填入表中得到真值表。

3. 分析逻辑功能

通常是通过分析真值表的特点，归纳出电路所能实现的逻辑功能。

3.2.2　组合逻辑电路分析举例

【例 3-1】　分析图 3-4 所示电路的逻辑功能。

解：根据逻辑图，逐级写出输出逻辑表达式

$$Y_1 = \overline{A},\quad Y_2 = \overline{B},\quad Y_3 = \overline{AB},\quad Y_4 = \overline{Y_1 Y_2} = \overline{\overline{A}\,\overline{B}}$$

$$Y = \overline{Y_3 Y_4} = \overline{\overline{AB}\ \overline{\overline{A}\,\overline{B}}} = AB + \overline{A}\,\overline{B} = A \odot B \tag{3-2}$$

从逻辑表达式(3-2)可以看出，图 3-4 所示电路是判断 A 和 B 是否相等的电路，即 $A = B$ 时，Y 为 **1**，否则 Y 为 **0**。

【例 3-2】　分析图 3-5 所示电路的逻辑功能。

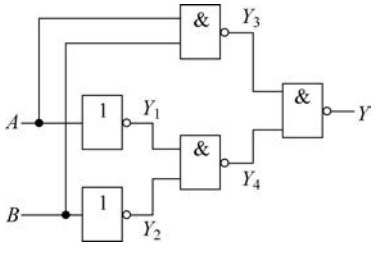


图 3-4　例 3-1 的电路图

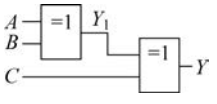


图 3-5　例 3-2 的电路图

解：根据逻辑图，逐级写出输出逻辑表达式

$$Y_1 = A \oplus B$$

$$Y = Y_1 \oplus C = A \oplus B \oplus C \tag{3-3}$$

将输入变量 A、B 和 C 的各种取值组合代入逻辑表达式(3-3)中，求出逻辑函数 Y 的值，由此得出真值表如表 3-1 所示。

表 3-1　例 3-2 的真值表

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

由真值表看出，在 3 个输入变量 A、B、C 中，有奇数个 **1** 时，输出 Y 为 **1**；否则为 **0**。由此可以判断出图 3-5 所示电路的逻辑功能为 3 位判奇电路，又称为“奇校验电路”，是判断输入变量中 **1** 的个数是否为奇数的电路。

【例 3-3】 分析图 3-6 所示电路的逻辑功能，并指出该电路设计是否合理。

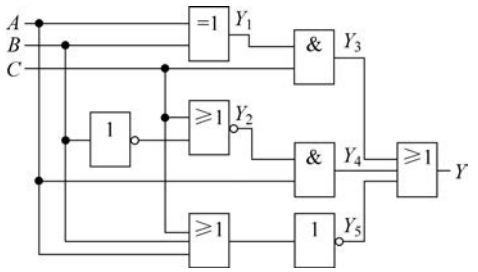


图 3-6　例 3-3 的电路图

解：逐级写出输出逻辑函数表达式

$$Y_1 = A \oplus B;\ Y_2 = \overline{\overline{B} + C};\ Y_3 = Y_1 \cdot C = (A \oplus B) \cdot C$$

$$Y_4 = Y_2 \cdot A = \overline{\overline{B} + C} \cdot A;\ Y_5 = \overline{A + B + C}$$

$$\begin{aligned} Y &= Y_3 + Y_4 + Y_5 = (A \oplus B) \cdot C + \overline{\overline{B} + C} \cdot A + \overline{A + B + C} \\ &= C(A\overline{B} + \overline{A}B) + AB\overline{C} + \overline{A}\,\overline{B}\,\overline{C} \\ &= A\overline{B}C + \overline{A}BC + AB\overline{C} + \overline{A}\,\overline{B}\,\overline{C} \\ &= \sum(0,3,5,6) \end{aligned} \tag{3-4}$$

将 A、B 和 C 取值的各种组合代入最终表达式(3-4)中(或直接依据最小项表达式)，可以得到如表 3-2 所示的真值表。

表 3-2　例 3-3 的真值表

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

续表

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

由真值表看出，当输入均为 **0** 或有偶数个 **1** 时，输出 Y 为 **1**；否则 Y 为 **0**。所以该电路为 3 位判偶电路，又称为“偶校验电路”。这个电路使用门的数量太多，设计并不合理，可用较少的门电路来实现。对表达式进行变换得

$$\begin{aligned} Y &= A\overline{B}C + \overline{A}BC + AB\overline{C} + \overline{A}\,\overline{B}\overline{C} \\ &= (A\overline{B} + \overline{A}B)C + (AB + \overline{A}\,\overline{B})\overline{C} \\ &= (A \oplus B)C + \overline{(A \oplus B)}\overline{C} \\ &= A \oplus B \odot C \end{aligned} \tag{3-5}$$

由式(3-5)可以看出，图 3-6 所示电路可以用“异或”门和“同或”门实现，其电路如图 3-7 所示。

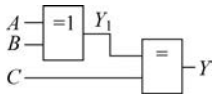


图 3-7　用“异或”门和“同或”门实现偶校验电路

组合电路的分析过程不是一成不变的，实际分析组合电路时，可以根据电路的复杂程度灵活取舍。对较简单的电路，可以从表达式中直接指出电路的逻辑功能。较复杂的电路要借助真值表，这样能较直观地分析出电路的逻辑功能。

微课视频

3.3　组合逻辑电路的设计

组合逻辑电路的设计过程与分析过程相反，是根据已知的逻辑问题，画出能实现其逻辑功能的最简逻辑电路图的过程。

3.3.1　组合逻辑电路的设计方法

1. 基本设计过程

组合逻辑电路的设计过程如图 3-8 所示。

图 3-8　组合电路的设计过程

(1) 逻辑抽象。逻辑抽象是将文字描述的逻辑命题(设计要求)转换成逻辑函数表达式的过程。

(2) 逻辑化简。逻辑化简是指采用代数法(公式法)或卡诺图法将逻辑函数化简为最简“与或”表达式，通常使用卡诺图法来完成。

(3) 逻辑变换。逻辑变换是指根据选用的逻辑器件类型，将最简“与或”表达式变换为所需形式。

(4) 画逻辑图。画逻辑图是指根据变换后的逻辑表达式绘制逻辑电路图。

上述过程中，除逻辑抽象外，其他内容均在第 1 章中做过介绍，这里不再重复。下面仅对逻辑抽象的方法做简要介绍。

2. 逻辑抽象

在设计组合电路时,要将文字描述的设计要求转化为逻辑函数的某种表达方式,这样才能设计出满足要求的逻辑电路。

由于实际逻辑问题各种各样,逻辑抽象没有规范的方法,往往要凭借设计者的经验去完成。通常的思路是:

(1) 确定输入、输出变量;

(2) 用二值逻辑的**0**、**1**两种状态分别对输入、输出变量进行逻辑赋值,即确定**0**、**1**的具体含义;

(3) 根据输入、输出之间的逻辑关系列出真值表或直接写出逻辑表达式。当变量较多时,可以建立简化的真值表。变量更多时,可根据设计要求直接列写逻辑表达式。

【例 3-4】 写出3人多数表决电路的逻辑表达式,当A、B、C三人中有多数人赞同时表决通过,且A有否决权。

解:参与表决的人A、B和C为输入变量,赞同时用**1**表示;不赞同时用**0**表示。设Y为代表表决结果的输出变量,表决通过用**1**表示;未通过用**0**表示。由此可列出如表3-3所示的真值表。

表 3-3 例 3-4 的真值表

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

由真值表可以抽象出逻辑函数Y的最小项表达式

$$Y=\sum(5,6,7) \tag{3-6}$$

【例 3-5】 已知$M=m_1m_2$和$N=n_1n_2$是两个二进制正整数,写出判断$M<N$的逻辑函数表达式。

解:分析逻辑问题可知,判断式中应该有4个输入变量,即m_1、m_2、n_1和n_2。设判断结果用Y表示,即输出变量。

在比较两个二进制正整数大小时,通常是从高位到低位逐位比较,即当高位$m_1=0$,$n_1=1$时,不论m_2和n_2为何值,都有$M<N$;而在高位相等时,比较低位即可。于是可以列出简化的真值表,如表3-4所示。表中"×"表示取**0**和**1**两种逻辑值。

表 3-4 例 3-5 的简化真值表

M		N		Y
m_1	m_2	n_1	n_2	
0	×	1	×	1
1	0	1	1	1
0	0	0	1	1

由简化的真值表可以抽象出逻辑函数表达式

$$Y=\bar{m}_1 n_1+m_1\bar{m}_2 n_1 n_2+\bar{m}_1\bar{m}_2\bar{n}_1 n_2 \tag{3-7}$$

【提示】 简化真值表得出的表达式不是最小项表达式。

3.3.2 组合逻辑电路设计举例

下面举例说明组合电路的设计过程。

【例 3-6】 用“非”门和“与或非”门完成例 3-4 中的 3 人多数表决器设计。

解：第一步，逻辑抽象。例 3-4 中已经由真值表抽象出了逻辑函数表达式(3-6)，即

$$Y=\sum(5,6,7)$$

第二步，逻辑化简。用卡诺图化简逻辑函数(由于化简过程较简单，这里略去)，可得最简“与或”表达式为

$$Y=AC+AB \tag{3-8}$$

第三步，逻辑变换。根据题意，通过两次求反，将式(3-8)变换成“与或非”形式的表达式

$$Y=\overline{\overline{AB+AC}}=\overline{\overline{A(B+C)}}=\overline{\bar{A}+\bar{B}\bar{C}} \tag{3-9}$$

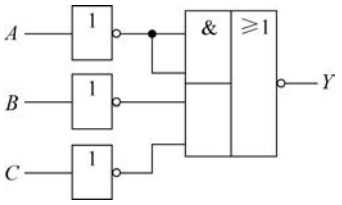


图 3-9　例 3-6 的逻辑图

第四步，画逻辑图。根据“与或非”表达式(3-9)可以画出逻辑图，如图 3-9 所示。

【例 3-7】 用“与非”门和“非”门设计一个交通信号灯工作状态监视电路，正常情况下，任何时刻有且仅有一盏灯点亮。当出现所有的灯都熄灭或有两盏及两盏以上的灯都亮的情况，则说明电路出现了故障，需发出报警信号以通知维修人员处理。

解：第一步，逻辑抽象。设变量 A、B、C 表示红、黄、绿三个信号灯，将灯的亮、灭分别用 **1** 和 **0** 表示，电路工作状态指示信号用 Y 来表示，需要报警时 Y 为 **1**，正常工作时 Y 为 **0**。真值表如表 3-5 所示。

表 3-5　例 3-7 的真值表

A	*B*	*C*	*Y*
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**

根据表 3-7，可以抽象出逻辑函数表达式

$$Y=\sum(0,3,5,6,7) \tag{3-10}$$

第二步，逻辑化简。用卡诺图对式(3-10)进行化简

$$Y=\bar{A}\bar{B}\bar{C}+AB+BC+AC \tag{3-11}$$

第三步，逻辑变换。根据题意，将式(3-11)变换成“与非-与非”表达式

$$Y=\overline{\overline{\bar{A}\bar{B}\bar{C}}\cdot\overline{AB}\cdot\overline{BC}\cdot\overline{AC}} \tag{3-12}$$

第四步，画逻辑图。依据式(3-12)，用“与非”门和“非”门绘制逻辑电路图，如图 3-10 所示。

【例 3-8】 人有 O、A、B、AB 四种基本血型。输血者与献血者的血型必须符合下述原则：O 型血是万能输血者，可以输给任意血型的人，但 O 型血的人只接受 O 型血；AB 型血是万能受血者，可以接受所有血型的血。输血者和受血者之间的血型关系如图 3-11 所示。试用“非”门和“与非”门设计一个组合电路，以判别一对输、受血者是否相容。

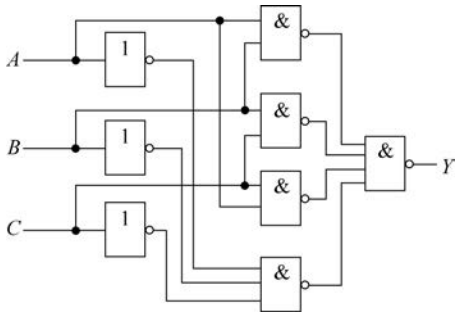


图 3-10 例 3-7 的逻辑图

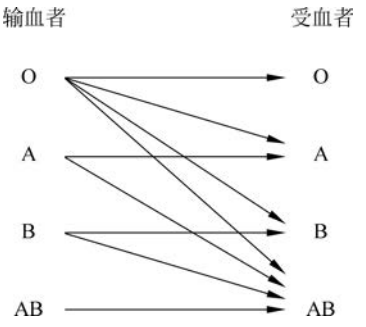


图 3-11 血型关系图

解：第一步，逻辑抽象。设用 C、D 的四种变量组合表示输血者的四种血型，用 E、F 的四种变量组合表示受血者的四种血型，如表 3-6 所示。

表 3-6 用字母表示血型关系

输血者		受血者		血型
C	D	E	F	
0	0	0	0	O
0	1	0	1	A
1	0	1	0	B
1	1	1	1	AB

根据表 3-6 可以列出输出逻辑函数 Y 与输入变量 C、D、E、F 之间关系的简化真值表，如表 3-7 所示。

表 3-7 例 3-8 的简化真值表

C	D	E	F	Y
0	0	×	×	1
0	1	0	1	1
1	0	1	0	1
×	×	1	1	1

根据表 3-7，可以抽象出逻辑函数表达式为

$$Y=\overline{C}\,\overline{D}+\overline{C}D\overline{E}F+C\overline{D}E\overline{F}+EF \tag{3-13}$$

第二步，逻辑化简。用图 3-12 所示的卡诺图化简式(3-13)，可得最简“与或”式为

$$Y=\overline{C}\,\overline{D}+EF+\overline{C}F+\overline{D}E \tag{3-14}$$

第三步，逻辑变换。对最简“与或”式(3-14)进行“与非-与非”变换得

$$\begin{aligned}Y&=\overline{C}\,\overline{D}+EF+\overline{C}F+\overline{D}E\\&=\overline{\overline{\overline{C}\,\overline{D}+EF+\overline{C}F+\overline{D}E}}\\&=\overline{\overline{\overline{C}\,\overline{D}}\cdot\overline{EF}\cdot\overline{\overline{C}F}\cdot\overline{\overline{D}E}}\end{aligned} \tag{3-15}$$

第四步，画逻辑图。根据 Y 的最简“与非-与非”表达式(3-15)，可绘制如图 3-13 所示的逻辑图。

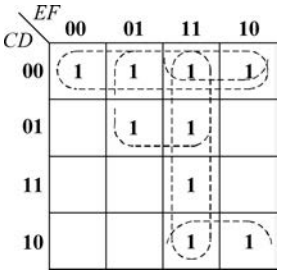


图 3-12　例 3-8 的卡诺图

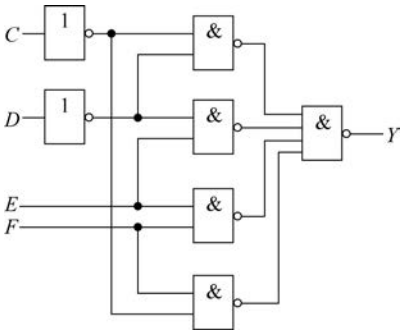


图 3-13　例 3-8 的逻辑图

3.3.3　含有无关项的组合逻辑电路设计

在第 1 章中介绍过含有无关项的逻辑函数的化简方法，利用无关项的特性可以使逻辑函数表达式化简得更简单，这意味着设计出的逻辑电路所用的门电路更少，性价比更高。下面举例说明含有无关项的组合逻辑电路设计方法。

【例 3-9】　用“与非”门、“非”门和“异或”门设计一个组合电路，以实现余三码到 8421 码的转换。

解：第一步，逻辑抽象。由题意可知，组合电路的输入为余三码，有四个输入变量，设为 A、B、C、D；输出是 8421 码，有四个输出变量，设为 Y_4、Y_3、Y_2、Y_1。由于输入变量 A、B、C、D 的取值组合不可能为 **0000～0010** 和 **1101～1111** 这 6 种组合，即有 6 个约束项，故约束方程为

$$\sum d(0,1,2,13,14,15)=\mathbf{0} \tag{3-16}$$

根据上述分析可以列出所设计电路的真值表，如表 3-8 所示。

表 3-8　例 3-9 的真值表

A	B	C	D	Y_4	Y_3	Y_2	Y_1
0	0	0	0	×	×	×	×
0	0	0	1	×	×	×	×
0	0	1	0	×	×	×	×
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1
1	1	0	1	×	×	×	×
1	1	1	0	×	×	×	×
1	1	1	1	×	×	×	×

根据表 3-8 可以抽象出逻辑函数表达式为

$$\begin{aligned} Y_4 &= \sum(11,12) + \sum d(0,1,2,13,14,15) \\ Y_3 &= \sum(7,8,9,10) + \sum d(0,1,2,13,14,15) \\ Y_2 &= \sum(5,6,9,10) + \sum d(0,1,2,13,14,15) \\ Y_1 &= \sum(4,6,8,10,12) + \sum d(0,1,2,13,14,15) \end{aligned} \tag{3-17}$$

第二步，逻辑化简。用图 3-14 所示的各卡诺图化简式(3-17)中的各逻辑函数，可得逻辑函数的最简“与或”表达式

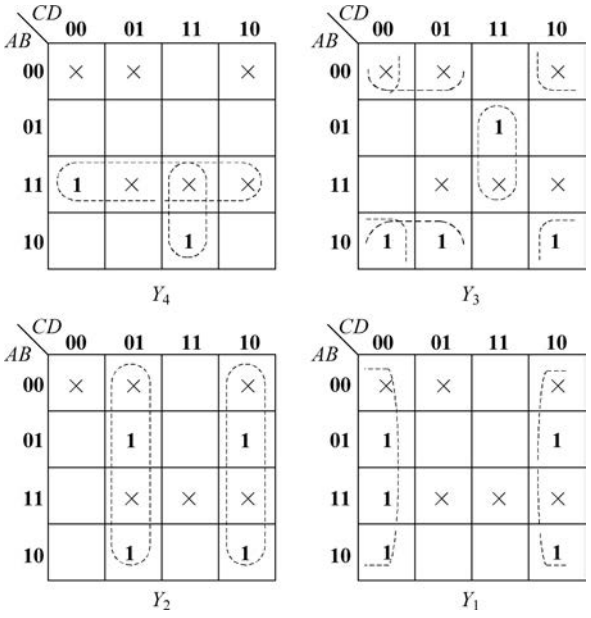


图 3-14　例 3-9 的卡诺图

$$\begin{aligned} Y_4 &= AB + ACD \\ Y_3 &= \overline{B}\overline{C} + \overline{B}\overline{D} + BCD \\ Y_2 &= C\overline{D} + \overline{C}D \\ Y_1 &= \overline{D} \end{aligned} \tag{3-18}$$

第三步，逻辑变换。对最简“与或”式(3-18)进行“与非”变换或“异或”变换得

$$\begin{aligned} Y_4 &= \overline{\overline{AB + ACD}} = \overline{\overline{AB} \cdot \overline{ACD}} \\ Y_3 &= \overline{B}(\overline{C} + \overline{D}) + BCD = \overline{B}\overline{CD} + BCD = B \oplus \overline{CD} \\ Y_2 &= C\overline{D} + \overline{C}D = C \oplus D \\ Y_1 &= \overline{D} \end{aligned} \tag{3-19}$$

第四步，画逻辑图。根据变换后的表达式(3-19)可绘制如图 3-15 所示的逻辑图。

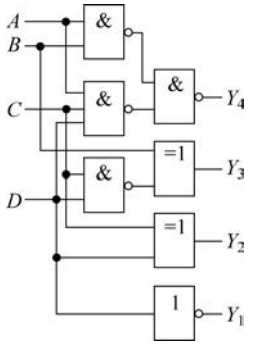


图 3-15　例 3-9 的逻辑图

【例 3-10】 试用“与或非”门设计一个操作码形成电路，当按

下“×、+、−”各操作键时，要求分别产生乘法、加法和减法的操作码 **01**、**10** 和 **11**。

解：第一步，逻辑抽象。设电路的输入变量为 A、B、C；输出变量为 Y_2、Y_1。当按下某一操作键时，相应输入变量的取值为 **1**，否则为 **0**。由于正常操作下，某一时刻只按下一个操作键，所以输入变量 A、B、C 对取值 **1** 互斥，由此可得真值表如表 3-9 所示。

表 3-9　例 3-10 的真值表

A	B	C	Y_2	Y_1
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	×	×
1	0	0	0	1
1	0	1	×	×
1	1	0	×	×
1	1	1	×	×

由表 3-9 可以写出逻辑函数表达式

$$\begin{aligned} Y_2 &= \sum(1,2) + \sum d(3,5,6,7) \\ Y_1 &= \sum(1,4) + \sum d(3,5,6,7) \end{aligned} \tag{3-20}$$

第二步，逻辑化简。用图 3-16 所示的卡诺图化简式(3-20)，可以得到逻辑函数的最简“与或”表达式

$$\begin{aligned} Y_2 &= B + C \\ Y_1 &= A + C \end{aligned} \tag{3-21}$$

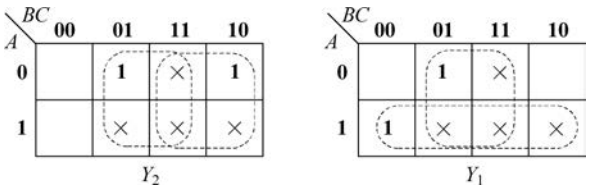


图 3-16　例 3-10 的卡诺图

比较化简结果式(3-21)和原始函数式(3-20)，可以发现，若逻辑函数的输入变量对取值 **1** 互斥，则仅包含有一个互斥变量的最小项可以化简为该互斥变量。例如 m_2，即 $\overline{A}B\overline{C}$ 可化简为 B。

第三步，逻辑变换。对最简“与或”式(3-21)进行“与或非”变换得

$$\begin{aligned} Y_2 &= \overline{\overline{B + C}} \\ Y_1 &= \overline{\overline{A + C}} \end{aligned} \tag{3-22}$$

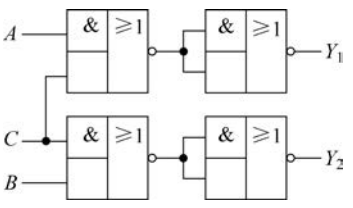


图 3-17　例 3-10 的逻辑图

第四步，画逻辑图。根据变换后的表达式(3-22)可绘制如图 3-17 所示的逻辑图。

由以上实例可以看出，在设计组合逻辑电路时，若有无关项可以利用，则设计的电路会更简单。

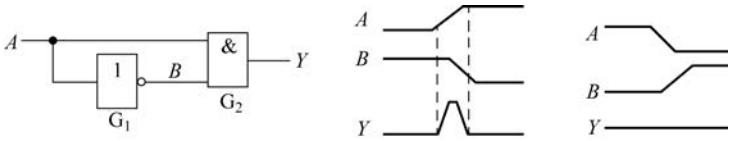
*3.4 组合逻辑电路的竞争冒险

前面在分析和设计组合逻辑电路时,讨论的都是电路的逻辑输出和输入处于稳定的状态下。而组合电路实际应用时,由于门电路传输延迟的影响,会导致电路在某些情况下,在输出端产生错误信号,从而造成逻辑关系的混乱,出现竞争冒险现象,使电路无法正常工作。

3.4.1 竞争冒险现象

在组合电路中,当电路从一种稳定状态转换到另一种稳定状态的瞬间,某个门电路的两个输入信号同时向相反方向变化,由于传输延迟时间不同,所以到达输出门的时间有先有后,这种现象称为竞争。

如图 3-18(a)所示的组合电路,当输入变量 A 由 **0** 变为 **1** 时,由于经过 G_1 门的传输延迟,G_2 门的两个输入信号 A、B 会向相反方向变化,因此 A 和 B 存在竞争。由于竞争,使电路的逻辑关系受到短暂的破坏,并在输出端产生极窄的尖峰脉冲。



(a) 逻辑图　(b) 出现冒险现象　(c) 未出现冒险

图 3-18　组合电路中的竞争冒险现象

由于输出 $Y=A\cdot B=A\cdot\overline{A}=\mathbf{0}$,即输出应恒为 **0**。但由于存在门电路的传输延迟时间,B 的变化落后于 A 的变化。当 A 已由 **0** 变为 **1**,而 B 尚未由 **1** 变为 **0** 时,在输出端 Y 就产生一个瞬间的正尖峰脉冲,如图 3-18(b)所示。这个尖峰脉冲会对后面电路产生干扰。

在图 3-18(c)中,当 A 已由 **1** 变为 **0**,而 B 尚未由 **0** 变为 **1** 时,这样在输出端 Y 仍为 **0**,符合电路逻辑关系,不会产生尖峰脉冲。

【提示】 有竞争现象时不一定都会产生尖峰脉冲。

在“与”门和“或”门组成的复杂数字系统中,由于输入信号经过不同途径到达输出门,在设计时往往难以准确知道到达的先后次序,以及门电路两个输入端在上升时间和下降时间产生的细微差别,因此都会存在竞争现象。这种由于竞争而在输出端可能出现违背稳态下逻辑关系的尖峰脉冲现象叫作冒险。

3.4.2 竞争冒险的判断

判断组合电路是否存在竞争冒险有以下几种方法。

1. 代数法

逻辑电路中存在竞争就可能产生冒险,这可以从逻辑函数表达式的结构出发来判断。经分析得知,若输出逻辑函数表达式在一定条件下最终能化简为 $Y=A+\overline{A}$ 或 $Y=A\cdot\overline{A}$ 的形式时,则可能有竞争冒险出现。例如,有两个逻辑函数 $Y_1=AB+\overline{A}C$,$Y_2=(A+B)(\overline{B}+C)$,显然,函数 Y_1 在 $B=C=\mathbf{1}$ 时,$Y_1=A+\overline{A}$。因此,按此逻辑函数实现的组合电路会出现竞争冒险现象。同理,当 $A=C=\mathbf{0}$ 时,$Y_2=B\cdot\overline{B}$,所以此函数也存在竞争冒险。

【例 3-11】 用代数法判断逻辑函数 $Y=(\overline{A}+B)(A+C)(B+\overline{C})$ 的竞争冒险情况。

解：变量 A 和 C 存在原变量和反变量，具有竞争能力，冒险判断如表 3-10 所示。

表 3-10　例 3-11 的冒险判断

A 变量			*C* 变量		
B	***C***	***Y***	***A***	***B***	***Y***
0	**0**	$A\overline{A}$	**0**	**0**	$C\overline{C}$
0	**1**	**0**	**0**	**1**	C
1	**0**	A	**1**	**0**	**0**
1	**1**	**1**	**1**	**1**	**1**

由表 3-10 可以看出，$B=C=\mathbf{0}$ 时，$Y=A\overline{A}$；$A=B=\mathbf{0}$ 时，$Y=C\overline{C}$，所以 A、C 变量分别可能产生冒险。

2. 卡诺图法

在用卡诺图法化简逻辑函数时，为了使逻辑函数最简而画的包围圈中，若有两个包围圈之间相切而不交，则在相邻处也可能存在竞争冒险。

将上述逻辑函数 Y_1 和 Y_2 用卡诺图表示，如图 3-19 所示。Y_1 是最简“与或”式，两个包围圈在 A 和 $\overline{A}$ 处相切，Y_2 是“或与”式（画 **0** 的包围圈再取反），两包围圈在 B 和 $\overline{B}$ 处相切。所以 Y_1 和 Y_2 都存在竞争冒险。

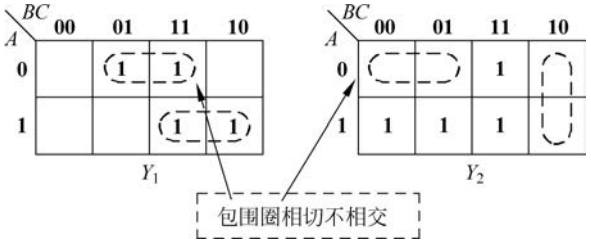


图 3-19　卡诺图包围圈相切不相交的情况

【例 3-12】 用卡诺图法判断函数 $Y_1=A\,B\,\overline{C}+A\,\overline{B}\overline{C}+\overline{A}\,BCD+\overline{A}\,BC\overline{D}$、$Y_2=B\,\overline{C}+\overline{A}CD+A\,\overline{B}\overline{C}$ 的冒险情况。

解：绘制函数的卡诺图如图 3-20 所示。

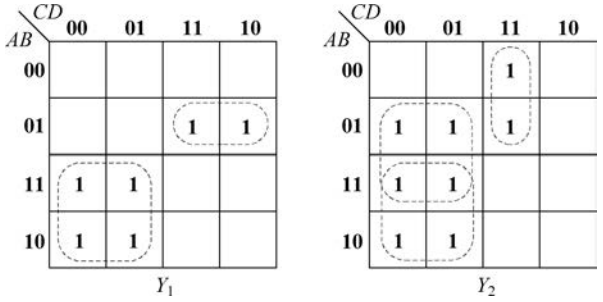


图 3-20　例 3-12 的卡诺图

在图 3-20 所示的卡诺图上，按卡诺图化简法绘制包围圈。可以判断，Y_1 的包围圈不相切，无冒险。Y_2 的包围圈 $\overline{A}CD$ 与 $B\overline{C}$ 相切，相切处 $B=D=\mathbf{1}$，$A=\mathbf{0}$，此时变量 C 变化时可能

产生冒险。

3. 计算机仿真方法

用计算机仿真判断组合逻辑电路的竞争冒险也是一种可行的方法。目前有多种计算机电路仿真软件,将设计好的逻辑电路通过仿真软件,可以观察到输出有无竞争冒险。

4. 实验法

利用实验手段检查冒险,即在逻辑电路中的输入端,加入信号所有可能的组合状态,用逻辑分析仪或示波器,捕捉输出端可能产生的冒险现象。实验法检查的结果是最终的结果,这种方法是检验电路是否存在冒险现象的最有效、最可靠的方法。

3.4.3 竞争冒险的消除

当组合逻辑电路存在着竞争冒险时,会对电路的正常工作造成威胁。因此,必须设法予以消除。常采用以下几种方法消除竞争冒险。

1. 修改逻辑设计

(1) 在逻辑表达式中添加多余项来消除竞争冒险。

【例 3-13】 判断逻辑函数 $Y=AC+\overline{A}B+\overline{A}\overline{C}$ 是否存在竞争冒险?如何消除?

解:分析 Y 的表达式可知,当 $B=C=1$ 时,$Y=A+\overline{A}$,A 可能产生竞争冒险。而 C 虽然具有竞争能力,但始终不会产生冒险。

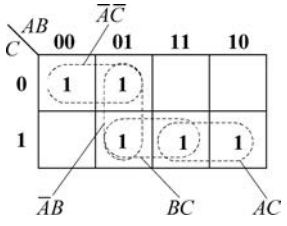


图 3-21 添加多余项消除竞争冒险

若在逻辑表达式中增加多余项 BC,则当 $B=C=1$ 时,Y 恒为 1,即消除了竞争冒险。

Y 的卡诺图如图 3-21 所示。添加多余项意味着在相切处多画一个包围圈 BC,使相切变为相交,从而消除了竞争冒险。为了简化电路,多余项通常会被舍去。但在图 3-21 中,为了保证逻辑电路能够可靠工作,又需要添加多余项消除竞争冒险。这说明最简的设计并不一定是最可靠的设计。

(2) 对逻辑表达式进行逻辑变换,以消掉互补变量。

【例 3-14】 试消除逻辑函数 $Y=(\overline{A}+\overline{C})(A+B)(B+C)$ 中的竞争冒险。

解:对逻辑表达式进行变换得

$$\begin{aligned} Y&=(\overline{A}+\overline{C})(A+B)(B+C)\\ &=\overline{A}B+AB\overline{C}+B\overline{C}+\overline{A}BC\\ &=\overline{A}B+B\overline{C} \end{aligned} \tag{3-23}$$

在上述逻辑变换过程中,消去了表达式中隐含的 $A\cdot\overline{A}$ 和 $C\cdot\overline{C}$ 项,所以由表达式 $\overline{A}B+B\overline{C}$ 确定的逻辑电路,就不会出现竞争冒险了。

修改逻辑设计的方法简便,但局限性大,不适合于输入变量较多及较复杂的电路。

2. 加滤波电容

由于冒险现象产生的尖峰脉冲一般都很窄,所以如果组合逻辑电路在较慢的速度下工作,只要在逻辑电路的输出端并联一个很小的滤波电容,其容量为 4～20pF,就可以把尖峰脉冲的幅度削弱至门电路的阈值以下,使输出端不会出现逻辑错误。

加入小电容滤波的方法简单易行,但输出电压波形边沿会随之变形,仅适合于对输出波形前、后沿要求不高的电路。

3. 引入选通脉冲

在组合逻辑电路中引入选通脉冲信号，使电路在输入信号变化时处于禁止状态，待输入信号稳定后，令选通脉冲信号有效，使电路输出正常结果，这样可以有效地消除任何竞争冒险。图 3-22 所示电路就是利用选通脉冲信号消除竞争冒险的一个例子，此电路输出信号的有效时间与选通脉冲信号的宽度相同。

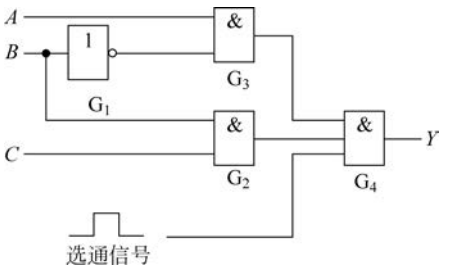


图 3-22　利用选通脉冲信号消除竞争冒险的电路

引入选通脉冲的方法简单且不需要增加电路元件，但要求选通脉冲与输入信号同步，而且对选通脉冲的宽度、极性、作用时间均有严格要求。

3.5　常用组合逻辑电路及其应用

在计算机等数字系统中，有些组合电路经常大量出现。为了使用方便，通常将这些电路制作成中、小规模的集成电路产品。编码器、译码器、加法器、数值比较器、数据选择器和数据分配器等都是这样的组合电路。本节介绍这些组合逻辑电路的原理、功能和简单应用。

3.5.1　编码器


微课视频

为区分不同事物，常常需要将某一信息（输入）变换为某一特定代码（输出）。通常将用数字或某种文字、符号来表示某一对象或信号的过程称为编码，具有编码功能的逻辑电路称为编码器。

在数字系统中，通常采用若干位二进制代码对编码对象进行编码。要表示的信息越多，二进制代码的位数就越多。n 位二进制代码有 2^n 个信息，对 N 个信号进行编码时，应按公式 $2^n \geqslant N$ 来确定需要使用的二进制代码的位数 n。

常用的编码器有二进制编码器和二-十进制编码器，每种编码器又有普通编码器和优先编码器之分。

1. 二进制编码器

二进制编码器是用 n 位二进制数把某种信号变成 2^n 个二进制代码的逻辑电路，通常命名为“2^n 线-n 线”编码器。二进制编码器的框图如图 3-23 所示，其中 $I_0 \sim I_{2^n-1}$ 为输入端，$Y_0 \sim Y_{n-1}$ 为 n 位二进制码输出端。

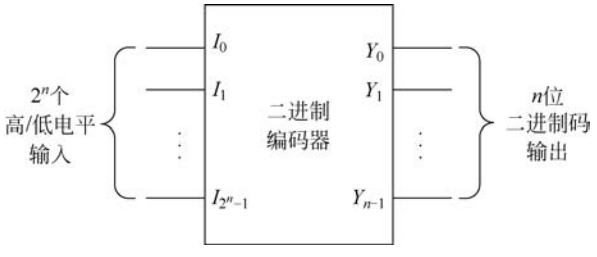


图 3-23　二进制编码器的框图

1) 普通二进制编码器

普通二进制编码器是用 n 位二进制数把某种信号变成 2^n 个二进制代码的逻辑电路。

这里以 3 位二进制(8 线-3 线)编码器为例,说明这类编码器的工作原理。编码器的三位输出 Y_2、Y_1、Y_0 的不同取值组合分别代表 8 个输入信号 $\bar{I}_0 \sim \bar{I}_7$,输入信号低电平有效,其真值表如表 3-11 所示。

表 3-11 3 位二进制编码器的真值表

$\bar{I}_0$	$\bar{I}_1$	$\bar{I}_2$	$\bar{I}_3$	$\bar{I}_4$	$\bar{I}_5$	$\bar{I}_6$	$\bar{I}_7$	Y_2	Y_1	Y_0
0	1	1	1	1	1	1	1	0	0	0
1	0	1	1	1	1	1	1	0	0	1
1	1	0	1	1	1	1	1	0	1	0
1	1	1	0	1	1	1	1	0	1	1
1	1	1	1	0	1	1	1	1	0	0
1	1	1	1	1	0	1	1	1	0	1
1	1	1	1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	0	1	1	1

由表 3-11 可知,当仅有某一个输入端为低电平时,就输出与该输入端相对应的代码。例如,当 $\bar{I}_3$ 为低电平 **0**,而其他输入端均为高电平 **1** 时,输出 $Y_2Y_1Y_0$ 为 **011**。表中列出了 **8** 种输入信号的组合状态,每种状态的输入变量仅有一个取值为 **0**,其他未列出的状态是无关项,即任意时刻只能对一个输入信号进行编码。为避免产生乱码,该编码器不能接收两个或两个以上的编码信号请求。

根据表 3-11 可以得到该编码器三个输出信号的逻辑表达式

$$\begin{aligned}
Y_2 &= \bar{I}_0\bar{I}_1\bar{I}_2\bar{I}_3I_4\bar{I}_5\bar{I}_6\bar{I}_7 + \bar{I}_0\bar{I}_1\bar{I}_2\bar{I}_3\bar{I}_4I_5\bar{I}_6\bar{I}_7 \\
&\quad + \bar{I}_0\bar{I}_1\bar{I}_2\bar{I}_3\bar{I}_4\bar{I}_5I_6\bar{I}_7 + \bar{I}_0\bar{I}_1\bar{I}_2\bar{I}_3\bar{I}_4\bar{I}_5\bar{I}_6I_7 \\
Y_1 &= \bar{I}_0\bar{I}_1I_2\bar{I}_3\bar{I}_4\bar{I}_5\bar{I}_6\bar{I}_7 + \bar{I}_0\bar{I}_1\bar{I}_2I_3\bar{I}_4\bar{I}_5\bar{I}_6\bar{I}_7 \\
&\quad + \bar{I}_0\bar{I}_1\bar{I}_2\bar{I}_3\bar{I}_4\bar{I}_5I_6\bar{I}_7 + \bar{I}_0\bar{I}_1\bar{I}_2\bar{I}_3\bar{I}_4\bar{I}_5\bar{I}_6I_7 \\
Y_0 &= \bar{I}_0I_1\bar{I}_2\bar{I}_3\bar{I}_4\bar{I}_5\bar{I}_6\bar{I}_7 + \bar{I}_0\bar{I}_1\bar{I}_2I_3\bar{I}_4\bar{I}_5\bar{I}_6\bar{I}_7 \\
&\quad + \bar{I}_0\bar{I}_1\bar{I}_2\bar{I}_3\bar{I}_4I_5\bar{I}_6\bar{I}_7 + \bar{I}_0\bar{I}_1\bar{I}_2\bar{I}_3\bar{I}_4\bar{I}_5\bar{I}_6I_7
\end{aligned} \tag{3-24}$$

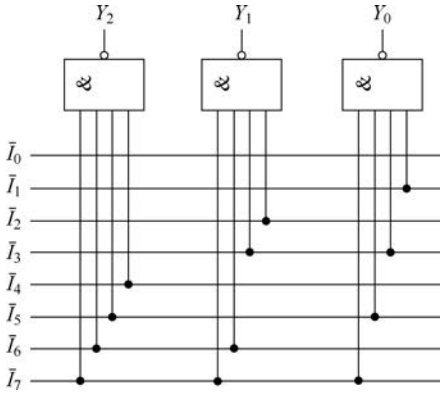


图 3-24 "与非"门构成的 3 位二进制编码器逻辑图

利用约束项化简式(3-24)得

$$\begin{aligned}
Y_2 &= I_4 + I_5 + I_6 + I_7 = \overline{\bar{I}_4\bar{I}_5\bar{I}_6\bar{I}_7} \\
Y_1 &= I_2 + I_3 + I_6 + I_7 = \overline{\bar{I}_2\bar{I}_3\bar{I}_6\bar{I}_7} \\
Y_0 &= I_1 + I_3 + I_5 + I_7 = \overline{\bar{I}_1\bar{I}_3\bar{I}_5\bar{I}_7}
\end{aligned} \tag{3-25}$$

图 3-24 所示的 8 线-3 线编码器逻辑图就是按照式(3-25)得出的。

2) 二进制优先编码器

普通二进制编码器虽然比较简单,但当两个或更多个输入信号同时有效时,其输出将是混乱的。而优先编码器则不同,它允许几个信号同时输入,但每一时刻输出端只给出优先级别较高的那个输入信号所对应

的代码，不处理优先级别低的信号。至于优先级别的高低，完全是由设计人员根据各输入信号的轻重缓急情况而决定的。对多个请求信号的优先级别进行编码的逻辑部件称为优先编码器。下面以 3 位二进制优先编码器为例分析其原理特性。

编码器的 8 个输入信号为 $I_0 \sim I_7$（高电平有效），设 I_7 的优先级最高，I_0 的优先级最低，Y_2、Y_1、Y_0 为三位代码输出，其真值表如表 3-12 所示。

表 3-12　3 位二进制优先编码器的真值表

I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Y_2	Y_1	Y_0
1	×	×	×	×	×	×	×	1	1	1
0	1	×	×	×	×	×	×	1	1	0
0	0	1	×	×	×	×	×	1	0	1
0	0	0	1	×	×	×	×	1	0	0
0	0	0	0	1	×	×	×	0	1	1
0	0	0	0	0	1	×	×	0	1	0
0	0	0	0	0	0	1	×	0	0	1
0	0	0	0	0	0	0	1	0	0	0

由表 3-12 求出该编码器三个输出信号的逻辑表达式，并化简得

$$
\begin{aligned}
Y_2 &= I_4\bar{I}_5\bar{I}_6\bar{I}_7 + I_5\bar{I}_6\bar{I}_7 + I_6\bar{I}_7 + I_7 \\
&= I_4 + I_5 + I_6 + I_7 \\
Y_1 &= I_2\bar{I}_3\bar{I}_4\bar{I}_5\bar{I}_6\bar{I}_7 + I_3\bar{I}_4\bar{I}_5\bar{I}_6\bar{I}_7 + I_6\bar{I}_7 + I_7 \\
&= I_2\bar{I}_4\bar{I}_5 + I_3\bar{I}_4\bar{I}_5 + I_6 + I_7 \\
Y_0 &= I_1\bar{I}_2\bar{I}_3\bar{I}_4\bar{I}_5\bar{I}_6\bar{I}_7 + I_3\bar{I}_4\bar{I}_5\bar{I}_6\bar{I}_7 + I_5\bar{I}_6\bar{I}_7 + I_7 \\
&= I_1\bar{I}_2\bar{I}_4\bar{I}_6 + I_3\bar{I}_4\bar{I}_6 + I_5\bar{I}_6 + I_7
\end{aligned}
\tag{3-26}
$$

由式(3-26)可绘制 3 位二进制优先编码器的逻辑图，如图 3-25 所示。

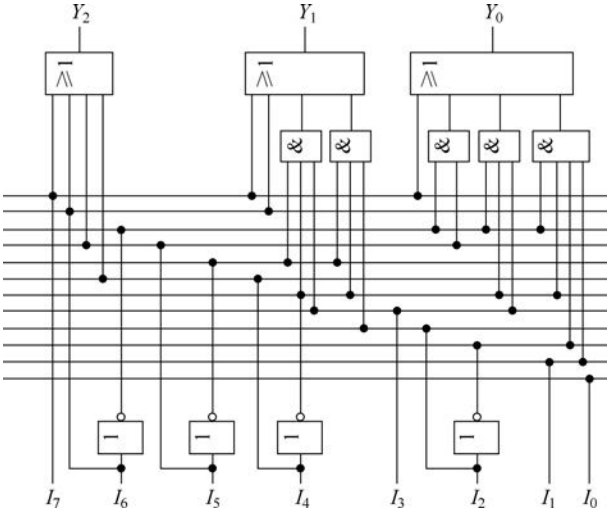


图 3-25　3 位二进制优先编码器逻辑图

常用的二进制优先编码器是 8 线-3 线优先编码器 74LS148，其简易图形符号如图 3-26 所示。为了便于级联扩展，74LS148 增加了使能端 $\overline{ST}$ 及优先扩展端 $\bar{Y}_{EX}$ 和 $\bar{Y}_S$，均为低电平有效。74LS148 的内部电路不再给出，读者可查阅相关资料。

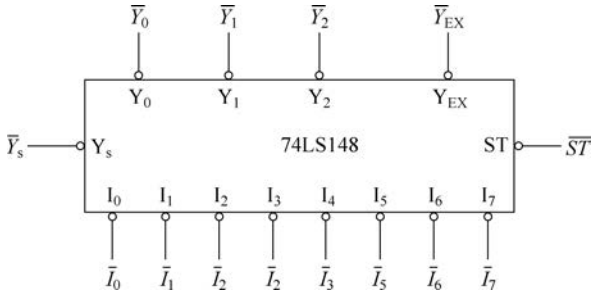


图 3-26　74LS148 的简易图形符号

【提示】 图形符号框外的引脚标注的变量符号，反变量表示低电平有效，并在近框处画有小圆圈。原变量表示高电平有效，近框处无小圆圈。

表 3-13 为 74LS148 的功能表。当 $\overline{ST}=\mathbf{1}$ 时，电路处于禁止状态，即禁止编码，输出端均为高电平。当 $\overline{ST}=\mathbf{0}$ 时，电路处于编码状态，即允许编码。只有当 $\overline{I}_0\sim\overline{I}_7$ 都为 **1** 时，$\overline{Y}_S$ 才为 **0**，其余情况 $\overline{Y}_S$ 均为 **1**，故 $\overline{Y}_S=\mathbf{0}$ 表示“电路工作，但无编码输入”；当编码输入至少有一个为有效电平时，$\overline{Y}_{EX}=\mathbf{0}$，表示“电路工作，且有编码输入”。

表 3-13　74LS148 的功能表

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

当 $\overline{ST}=\mathbf{0}$ 时，只有当 $\overline{I}_1$、$\overline{I}_2$、$\overline{I}_3$、$\overline{I}_4$、$\overline{I}_5$、$\overline{I}_6$、$\overline{I}_7$ 均为 **1**，即均为无效电平输入，且 $\overline{I}_0$ 为 **0** 时，输出为 **111**；当 $\overline{I}_7$ 为 **0** 时，无论其他 7 个输入是否为有效电平输入，输出均为 **000**。由此可知 $\overline{I}_7$ 的优先级别高于 $\overline{I}_0$ 的优先级别，且这 8 个输入优先级别的高低次序依次为 $\overline{I}_7\sim\overline{I}_0$，下角标号码越大的优先级别越高。

【例 3-15】 用两片 8 线-3 线优先编码器 74LS148 组成一个 16 线-4 线优先编码器，将 $\overline{A}_{15}\sim\overline{A}_0$16 个低电平有效的输入信号编为 **0000**～**1111** 16 个 4 位二进制代码，其中 $\overline{A}_{15}$ 优先级别最高，$\overline{A}_0$ 优先级别最低。

解：根据 74LS148 的功能表，将 16 个输入信号分别接到两个芯片上，其中优先级别高的 $\overline{A}_{15}\sim\overline{A}_8$ 接 74LS148 片(2)的 $\overline{I}_7\sim\overline{I}_0$，而将优先级别低的 $\overline{A}_7\sim\overline{A}_0$ 接到 74LS148 片(1)的 $\overline{I}_7\sim\overline{I}_0$。

按照优先次序的要求，只有片(2)的输入端无信号时，才允许对片(1)的输入信号编码。因此，只要将片(2)的 $\overline{Y}_S$ 作为片(1)的选通输入信号 $\overline{ST}$ 即可。

另外，当片(2)有编码信号输入时，它的 $\overline{Y}_{EX}=\mathbf{0}$，无编码输入时 $\overline{Y}_{EX}=\mathbf{1}$，正好可以用它作为输出编码的最高位。编码输出的低 3 位为两片相应原码输出的逻辑“或”。连接图如

图 3-27 所示。

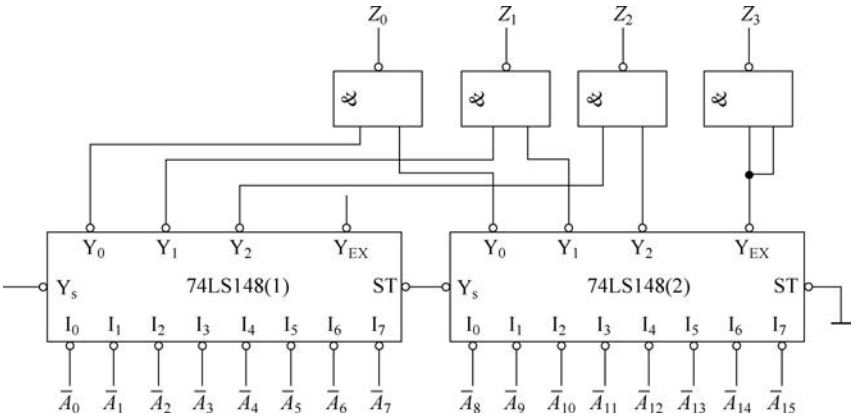


图 3-27　用两片 74LS148 组成的 16 线-4 线优先编码器

图 3-27 中，当 $\overline{A}_{15}\sim\overline{A}_8$ 中有低电平输入时，片(2)的输出端 $\overline{Y}_S=\mathbf{1}$，$\overline{Y}_{EX}=\mathbf{0}$，使片(1)的选通端 $\overline{ST}=\mathbf{1}$，片(1)不编码，其输出 $\overline{Y}_2\overline{Y}_1\overline{Y}_0=\mathbf{111}$，不影响片(2)对 $\overline{A}_{15}\sim\overline{A}_8$ 的编码操作。当 $\overline{A}_{15}\sim\overline{A}_8$ 均为高电平时，片(1)才能对 $\overline{A}_7\sim\overline{A}_0$ 进行优先编码操作，所以片(2)的优先级别高于片(1)。$Z_3\sim Z_0$ 将反码输出转换为原码输出。

2. 二-十进制优先编码器

用 4 位二进制代码表示 1 位十进制数(也可以是十种其他信息)称为二-十进制编码。完成二-十进制编码的电路称为二-十进制编码器，它能将 $I_0\sim I_9$(对应 0～9)10 个有效的输入信号编成 8421BCD 码。表 3-14 是二-十进制编码器的真值表。

表 3-14　二-十进制编码器的真值表

I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Y_3	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	1	0	0	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1	0	0	0	0

由表 3-14 可以求出二-十进制编码器四个输出信号的逻辑表达式，并进行“与非-与非”变换得

$$
\begin{aligned}
Y_3&=I_8+I_9=\overline{\overline{I}_8\overline{I}_9}\\
Y_2&=I_4+I_5+I_6+I_7=\overline{\overline{I}_4\overline{I}_5\overline{I}_6\overline{I}_7}\\
Y_1&=I_2+I_3+I_6+I_7=\overline{\overline{I}_2\overline{I}_3\overline{I}_6\overline{I}_7}\\
Y_0&=I_1+I_3+I_5+I_7+I_9=\overline{\overline{I}_1\overline{I}_3\overline{I}_5\overline{I}_7\overline{I}_9}
\end{aligned}
\tag{3-27}
$$

由式(3-27)可绘制二-十进制编码器的逻辑图,如图 3-28 所示。

图 3-29 是集成二-十进制优先编码器 74LS147 的简易图形符号,表 3-15 是 74LS147 的功能表。

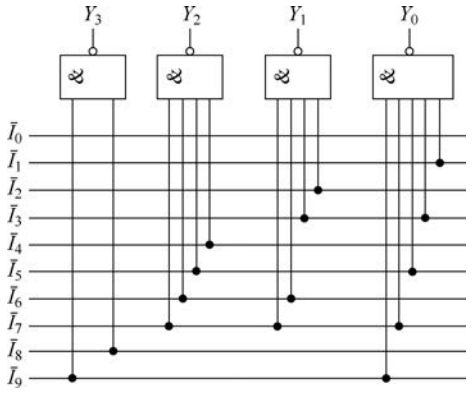


图 3-28 二-十进制编码器逻辑图

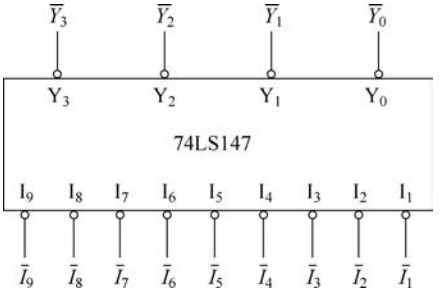


图 3-29 74LS147 的简易图形符号

由表 3-15 可以看出,编码器的输入信号低电平有效,输出是 8421 码的反码。$\bar{I}_9$ 的优先级最高,$\bar{I}_0$ 的优先级最低,即只要 $\bar{I}_9$ 有低电平输入,无论其他输入端是什么,输出都是 **0110**。电路中没有 $\bar{I}_0$ 输入端,当所有的输入端都为高电平时,相当于 $\bar{I}_0$ 端有效,这时四个输出端输出的是 **1111**。

表 3-15 74LS147 的功能表

$\bar{I}_9$	$\bar{I}_8$	$\bar{I}_7$	$\bar{I}_6$	$\bar{I}_5$	$\bar{I}_4$	$\bar{I}_3$	$\bar{I}_2$	$\bar{I}_1$	$\bar{Y}_3$	$\bar{Y}_2$	$\bar{Y}_1$	$\bar{Y}_0$
1	1	1	1	1	1	1	1	1	1	1	1	1
0	×	×	×	×	×	×	×	×	0	1	1	0
1	0	×	×	×	×	×	×	×	0	1	1	1
1	1	0	×	×	×	×	×	×	1	0	0	0
1	1	1	0	×	×	×	×	×	1	0	0	1
1	1	1	1	0	×	×	×	×	1	0	1	0
1	1	1	1	1	0	×	×	×	1	0	1	1
1	1	1	1	1	1	0	×	×	1	1	0	0
1	1	1	1	1	1	1	0	×	1	1	0	1
1	1	1	1	1	1	1	1	0	1	1	1	0

微课视频

3.5.2 译码器

译码是将表示特定意义信息的二进制代码翻译出来,是编码的逆过程。实现译码操作的电路称为“译码器”,它输入的是二进制代码,输出的是与输入代码对应的特定信息。

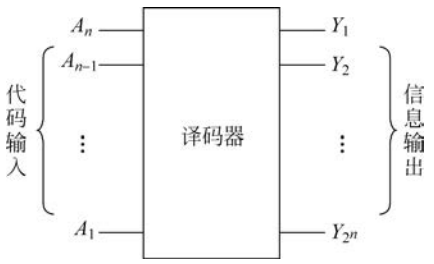


图 3-30 二进制译码器框图

常用的译码器有二进制译码器、二-十进制译码器和显示驱动译码器等。

1. 二进制译码器

图 3-30 是二进制译码器的框图。图中 $A_1 \sim A_n$ 是 n 个输入信号,组成 n 位二进制代码,A_n 是代码的

最高位，A_1 是代码的最低位。代码可能是原码，也可能是反码。若为反码，则字母 A 上面要带反号。$Y_1 \sim Y_{2^n}$ 是 2^n 个输出信号，可能是高电平有效，也可能是低电平有效。若为低电平有效，则字母 Y 上面要带反号，这种译码器称为 n 线-2^n 线译码器。

对于 n 线-2^n 线译码器的每一种输入代码，输出只能有一个有效，其余均无效。二进制译码器可以译出输入变量的全部状态，所以又称为变量译码器或全译码器。表 3-16 是 3 位二进制译码器的真值表，输入是 3 位二进制代码，输出是 8 个互斥的信号。

表 3-16　3 位二进制译码器的真值表

A_2	A_1	A_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

由真值表 3-16 可以写出输出逻辑表达式

$$
\begin{aligned}
Y_7 &= A_2A_1A_0 = m_7 \\
Y_6 &= A_2A_1\overline{A}_0 = m_6 \\
Y_5 &= A_2\overline{A}_1A_0 = m_5 \\
Y_4 &= A_2\overline{A}_1\overline{A}_0 = m_4 \\
Y_3 &= \overline{A}_2A_1A_0 = m_3 \\
Y_2 &= \overline{A}_2A_1\overline{A}_0 = m_2 \\
Y_1 &= \overline{A}_2\overline{A}_1A_0 = m_1 \\
Y_0 &= \overline{A}_2\overline{A}_1\overline{A}_0 = m_0
\end{aligned}
\tag{3-28}
$$

由式(3-28)可以看出，译码器的每个输出都与输入代码的一个最小项对应。依据式(3-28)可以画出 3 位二进制译码器的逻辑图，如图 3-31 所示。

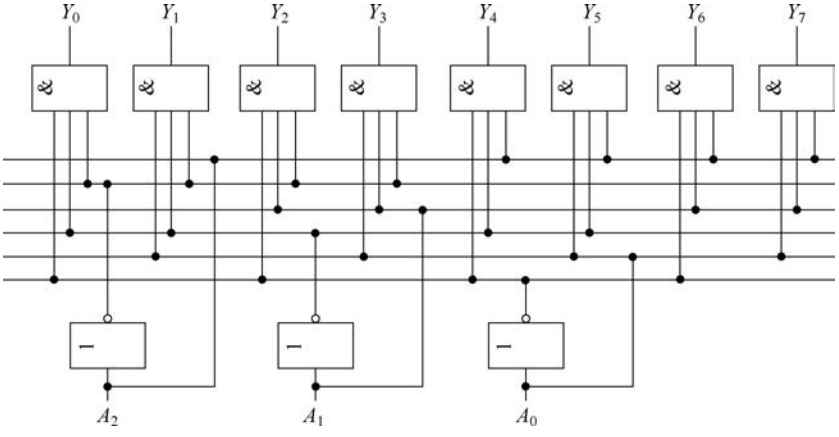


图 3-31　3 位二进制译码器的逻辑图

集成3线-8线译码器74LS138的输出采用低电平有效方式,即输出为反变量,而且为了功能扩展和应用,增加了使能控制信号。74LS138的功能如表3-17所示,其中ST_A、$\overline{ST}_B$、$\overline{ST}_C$是使能端。当$ST_A=1$且$\overline{ST}_B=\overline{ST}_C=0$时,译码器才工作,否则译码器处于禁止状态。

表3-17 74LS138的功能表

ST_A	$\overline{ST}_B$	$\overline{ST}_C$	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
1	1	×	×	×	×	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

由表3-17可以看出,其输入信号为原码,A_2是最高位。译码过程中,根据A_2、A_1、A_0的取值组合,$\overline{Y}_0\sim\overline{Y}_7$中某一个输出为低电平。译码输出表达式为

$$\overline{Y}_i=ST_A\cdot\overline{\overline{ST}_B}\cdot\overline{\overline{ST}_C}\cdot\overline{m}_i \tag{3-29}$$

74LS138的逻辑图和简易图形符号分别如图3-32、图3-33所示。

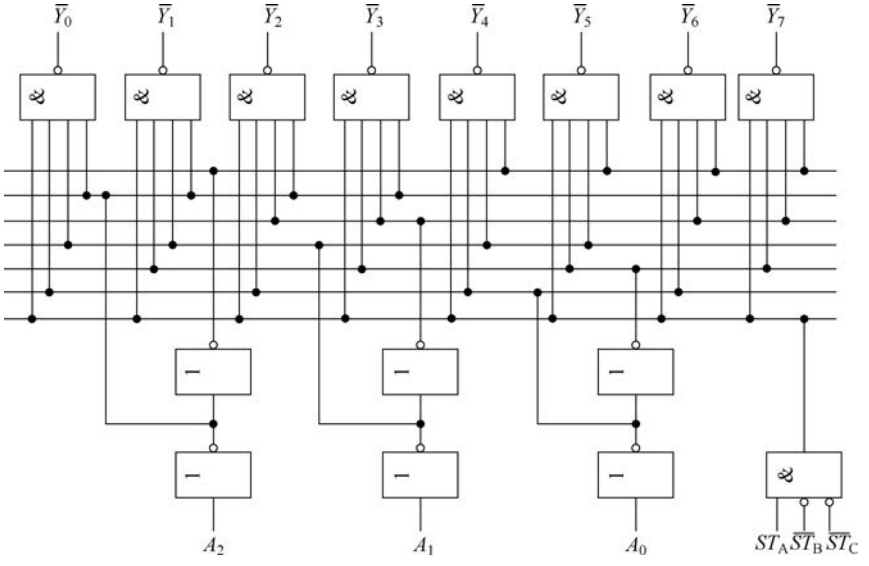


图3-32 74LS138的逻辑图

【提示】 译码输出的是三个变量的全部最小项,这一特点是全译码器所共有的,据此可以用集成译码器实现组合逻辑函数。当代码位数较多时,可以利用使能端将多个译码器级联使用。

2. 二-十进制译码器

将输入的四位 8421BCD 码翻译成十个对应的高、低电平输出信号(用来表示 0～9 共十个数字)的逻辑电路称为二-十进制译码器,又称 4 线-10 线译码器。常用的 4 线-10 线译码器是 74LS42,表 3-18 是其功能表,输入的四位 8421BCD 码用 D、C、B、A 表示,输出的 0～9 十个十进制数对应的信号用 $\overline{Y}_0 \sim \overline{Y}_9$ 表示。

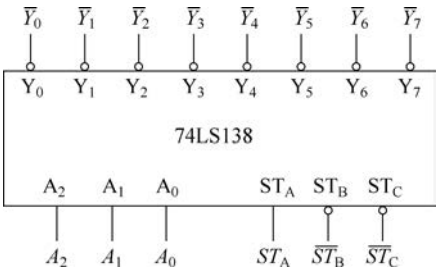


图 3-33　74LS138 的简易图形符号

表 3-18　74LS42 的功能表

数字	D	C	B	A	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
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
无效码	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

由表 3-18 可见,该电路输入端 D、C、B、A 输入的是 8421BCD 码,输出端有译码输出时为 **0**,没有译码输出时为 **1**,即低电平为有效输出信号。所以,当输入为 **1010～1111** 六个无效信号时,译码器输出全 **1**,即对无效信号拒绝译码。

由功能表 3-18 可以写出“与非”形式的输出表达式

$$
\begin{aligned}
\overline{Y}_0 &= \overline{\overline{D}\overline{C}\overline{B}\overline{A}} \qquad \overline{Y}_5 = \overline{D\overline{C}B\overline{A}} \\
\overline{Y}_1 &= \overline{\overline{D}\overline{C}\overline{B}A} \qquad \overline{Y}_6 = \overline{D\overline{C}BA} \\
\overline{Y}_2 &= \overline{\overline{D}\overline{C}B\overline{A}} \qquad \overline{Y}_7 = \overline{\overline{D}CBA} \\
\overline{Y}_3 &= \overline{\overline{D}\overline{C}BA} \qquad \overline{Y}_8 = \overline{D\overline{C}\overline{B}\overline{A}} \\
\overline{Y}_4 &= \overline{\overline{D}C\overline{B}\overline{A}} \qquad \overline{Y}_9 = \overline{D\overline{C}BA}
\end{aligned}
\tag{3-30}
$$

根据式(3-30),可以画出用“与非”门组成的 74LS42 的逻辑图,如图 3-34 所示。

图 3-35 是二-十进制译码器 74LS42 的简易图形符号。

3. 显示驱动译码器

显示驱动译码器不同于上述的译码器,它的主要功能是译码驱动数字显示器件。数字显示的方式一般分为字形重叠式、分段式和点阵式三种。

字形重叠式显示器是将不同字符的电极重叠起来,使相应的电极发亮,则可显示需要的字符。

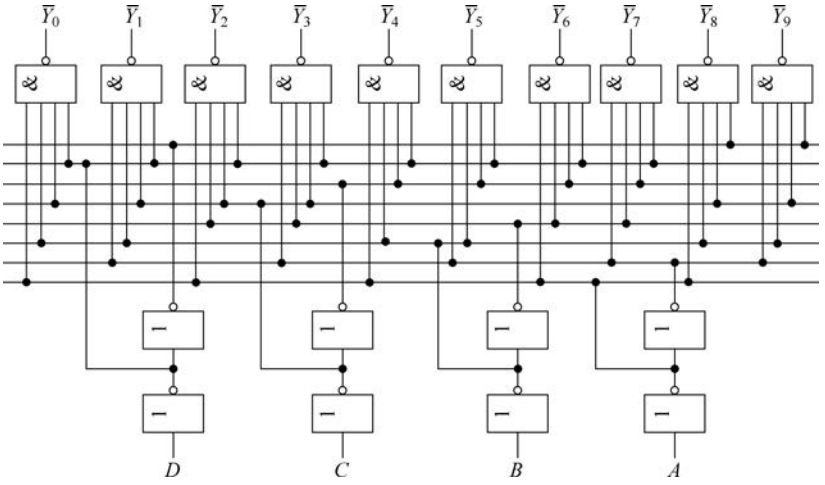


图 3-34　74LS42 的逻辑图

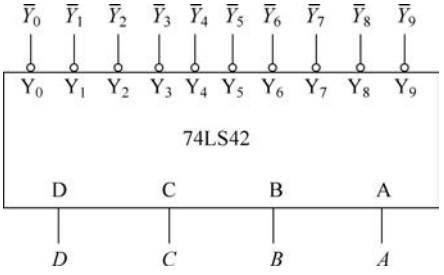


图 3-35　74LS42 的简易图形符号

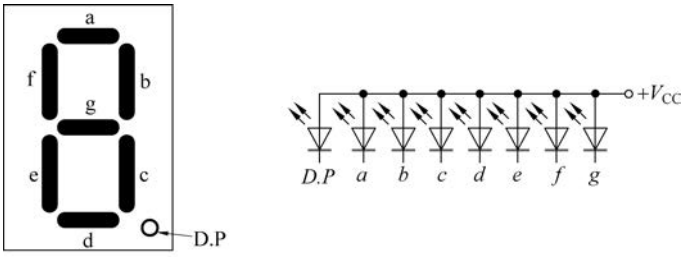
分段式显示器是在同一平面上按笔画分布发光段，利用不同发光段组合，显示不同的数码。

点阵式显示器是由按一定规律排列的可发光的点阵组成，通过发光点组合显示不同的数码。

数字显示方式以分段式应用最为普遍，下面先对常用的分段式显示器作一些介绍，然后对显示驱动译码器的原理进行分析。

1）七段 LED 数码管显示器

某些特殊的半导体材料做成的 PN 结，在外加一定的电压时，具有能将电能转化成光能的特性。利用这种 PN 结发光特性制作成显示器件，称为半导体显示器。多个发光二极管组成的七段 LED 数码管显示器就是半导体显示器，其外观及其等效电路如图 3-36 所示。



(a) 外观图　　(b) 等效电路

图 3-36　七段 LED 数码管显示器

LED 数码管有共阴极与共阳极两种，共阳极 LED 数码管的等效电路如图 3-36(b)所示，各字段发光二极管的阳极接在一起。在构成显示驱动译码器时，对于共阳极 LED 数码管，要使某段发光，该段应接低电平；对于共阴极 LED 数码管，要使某段发光，该段应接高电平。

半导体显示器的优点是体积小、工作可靠、寿命长、响应速度快、颜色丰富；其缺点是功耗较大。

2）七段 LED 显示驱动译码器

现以 8421BCD 码七段显示译码器为例，说明显示驱动译码器的工作原理。

8421BCD码七段显示译码器的真值表如表3-19所示。设$A_3 \sim A_0$是8421BCD码的输入端，经译码后产生驱动LED数码管的7个高电平有效的输出信号$Y_a \sim Y_g$，用于连接LED数码管的7个数码显示段a～g(不包括小数点的驱动)。这里规定输出**1**时为数码显示段的点亮状态，**0**为熄灭状态。

除了可对输入的**0000～1001**的8421BCD码进行显示译码外，真值表还规定了输入为**1010～1111**这六个状态下显示的字形。但由于这些字形比较奇异，故在实际应用中很少使用。

表3-19　8421BCD码七段显示译码器的真值表

数码	A_3	A_2	A_1	A_0	Y_a	Y_b	Y_c	Y_d	Y_e	Y_f	Y_g	字形
0	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	0
1	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	1
2	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	2
3	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	3
4	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	4
5	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	5
6	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	6
7	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	7
8	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	8
9	**1**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	9
10	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**1**	c
11	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	ɔ
12	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	u
13	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	ɔ
14	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	E
15	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	

由表3-19可以得出输出的最小项表达式，利用卡诺图化简**0**项，得到反函数的“与或”表达式，然后对反函数取反得到“与或非”表达式。这里省略化简和变换步骤，直接给出输出逻辑函数的“与或非”表达式。

$$
\begin{aligned}
Y_a &= \sum(0,2,3,5,7,8,9,13) = \overline{A_3A_1 + A_2\overline{A}_0 + \overline{A}_3\overline{A}_2\overline{A}_1A_0} \\
Y_b &= \sum(0,1,2,3,4,7,8,9,12) = \overline{A_3A_1 + A_2\overline{A}_1A_0 + A_2A_1\overline{A}_0} \\
Y_c &= \sum(0,1,3,4,5,6,7,8,9,11) = \overline{A_3A_2 + \overline{A}_2A_1\overline{A}_0} \\
Y_d &= \sum(0,2,3,5,6,8,10,11,13,14) = \overline{A_2\overline{A}_1\overline{A}_0 + A_2A_1A_0 + \overline{A}_2\overline{A}_1A_0} \\
Y_e &= \sum(0,2,6,8,10,14) = \overline{A_0 + A_2\overline{A}_1} \\
Y_f &= \sum(0,4,5,6,8,9,12,13,14) = \overline{A_1A_0 + \overline{A}_2A_1 + \overline{A}_3\overline{A}_2A_0} \\
Y_g &= \sum(2,3,4,5,6,8,9,10,11,12,13,14) = \overline{\overline{A}_3\overline{A}_2\overline{A}_1 + A_2A_1A_0}
\end{aligned}
\tag{3-31}
$$

由式(3-31)可以画出译码器的逻辑图，如图3-37所示。

集成显示驱动译码器74LS48是用于驱动共阴极LED数码管的BCD-七段显示译码器，它的内部有上拉电阻，输出状态为高电平有效。74LS48的简易图形符号如图3-38所示，符号中$A_3 \sim A_0$是8421BCD码的输入端，$Y_a \sim Y_g$是经译码后产生驱动共阴极LED数码管的7个

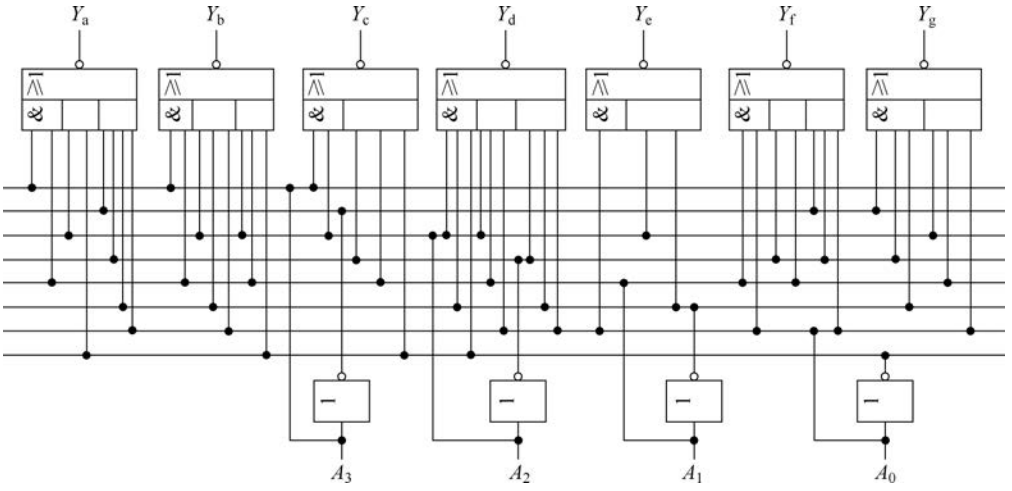


图 3-37　8421BCD 码七段显示译码器逻辑图

输出信号。74LS48 中还提供了附加控制电路,有三个附加控制端 $\overline{LT}$、$\overline{RBI}$ 和 $\overline{BI}/\overline{RBO}$,用以扩展电路的功能。

图 3-39 给出了 74LS48 与共阴极 LED 数码管的基本连接方法,供读者参考。

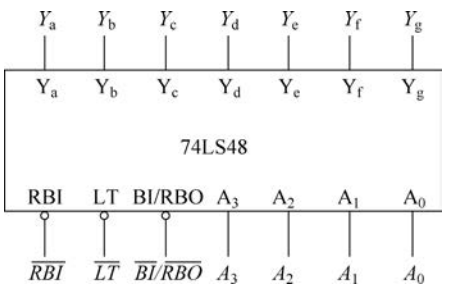


图 3-38　74LS48 的简易图形符号

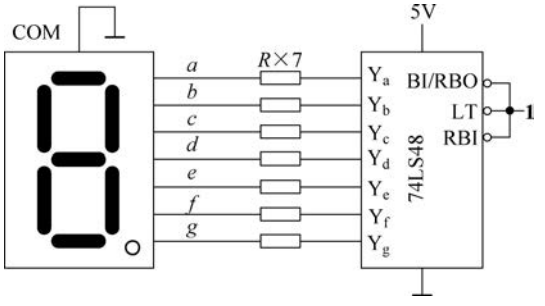


图 3-39　用 74LS48 驱动共阴极 LED 数码管的连接方法

74LS48 中附加控制端的功能如下:

(1) 灯测试输入端 $\overline{LT}$。$\overline{LT}$ 为低电平有效信号。当 $\overline{LT}=0$ 时,无论输入的 8421BCD 码为何状态,将强行使译码器的输出信号全部置成高电平,从而使 LED 数码管的 7 个数码显示段全部点亮。此输入端的功能主要是对数码管的七个显示段进行测试。正常工作时应将 $\overline{LT}$ 置为高电平。

(2) 灭零输入端 $\overline{RBI}$。$\overline{RBI}$ 为低电平有效信号。一般情况下,当显示译码器的输入 8421BCD 码为 **0000** 时,显示译码器的输出信号将使数码管显示为 **0**。但 $\overline{RBI}=$**0** 时,如果显示译码器的输入为 **0000**,则输出的信号将全部置为高阻态,数码管的七个显示段全部不亮,这就是所谓的"灭零"。然而,$\overline{RBI}$ 有效时,仅仅是对输入的 8421BCD 码为 **0000** 时产生"灭零"效果,如果输入的 8421BCD 码是其他数值,则显示译码器的输出仍同于一般情况,使数码管显示出相应数字。

(3) 灭灯输入/灭零输出端 $\overline{BI}/\overline{RBO}$。这是一个双功能的输入/输出端,其输入和输出均是低电平有效。$\overline{BI}/\overline{RBO}$ 端作为输入端时,称灭灯输入控制端。当将此端加上低电平时,则无论显示译码器的输入为什么状态,输出信号将被全部置为高阻态,使数码管的各个显示段全

部熄灭。$\overline{BI}/\overline{RBO}$ 端作为输出端时，称灭零输出端。当灭零输入端 $\overline{RBI}$ 处于有效状态且8421BCD码的输入为 **0000** 时，显示译码器实现“灭零”。此时，灭零输出端 $\overline{BI}/\overline{RBO}$ 输出低电平，表示本显示译码器处于“灭零”状态，将本应显示的0给熄灭了。

将 $\overline{RBI}$ 与 $\overline{RBO}$ 配合使用，可实现多位十进制数码显示系统的整数前和小数后的灭零控制。图3-40给出了灭零控制的连接方法，整数部分将高位的 $\overline{RBO}$ 与后一位的 $\overline{RBI}$ 相连。小数部分将低位的 $\overline{RBO}$ 与前一位的 $\overline{RBI}$ 相连。整数显示部分最高位译码器的 $\overline{RBI}$ 接地，始终处于有效状态，输入为0时将进行灭零操作，并通过 $\overline{RBO}$ 将灭零输出低电平向后传递，开启后一位灭零功能。小数显示部分最低位译码器的 $\overline{RBI}$ 始终处于有效状态，输入为0时将进行灭零操作，并通过 $\overline{RBO}$ 将灭零输出的低电平向前传递，开启前一位的灭零功能。

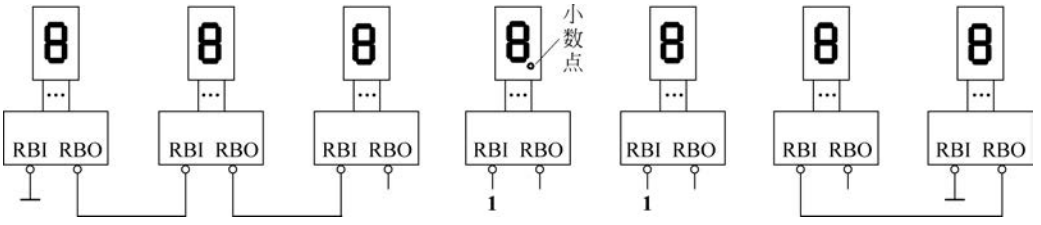


图3-40　有灭零控制的数码显示系统

4. 用译码器实现组合逻辑函数

任何逻辑函数都可以写成最小项表达式的形式，而对于具有 n 个输入的二进制译码器来说，它的 2^n 个输出恰恰对应输入信号的 2^n 个最小项，即 $Y_i=m_i$。所以，可以利用译码器的这一特点，并配合门电路构成组合逻辑电路。具体步骤如下：

(1) 由函数的自变量数确定译码器的线数，自变量数应与译码器的输入线数相等。

(2) 将组合逻辑函数转换成最小项表达式形式。

(3) 将函数的最小项表达式与译码器输出对比，并进行相应变换。

(4) 画出用译码器和门电路组成的逻辑图。

【例3-16】 用二进制译码器和“与非”门实现逻辑函数 $Y=AB+BC+AC$。

解：(1) 给定的组合逻辑函数 Y 为三变量逻辑函数，所以选择3线-8线译码器74LS138来设计实现该组合逻辑函数。

(2) 把逻辑函数 Y 写成最小项表达式的形式

$$\begin{aligned}Y&=AB+BC+AC\\&=AB(C+\overline{C})+(A+\overline{A})BC+A(B+\overline{B})C\\&=ABC+AB\overline{C}+\overline{A}BC+A\overline{B}C\\&=m_3+m_5+m_6+m_7\end{aligned}\tag{3-32}$$

(3) 由译码器74LS138功能可知，只要令 $A_2=A$，$A_1=B$，$A_0=C$，则它的输出 $\overline{Y}_0\sim\overline{Y}_7$ 即为三变量逻辑函数的8个最小项 $\overline{m}_0\sim\overline{m}_7$。由于这些最小项以反函数的形式给出，所以还需将式(3-32)变换为由 $\overline{m}_0\sim\overline{m}_7$ 表示的函数式

$$\begin{aligned}Y&=m_3+m_5+m_6+m_7\\&=\overline{\overline{m_3+m_5+m_6+m_7}}\\&=\overline{\overline{m}_3\cdot\overline{m}_5\cdot\overline{m}_6\cdot\overline{m}_7}\end{aligned}\tag{3-33}$$

对比译码器74LS138的输出，式(3-33)可以写成

$$Y=\overline{\overline{Y}_3\cdot\overline{Y}_5\cdot\overline{Y}_6\cdot\overline{Y}_7} \tag{3-34}$$

(4) 由式(3-34)画出逻辑图,如图 3-41 所示。

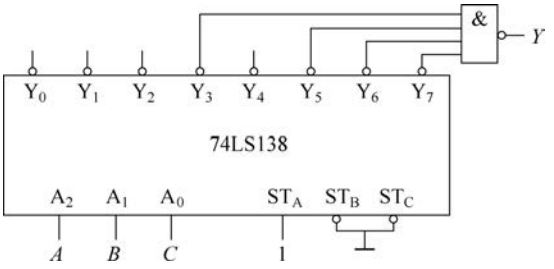


图 3-41 例 3-16 的逻辑图


微课视频

3.5.3 加法器

二进制加法器是数字系统的基本逻辑部件之一。两个二进制数之间的加、减、乘、除等算术运算,最后都可以化作加法运算来实现。能够实现加法运算的电路称为加法器,加法器是算术运算的基本单元电路。下面先讨论能实现 1 位二进制数相加的半加器和全加器,然后探讨多位二进制数加法器。

1. 半加器和全加器

如果不考虑来自低位的进位而将两个一位二进制数相加,称为半加。实现半加运算的逻辑电路叫作半加器。

若用 A、B 表示两个加数输入,S、CO 分别表示和与进位输出。根据半加器的逻辑功能,可以得出其真值表,如表 3-20 所示。

表 3-20 半加器的真值表

A	*B*	*S*	*CO*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

由真值表可以求出 S 和 CO 的表达式

$$\begin{cases} S=A\overline{B}+\overline{A}B=A\oplus B \\ CO=AB \end{cases} \tag{3-35}$$

式(3-35)可用图 3-42(a)所示的逻辑电路实现。半加器的逻辑符号如图 3-42(b)所示。

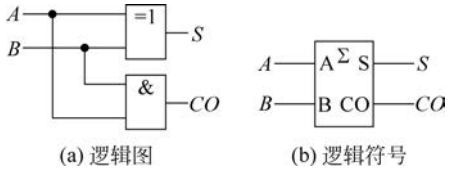


图 3-42 半加器的逻辑图和逻辑符号

如果不仅考虑两个一位二进制数相加,而且考虑来自低位进位的加法运算称为全加。实现全加运算的逻辑电路叫作全加器。设 A、B 为两个加数,CI 是来自低位的进位,S 为本位的和,CO 是向高位的进位,根据全加器的逻辑功能,可以得到其真值表,如表 3-21 所示。

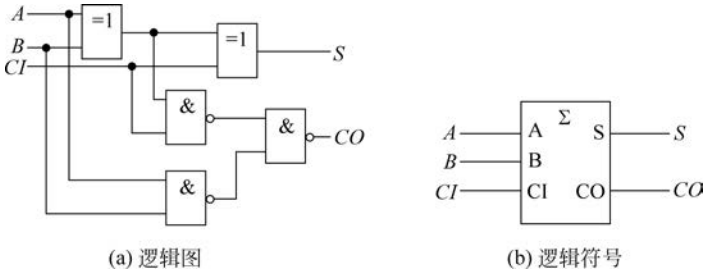
表 3-21　全加器的真值表

A	*B*	*CI*	*CO*	*S*
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

由表 3-21 可以写出全加器的逻辑函数表达式，并进行相应变换得

$$
\begin{cases}
S &= \sum(1,2,4,7) \\
&= \overline{A}\,\overline{B}CI + \overline{A}B\overline{CI} + A\overline{B}\,\overline{CI} + ABCI \\
&= (\overline{A}\,\overline{B} + AB)CI + (\overline{A}B + A\overline{B})\overline{CI} \\
&= A \oplus B \oplus CI \\
CO &= \sum(3,5,6,7) \\
&= \overline{A}BCI + A\overline{B}CI + AB\overline{CI} + ABCI \\
&= AB + (A \oplus B)CI \\
&= \overline{\overline{AB + (A \oplus B)CI}} \\
&= \overline{\overline{AB} \cdot \overline{(A \oplus B)CI}}
\end{cases}
\tag{3-36}
$$

全加器的电路结构有多种类型，图 3-43(a)是用“异或”门和“与非”门构成的全加器。不论哪种电路结构，其功能必须符合表 3-21 给出的全加器真值表。全加器的逻辑符号如图 3-43(b)所示。



(a) 逻辑图　　(b) 逻辑符号

图 3-43　全加器的逻辑图和逻辑符号

2. 多位加法器

两个多位二进制数进行加法运算时，前面讲的全加器是不能完成的。必须把多个这样的全加器连接起来使用。即把相邻的第一位全加器的 *CO* 连接到高一位全加器的 *CI* 端，最低一位相加时可以使用半加器，也可以使用全加器。使用全加器时，需要把 *CI* 端接低电平 **0**，这样组成的加法器称为串行进位加法器，如图 3-44 所示。

由于电路的进位是从低位到高位依次连接而成的，所以必须等到低位的进位产生并送到相邻的高位以后，相邻的高一位才能产生相加的结果和进位输出。所以，串行进位加法器的缺

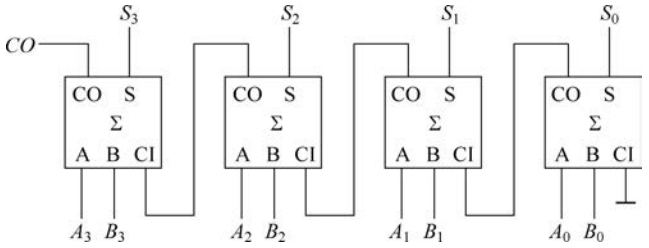
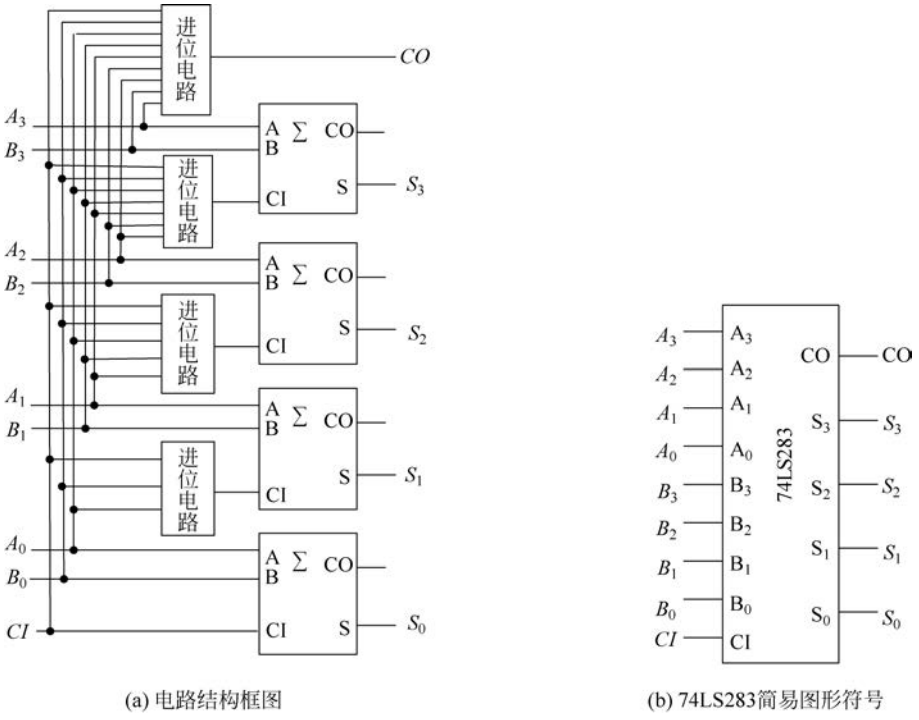


图 3-44　4 位串行进位加法器

点是运行速度慢,只能用在对工作速度要求不太高的场合。串行进位加法器的优点是电路简单。TTL 集成电路中的 T692 就属于此类加法器。

为了提高运算速度,通常使用超前进位并行加法器。图 3-45(a)是 4 位二进制超前进位加法器的电路结构框图,图 3-45(b)是中规模集成电路 4 位二进制超前进位加法器 74LS283 的简易图形符号。其中 $A_3 \sim A_0$、$B_3 \sim B_0$ 分别为 4 位加数和被加数的输入端,$S_3 \sim S_0$ 为四位和的输出端,CI 为最低进位输入端,CO 为向高位输送进位的输出端。



(a) 电路结构框图　　(b) 74LS283简易图形符号

图 3-45　4 位二进制超前进位加法器

在两个多位数相加时,任何一位的进位输入信号都取决于两个加数中低于该位的各位数值,在给出两个多位数后,可以通过进位电路直接判断出每一位的进位输入信号,使进位信号不再逐级传递,从而提高运算速度,这种方法称为超前进位。超前进位加法器 74LS283 实现的加法关系为

$$(CO, S_3S_2S_1S_0) = A_3A_2A_1A_0 + B_3B_2B_1B_0 + CI \tag{3-37}$$

各级进位信号仅由加数、被加数和最低位信号 CI 决定,而与其他进位无关,这就有效地提高了运算速度。需要注意的是,加法器速度越高,位数越多,电路越复杂。目前中规模集成超前进位加法器多为四位。若实现更多位的加法运算,需将多个四位加法器串接使用。

3. 用加法器实现组合逻辑函数

加法器能实现两个二进制数相加，如果某个逻辑函数能表示为某些输入变量相加或输入变量与常量相加的形式，则用加法器来设计组合逻辑电路会更简单。

【例 3-17】 用超前进位加法器 74LS283 设计一个代码转换电路，以将余 3 码转换为 8421 码。

解：根据设计要求，电路的输入为余 3 码，用 $ABCD$ 表示；电路的输出为 8421 码，用 $Y_3Y_2Y_1Y_0$ 表示。由代码的编码规则可知，余 3 码是 8421 码加 3 得到的，即 8421 码可以由余 3 码加(−3)得到。所以只要将 $ABCD$ 和(−3)的补码 **1101** 作为加数和被加数接入 74LS283 的输入端 $A_3 \sim A_0$、$B_3 \sim B_0$，即可从 $S_3 \sim S_0$ 端得到 8421 码，这时在 CO 端会产生进位，忽略即可。电路连接如图 3-46 所示。

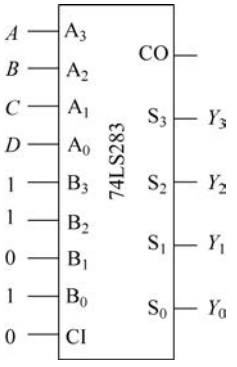


图 3-46　例 3-17 的电路连接图

3.5.4　数据选择器

微课视频

能从一组输入数据中选择出某一数据的电路叫数据选择器。数据选择器由地址译码器和多路数字开关组成，如图 3-47 所示。它有 n 个选择输入端(也称为地址输入端)，2^n 个数据输入端，一个数据输出端。数据输入端与选择输入端输入的地址码有一一对应关系，当地址码确定后，输出端就输出与该地址码有对应关系的数据输入端的数据，即将与该地址码有对应关系的数据输入端和输出端相接。

1. 4 选 1 数据选择器

图 3-48 是 4 选 1 数据选择器的功能示意图。图中 $D_0 \sim D_3$ 为 4 个数据输入端；Y 为输出端；A_1、A_0 为地址输入端，其真值表如表 3-22 所示。

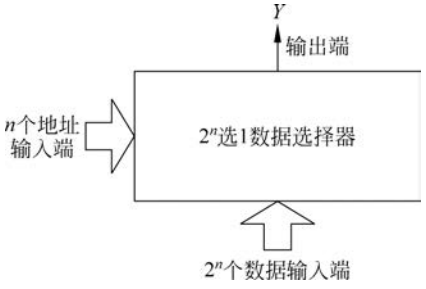


图 3-47　数据选择器的框图

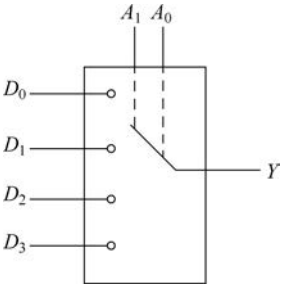


图 3-48　4 选 1 数据选择器功能示意图

表 3-22　4 选 1 数据选择器真值表

A_1	A_0	D_0	D_1	D_2	D_3	Y
0	**0**	D_0	×	×	×	D_0
0	**1**	×	D_1	×	×	D_1
1	**0**	×	×	D_2	×	D_2
1	**1**	×	×	×	D_3	D_3

由真值表 3-22 可以写出输出信号 Y 的表达式

$$Y=\overline{A}_1\overline{A}_0D_0+\overline{A}_1A_0D_1+A_1\overline{A}_0D_2+A_1A_0D_3=\sum_{i=0}^{3}m_iD_i \tag{3-38}$$

式中，m_i 为地址输入变量 A_1、A_0 的最小项，依据表达式(3-38)可以画出 4 选 1 数据选择器的逻辑图，如图 3-49 所示。

集成 4 选 1 数据选择器的典型电路是 74LS153。74LS153 内部有两片功能完全相同的 4 选 1 数据选择器，通常称为双 4 选 1 数据选择器。

74LS153 中设有选通端 $\overline{S}$，低电平有效。当 $\overline{S}=\mathbf{1}$ 时，$Y=\mathbf{0}$，数据选择器不工作。当 $\overline{S}=\mathbf{0}$ 时，数据选择器才工作。于是，式(3-38)应改写为

$$Y=(\overline{A}_1\overline{A}_0D_0+\overline{A}_1A_0D_1+A_1\overline{A}_0D_2+A_1A_0D_3)\overline{\overline{S}}=\overline{\overline{S}}\sum_{i=0}^{3}m_iD_i \tag{3-39}$$

当 $\overline{S}=\mathbf{0}$ 时，根据地址码 A_1A_0 的不同，将从 $D_0\sim D_3$ 中选出一个数据输出。即地址码 A_1A_0 分别为 **00**、**01**、**10**、**11** 时，输出分别为 D_0、D_1、D_2、D_3。

表 3-23 是 74LS153 中一片 4 选 1 数据选择器的功能表。74LS153 的简易图形符号如图 3-50 所示。

表 3-23　74LS153 中一片 4 选 1 数据选择器的功能表

$\overline{S}$	A_1	A_0	Y
1	×	×	**0**
0	**0**	**0**	D_0
0	**0**	**1**	D_1
0	**1**	**0**	D_2
0	**1**	**1**	D_3

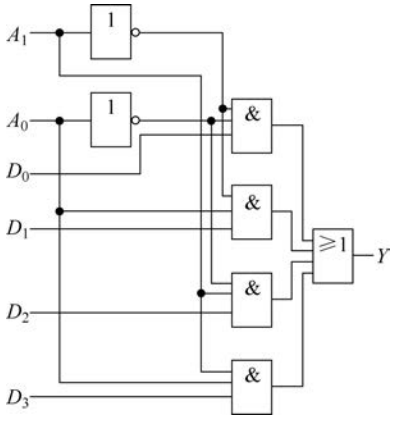


图 3-49　4 选 1 数据选择器的逻辑图

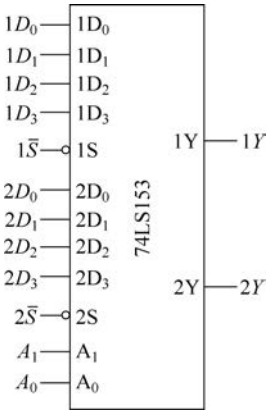


图 3-50　74LS153 的简易图形符号

2. 8 选 1 数据选择器

集成 8 选 1 数据选择器 74LS151 的功能如表 3-24 所示。可以看出，74LS151 有一个使能端 $\overline{S}$，低电平有效；A_2、A_1、A_0 为地址输入端；有两个互补输出端 Y 和 $\overline{Y}$，其输出信号相反。

表 3-24　74LS151 的功能表

$\overline{S}$	A_2	A_1	A_0	Y	$\overline{Y}$
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	$\overline{D}_3$

续表

$\overline{S}$	A_2	A_1	A_0	Y	$\overline{Y}$
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	$\overline{D}_7$

由功能表可以写出输出逻辑函数 Y 的表达式

$$Y=\overline{S}\sum_{i=0}^{7}m_iD_i \tag{3-40}$$

式中，m_i 为地址输入变量 A_2、A_1、A_0 的最小项。当 $\overline{S}=\mathbf{1}$ 时，$Y=\mathbf{0}$，数据选择器不工作；当 $\overline{S}=\mathbf{0}$ 时，根据地址码 $A_2A_1A_0$ 的不同取值，将从 $D_0\sim D_7$ 中选出一个数据输出。74LS151 的简易图形符号如图 3-51 所示。

3. 用数据选择器实现组合逻辑函数

从前面的分析可知，数据选择器输出信号的逻辑表达式具有以下特点：

（1）具有标准“与或”表达式（最小项表达式）的形式；

（2）提供了地址变量的全部最小项；

（3）一般情况下，输入信号 D_i 可以当成一个变量处理。

任何组合逻辑函数都可以写成唯一的最小项表达式的形式，从原理上讲，应用对照比较的方法，用数据选择器可以不受限制地实现任何组合逻辑函数。具体步骤如下：

（1）根据逻辑函数中变量的个数确定数据选择器的类型。若变量数为 n，则一般应选择 2^n 选 1 数据选择器或 2^{n-1} 选 1 数据选择器。

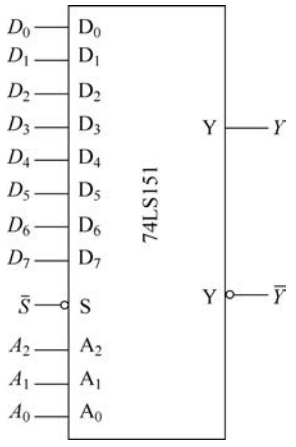


图 3-51　74LS151 的简易图形符号

（2）确定地址输入。如果选择 2^n 选 1 数据选择器，则 n 个变量全部设成地址输入；如果选择 2^{n-1} 选 1 数据选择器，则任选 $n-1$ 个变量设成地址输入。

（3）确定数据输入。对比逻辑函数最小项表达式和数据选择器的输出表达式来确定数据输入。

【例 3-18】 用数据选择器实现二进制全加器。

解：二进制全加器有三个输入变量，两个输出变量，可以选择双 4 选 1 数据选择器 74LS153 来实现。

设 A 和 B 为二进制全加器的加数和被加数，C 为低位来的进位，S 为本位的和，CO 为向高位的进位。设地址输入 $A_1=A$，$A_0=B$。

式(3-36)已经给出了 S 和 CO 的逻辑表达式，这里重写出来，并稍作变换得

$$\begin{aligned}S&=\sum(1,2,4,7)\\&=\overline{A}\,\overline{B}\cdot C+\overline{A}B\cdot\overline{C}+A\overline{B}\cdot\overline{C}+AB\cdot C\\CO&=\sum(3,5,6,7)\\&=\overline{A}BC+A\overline{B}C+AB\overline{C}+ABC\\&=\overline{A}\,\overline{B}\cdot\mathbf{0}+\overline{A}B\cdot C+A\overline{B}\cdot C+AB\cdot\mathbf{1}\end{aligned}\tag{3-41}$$

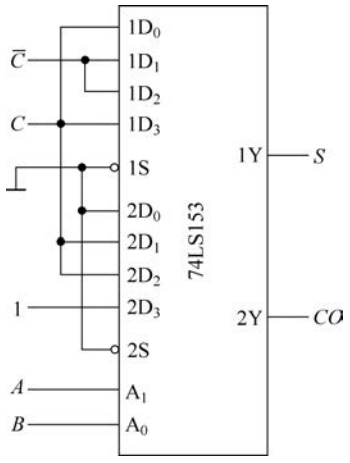


图 3-52　例 3-18 的电路连接图

比较式(3-41)与 4 选 1 数据选择器 74LS153 的表达式(3-38),可以确定数据输入及输出为

$$1D_0=C、1D_1=\overline{C}、1D_2=\overline{C}、1D_3=C、S=1Y$$

$$2D_0=\mathbf{0}、2D_1=C、2D_2=C、2D_3=\mathbf{1}、CO=2Y$$

依据各数据输入和输出画出连接图,如图 3-52 所示。

【例 3-19】 用数据选择器实现逻辑函数 $Y(A,B,C)=\overline{A}BC+A\overline{B}C+AB$。

解:根据逻辑函数的变量数,选择 8 选 1 数据选择器 74LS151 来实现。令地址输入 $A_2=C$,$A_1=B$,$A_0=A$。在 $\overline{S}=\mathbf{0}$ 时,74LS151 的输出逻辑表达式为

$$\begin{aligned}Y=&\overline{A}_2\overline{A}_1\overline{A}_0D_0+\overline{A}_2\overline{A}_1A_0D_1+\overline{A}_2A_1\overline{A}_0D_2\\&+\overline{A}_2A_1A_0D_3+A_2\overline{A}_1\overline{A}_0D_4+A_2\overline{A}_1A_0D_5\\&+A_2A_1\overline{A}_0D_6+A_2A_1A_0D_7\end{aligned}\tag{3-42}$$

把待实现的逻辑函数变换成与此表达式相同的形式

$$\begin{aligned}Y(A,B,C)&=\overline{A}BC+A\overline{B}C+AB\\&=CB\overline{A}+C\overline{B}A+\overline{C}BA+CBA\\&=\mathbf{0}(\overline{C}\overline{B}\overline{A})+\mathbf{0}(\overline{C}\overline{B}A)+\mathbf{0}(\overline{C}B\overline{A})+\mathbf{1}(\overline{C}BA)+\\&\quad\mathbf{0}(C\overline{B}\overline{A})+\mathbf{1}(C\overline{B}A)+\mathbf{1}(CB\overline{A})+\mathbf{1}(CBA)\end{aligned}\tag{3-43}$$

比较式(3-42)和式(3-43),可以确定数据输入为

$$D_0=D_1=D_2=D_4=\mathbf{0},\quad D_3=D_5=D_6=D_7=\mathbf{1}$$

输出可以从同相输出端 Y 输出,也可以从反相输出端 $\overline{Y}$ 加反相器以后输出。电路的连接方法如图 3-53 所示。

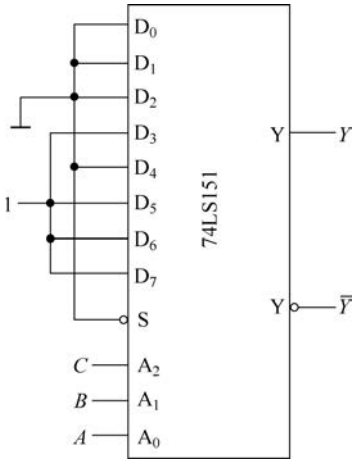


图 3-53　例 3-19 的电路连接图

【提示】 数据选择器又称为多路开关,多用在需要有选择、分时地传送数据的场合。利用选通端可以扩展数据选择器的功能。

3.5.5　数据分配器

根据 m 个地址输入,将一个输入信号传送到 2^m 个输出端中的某一个的器件称为数据分配器。数据分配器示意图如图 3-54 所示。下面以 1 路-4 路数据分配器为例,说明数据分配器的工作原理。

1 路-4 路数据分配器有 1 个信号输入端 D,2 个地址输入端 A_1、A_0,4 个数据输出端 Y_3、Y_2、Y_1、Y_0,如图 3-55 所示。

根据数据分配器的定义及图 3-55,可以列出 1 路-4 路数据分配器的真值表,如表 3-25 所示。

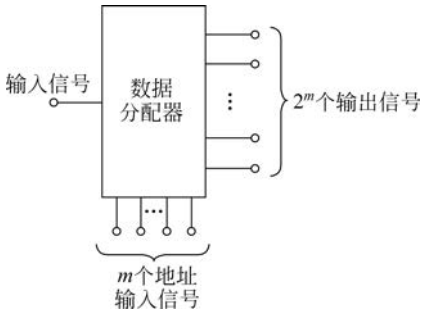


图 3-54　数据分配器示意图

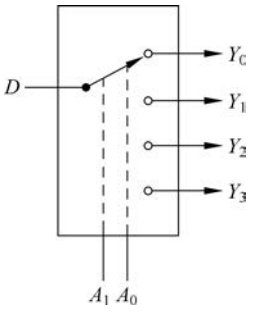


图 3-55　1 路-4 路数据分配器功能示意图

表 3-25　1 路-4 路数据分配器的真值表

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

根据表 3-25 可以写出输出逻辑表达式

$$\begin{aligned} Y_0 &= D\overline{A}_1\overline{A}_0 \\ Y_1 &= D\overline{A}_1 A_0 \\ Y_2 &= DA_1\overline{A}_0 \\ Y_3 &= DA_1 A_0 \end{aligned} \tag{3-44}$$

根据式(3-44)可以画出 1 路-4 路数据分配器的逻辑图，如图 3-56 所示。

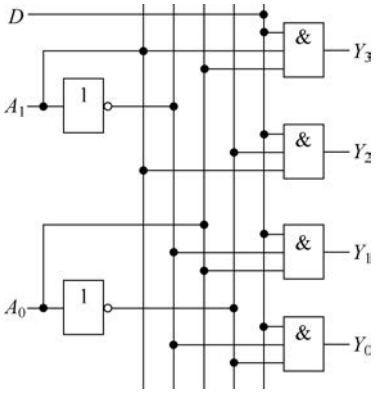


图 3-56　1 路-4 路数据分配器的逻辑图

从图 3-56 可以看出，如果将地址输入 A_1、A_0 作为二进制编码输入，D 作为选通控制信号，则数据分配器就成为二进制译码器了。所以数据分配器完全可以用二进制译码器来代替。

由于数据分配器可以用二进制译码器代替，所以集成二进制译码器也是集成数据分配器。如集成 2 线-4 线二进制译码器 74LS139 也是集成 1 路-4 路数据分配器；集成 3 线-8 线二进制译码器 74LS138 也是集成 1 路-8 路数据分配器。

【提示】 数据分配器多用在需要数据分时传送的场合。

3.5.6　数值比较器

微课视频

数字电路中，用于比较两个二进制数 A 和 B 数值大小的逻辑电路称为数值比较器。下面首先讨论 1 位数值比较器，然后探讨多位数值比较器。

1. 1 位数值比较器

当两个一位二进制数 A 和 B 比较时，其结果有 3 种情况，即 $A<B$、$A=B$、$A>B$，比较结果分别用 M、G 和 L 表示。设 $A<B$ 时，$M=\mathbf{1}$；$A=B$ 时，$G=\mathbf{1}$；$A>B$ 时，$L=\mathbf{1}$，由此可得 1 位数值比较器的真值表，如表 3-26 所示。

根据表 3-26 可以写出逻辑函数表达式

$$\begin{cases} M=\overline{A}B \\ G=\overline{A}\,\overline{B}+AB=\overline{\overline{A}B+A\overline{B}} \\ L=A\overline{B} \end{cases} \tag{3-45}$$

根据式(3-45)可以画出 1 位数值比较器的逻辑图,如图 3-57 所示。

表 3-26　1 位数值比较器的真值表

A	*B*	*M*	*G*	*L*
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

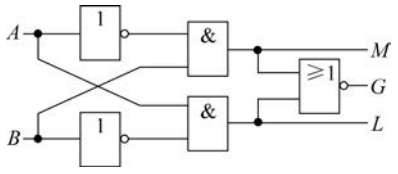


图 3-57　1 位数值比较器的逻辑图

2. 多位数值比较器

如果比较两个多位二进制数,必须逐位比较,使用多位数值比较器。下面以 4 位数值比较器为例说明其工作原理。

设两个 4 位二进制数为 $A=A_3A_2A_1A_0$,$B=B_3B_2B_1B_0$,因此 4 位数值比较器有 8 个数值输入信号。同样,A 与 B 的比较有三种结果:大于、等于、小于,对应的 3 个输出信号分别为 $Y_{A>B}$、$Y_{A=B}$ 和 $Y_{A<B}$。

(1) 如果 $A>B$,则必须使 $A_3>B_3$;或者 $A_3=B_3$ 且 $A_2>B_2$;或者 $A_3=B_3$,$A_2=B_2$ 且 $A_1>B_1$;或者 $A_3=B_3$,$A_2=B_2$,$A_1=B_1$ 且 $A_0>B_0$。

设 A,B 的第 i 位($i=0,1,2,3$)二进制数比较结果的大于、等于、小于用 L_i、G_i、M_i 表示,则

$$Y_{A>B}=L_3+G_3L_2+G_3G_2L_1+G_3G_2G_1L_0 \tag{3-46}$$

(2) 如果 $A=B$,则必须使 $A_3=B_3$,$A_2=B_2$,$A_1=B_1$ 且 $A_0=B_0$,所以

$$Y_{A=B}=G_3G_2G_1G_0 \tag{3-47}$$

(3) 如果 $A<B$,则必须使 $A_3<B_3$;或者 $A_3=B_3$ 且 $A_2<B_2$;或者 $A_3=B_3$,$A_2=B_2$ 且 $A_1<B_1$;或者 $A_3=B_3$,$A_2=B_2$,$A_1=B_1$ 且 $A_0<B_0$,则

$$Y_{A<B}=M_3+G_3M_2+G_3G_2M_1+G_3G_2G_1M_0 \tag{3-48}$$

另外,也可以由排除法推导出:如果 A 不大于且不等于 B,则 $A<B$,由此得出 $Y_{A<B}$ 的表达式为

$$Y_{A<B}=\overline{Y}_{A>B}\cdot\overline{Y}_{A=B}=\overline{Y_{A>B}+Y_{A=B}} \tag{3-49}$$

由式(3-45)可得 L_i、G_i、M_i 的表达式

$$\begin{cases} L_i=A_i\overline{B_i} \\ G_i=\overline{A_iB_i}+A_iB_i=\overline{\overline{A_i}B_i+A_i\overline{B_i}} \\ M_i=\overline{A_i}B_i \end{cases} \tag{3-50}$$

根据式(3-46)、式(3-47)、式(3-49)和图 3-57 可以画出 4 位数值比较器的逻辑图,如图 3-58 所示。图中 1 位数值比较器是按照式(3-50)得出的,与图 3-57 相同。

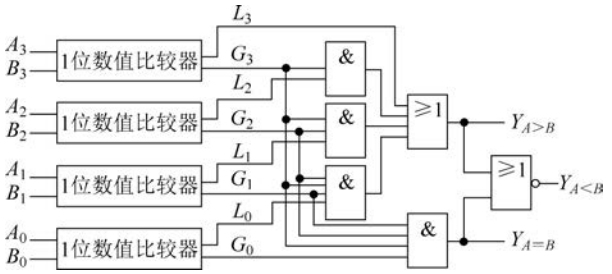


图 3-58　4 位数值比较器的逻辑图

集成 4 位数值比较器的典型电路是 74LS85，其简易图形符号如图 3-59 所示。$I_{A>B}$、$I_{A<B}$、$I_{A=B}$ 是扩展端，是低位来的比较结果，用于多个芯片之间扩展连接时使用。只比较两个 4 位二进制数时，将 $I_{A>B}$、$I_{A<B}$ 接低电平，同时将 $I_{A=B}$ 接高电平。74LS85 的功能如表 3-27 所示。

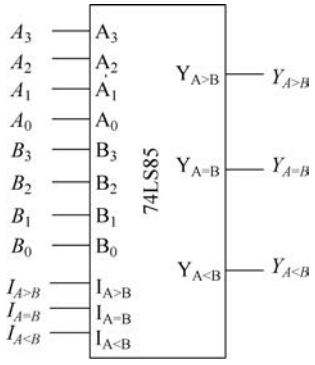


图 3-59　74LS85 的简易图形符号

表 3-27　74LS85 的功能表

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

*3.6　利用 Multisim 分析组合逻辑电路

本节用 Multisim 13.0 分别分析由小规模集成门电路构成的组合逻辑电路和中规模组合逻辑电路。

3.6.1　小规模门电路构成的组合逻辑电路的仿真分析

在 Multisim 13.0 中构建如图 3-60 所示的组合逻辑电路。从元件工具栏的 TTL 器件库中找出“与非”门 74LS00N 和“与非”门 74LS10N，也可以使用快捷键 Ctrl＋W，在弹出的对话框中 Group 栏选择 TTL 找出相应器件；逻辑转换仪 XLC1 从虚拟仪器工具栏中找出。

双击逻辑转换仪 XLC1 的图标打开面板图，如图 3-61 所示。单击面板图上的“由电路转换为真值表”按钮，在面板图的真值表区弹出所分析电路的真值表。再单击面板图上的“由真值表转换为最简逻辑函数表达式”按钮，在面板图底部的逻辑函数表达式栏，显示最简逻辑表达式“$Y=AC+AB+BC$”。

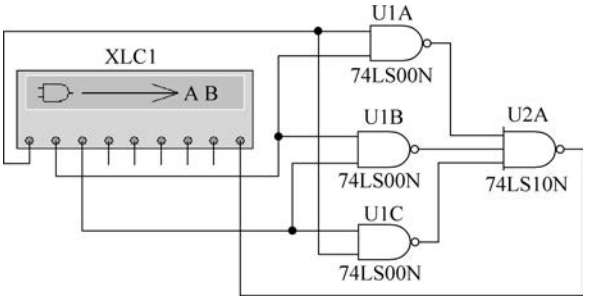


图 3-60 门电路构成的组合逻辑电路

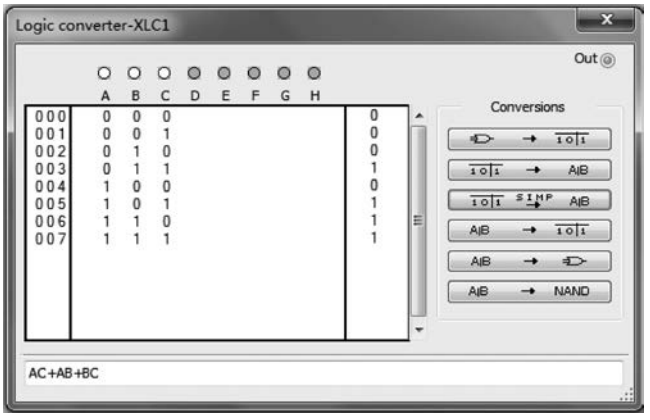


图 3-61 组合逻辑电路的分析

分析真值表可知,当 A、B、C 输入变量取值中 **1** 的个数占多数时输出 Y 为 **1**,否则输出 Y 为 **0**,电路实现三人多数表决器的功能。

3.6.2 中规模组合逻辑电路的仿真分析

1. 二进制译码器 74LS138 的仿真分析

从 Multisim 13.0 元件工具栏的 TTL 器件库中找出 74LS138N,连接设置 74LS138N 的使能端,在虚拟仪器工具栏中调用字信号发生器(Word Generator)XWG1 和逻辑分析仪(Logic Analyzer)XLA1,组成译码器的仿真电路,如图 3-62 所示。

在图 3-62 中单击字信号发生器图标 XWG1,得到 Word generator-XWG1 对话框,在 Controls 选项组中单击 Cycle 按钮,在 Display 选项组中选中 Dec(十进制)复选框,在字信号编辑区写 0、1、2、3、4、5、6、7。单击 Set 按钮,弹出 Setting(设置)对话框,将 Buffer Size 的值设置为 8。

单击“运行”按钮,双击逻辑分析仪 XLA1 的图标,显示运行结果如图 3-63 所示。

从仿真分析结果中看出,当 74LS138N 的输入代码分别为 **000～111** 时,对应的输出端依次输出低电平。该仿真结果符合二进制译码器 74LS138 的逻辑功能。

2. 集成 BCD 码七段显示译码器 74LS248 仿真分析

在 Multisim 13.0 中构建集成 BCD 码七段显示译码器逻辑功能仿真电路,如图 3-64 所示。从元件工具栏的 TTL 器件库中找出显示译码器 74LS248N;从元件工具栏的基本元件库中找出电阻及单刀双掷开关;从元件工具栏的电源/信号源库中找出电压源及接地端;从元件工具栏的指示元件库中找出共阴极 LED 数码显示器;也可以使用快捷键 Ctrl+W 调出选用元件对话框,再找出相应的元件。

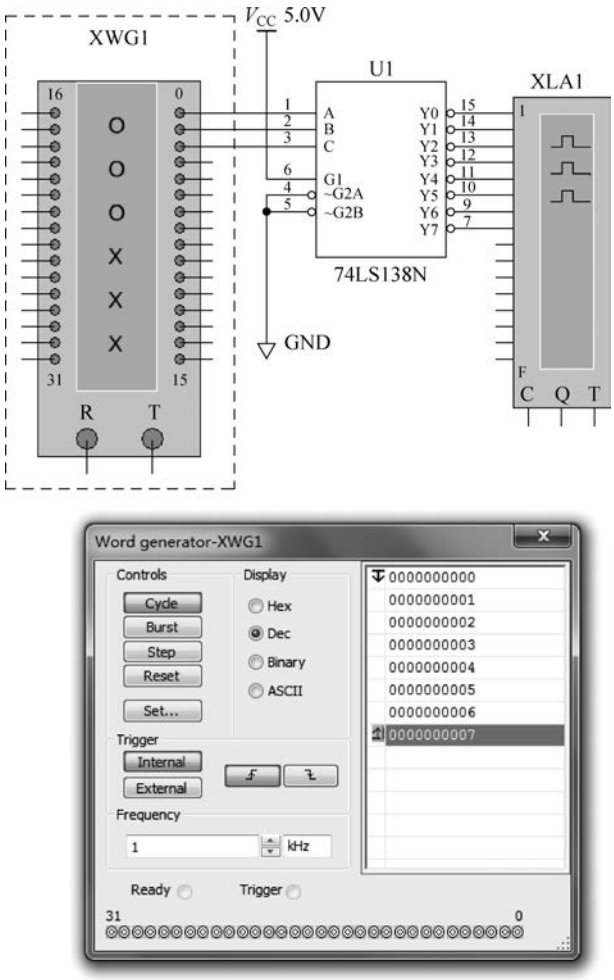


图 3-62　74LS138 测试电路及字函数发生器的设置

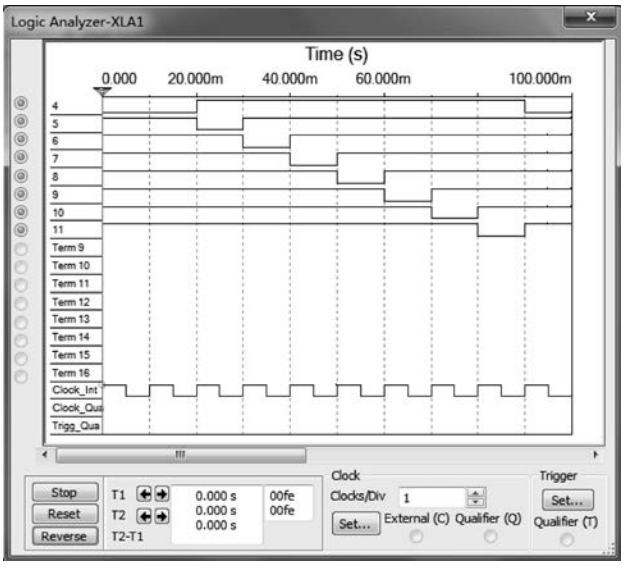


图 3-63　译码器 74LS138 电路的仿真波形

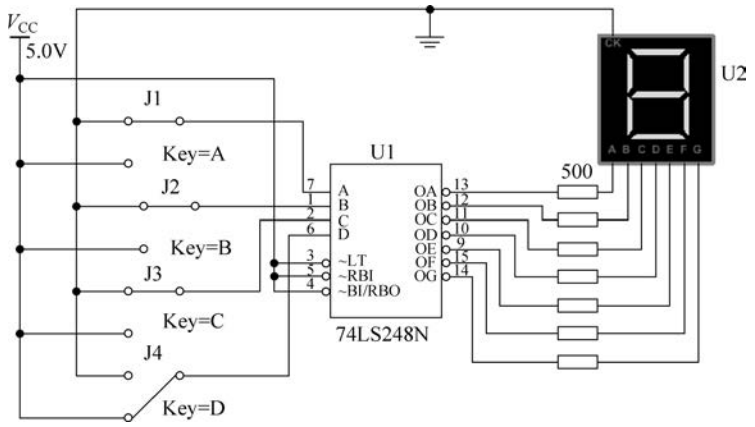


图 3-64　集成 BCD 码七段显示译码器 74LS248 的测试电路

单击仿真开关后，通过键盘上的开关 A、B、C、D 键改变开关的状态，当输入信号的输入组合为 **0000～1001** 时，数码显示器所显示的十进制数字分别是 0～9。图中显示的是输入 8421BCD 码为 **1000**，显示字符为 8 的情况。可见，电路的测试结果符合显示驱动译码器 74LS248 的特性。

【提示】 74LS248N 代码输入变量由高位到低位的排列顺序为 D、C、B、A。

*3.7　利用 VHDL 设计组合逻辑电路

本节用 VHDL 对编码器、译码器和数据选择器进行设计和仿真。

1. 8 线-3 线优先编码器的设计及仿真

设 D(7)～D(0)为输入信号，D(7)的优先级最高，D(0)的优先级最低；A(0)～A(2)为输出编码。源代码为

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity encoder is
port( D:in std_logic_vector(0 to 7);
      A:out std_logic_vector(0 to 2) );
end ;
architecture xiani of encoder is
begin
      process(D)
         begin
           if (D(7) = '1')then A<= "111";
             elsif (D(6) = '1')then A<= "110";
             elsif (D(5) = '1')then A<= "101";
             elsif (D(4) = '1')then A<= "100";
             elsif (D(3) = '1')then A<= "011";
             elsif (D(2) = '1')then A<= "010";
             elsif (D(1) = '1')then A<= "001";
             elsif (D(0) = '1')then A<= "000";
             else A<= "zzz";
```

```
            end if;
        end process;
end;
```

对源代码进行仿真，仿真结果如图 3-65 所示。

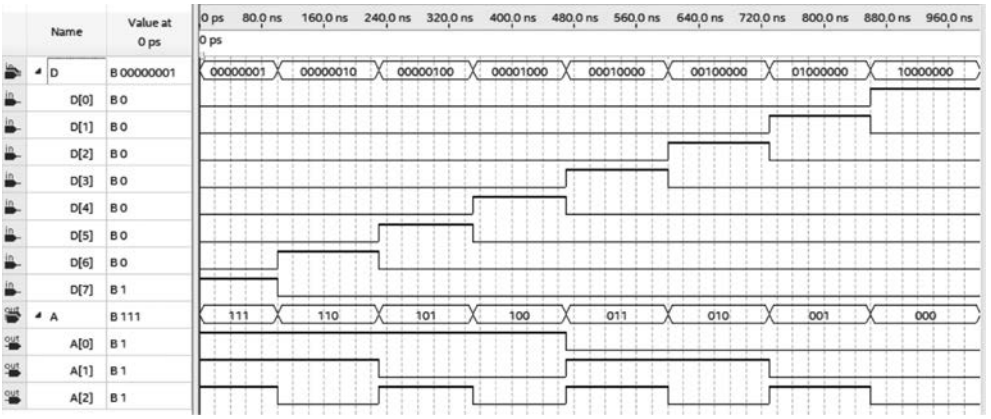


图 3-65　8 线-3 线编码器的仿真图

2. 3 线-8 线译码器的设计及仿真

设 a2、a1、a0 为编码输入，STa、STb、STc 为控制信号，Y(0)～Y(7)为输出信号。当 STa＝1 且 STb＝STc＝0 时，译码器进行译码，否则译码器处于禁止译码状态。源代码为

```
library ieee;
use ieee.std_logic_1164.all;
entity decoder3 is
    port(a0,a1,a2,STa,STb,STc:in std_logic;
        Y:out std_logic_vector (0 to 7));
end decoder3 ;
architecture rtl of decoder3 is
signal indata :STD_logic_vector (2 downto 0);
begin
    indata <= a2 & a1 & a0;
    process (indata,STa,STb,STc)
        begin
        if (STa = '1' and STb = '0' and STc = '0') then
            case indata is
                when "000" => Y <= "01111111";
                when "001" => Y <= "10111111";
                when "010" => Y <= "11011111";
                when "011" => Y <= "11101111";
                when "100" => Y <= "11110111";
                when "101" => Y <= "11111011";
                when "110" => Y <= "11111101";
                when "111" => Y <= "11111110";
                when others => null;
            end case;
        else
            Y <= "11111111";
```

```
        end if;
    end process;
end rtl;
```

对源代码进行仿真,仿真结果如图 3-66 所示。

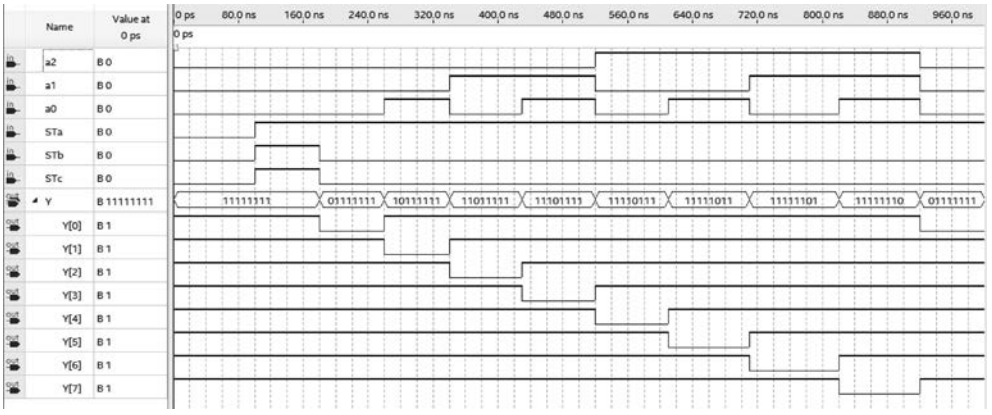


图 3-66　3 线-8 线译码器的仿真图

3.4 选 1 数据选择器的设计及仿真

设数据选择器的地址输入为 a2、a1,数据输入为 d3、d2、d1、d0,s 为使能端,高电平有效,y 为输出端。源代码为

```
library ieee;
use ieee.std_logic_1164.all;
entity my4s1 is
port ( d0,d1,d2,d3,s,a1,a2: in std_logic;
                            y: out std_logic);
end my4s1;
architecture Behavioral of my4s1 is
  signal a:std_logic_vector(1 downto 0);
  signal y1:std_logic;
    begin
      process(s,y1)
        begin
          if(s = '1')then
            y <= y1;
             else
            y <= '0';
          end if;
      end process;
        a <= a2&a1;
        y1 <= d0 when a = "00" else
        d1 when a = "01" else
        d2 when a = "10" else
        d3;
end Behavioral;
```

对源代码进行仿真,仿真结果如图 3-67 所示。

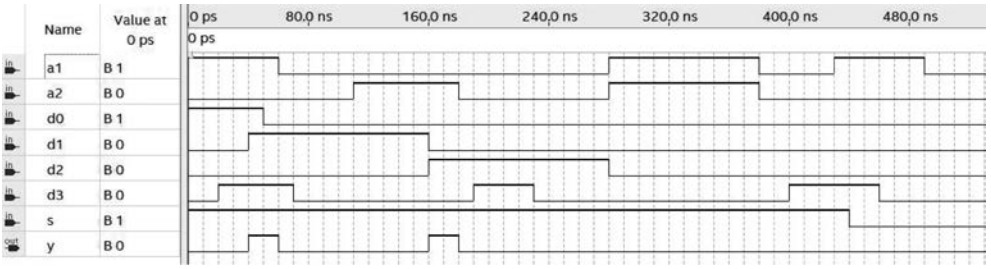


图 3-67　4 选 1 数据选择器的仿真图

本章小结

本章介绍了组合逻辑电路的特点、分析和设计方法，组合逻辑电路的竞争冒险现象以及数字系统中常用组合逻辑电路的原理及应用。本章主要讲述了如下内容。

(1) 组合逻辑电路任何时刻的输出仅取决于该时刻的各种输入变量的状态组合，而与电路过去的状态无关。在电路结构上只包含门电路，没有存储(记忆)单元。

(2) 分析组合逻辑电路的目的是确定已知电路的逻辑功能，可通过写逻辑表达式、列真值表等手段来完成。对于表达式较简单的电路，可以直接通过表达式得知电路的逻辑功能；对于表达式较复杂的电路，要借助于真值表来归纳电路的逻辑功能。

(3) 利用小规模集成电路(门电路)设计组合逻辑电路，常以电路简单、所用器件个数以及种类最少为设计原则。设计过程包括逻辑抽象、逻辑化简、逻辑变换、画出逻辑图。逻辑抽象的方法要视逻辑函数中逻辑变量的多少而定，对于具有较少逻辑变量的逻辑函数，采用列真值表来抽象逻辑表达式；当变量数较多时，可采用列简化真值表的方法抽象逻辑表达式。

(4) 竞争冒险是组合逻辑电路工作状态转换过程中经常会出现的一种现象。如果负载是一些对尖峰脉冲不敏感(例如光电显示器)的器件，就不必考虑冒险问题。

(5) 常用组合逻辑电路有编码器、译码器、加法器、数据选择器和数据分配器、数值比较器等。为使用方便，它们常被做成中规模集成电路组件，利用这些组合逻辑电路可以实现组合逻辑函数。使用中规模集成器件可以大大简化组合逻辑电路的设计。

习题

1. 填空题

(1) 组合电路逻辑功能上的特点是，任意时刻的_____状态仅取决于该时刻_____的状态，而与以前时刻的_____无关。

(2) 组合逻辑电路的功能描述方法主要有_____、_____、_____、_____ 等。

(3) 将文字描述的逻辑命题转换成逻辑表达式的过程称为_____。

(4) 不考虑来自低位的进位而将两个一位二进制数相加，称为_____；不仅考虑两个一位二进制数相加，而且考虑来自低位进位的加法运算称为_____。

(5) 由于竞争而在输出端可能出现违背稳态下逻辑关系的尖峰脉冲现象叫作_____。

(6) 消除或减弱组合逻辑电路中的竞争冒险,常用的方法是发现并消掉互补变量,增加_____,并在输出端并联_____。

(7) 组合电路由门电路组成,不包含任何_____,没有_____能力。

(8) 常见的中规模组合逻辑器件有_____。

2. 选择题

(1) 组合逻辑电路是由_____组成的。

A. 触发器　　B. 门电路　　C. 二极管　　D. 场效应管

(2) 16 选 1 路数据选择器的地址输入端有_____个。

A. 16　　B. 4　　C. 32　　D. 2

(3) 下列逻辑电路中不是组合电路的是_____。

A. 编码器　　B. 数据选择器　　C. 计数器　　D. 全加器

(4) 用四位数值比较器比较两个四位二进制数时,应先比较_____位。

A. 最低　　B. 次低　　C. 最高　　D. 次高

(5) 逻辑函数 $Y=\overline{A}C+AB+\overline{B}\overline{C}$,当变量取值为_____时,将出现冒险现象。

A. $B=C=\mathbf{1}$　　B. $B=C=\mathbf{0}$　　C. $A=\mathbf{1},C=\mathbf{0}$　　D. $A=B=\mathbf{0}$

(6) 16 位输入的二进制编码器,其输出端有_____位。

A. 256　　B. 128　　C. 4　　D. 3

(7) 可以用作数据分配器的是_____。

A. 编码器　　B. 译码器　　C. 数据选择器　　D. 数值比较器

(8) 将两片 8 线-3 线编码器进行级联,可以构成_____编码器。

A. 8 线-3 线　　B. 16 线-4 线　　C. 16 线-3 线　　D. 8 线-4 线

3. 分析图 3-68 所示的电路,写出逻辑函数表达式,列出真值表,指出电路完成的逻辑功能。

4. 分析图 3-69 所示的电路,写出 Y_1、Y_2 的逻辑函数表达式,列出真值表,指出电路完成的逻辑功能。

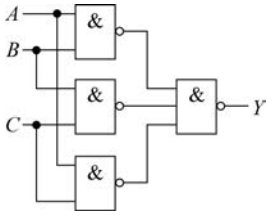


图 3-68　题 3 电路图

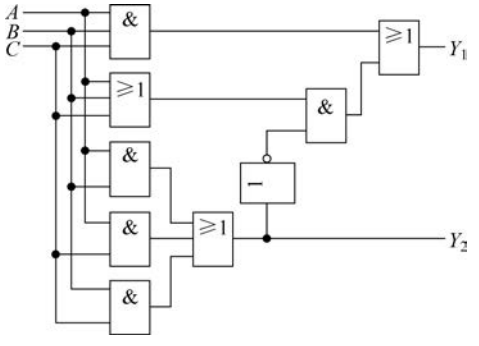


图 3-69　题 4 电路图

5. 已知图 3-70 所示电路的输入、输出都是 8421 码,写出输出逻辑函数表达式,列出真值表,指出电路完成的逻辑功能。

6. 已知逻辑电路如图 3-71 所示,试分析其逻辑功能。

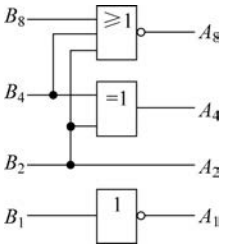


图 3-70　题 5 电路图

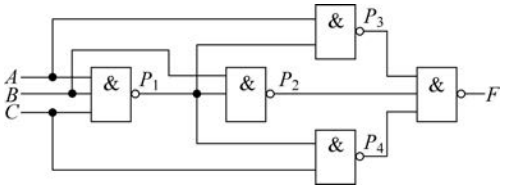


图 3-71　题 6 电路图

7. 用“与非”门设计四变量的多数表决器。当输入变量 A、B、C、D 有 3 个或 3 个以上为 **1** 时输出为 **1**，输入为其他状态时输出为 **0**。

8. 某雷达站有 3 部雷达 A、B 和 C，其中 A 和 B 功率消耗相等，C 的消耗功率是 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 的功率消耗，发电机 Y 的最大输出功率是雷达 A 和 C 的功率消耗总和。要求设计一个组合逻辑电路，能够根据各雷达的启动、关闭信号，以最省电的方式开、停发电机。

9. 设计一个能被 2 或 3 整除的逻辑电路，其中被除数 A、B、C、D 是 8421BCD 编码。规定能整除时，输出为高电平，否则，输出为低电平。要求用最少的“与非”门实现。（设 **0** 能被任何数整除。）

10. 试用卡诺图法判断逻辑函数式

$$Y(A,B,C,D)=\sum m(0,1,4,5,12,13,14,15)$$

是否存在竞争冒险？若有，则采用增加冗余项的方法消除。

11. 用一个 8 线-3 线优先编码器 74LS148 和一个 3 线-8 线译码器 74LS138 实现 3 位格雷码到 3 位二进制码的转换。

12. 图 3-72 是由双 4 选 1 数据选择器 74LS153 和门电路组成的组合电路。试分析输出 Z 与输入 X_3、X_2、X_1、X_0 之间的逻辑关系。

13. 图 3-73 所示是用两个 4 选 1 数据选择器组成的逻辑电路，试写出输出 Z 与输入 M、N、P、Q 之间的逻辑函数式。

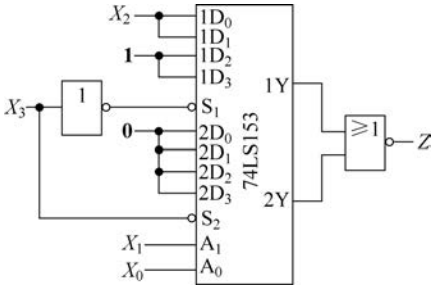


图 3-72　题 12 电路图

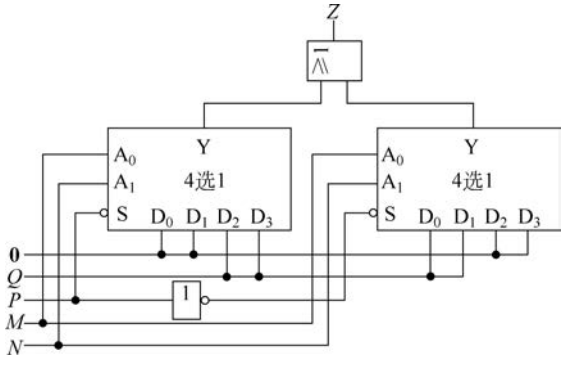


图 3-73　题 13 电路图

14. 图 3-74 所示电路是 3 线-8 线译码器 74LS138 和 8 选 1 数据选择器 74LS151 组成的电路，试分析电路的逻辑功能。

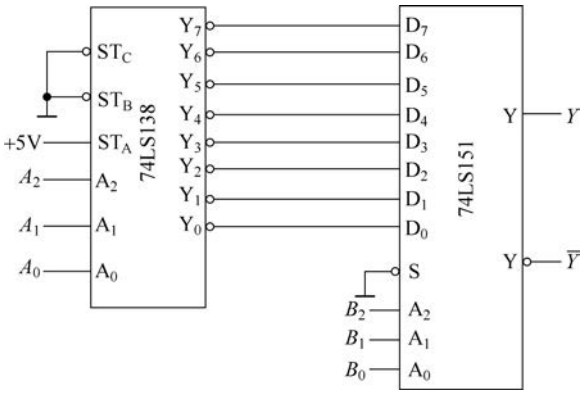


图 3-74　题 14 电路图

15. 图 3-75 所示电路中，$A_3 \sim A_0$、$B_3 \sim B_0$ 是两个 4 位二进制数，M 是控制信号。分析电路的逻辑功能。

16. 分析图 3-76 所示电路，写出输出函数 Y 的表达式，并指出电路的逻辑功能。

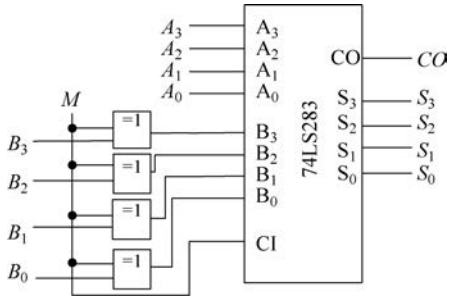


图 3-75　题 15 电路图

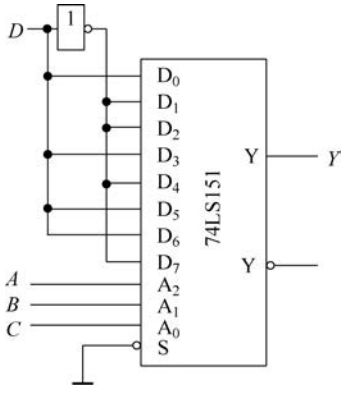


图 3-76　题 16 电路图

17. 图 3-77 电路中，74LS283 是 4 位二进制加法器，74LS85 是 4 位二进制数比较器，74LS47 是显示驱动译码器。加法器的输入 $A_3 \sim A_0$、$B_3 \sim B_0$ 为 8421 码。分析电路，解答下列问题。

(1) 当加法器的输入最大时，求输出 $S_3 \sim S_0$。

(2) $S_3 \sim S_0$ 为何种取值时，$Y_{A=B}=1$?

(3) 数码管应为共阴极数码管还是共阳极数码管?

(4) 数码管显示的十进制数范围是什么?

(5) 该电路实现何种功能?

18. 电路如图 3-78 所示，A_1B_1、A_0B_0 是两位二进制数。

(1) 分别写出 P 和 H 关于 A_0 和 B_0 的表达式和真值表，并指出由 74LS153 构成的电路的逻辑功能；

(2) 分别写出 M 和 N 关于 A_1、B_1 和 P 的表达式和真值表，并指出由 74LS138 构成的电路的逻辑功能；

(3) 指出整个电路的逻辑功能。

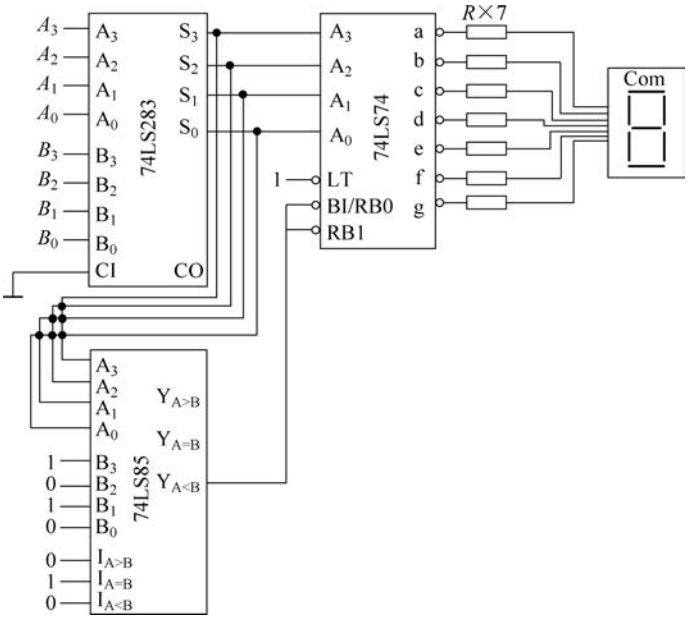


图 3-77　题 17 电路图

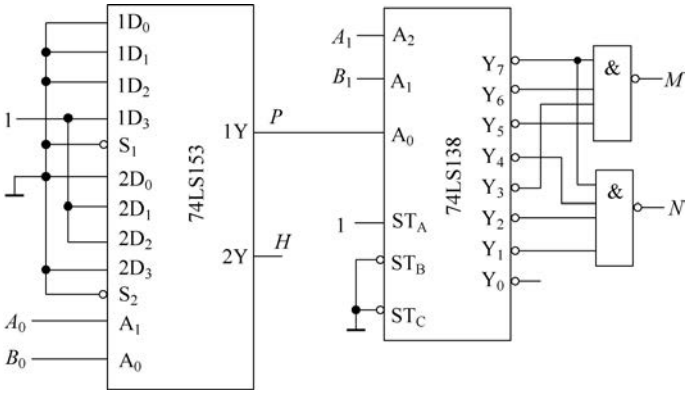


图 3-78　题 18 电路图

19. 设计一个 8421 码到余 3 码的转换电路。

(1) 写出真值表和表达式；

(2) 用基本逻辑门实现；

(3) 用加法器 74LS283 实现。

20. 分别用下列方法设计全加器和全减器。

(1) 用基本逻辑门；

(2) 用半加器和门电路；

(3) 用译码器 74LS138，可以附加必要的门电路；

(4) 用双 4 选 1 数据选择器 74LS153，可以附加必要的门电路。

21. 设计一个火灾监测报警系统。当温度传感器、烟雾传感器和红外传感器这 3 个传感器中有两个或两个以上探测出超限信号时，系统才发出火灾报警信号。

(1) 写出真值表和表达式；

(2) 用与非门实现,要求电路尽量简单;

(3) 用 74LS138 译码器实现,可以附加必要的门电路;

(4) 用 8 选 1 数据选择器 74LS151 实现。

22. 设计一个用 4 个开关控制一个灯的逻辑电路。当总控制开关 S 闭合时,改变 A、B、C 任何一个开关的状态都能控制灯由亮变灭或者由灭变亮。当总控制开关 S 断开时,灯始终处于熄灭状态。

(1) 写出真值表和表达式;

(2) 用门电路实现,要求电路尽量简单;

(3) 用 74LS138 译码器实现,可以附加必要的门电路;

(4) 用 8 选 1 数据选择器 74LS151 实现,可以附加必要的门电路。