Lecture 06 常用组合逻辑电路*OK*

一、加法器

（一）加法器工作原理及实现

1、1 位全加器

由真值表可得：

$$F_n = A_n \oplus B_n \oplus C_{n-1}$$

$$C_n = (A_n + B_n)C_{n-1} + A_n B_n$$

表4.4.1 1位全加器真值表

A_n	B_n	C_{n-1}	F_n	C_n
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

$$C_n = (A_n + B_n)C_{n-1} + A_nB_n = \overline{\overline{(A_n \oplus B_n)C_{n-1}} \cdot \overline{A_nB_n}}$$

根据上式，1 位全加器可以采用**异或门**和**与非门**实现如下图：

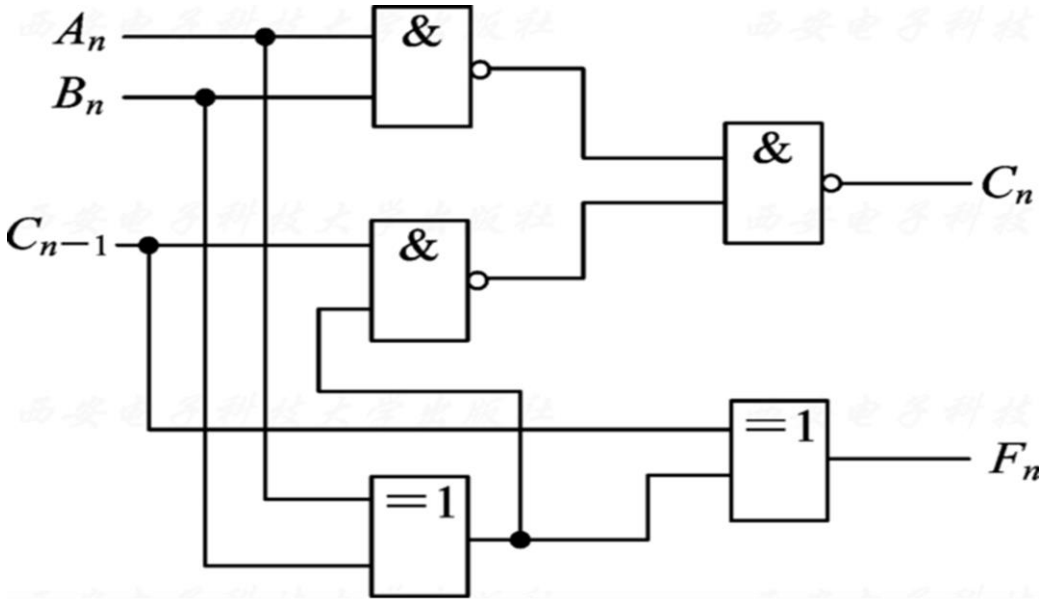


图4.4.2 1位全加器电路图

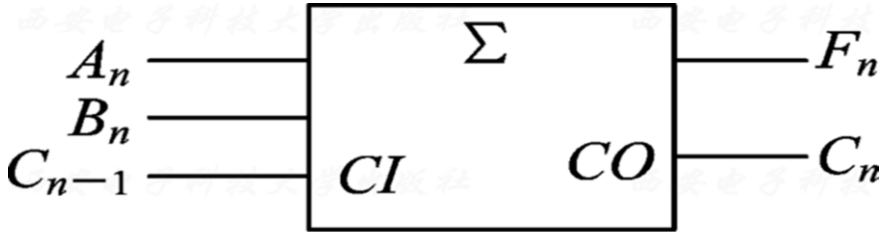


图4.4.1 全加器的逻辑符号

2、4 位全加器

4 位全加器可以用四个 1 位全加器以**串行进位**的方式连接组成：

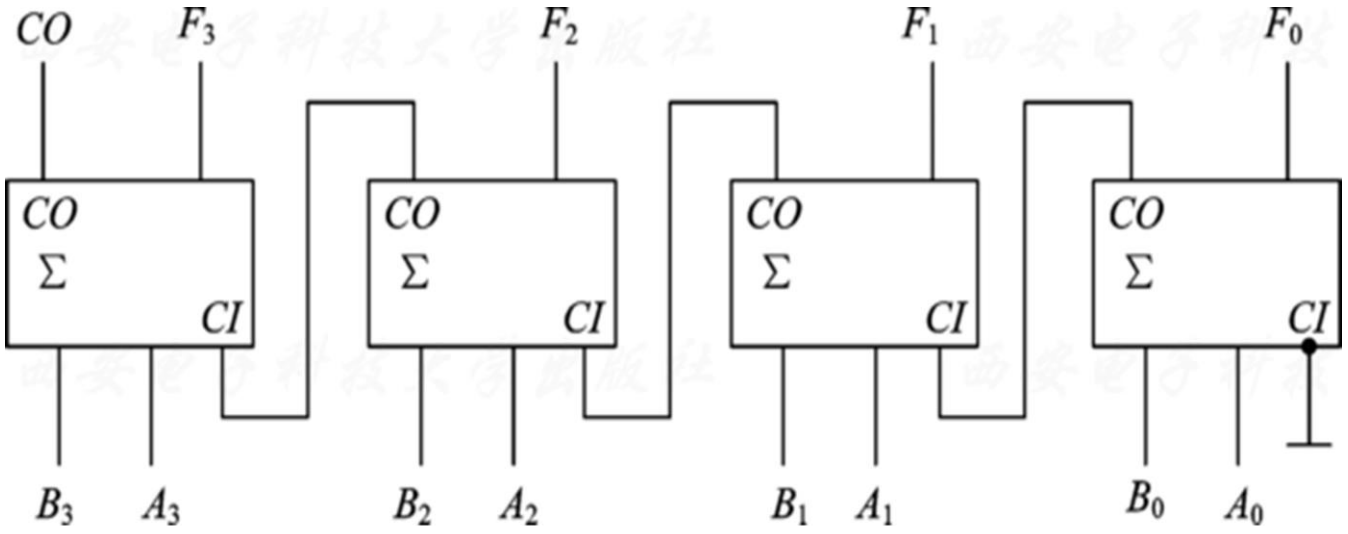


图4.4.3 4位串行进位加法器

特点：结构简单，速度慢

改进：定义从低位到高位四个 1 位全加器的进位输入信号依次是 C_{-1}、C_0、C_1 及 C_2，对应的进位输出依次是 C_0、C_1、C_2 和 C_3，由

$$C_n = (A_n + B_n)C_{n-1} + A_n B_n$$

可以得到各进位信号的逻辑表达式：

$$\begin{cases} C_0 = (A_0 + B_0)C_{-1} + A_0B_0 \\ C_1 = (A_1 + B_1)C_0 + A_1B_1 \\ C_2 = (A_2 + B_2)C_1 + A_2B_2 \\ C_3 = (A_3 + B_3)C_2 + A_3B_3 \end{cases}$$

设：$P_i = A_i + B_i$（进位传播函数），$G_i = A_iB_i$（进位生成函数），则有：

$$
\begin{cases}
C_{-1} = 0 \\
C_0 = P_0 C_{-1} + G_0 \\
C_1 = P_1 C_0 + G_1 = P_1 P_0 C_{-1} + P_1 G_0 + G_1 \\
C_2 = P_2 C_1 + G_2 = P_2 P_1 P_0 C_{-1} + P_2 P_1 G_0 + P_2 G_1 + G_2 \\
C_3 = P_3 C_2 + G_3 = P_3 P_2 P_1 P_0 C_{-1} + P_3 P_2 P_1 G_0 + P_3 P_2 G_1 + P_3 G_2 + G_3
\end{cases}
$$

每一位的进位信号 C_i 可以表示为加数、被加数和最低位进位信号 C_{-1} 的逻辑函数，因此每个进位信号 $C_0 \sim C_3$ 都可以据此采用与或门并行产生，

由此得到**超前进位**加法器：

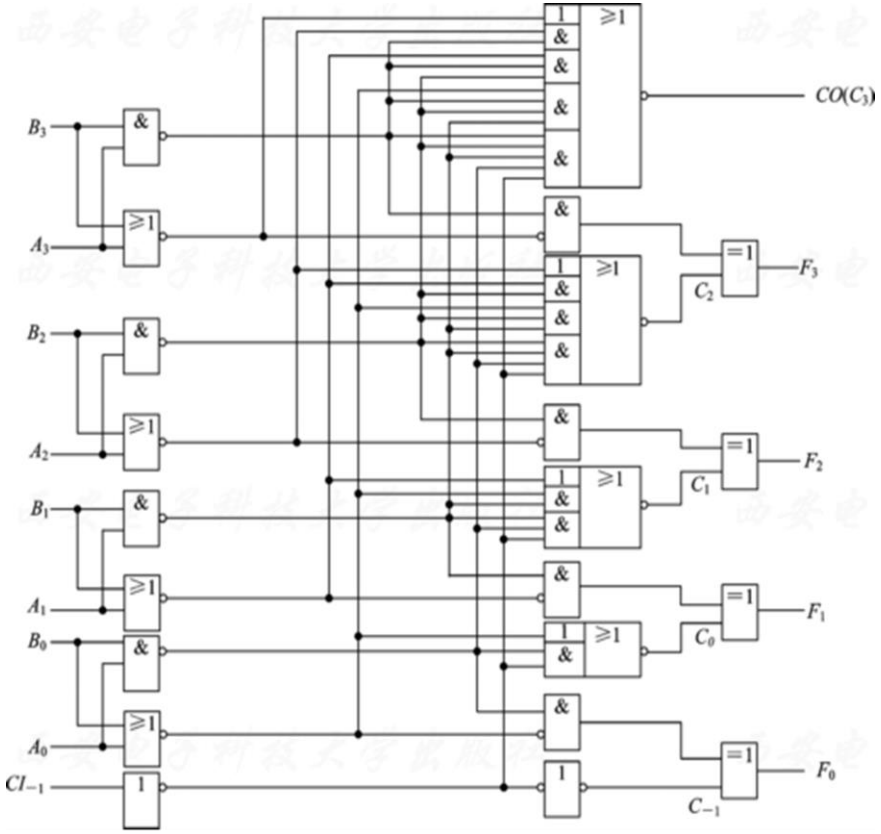


图4.4.4 4位超前进位加法器

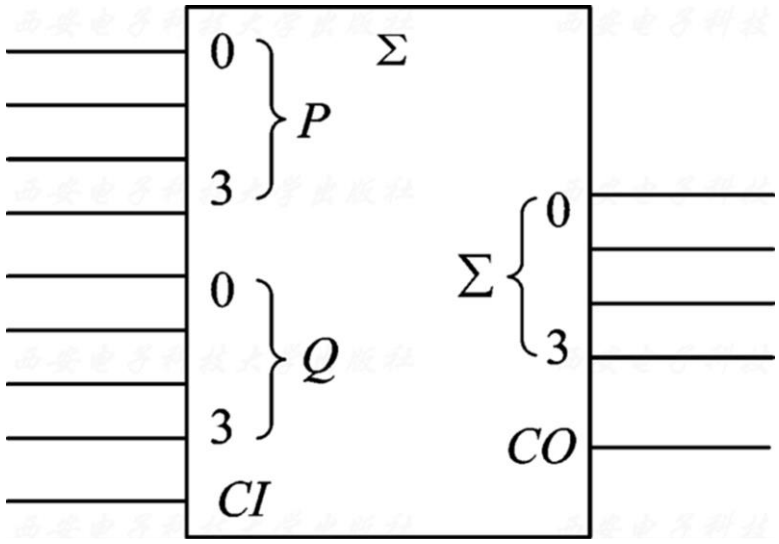


图4.4.5 4位全加器的逻辑符号

全加器：要考虑进位；**半加器**：不考虑进位

（二）加法器的应用举例

加法器除了进行二进制加法外，还可以用来构成**代码转换**、**减法器**、**十进制加法器**等电路。

【例 4.4.1】分析图 4.4.6 所示电路，说明其逻辑功能。

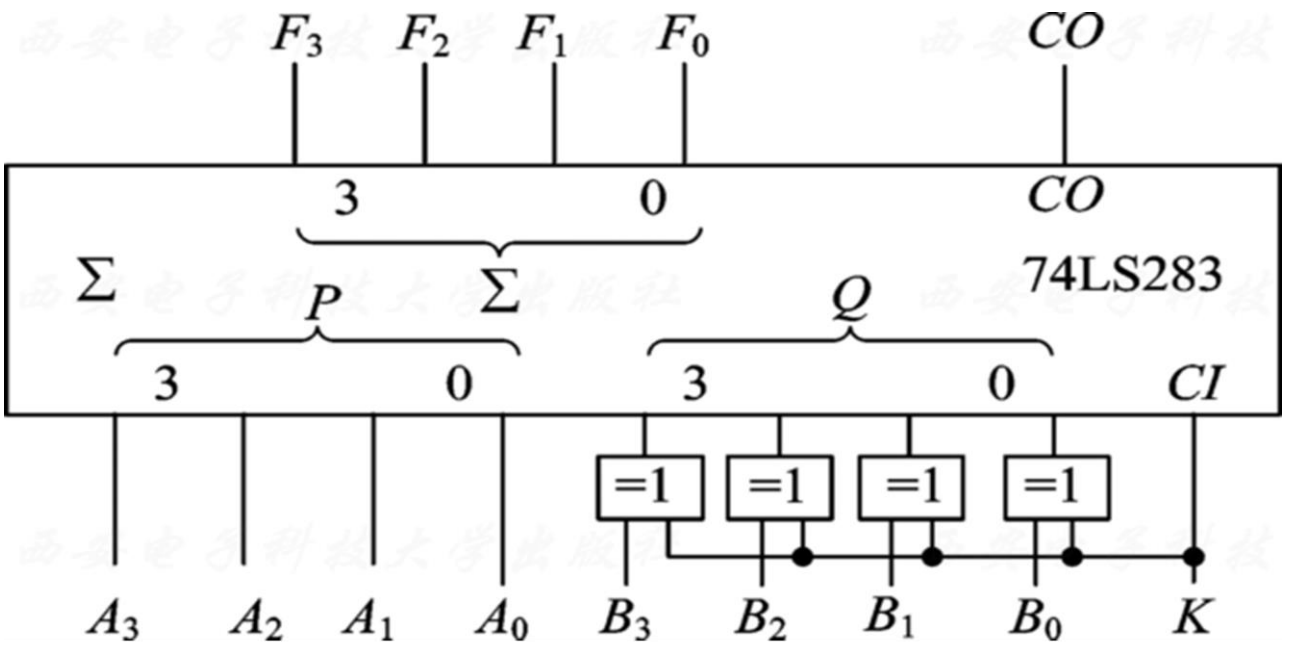


图4.4.6 例4.4.1的电路图

解答：图 4.4.6 中，74LS283 是 4 位超前进位加法器，*A*（$A_3A_2A_1A_0$）、*B*（$B_3B_2B_1B_0$）分别是两个 4 位的加数，*F*（$F_3F_2F_1F_0$）是运算的结果（本位和），*CI* 是进位输入，*CO* 是进位输出。

由图 4.4.6 有： $F_i = A_i + (B_i \oplus K), CI = K$

当 *K*=0 时：$F_i = A_i + B_i, CI = 0$，其功能是实现 4 位加法运算。

当 K=1 时：$F_i = A_i + \overline{B}_i, CI = 1$，其功能是实现 A 与 B 的反码相加再加 1 的功能，即实现 4 位二进制补码减法运算。

因此，图 4.4.6 的功能是**在输入 K 信号的控制下，实现补码的加或减运算**。

【例 4.4.2】试用一位全加器实现两位二进制**乘法器**。

解答：设有两个 2 位二进制数分别为 $A=A_1A_0$，$B=B_1B_0$，P 是其乘法运算结果，则有：

$$P=A\times B=A_1A_0\times B_1B_0$$

由于 2 位二进制的乘法运算的结果最多是 4 位二进制数，因此设 $P=P_3P_2P_1P_0$，其各位产生的计算过程如下：

$$\begin{array}{cccc} & & A_1 & A_0 \\ & \times & B_1 & B_0 \\ \hline & & A_1B_0 & A_0B_0 \\ + & A_1B_1 & A_0B_1 & \\ \hline P_3 & P_2 & P_1 & P_0 \end{array}$$

因此：$\begin{cases} P_0 = A_0 B_0 \\ P_1 = A_1 B_0 + A_0 B_1 \\ P_2 = A_1 B_1 + C_1 \\ P_3 = C_2 \end{cases}$，其中，$C_1$ 是 $A_1B_0+A_0B_1$ 的进位输出，C_2 是 $A_1B_1+C_1$ 的进位输出。

用 1 位加法器和与门实现的 2 位二进制乘法的电路如下：

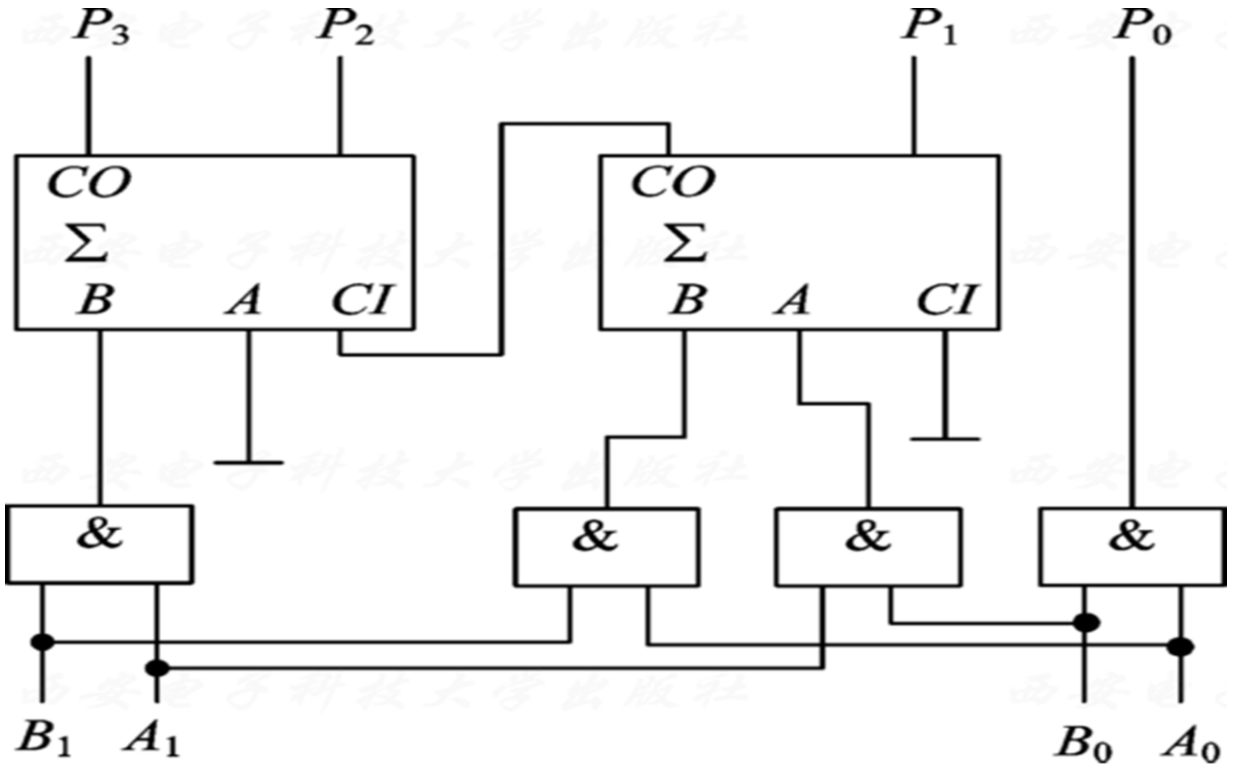


图4.4.7 两位乘法电路

二、译码器

（一）二进制译码器

1、二进制译码器的工作原理

二进制译码器的逻辑功能是把输入二进制代码表示的所有状态翻译成对应的输出信号。

若输入是 3 位二进制代码，则 3 位二进制代码可以表示 8 种状态，因此就有 8 个输出端，每个输出端分别表示一种输入状态。因此，又把 3

位二进制译码器称为 **3 线—8 线译码器**，简称 **3-8 译码器**，与此类似的还有 **2-4 译码器**等。

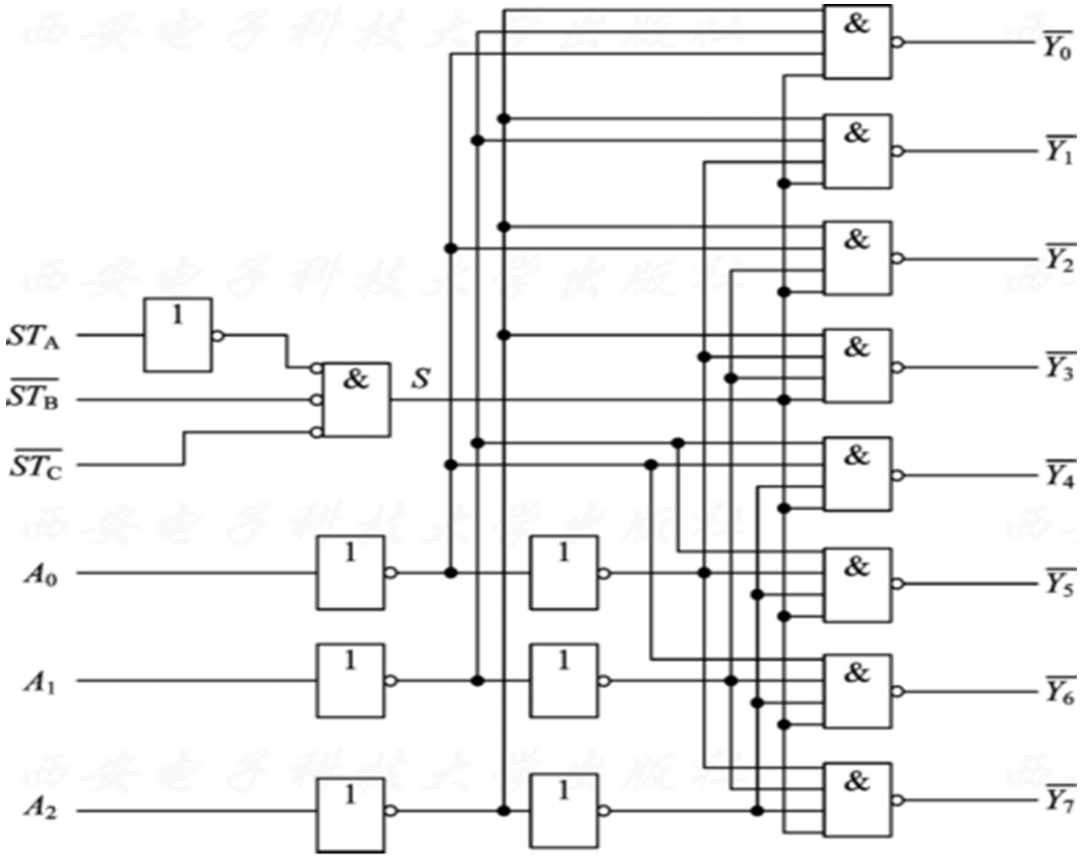


图4.4.16 74LS138的逻辑电路图

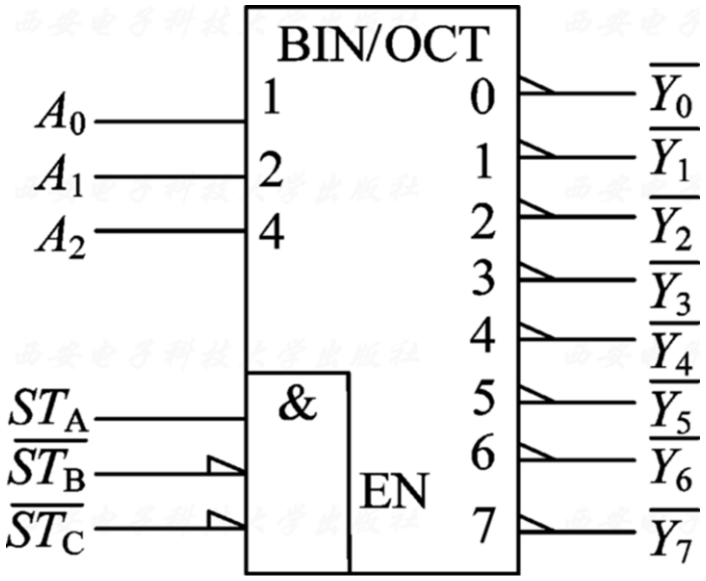


图 4.4.17 74LS138的逻辑符号

输入端：$A_2A_1A_0$，也称为地址输入端；

输出端：$\overline{Y}_0 \sim \overline{Y}_7$

控制端：$ST_A, \overline{ST}_B, \overline{ST}_C$

若设$S = ST_A \cdot \overline{\overline{ST}_B} \cdot \overline{\overline{ST}_C}$，则当 S=1 时译码器工作；当 S=0 时禁止译码器译码，输出全为 1。

输出逻辑表达式：

$$\overline{Y}_i(A_2, A_1, A_0) = \overline{S \cdot m_i} \quad (i = 0,1,2,\ldots 7)$$

m_i包含了 A_2、A_1、A_0 三个变量的全部最小项，所以把这种译码器又叫做**最小项译码器**。

表4.4.4 74LS138的真值表

输入					输出							
ST_A	$\overline{ST_A}+\overline{ST_B}$	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
×	1	×	×	×	1	1	1	1	1	1	1	1
0	×	×	×	×	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
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

2、二进制译码器的应用

1）译码器的扩展

【例 1】将两片 74LS138 扩展成 4-16 译码器。

解答：设 4-16 译码器输入为 $A_3A_2A_1A_0$，译码器的输出为 Y_0～Y_{15}，电路由两片 74LS138 组成，分别实现高 8 位和低 8 位译码输出。

当 A_3=1 时，使高位芯片工作，输出 Y_8～Y_{15}

有效，低位芯片禁止工作，Y_0～Y_7 输出均为 1；

当 A_3=0 时，高位芯片禁止工作，输出 Y_8～Y_{15} 均为 1，低位芯片允许译码，Y_0～Y_7 输出有效。

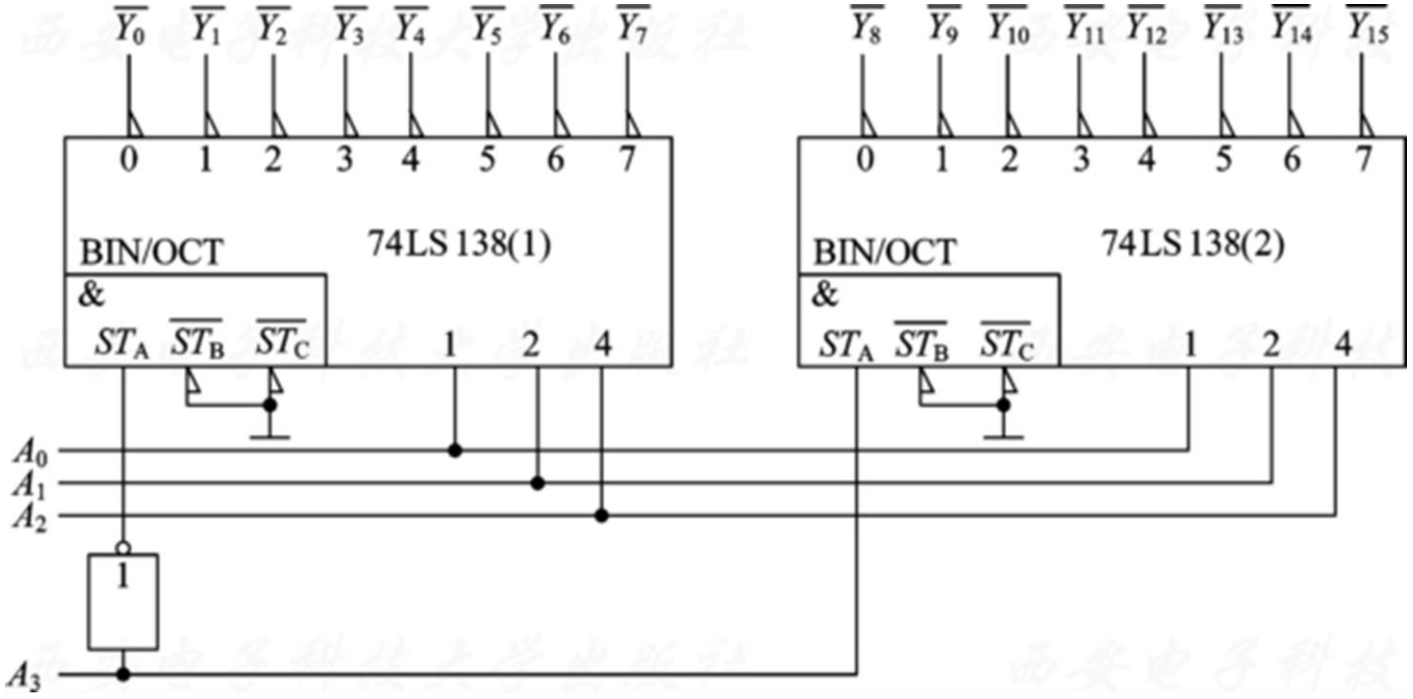


图4.4.18 例4.4.3的电路图

【例 2】用 74LS138 扩展实现 5-32 译码器。

解答：构成 5-32 译码器需要四个 74LS138 芯片，各芯片的控制信号可以通过门电路产生，更简单的方法是通过一个 2-4 译码器 74LS139 产生，实现电路如下图所示：

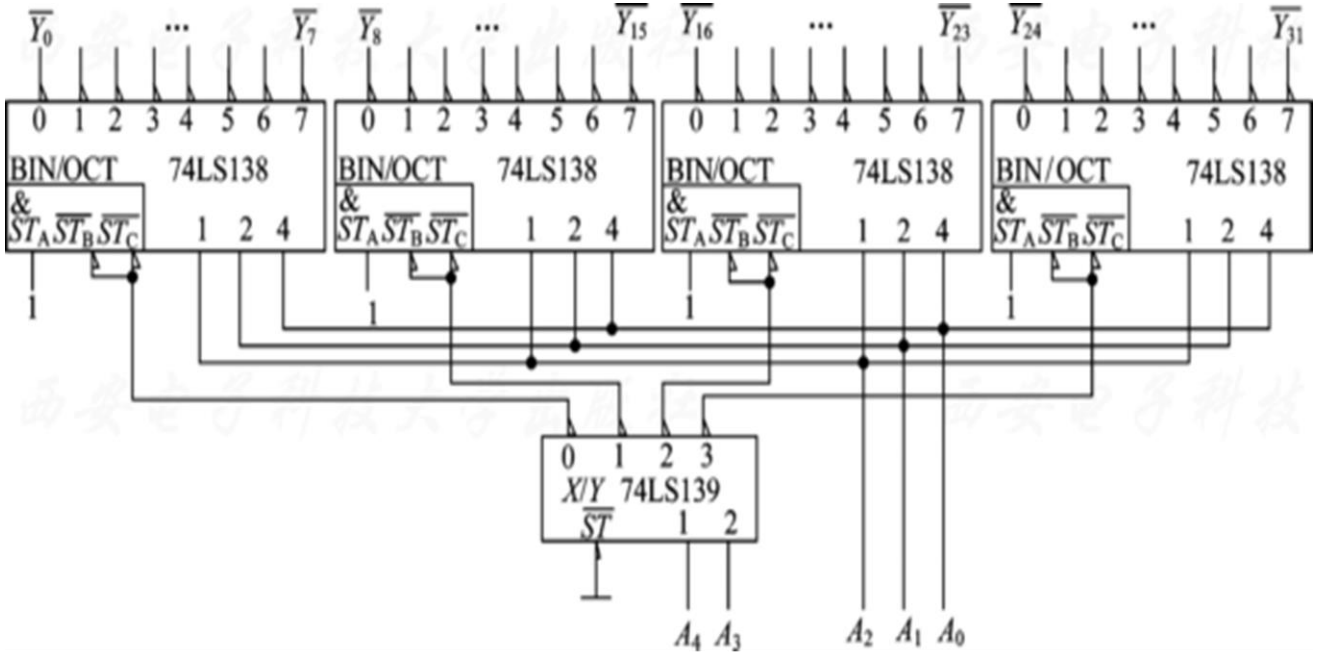


图4.4.19 用74LS138扩展成5-32译码器

2）利用译码器实现逻辑函数

【例 4.4.5】分析下图所示电路，写出输出函数 F 的逻辑表达式。

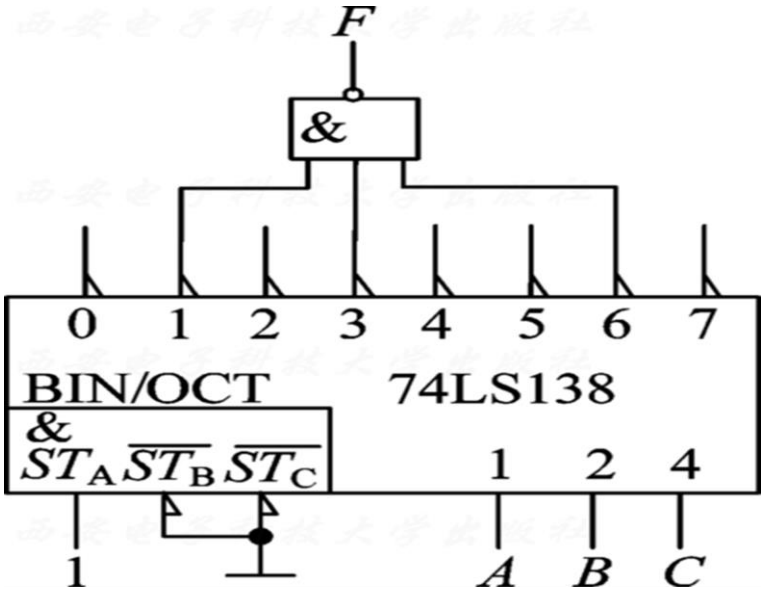


图4.4.20 例4.4.5的电路图

解答：由图可知三个使能输入端均有效，译码器正常工作。

$$F(C,B,A)=\overline{\overline{Y_1}\cdot\overline{Y_3}\cdot\overline{Y_6}}=\overline{\overline{m_1}\cdot\overline{m_3}\cdot\overline{m_6}}=m_1+m_3+m_6$$

$$F(C,B,A)=\overline{C}\,\overline{B}A+\overline{C}B\overline{A}+CB\overline{A}$$

一个 ***n*** 变量输入的译码器，其输出包含了这 ***n*** 个输入变量的全部最小项（或最小项的非）。

因此，利用 ***n*** 变量译码器和门电路就能实现任何形式的输入变量不大于 ***n*** 的组合逻辑函数。

【例 4.4.6】用译码器实现下面两个函数：

$$F_1=A\overline{B}+\overline{B}C+AC;\quad F_2=\overline{A}\,\overline{B}+B\overline{C}+ABC$$

解答：方法一：选用 **3-8 译码器**和**与非门**实现。

将输出函数写成最小项之和的形式，并变换为译码器反码输出形式，用与非门作为 F_1、F_2 的输出门。

也可以利用卡诺图，将输出函数写成最小项之和的形式。

$$F_1(A,B,C) = A\overline{B} + \overline{B}C + AC$$

$$= A\overline{B}\left(C+\overline{C}\right) + \overline{B}C\left(A+\overline{A}\right) + AC\left(B+\overline{B}\right)$$

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

$$= m_1 + m_4 + m_5 + m_7$$

$$= \overline{\overline{m_1 + m_4 + m_5 + m_7}}$$

$$= \overline{\overline{m_1} \cdot \overline{m_4} \cdot \overline{m_5} \cdot \overline{m_7}}$$

$$= \overline{\overline{Y_1} \cdot \overline{Y_4} \cdot \overline{Y_5} \cdot \overline{Y_7}}$$

$$F_2(A,B,C) = \overline{A}\,\overline{B} + B\overline{C} + ABC$$

$$= m_0 + m_1 + m_2 + m_6 + m_7$$

$$= \overline{\overline{m_0} \cdot \overline{m_1} \cdot \overline{m_2} \cdot \overline{m_6} \cdot \overline{m_7}}$$

$$= \overline{\overline{Y_0} \cdot \overline{Y_1} \cdot \overline{Y_2} \cdot \overline{Y_6} \cdot \overline{Y_7}}$$

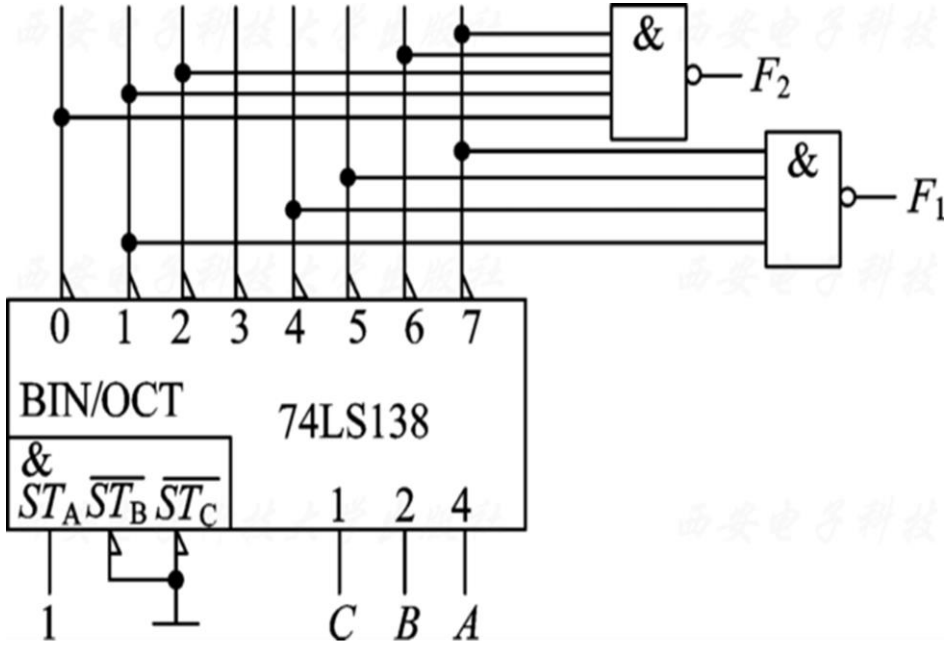


图4.4.21 用译码器和与非门实现例4.4.6的逻辑函数

连接电路时只要将输入变量 A、B、C 分别加到译码器地址输入端 A_2、A_1、A_0，然后用相关的译码输出信号作与非门的输入信号即可实现指定逻辑函数，实现电路如上图所示。

方法二：选用 **3-8 译码器**和**与门**实现。

将输出函数写成最大项之积的形式，然后进行如下变换：

$$F_1(A,B,C) = A\overline{B} + \overline{B}C + AC$$

$$= m_1 + m_4 + m_5 + m_7$$

$$= \sum \mathrm{m}(1,4,5,7)$$

$$= \prod \mathrm{M}(0,2,3,6)$$

$$= M_0 \cdot M_2 \cdot M_3 \cdot M_6$$

$$= \overline{Y}_0 \cdot \overline{Y}_2 \cdot \overline{Y}_3 \cdot \overline{Y}_6$$

$$F_2(A,B,C) = \overline{A}\,\overline{B} + B\overline{C} + ABC$$

$$= m_0 + m_1 + m_2 + m_6 + m_7$$

$$= \sum \mathrm{m}(0,1,2,6,7)$$

$$= \prod \mathrm{M}(3,4,5)$$

$$= M_3 \cdot M_4 \cdot M_5$$

$$= \overline{Y}_3 \cdot \overline{Y}_4 \cdot \overline{Y}_5$$

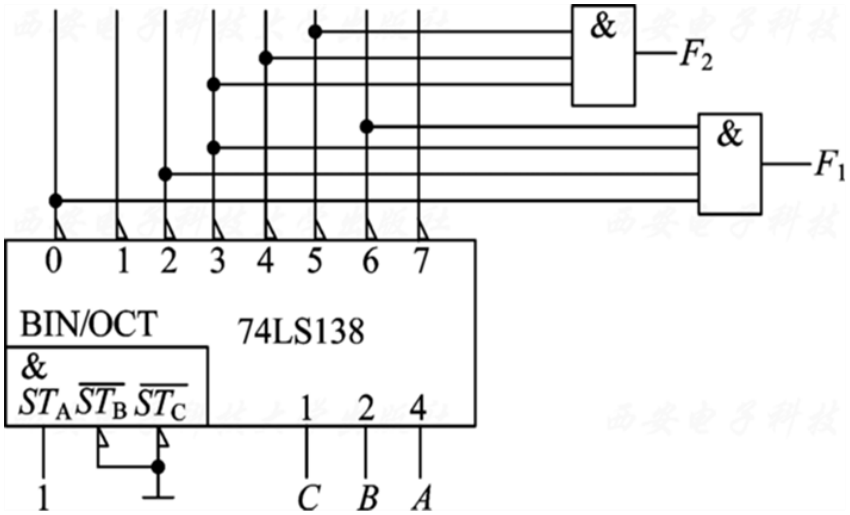


图4.4.22 用译码器和非门实现例4.4.6的逻辑函数

（二）二-十进制译码器

二-十进制译码器的逻辑功能是将输入的 BCD 码译成十个输出信号。常用的中规模二-十进制译码芯片有 74LS42，其逻辑符号如下：

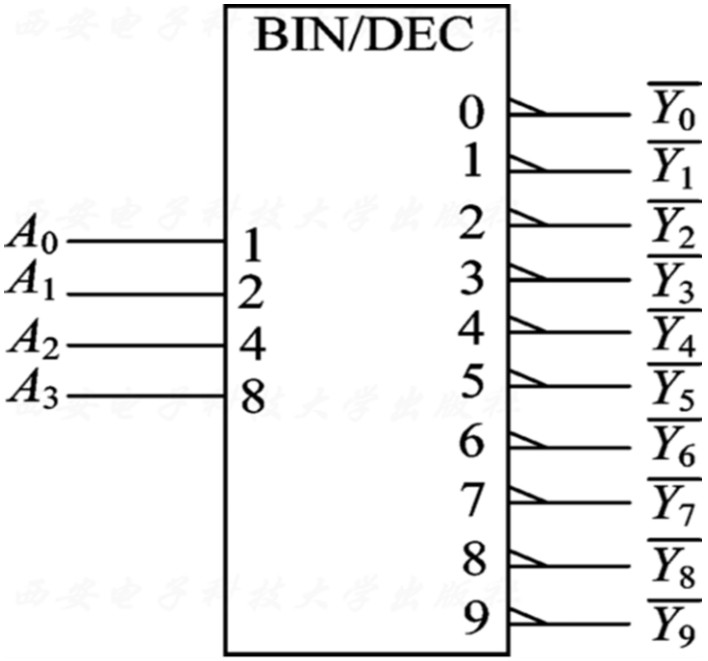


图4.4.24 74LS42的逻辑符号

$A_3A_2A_1A_0$ 是 **BCD 编码**输入端；

$\overline{Y}_0 \sim \overline{Y}_9$ 是输出端，低电平有效。

当 $A_3A_2A_1A_0$ 为**冗余码**输入时，

输出均为高电平。

这种译码器也称为**4 线-10 线译码器**。

（三）显示译码器

数字显示电路是数字系统不可缺少的部分，通常由显示译码器、驱动器和显示器等部分组成，如下图所示：

图4.4.26 数字显示电路的组成

1、显示译码器的工作原理

由于显示器件和显示方式的多样性，其译码电路也不相同，最简单和最常用的是**七段数码显示电路**，它由多个发光二极管 LED 按分段式封装制成。

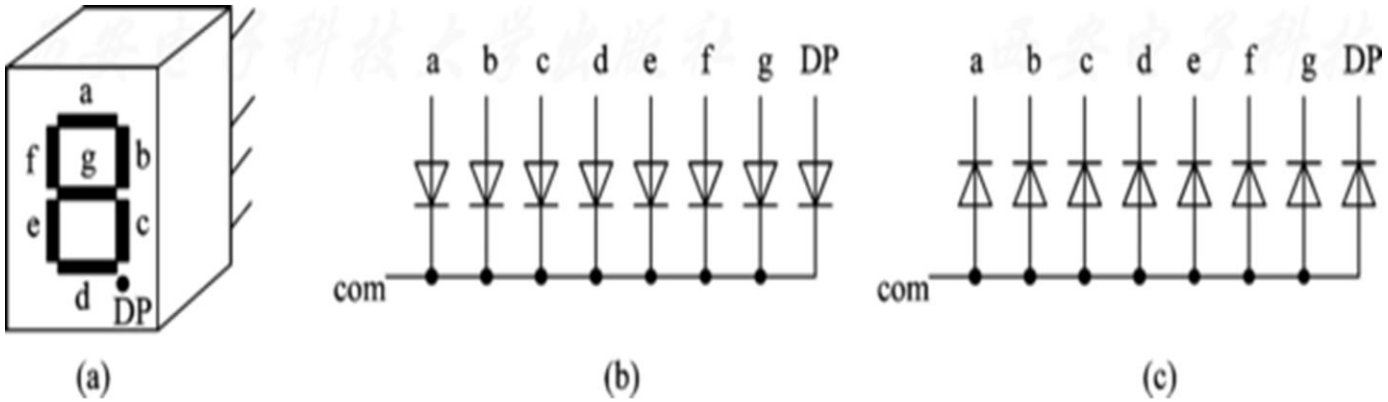


图4.4.27 七段显示LED数码管□

(a) 外形图; (b) 共阴极连接; (c) 共阳极连接

LED 数码管有**共阴型**和**共阳型**两种形式，下图分别给出了七段数码显示器件的外形图、共阴极和共阳极 LED 电路连接图。

图中，8 个 LED 分别用于显示数字和小数点，每个 LED 灯的亮灭由其对应的 a～g、DP 段位信号控制。

下图是常用七段译码器的输出与显示字形的对应关系：

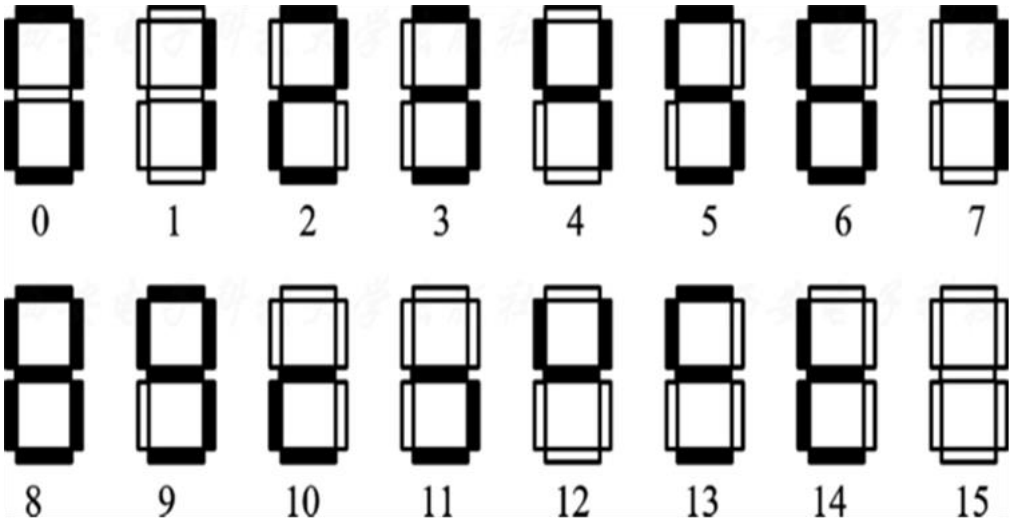


图4.4.28 常用七段译码器字形

例如，当显示数字“0”时，a～g 七个段中只有 g 段的 LED 灯是灭的，其余段的 LED 灯都应亮。

中规模数字译码器件很多，图 4.4.29(a)和(b)分别是**共阴极七段译码器 74LS48** 的电路图和逻辑符号。

74LS48 为有内部上拉电阻的**BCD 七段译码器/驱动器**，各信号功能分别如下：

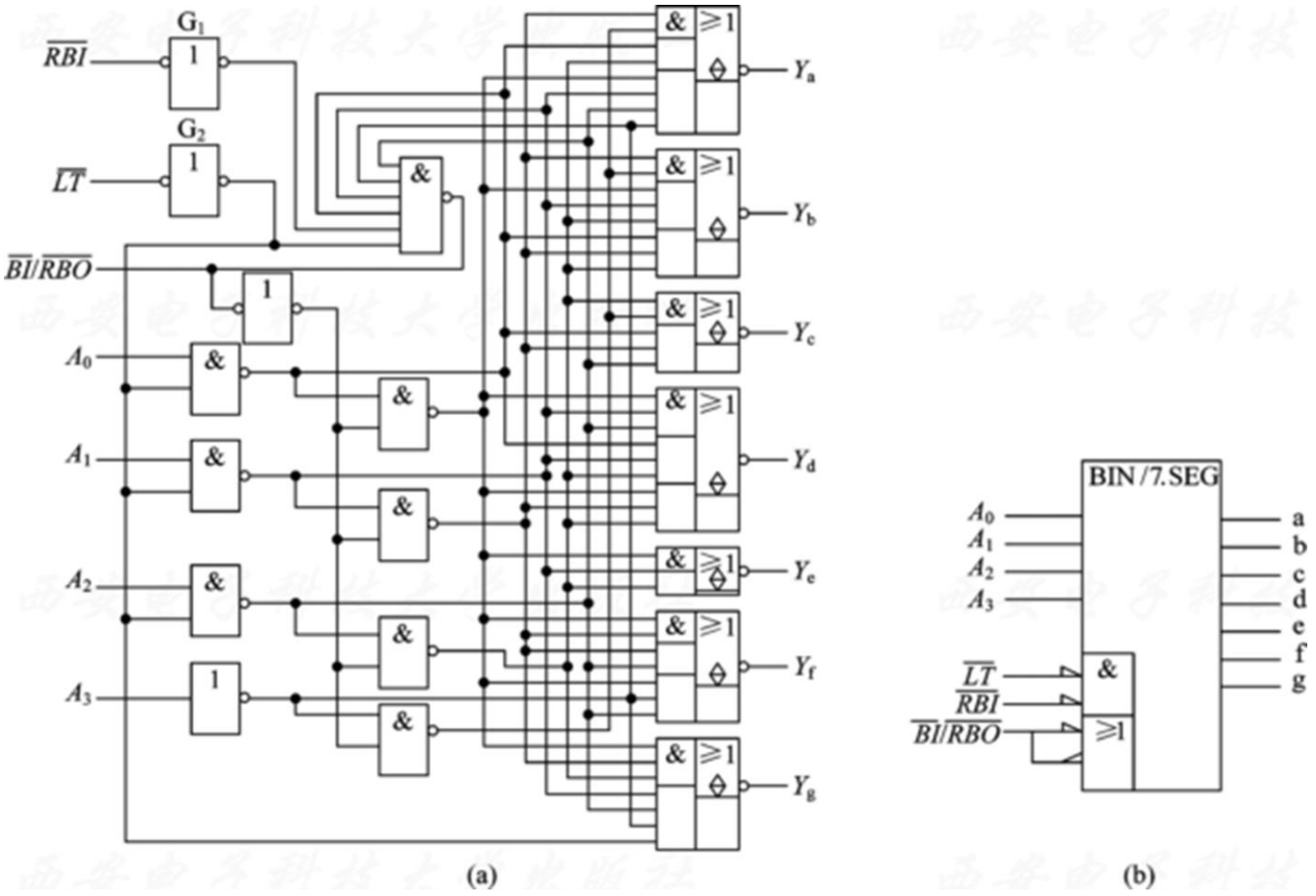


图4.4.29 七段译码器74LS48电路图和逻辑符号

(a) 74LS48电路图; (b) 74LS48逻辑符号

- A_3～A_0：8421BCD 输入端
- Y_a～Y_g：七段输出端，为高电平有效，可驱

动共阴极七段数码管。

•$\overline{LT}$：灯光测试输入端。当$\overline{LT}$=0 且$\overline{BI}$=1 时，与 A_3～A_0 状态无关，Y_a～Y_g 输出均为高电平，使七段数码管各段全部点亮。

• $\overline{RBI}$：串行灭零输入端。对不希望显示的数码“0”，可以通过控制$\overline{RBI}$=0 实现。即当$\overline{LT}$=1 且$\overline{RBI}$=0 时，若 A_3～A_0=0000，则不显示“0”。

• $\overline{BI}$ / $\overline{RBO}$：熄灭输入/串行灭零输出端。具有双重功能的端口，既可以作为输入信号$\overline{BI}$端，又可作为输出信号$\overline{RBO}$端口。

$\overline{BI}$为消隐输入，当$\overline{BI}$=0 时，与其他输入信号无关，Y_a～Y_g输出均为低电平，使七段 LED 灯处于熄灭状态。

$\overline{RBO}$为串行灭零输出端口。

2、数码显示器的应用

下图所示是用多片 74LS48 组成的一个数码显示系统。

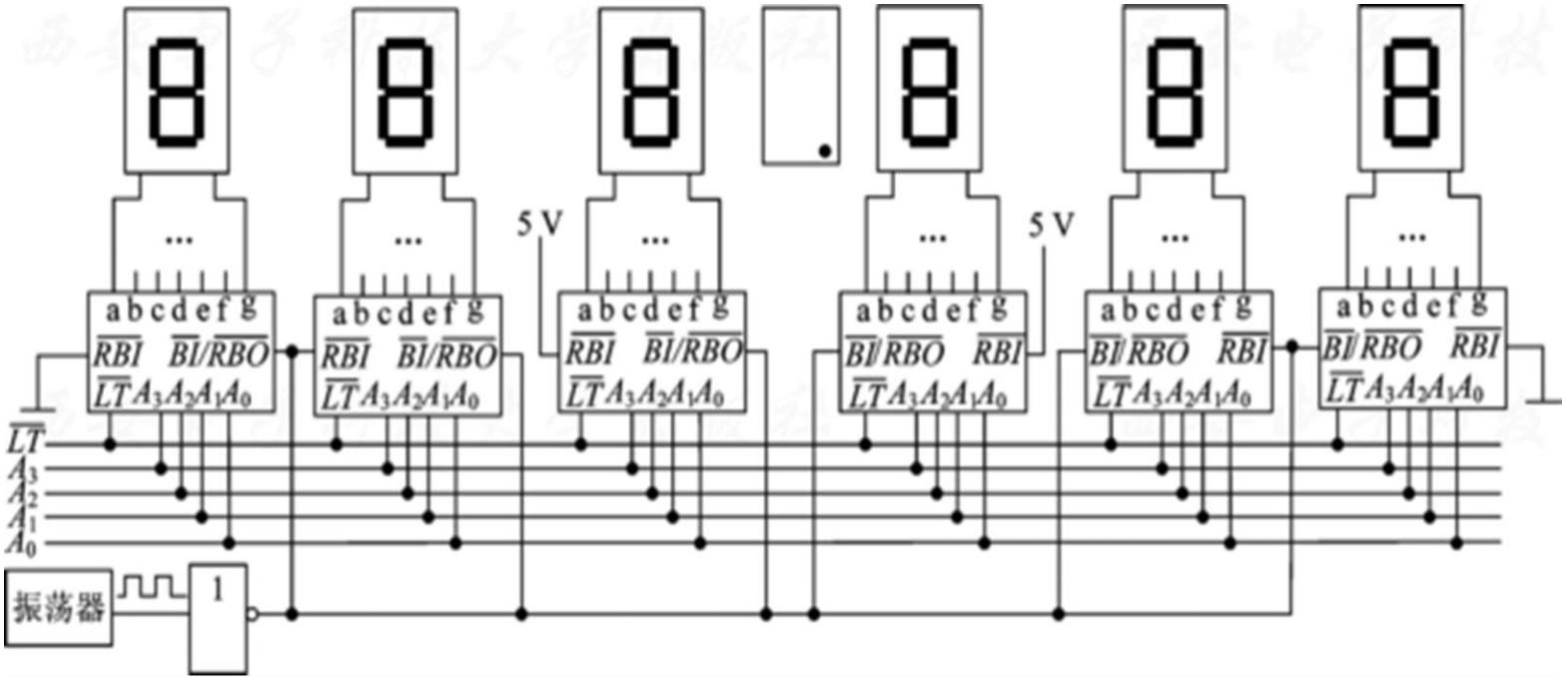


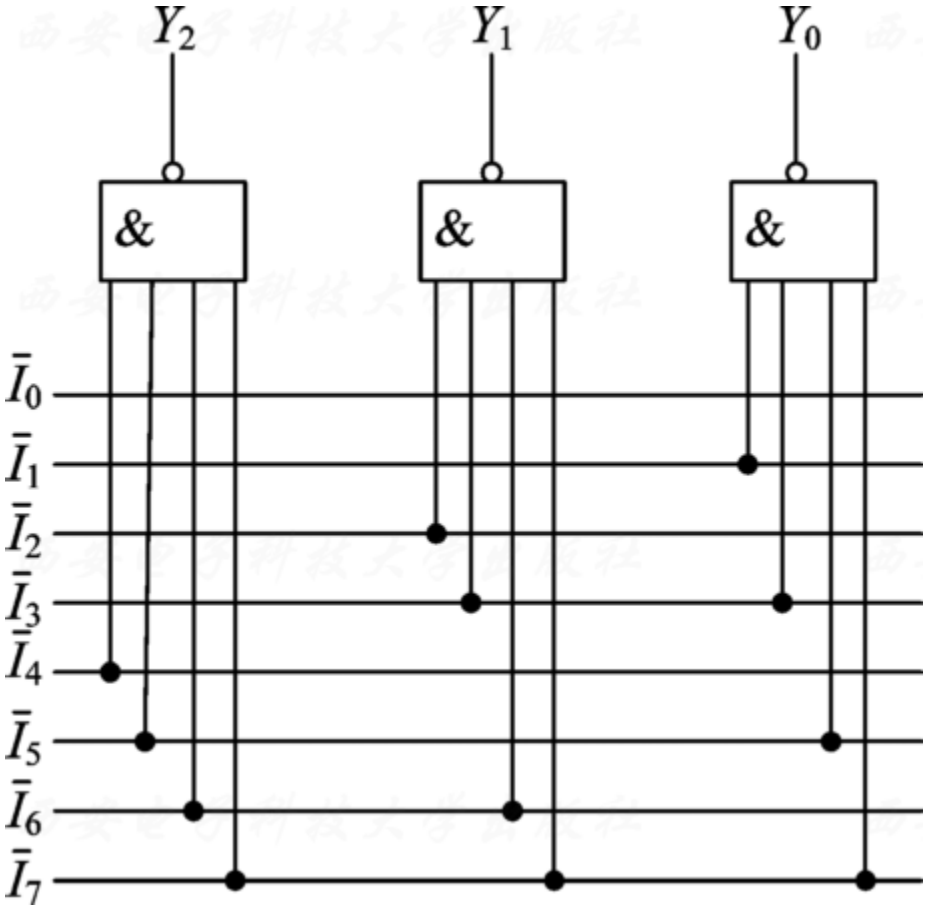
图4.4.30 七段译码器显示系统

三、编码器

（一）编码器的工作原理及实现

1、普通编码器

一个普通的 **8 线-3 线**编码器电路如下图所示。



其中：$\overline{I_0} \sim \overline{I_7}$ 是 8 个输入信号，$Y_2Y_1Y_0$ 是 3 位二进

制编码输出信号。输入与输出关系如下表所示：

4.4.2　普通 8 线—3 线编码器输入输出关系表

输　入								输　出		
$\bar{I}_0$	$\bar{I}_1$	$\bar{I}_2$	$\bar{I}_3$	$\bar{I}_4$	$\bar{I}_5$	$\bar{I}_6$	$\bar{I}_7$	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

输出逻辑表达式为：$\begin{cases} Y_2 = \overline{\overline{I_4}\,\overline{I_5}\,\overline{I_6}\,\overline{I_7}} \\ Y_1 = \overline{\overline{I_2}\,\overline{I_3}\,\overline{I_6}\,\overline{I_7}} \\ Y_0 = \overline{\overline{I_1}\,\overline{I_3}\,\overline{I_5}\,\overline{I_7}} \end{cases}$

2、优先编码器

普通编码器：不能同时有多个有效的输入

优先编码器：可以同时有多个有效的输入

下图所示为常用优先编码器 74LS148 的逻辑电路图：

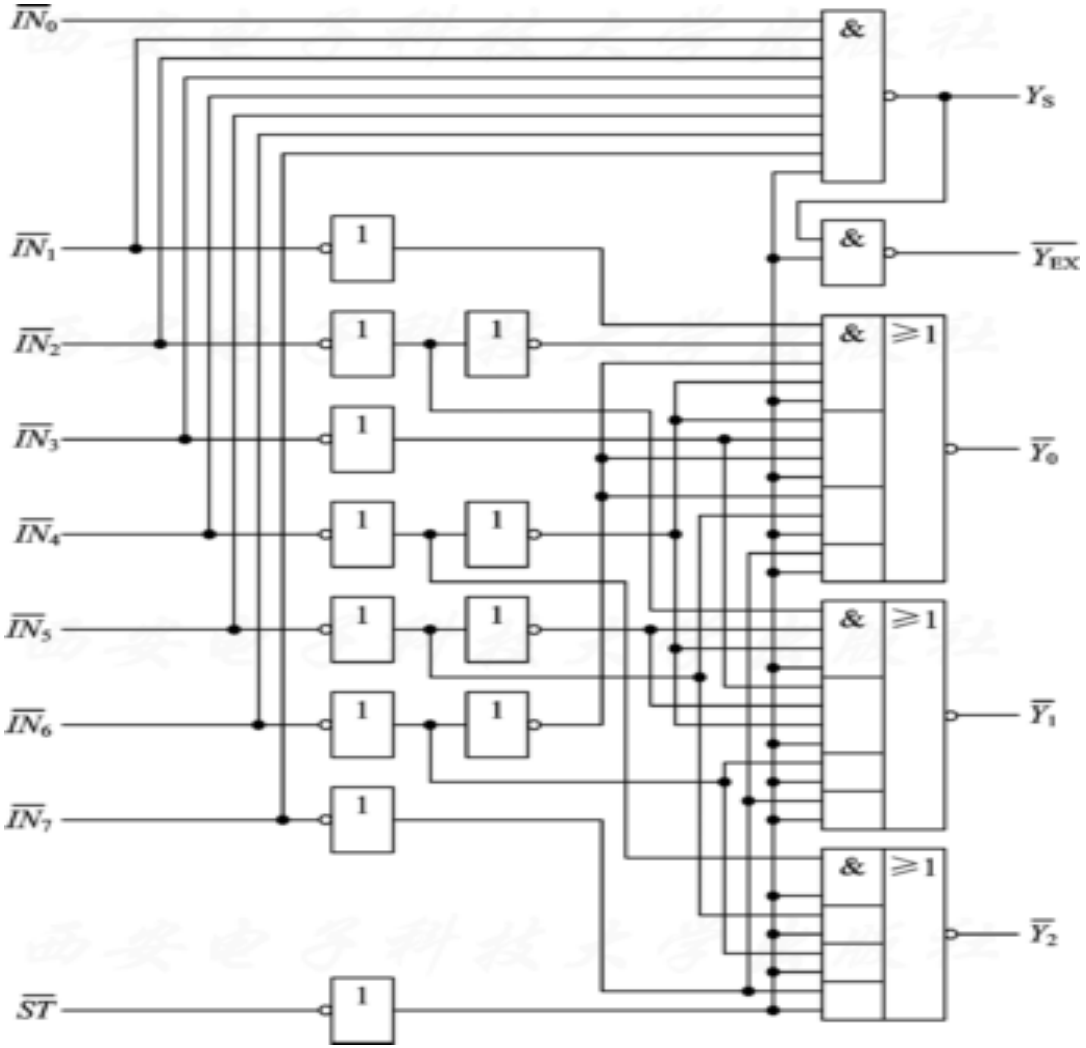


图 4.4.11　74LS148 逻辑电路图

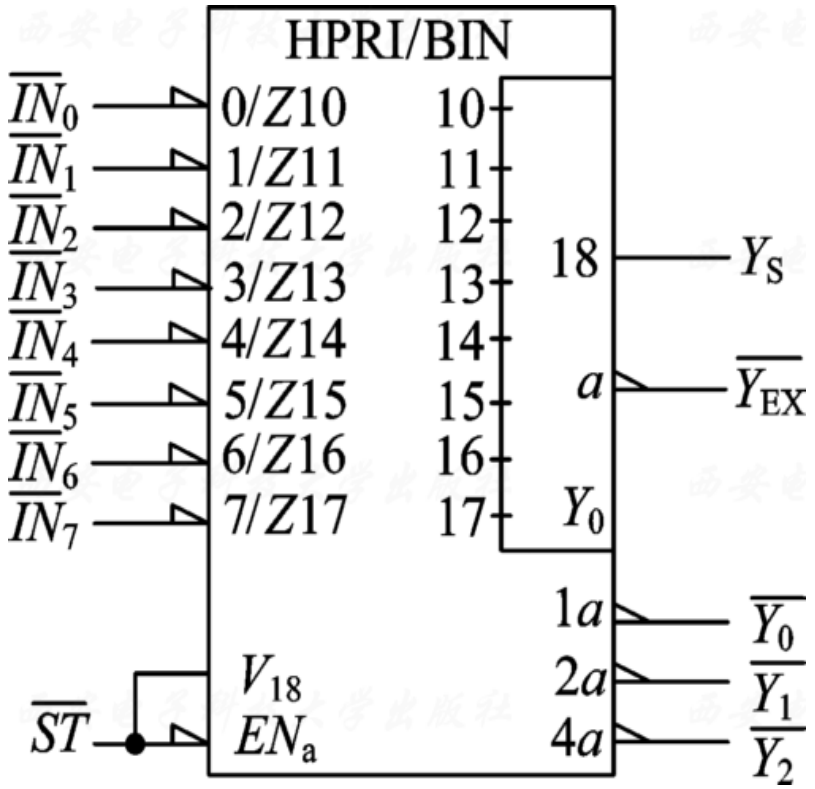


图 4.4.12　74LS148 逻辑符号

图中，$\overline{ST}$是选通控制输入端，当$\overline{ST}$=0 时，编

码器的输出取决于输入信号，当$\overline{ST}$=1 时，所有输出均被封锁为 1；

Y_S 为选通输出端，当$\overline{ST}=0, Y_S=0$时，表示编码电路工作，但所有输入信号均为无效状态；

$\overline{Y_{EX}}$是扩展端，当$\overline{ST}=0, \overline{Y_{EX}}=0$时，表示编码电路工作，有编码信号输入。

Y_S和$\overline{Y_{EX}}$常用于编码器的扩展连接；

8 个编码输入信号中，$\overline{IN_7}$优先权最高，$\overline{IN_0}$优先权最低。

若不考虑$\overline{ST}$、Y_S和$\overline{Y_{EX}}$，则电路输出方程为：

$$\begin{cases} \overline{Y_2} = \overline{\overline{IN_7} + \overline{IN_6} + \overline{IN_5} + \overline{IN_4}} = \overline{IN_7} \cdot \overline{IN_6} \cdot \overline{IN_5} \cdot \overline{IN_4} \\ \overline{Y_1} = \overline{IN_7} \cdot \overline{IN_6} \cdot \left(IN_5 + IN_4 + \overline{IN_3} \right) \cdot \left(IN_5 + IN_4 + \overline{IN_2} \right) \\ \overline{Y_0} = \overline{IN_7} \cdot \left(IN_6 + \overline{IN_5} \right) \cdot \left(IN_6 + IN_4 + \overline{IN_3} \right) \cdot \left(IN_6 + IN_4 + IN_2 + \overline{IN_1} \right) \end{cases}$$

其真值表如下：

表 4.4.3　优先编码器真值表表

输　入									输　出				
$\overline{ST}$	$\overline{IN_0}$	$\overline{IN_1}$	$\overline{IN_2}$	$\overline{IN_3}$	$\overline{IN_4}$	$\overline{IN_5}$	$\overline{IN_6}$	$\overline{IN_7}$	$\overline{Y}_2$	$\overline{Y}_1$	$\overline{Y}_0$	$\overline{Y_{EX}}$	Y_S
1	×	×	×	×	×	×	×	×	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	×	×	×	×	×	×	×	0	0	0	0	0	1
0	×	×	×	×	×	×	0	1	0	0	1	0	1
0	×	×	×	×	×	0	1	1	0	1	0	0	1
0	×	×	×	×	0	1	1	1	0	1	1	0	1
0	×	×	×	0	1	1	1	1	1	0	0	0	1
0	×	×	0	1	1	1	1	1	1	0	1	0	1
0	×	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

（二）编码器的应用举例

根据需要，可以用两片 8 线—3 线优先编码器连接成为一个 16 线-4 线优先编码

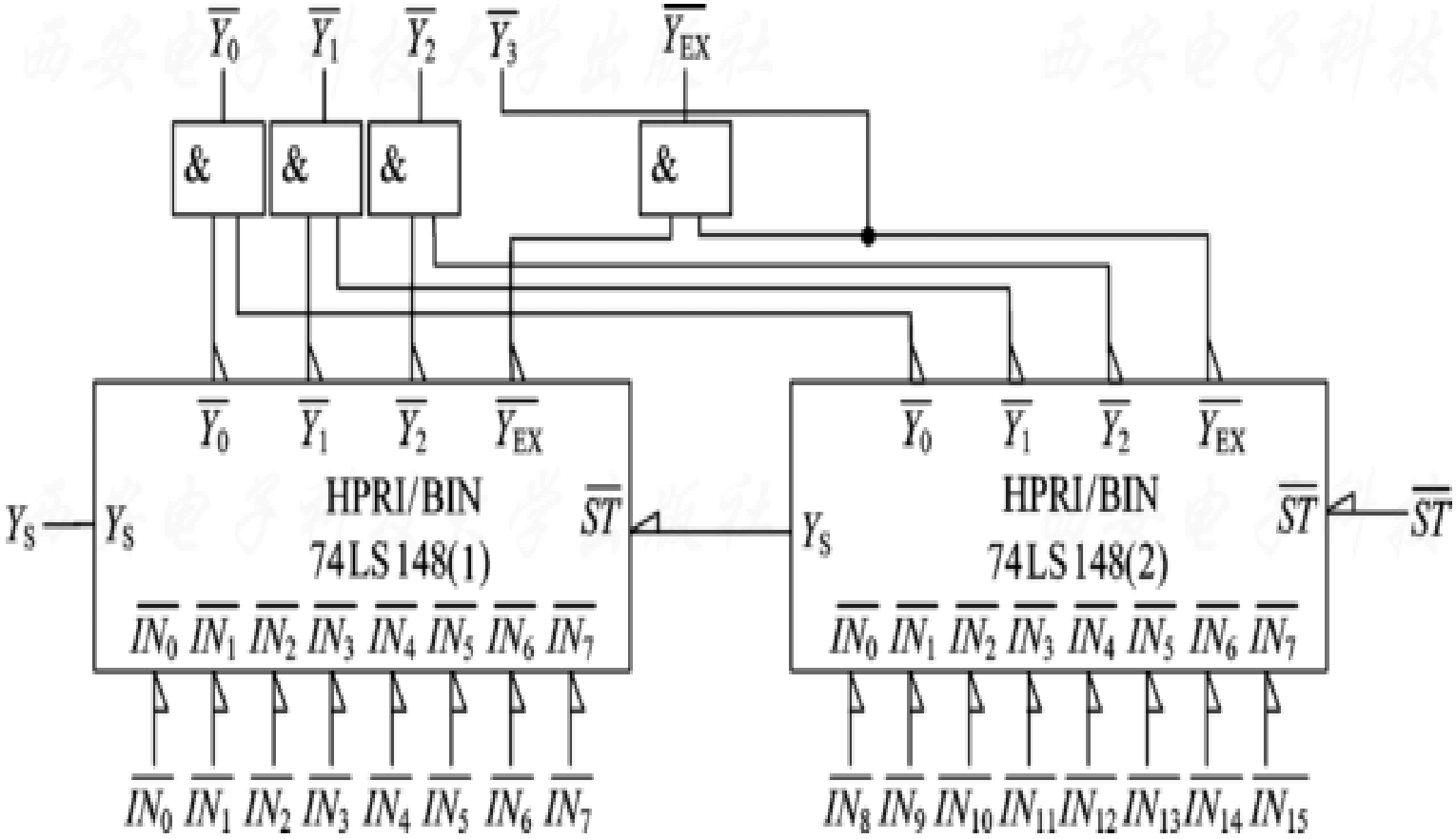


图 4.4.13　用 8 线—3 线优先编码器构成 16 线—4 线优先编码器

器，如上图所示。

四、数据选择器和数据分配器

（一）数据选择器

数据选择器的作用相当于多路开关，如下图所示：

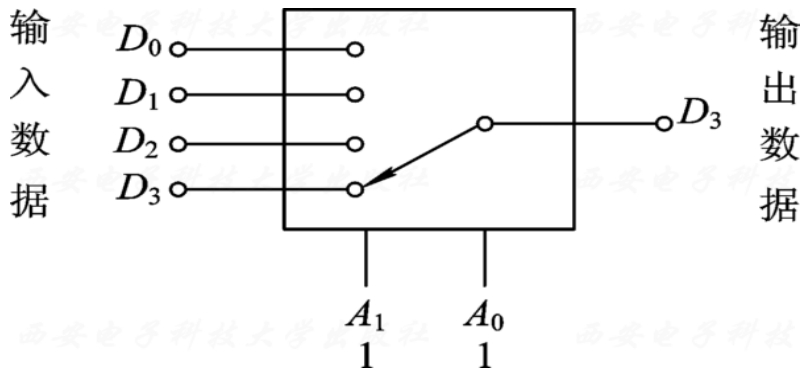


图 4.4.32　数据选择器工作原理示意图

根据输入信号A_1A_0的状态，从输入的四路数据$D_3 \sim D_0$中选择一个作为输出，因为图中$A_1A_0=11$，所以输出的数据是D_3。

1、数据选择器的工作原理

四选一数据选择器是在选择控制信号A_1A_0（又称为地址码）的控制下，从$D_3 \sim D_0$四个输入信号中选择一个送到公共输出端Y，其对应的真值表及电路图如下：

表 4.4.7　四选一数据选择器的真值表

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

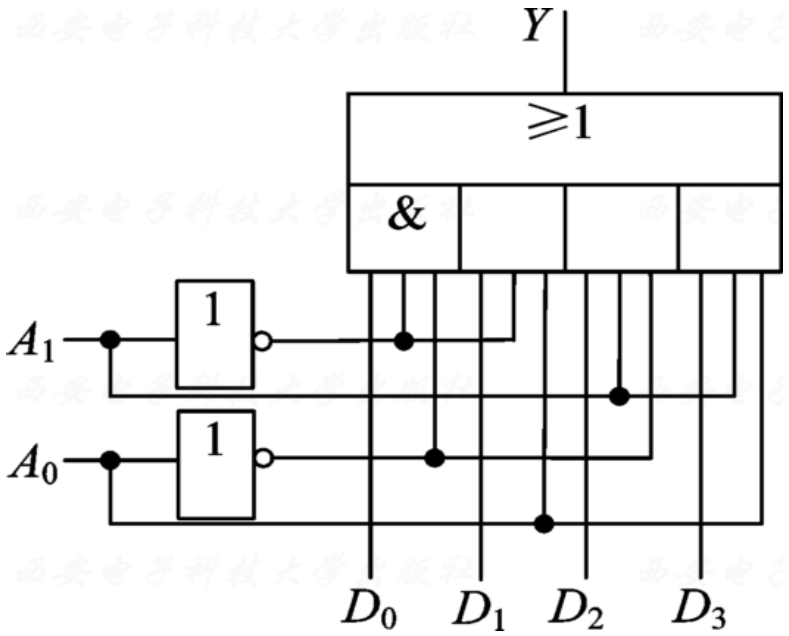


图 4.4.33　四选一数据选择器的电路

根据真值表可以写出输出Y的逻辑表达式：

$$Y = \overline{A}_1 \overline{A}_0 D_0 + \overline{A}_1 A_0 D_1 + A_1 \overline{A}_0 D_2 + A_1 A_0 D_3$$

中规模集成电路双四选一器件 74LS153 和互

补输出八选一数据选择器 74LS151 的逻辑符号：

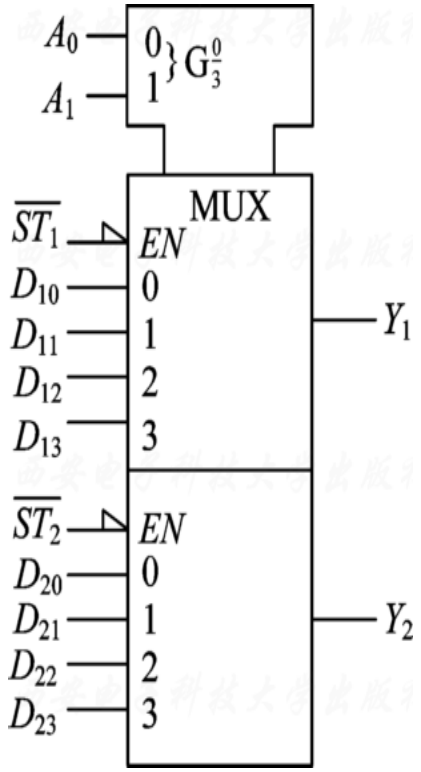
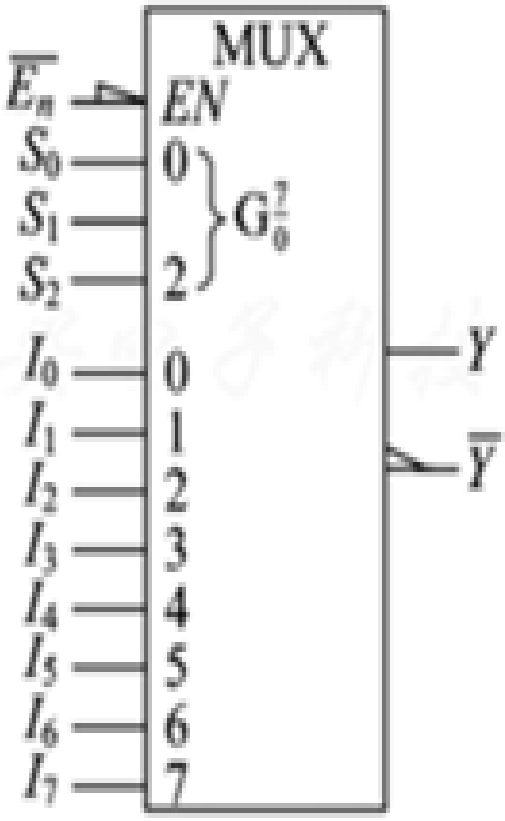


图 4.4.35　双四选一数据选择器逻辑符号



74LS153 的输出逻辑函数如下：

$$Y_1(A_1,A_0)$$

$$=ST_1(\overline{A_1}\,\overline{A_0}D_{10}+\overline{A_1}A_0D_{11}+A_1\overline{A_0}D_{12}+A_1A_0D_{13})$$

$$=ST_1(m_0D_{10}+m_1D_{11}+m_2D_{12}+m_3D_{13})$$

$$=ST_1\cdot\sum_{i=0}^{3}m_iD_{1i}$$

2、数据选择器的应用

1）数据选择器的级联

在实际应用中，如果给定的数据选择器不能满足数据输入端个数的要求，就需要利用已有的器件进行扩展。

常用的扩展方法是利用各数据选择芯片的选通端$\overline{EN}$。

【例 4.4.7】用 4 片八选一数据选择器构成三十二选一数据选择器。

解答：由于 $2^5=32$，因此三十二选一就需要 5 位地址，用 $A_4A_3A_2A_1A_0$ 来表示地址码。

进行芯片扩展的方法很多，这里介绍**使用译码器和选择器产生数据选择器控制信号**的方法。

方案一：采用 2-4 译码器实现扩展。

可以采用一片 2-4 译码器实现 4 个八选一芯片的选通控制，高位地址信号 A_4A_3 作 2-4 译码器的

地址输入，译码器输出分别接 4 片八选一数据选择器的选通端。A_4A_3 对 4 片八选一的控制如下表所示：

表 4.4.8　A_4A_3 对八选一芯片的控制情况

A_4	A_3	说明
0	0	MUX(1)工作
0	1	MUX(2)工作
1	0	MUX(3)工作
1	1	MUX(4)工作

未选中的芯片输出 Y=0，所以 4 个八选一芯片中只有选中的芯片输出有效。4 个八选一芯片的输出 Y 通过或门输出，电路如下图所示：

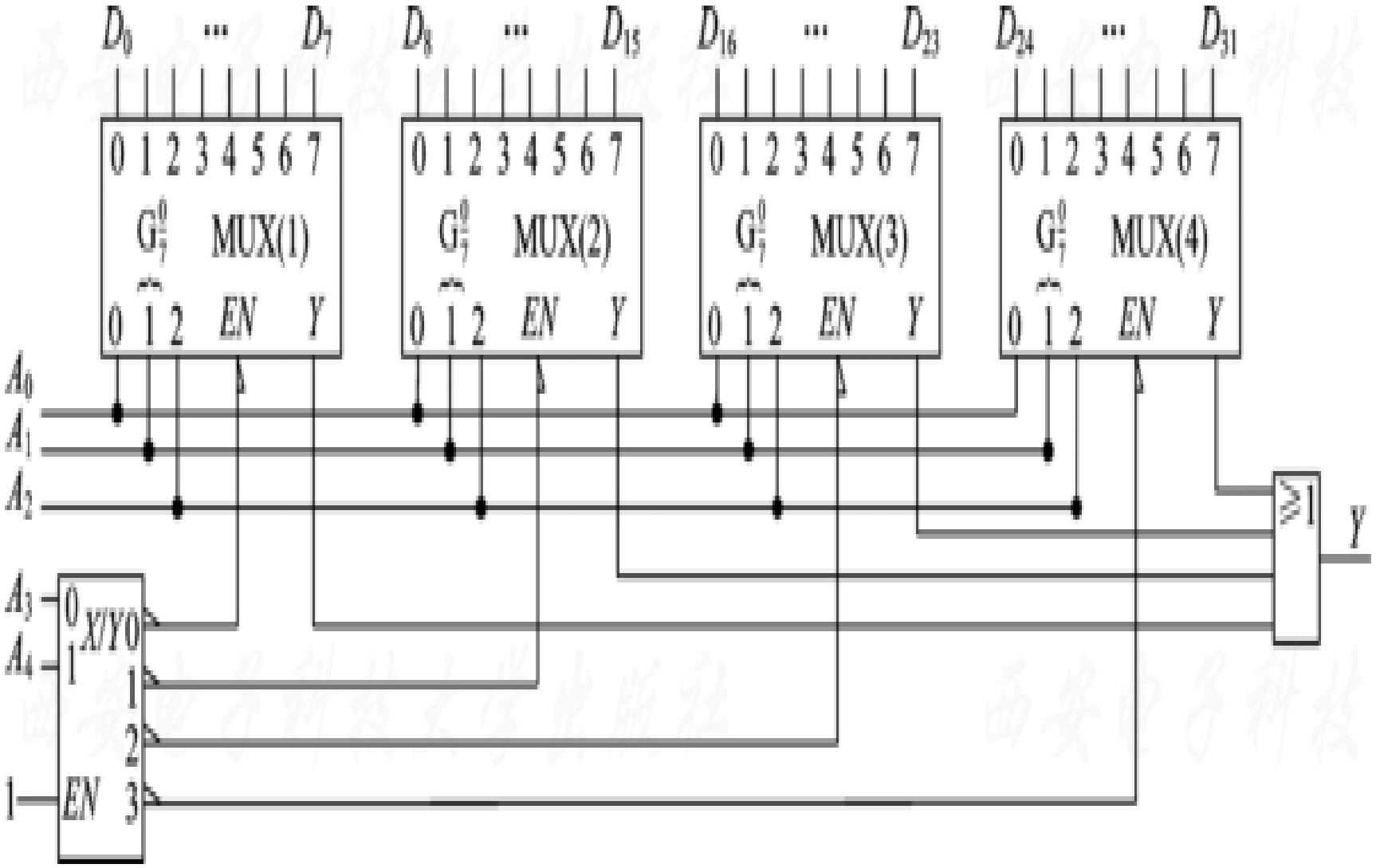


图 4.4.37　用译码器实现八选一扩展成三十二选一的电路

方案二：采用四选一数据选择器实现。

用 1 片四选一数据选择器和 4 个八选一芯片，4 个数据选择器始终处于选通状态，每个数据选择器在 $A_2A_1A_0$ 的控制下实现输入数据的八选一。

4 个数据选择器的输出又由 1 片四选一数据选择器在 A_4A_3 的控制下选择输出。

实现电路如下图所示：

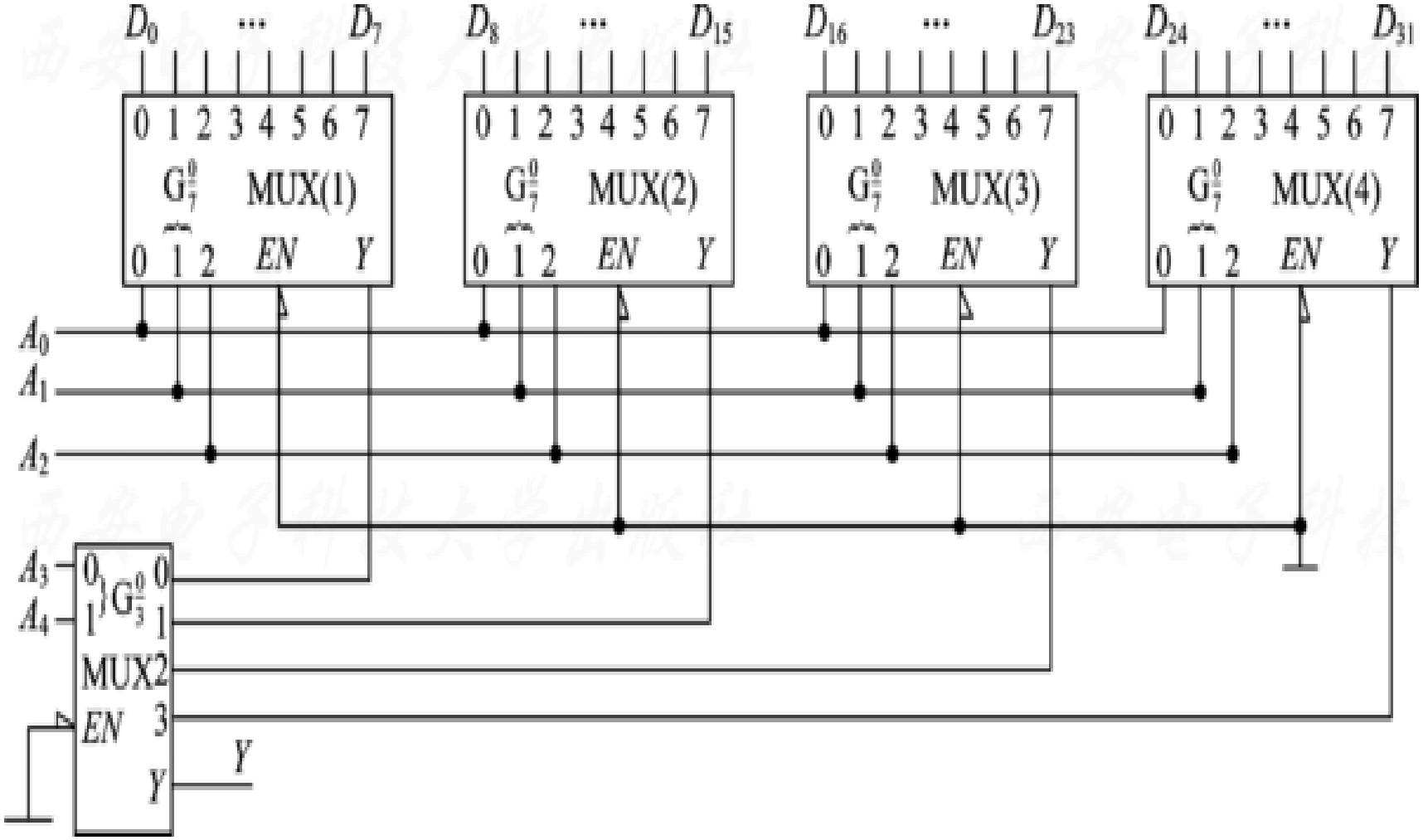


图 4.4.38　用数据选择器实现八选一扩展成三十二选一的电路

2）利用数据选择器实现逻辑函数

【例 4.4.8】分析下图所示电路，写出输出端 X、Y 的真值表，说明电路的逻辑功能。

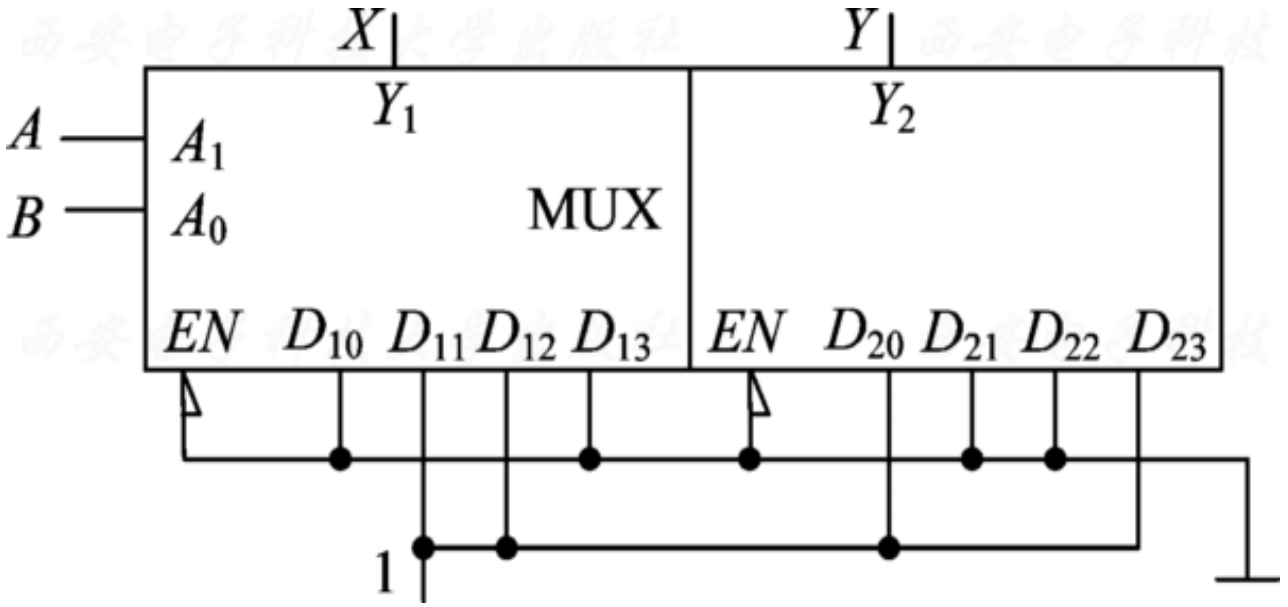


图 4.4.39　例 4.4.8 电路图

解答：根据图，可列出电路的真值表如下：

表 4.4.9　图 4.4.40 真值

A	B	X	Y
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

再根据真值表可以得到电路的输出逻辑函数：

$$X = \overline{A}B + A\overline{B},\quad Y = \overline{A}\,\overline{B} + AB$$

该电路中，X 实现了异或；Y 实现了同或。

【例 4.4.9】用八选一数据选择器实现函数：

$$F = A\overline{C} + A\overline{B} + \overline{B}C$$

解答：首先画出逻辑函数 F 和八选一选择器的卡诺图，如下图所示：

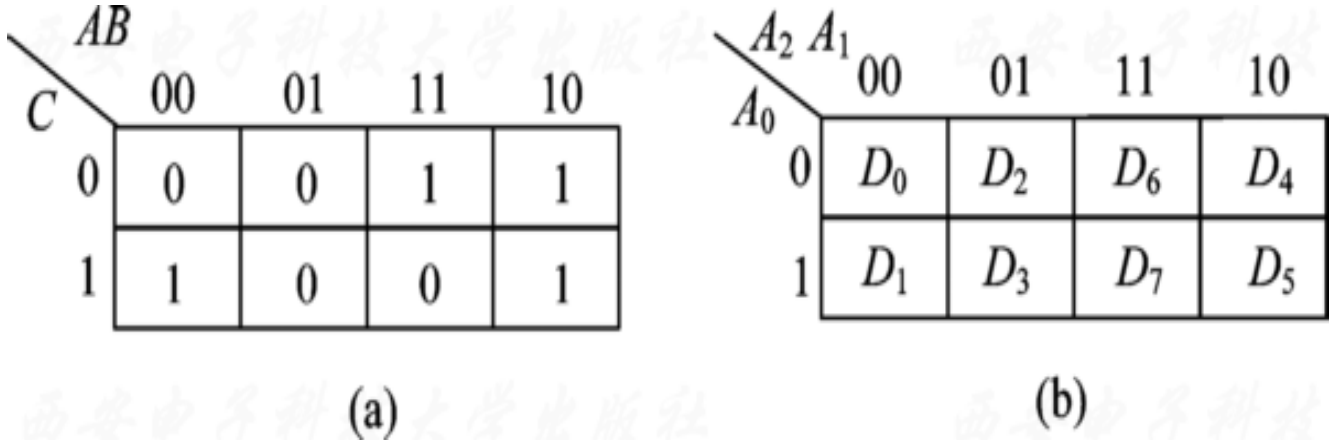


图 4.4.40　逻辑函数和八选一选择器卡诺图

通过对两个卡诺图进行比较发现，当 A、B、C 分别连接数据选择器的地址输入端 $A_2A_1A_0$ 时，数据选择器的数据输入端应分别为：

$D_0=D_2=D_3=D_7=0$，

$D_1=D_4=D_5=D_6=1$

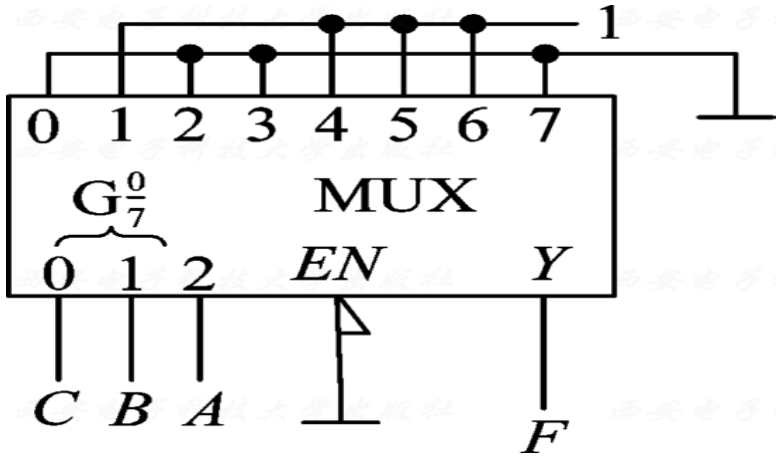


图 4.4.41　例 4.4.9 的逻辑电路图

用八选一数据选择器实现函数 F 的逻辑电路如图所示：

具有 ***N*** 个地址输入的数据选择器，可以实现 ***N*** 个变量的函数：只需要将输入变量加到选择器的地址端，选择器的数据输入端按卡诺图小方格中最小项的取值对应相连即可。

【例 4.4.10】用四选一数据选择器实现：

$$F = A\overline{C} + A\overline{B} + \overline{B}C$$

解答：用两地址端的数据选择器实现三变量

逻辑函数，即地址个数＜变量个数时，可以采用两种方法：**扩展法和降维法**。

扩展法实现函数 F 的方法是先将四选一数据选择器扩展为八选一，然后再实现逻辑函数，其电路如下图所示：

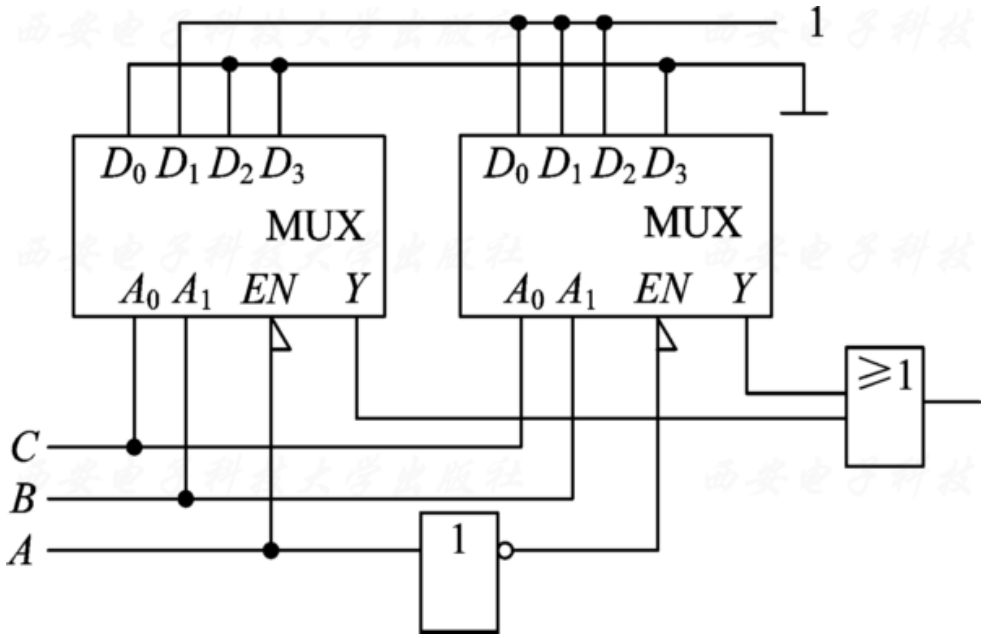


图 4.4.42　扩展法实现 3 变量逻辑函数

降维法的实质是将输入变量的数量减少，即

将三输入变量卡诺图转换为两变量表示，卡诺图的变量数称为该图的维数，将被减掉的变量填入相应的小方格中，称做**记图变量**。

下图所示是将函数 F 的三变量卡诺图变换为两变量卡诺图的过程：

$$F = A\overline{C} + A\overline{B} + \overline{B}C$$

$$= A\overline{C} + A\overline{B}C + A\overline{B}\overline{C} + A\overline{B}C + \overline{A}\overline{B}C$$

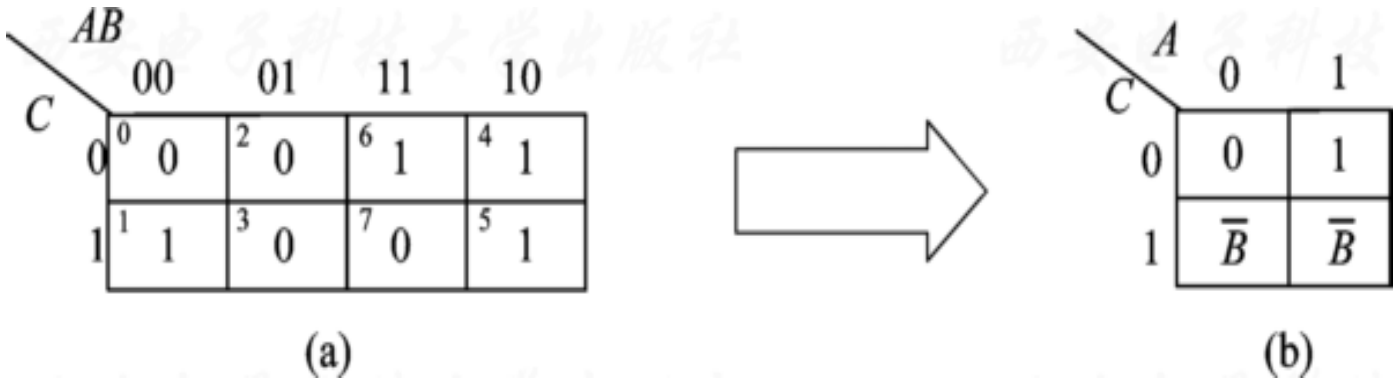


图 4.4.43　例 4.4.10 的卡诺图及降维后的卡诺图

(a) 三变量卡诺图；　(b) 降维后的两变量卡诺图

$$F = A\overline{C} + A\overline{B}C + \overline{A}\,\overline{B}C + \overline{A}\,\overline{C}\cdot 0$$

$$F = \overline{A}\,\overline{C}\cdot 0 + \overline{A}C\overline{B} + A\overline{C}\cdot 1 + AC\overline{B}$$

将降维后的卡诺图与四选一电路的卡诺图进行对比，四选一数据选择器的 A_1A_0 分别接变量 AC，数据输入端分别为 $D_0=0$，$D_1=\overline{B}$，$D_2=1$，$D_3=\overline{B}$，即可实现逻辑函数 F，电路如下：

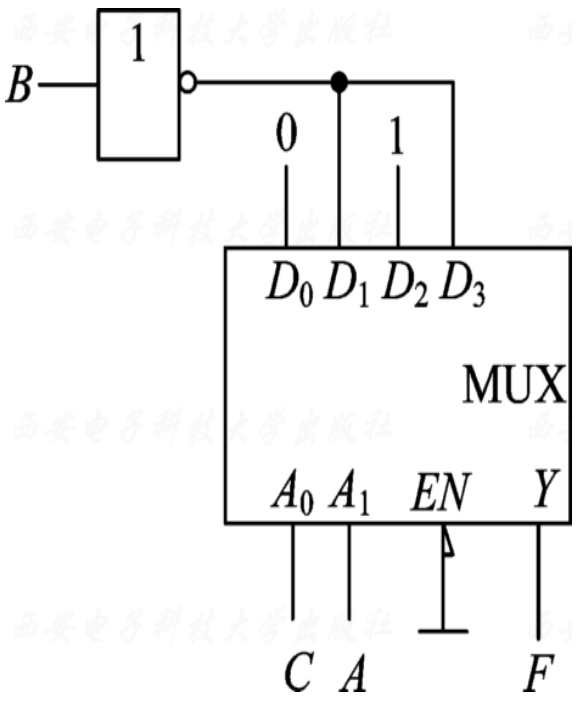


图 4.4.44　用四选一电路实现例 4.4.10 的逻辑函数

（二）数据分配器

在数据传送中，有时需要将某一路数据分配到不同的数据通道上，实现这种功能的电路称为数据分配器，也称多路分配器。

下面是一个四路数据分配器的示意图：

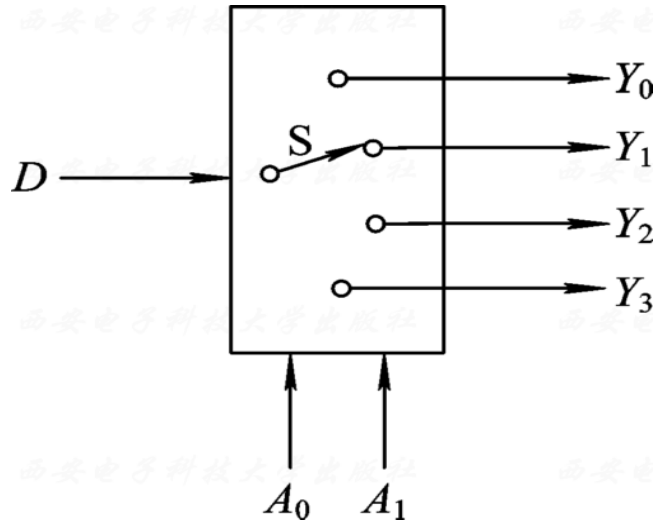


图 4.4.47　四路数据分配器功能示意图

图中，S 相当于一个由信号 A_1A_0 控制的单刀多掷输出开关，输入数据 D 在地址输入 A_1A_0 的控制下，传送到输出 Y_0～Y_3 的不同数据通道上。

市场上并没有专用的数据分配器器件，实际使用中，**通常用译码器来实现数据分配**的功能。

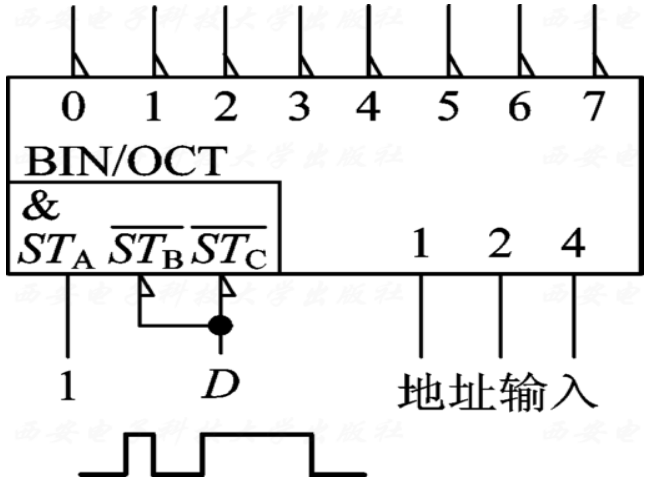


图 4.4.48　用 3-8 译码器实现数据分配

五、数值比较器

1. 数值比较器工作原理

首先，设计一个 **1 位数值比较器**电路，该电路能够对两个 1 位的二进制数据进行比较，并给出比较的结果。

输入变量：A、B，输出变量：$F_{A>B}$、$F_{A=B}$、$F_{A<B}$

表 4.4.10　1 位数值比较器的真值

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

由真值表写出逻辑表达式：

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

根据逻辑表达式画出的逻辑电路如下：

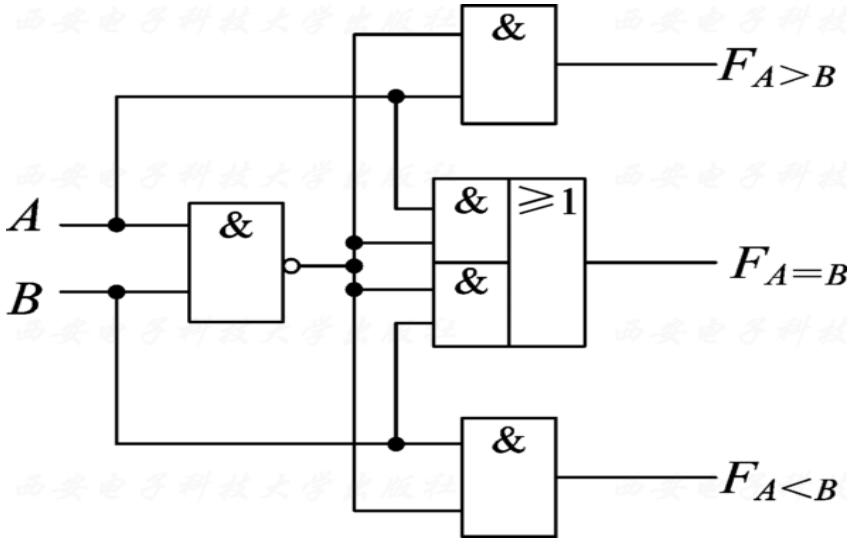


图 4.4.50　1 位数值比较器逻辑电路图

若要实现 **4 位数值比较器**，可以通过从高位到低位依次比较来实现。设两个 4 位数据分别为 $A_3A_2A_1A_0$ 和 $B_3B_2B_1B_0$，真值表如下：

表 4.4.11　4 位数值比较器的真值表

输　入							输　出		
$A_3\,B_3$	$A_2\,B_2$	$A_1\,B_1$	$A_0\,B_0$	$A>B$	$A<B$	$A=B$	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
$A_3>B_3$	××	××	××	××	××	××	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
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	××	××	××	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

为了能够实现更多位的级联，其中还增加了低位比较结果输入信号 ***A>B***、***A<B*** 和 ***A=B***。4 位比

较器的电路和逻辑符号如下：

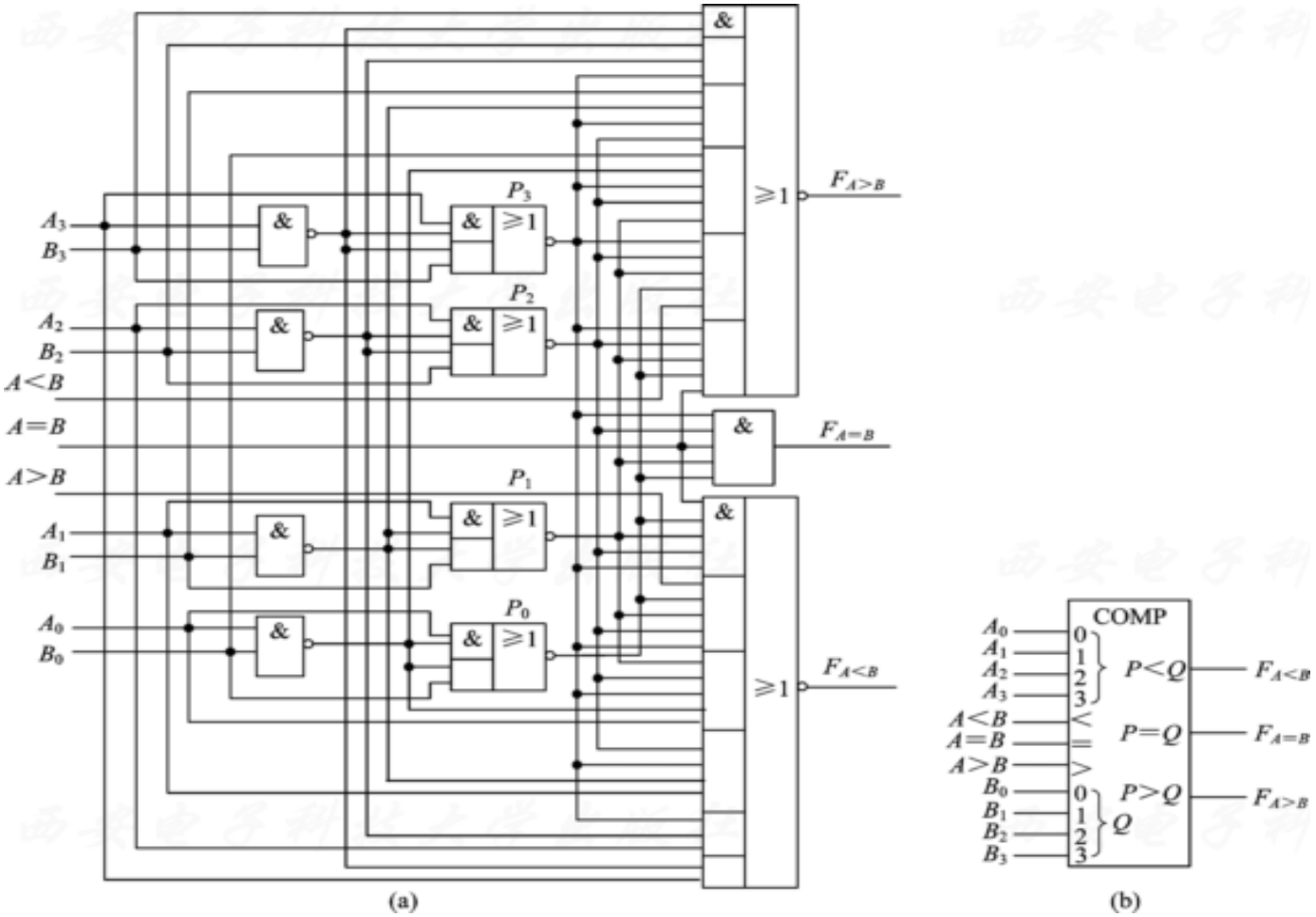


图 4.4.51　4 位比较器的电路图和逻辑符号

(a) 电路图；　(b) 逻辑符号

图(a)中**与或非门** P_3、P_2、P_1 和 P_0 的输出分别表示 $A_3 \odot B_3$、$A_2 \odot B_2$、$A_1 \odot B_1$ 和 $A_0 \odot B_0$。

2. 数据比较器的应用

两片 4 位数值比较器扩展为 8 位数值比较器的电路如下：

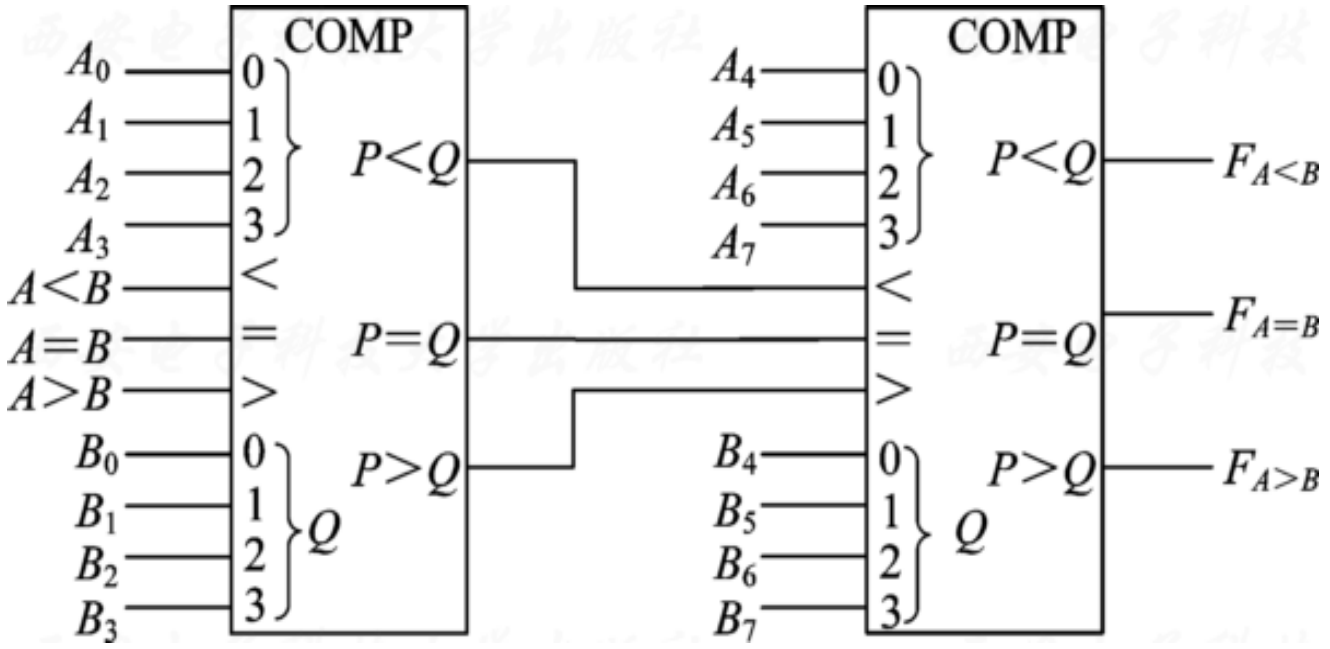


图 4.4.52　8 位数值比较器扩展

在连接时，应**将低位片的比较结果送入高位片的级联输入端**，使之参与高位片的比较。

六、奇偶产生/校验电路

1. 奇偶产生/校验电路工作原理

奇(或偶)校验码具有 1 位检错能力，其基本思想是通过在原数据信息后增加 1 位奇校验位(或偶校验码)，形成奇(或偶)校验码。

发送端发送奇(或偶)校验码，接收端对收到的奇(或偶)校验码中的数据位采用同样的方法产生

新的校验位，并将该校验位与收到的校验位进行比较：若一致则判定数据正确，否则判定数据错误。

具有产生检验码和奇偶检验功能的电路称为奇偶产生/校验器。

奇偶校验码包含 n 位数据位和 1 位校验位：对于奇校验码而言，其数据位加校验位后“1”的个数是奇数；对于偶校验码而言，数据位加校验

位后“1”的个数是偶数。

表 4.4.12　偶校验真值表

数据位				校验位	数据位				校验位
D_3	D_2	D_1	D_0	P	D_3	D_2	D_1	D_0	P
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

校验位 P 的逻辑表达式：$P = D_3 \oplus D_2 \oplus D_1 \oplus D_0$

生成校验位 P 的电路如下：

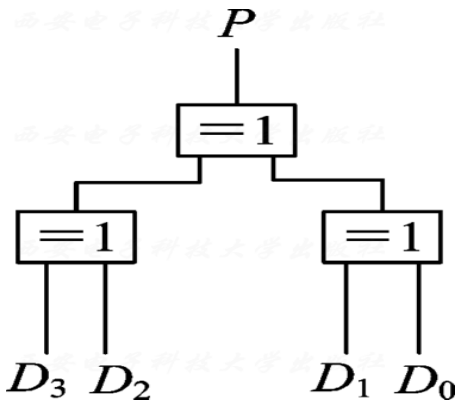


图 4.4.54　校验位产生电路

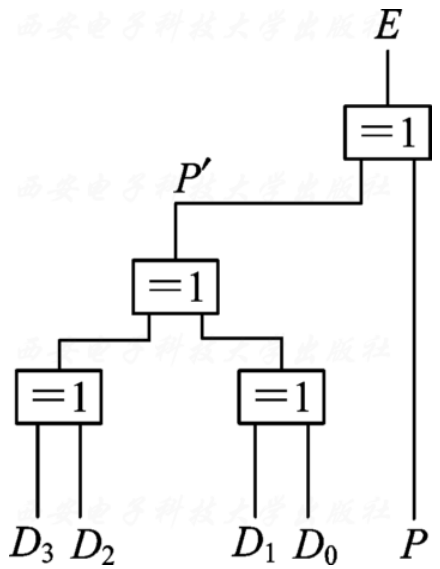


图 4.4.55　偶校验电路

为了检验所传送的数据位及偶校验位是否正确，还应设计偶校验检测器。根据接收的数据位产生校验位 P'与收到的校验位 P 进行比较就实现了校验功能，电路如上。

奇偶校验电路的逻辑符号如下：

图 4.4.56　奇偶校验单元逻辑符号

(a) 奇校验单元; (b) 偶校验单元

2. 奇偶产生/校验电路的应用

这里以 CT74180 器件为例，说明奇偶电路的应用。

该器件具有奇偶校验位产生和校验两种功能，其逻辑符号和真值表如下：

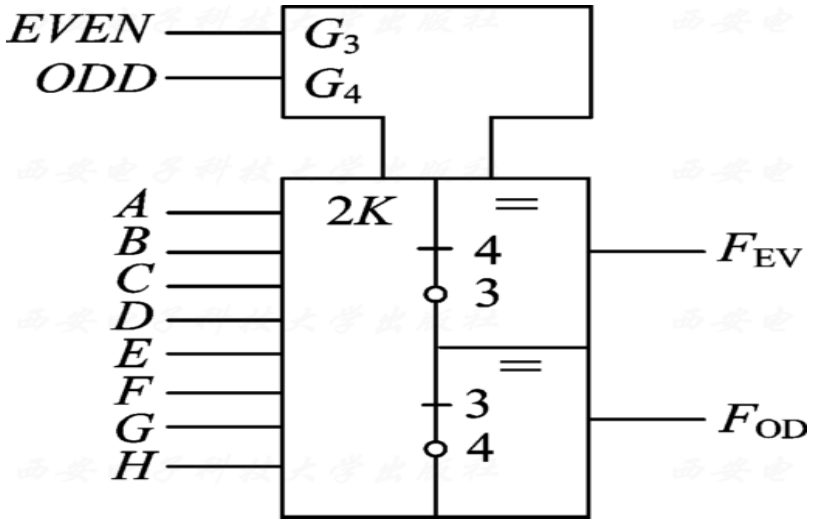


图 4.4.57　CT74180 的逻辑符号

表 4.4.13　CT74180 的功能表

输　　入			输　　出	
$A\sim H$ 中 1 的个数	$EVEN$	ODD	F_{EV}	F_{OD}
偶数	1	0	1	0
	0	1	0	1
奇数	1	0	0	1
	0	1	1	0
×	1	1	0	0
	0	0	1	1

各输入信号的含义如下：

- *A*～*H*：数据输入端
- *ODD*：奇校验控制输入端
- *EVEN*：偶校验控制输入端

ODD 和 *EVEN* 是一对互补输入端，不可以同时为 0 或 1。

各输出信号的含义如下：

- F_{OD}：奇校验输出端；• F_{EV}：偶校验输出端

F_{OD} 和 F_{EV} 是一对互补输出端，不可以同时为 0 或 1。

一个带奇偶校验功能的 8 位数据传输系统如下：

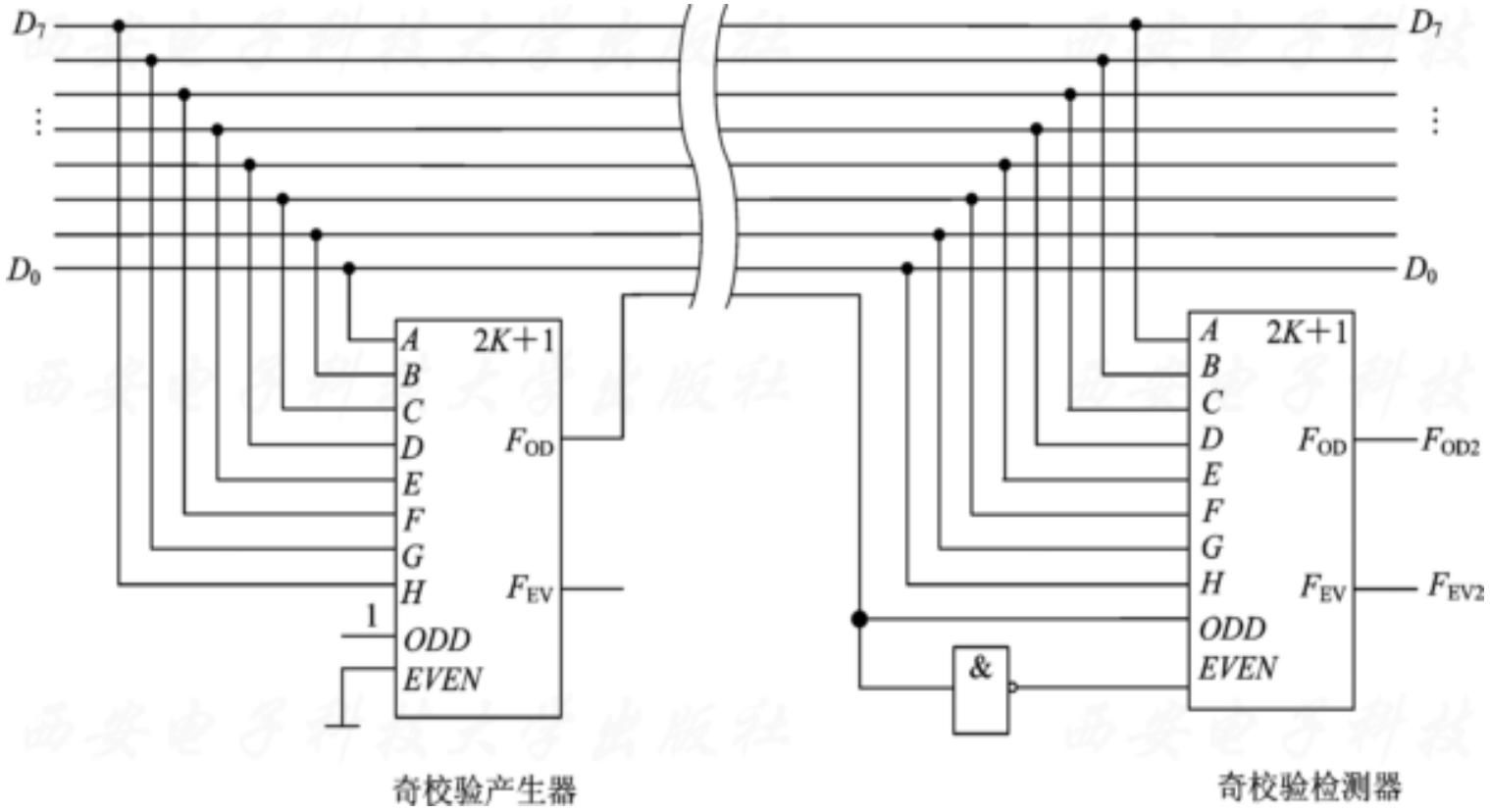


图 4.4.58　奇偶校验在数据传输系统的应用