

4 组合逻辑电路

4.1 组合逻辑电路的分析

4.2 组合逻辑电路的设计

4.3 组合逻辑电路中的竞争和冒险

4.4 常用组合逻辑电路模块

4.5 组合逻辑的可编程电路实现

4.6 用Verilog HDL描述组合逻辑电路

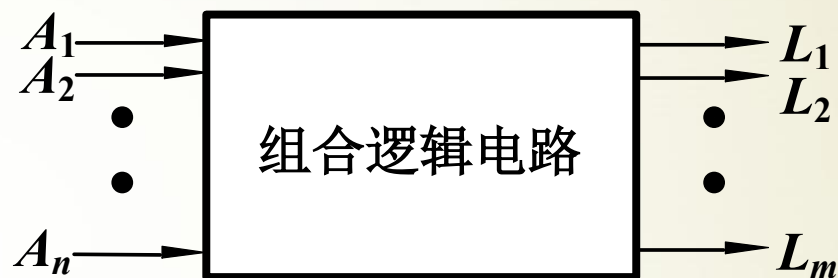
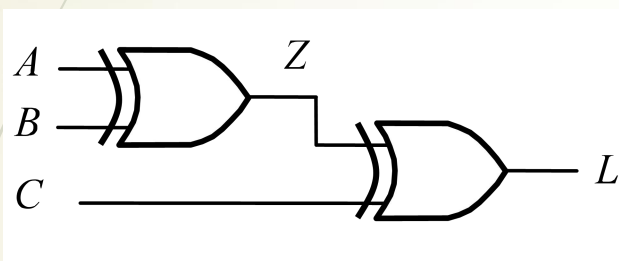
教学基本要求

- 1.熟练掌握组合逻辑电路的分析方法和设计方法
- 2.掌握编码器、译码器、数据选择器、数值比较器和加法器的逻辑功能及其应用；
- 3.学会阅读器件的功能表，并能根据设计要求完成电路的正确连接。
- 4.掌握可编程逻辑器件的表示方法,会用PLD实现组合逻辑电路

4.1 组合逻辑电路分析

4.1.1 组合逻辑电路的定义

组合逻辑电路的一般框图



$$L_i = f(A_1, A_2, \dots, A_n) \quad (i=1, 2, \dots, m)$$

结构特征:

- 1、输出、输入之间没有反馈延迟通路,
- 2、不含记忆单元

工作特征:

组合逻辑电路工作特点:在任何时刻, 电路的输出状态只取决于同一时刻的输入状态而与电路原来的状态无关。

4.1.2 组合逻辑电路的分析方法

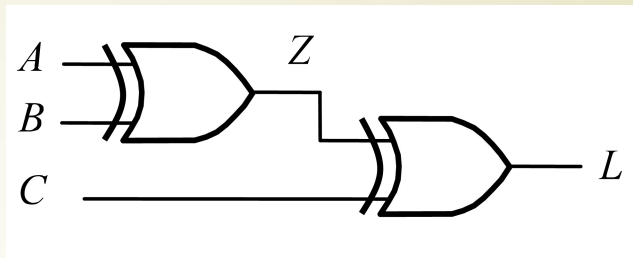
一. 组合逻辑电路分析

根据已知逻辑电路，经分析确定电路的逻辑功能。

二. 组合逻辑电路的分析步骤：

- 1、由逻辑图写出各输出端的逻辑表达式；
- 2、化简和变换逻辑表达式；
- 3、列出真值表；
- 4、根据真值表或逻辑表达式，经分析最后确定其功能。

三. 组合逻辑电路的分析举例



例1 分析如图所示逻辑电路的功能。

解：1.根据逻辑图写出输出函数的逻辑表达式

$$L = Z \oplus C$$

$$= (A \oplus B) \oplus C$$

$$= A \oplus B \oplus C$$

2. 列写真值表。

3. 确定逻辑功能：

输入变量取值中有奇数个1时， L 为1，否则 L 为0，电路具有为奇校验功能。

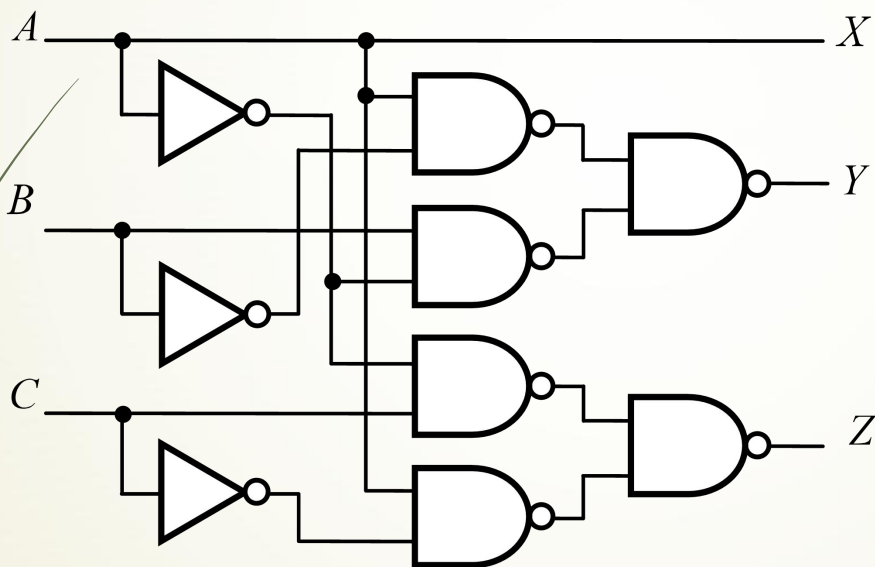
A	B	C	$Z = A \oplus B$	$L = (A \oplus B \oplus C)$
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1

奇校验由奇校验位产生电路和奇校验电路组成。见习题

如要实现偶校验，电路应做何改变？

例2 试分析下图所示组合逻辑电路的逻辑功能。

解：1. 根据逻辑电路写出各输出端的逻辑表达式，并进行化简和变换。



$$X = A$$

$$Y = \overline{\overline{A}B} \cdot \overline{\overline{A}B}$$

$$Z = \overline{\overline{A}C} \cdot \overline{\overline{A}C}$$

2. 列写真值表

$$X = A$$

$$Y = \overline{\overline{A} \overline{B}} \cdot \overline{\overline{A} \overline{B}} = A \overline{B} + \overline{A} B$$

$$Z = \overline{\overline{A} \overline{C}} \cdot \overline{\overline{A} \overline{C}} = A \overline{C} + \overline{A} C$$

真值表

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

3. 确定电路逻辑功能

这个电路逻辑功能是对输入的二进制码求反码。最高位为符号位，0表示正数，1表示负数，正数的反码与原码相同；负数的数值部分是在原码的基础上逐位求反。

真值表

<i>A</i>	<i>B</i>	<i>C</i>	<i>X</i>	<i>Y</i>	<i>Z</i>
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	0	0

4.2 组合逻辑电路的设计

4.2.1 组合逻辑电路的设计过程

一、组合逻辑电路的设计：根据实际逻辑问题，求出所要求逻辑功能的最简单逻辑电路。

二、组合逻辑电路的设计步骤

- 1、逻辑抽象：根据实际逻辑问题的因果关系确定输入、输出变量，并定义逻辑状态的含义；
- 2、根据逻辑描述列出真值表；
- 3、由真值表写出逻辑表达式；
- 4、简化和变换逻辑表达式，画出逻辑图。

例1 某办公楼空气净化设备是根据三台空气质量检测仪的检测结果进行控制。试设计一个逻辑电路，当有两台或以上检测结果超标时，要求其向控制系统输出高电平，以启动空气净化设备。

解：（1）逻辑抽象。

输入信号： A 、 B 、 C 分别表示三台空气质量检测仪，且检测结果超标时为1，没有超标时为0。

输出信号： L 表示逻辑电路输出，有两台或以上检测结果超标时1，否则为0。

(2) 根据题意列出真值表

输 入			输 出
<i>A</i>	<i>B</i>	<i>C</i>	<i>L</i>
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(3) 写出输出逻辑表达式,并化简

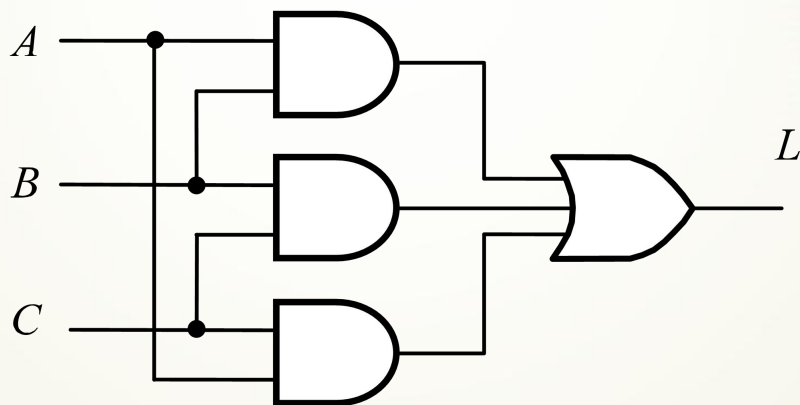
$$L = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$L = AB + AC + BC$$

(4) 根据输出逻辑表达式画出逻辑图。

$$L = AB + AC + BC$$

表达式为最简与或式，用与门和或门实现两级“与-或”结构的最简电路如图。



例2 某董事会有一位董事长和三位董事，就某项议题进行表决，当满足以下条件时决议通过：有三人或三人以上同意；或者有两人同意，但其中一人必须是董事长。试用两输入与非门设计满足上述要求的表决电路。

解：（1）逻辑抽象。

假设：用变量 A 、 B 、 C 、 D 表示输入， A 代表董事长， B 、 C 、 D 代表董事，1 → 同意，0 → 不同意；
用 L 表示输出， $L=1$ → 决议通过， $L=0$ → 不通过。

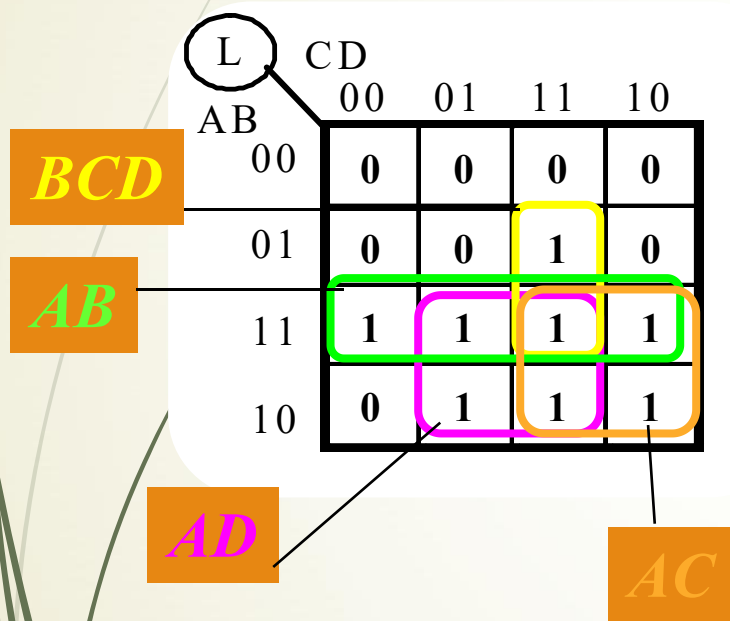
（2）列出真值表；

（3）画出卡诺图，求输出 L 的表达式；

（4）画出由与非门组成的逻辑电路。

(2) 列出真值表

(3) 画出输出L的卡诺图并化简得



输 入				出	输 入				出
A	B	C	D	L	A	B	C	D	L
0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	0	1	0	1
0	0	1	1	0	1	0	1	1	1
0	1	0	0	0	1	1	0	0	1
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	1

$$L = AB + AC + AD + BCD$$

(4) 画出由与非门组成的逻辑电路。

$$L = AB + AC + AD + BCD$$

画出由与非门组成的逻辑电路

$$L = \overline{\overline{AB} \cdot \overline{AC} \cdot \overline{AD} \cdot \overline{B} \cdot \overline{CD}}$$

例3 某工厂有A、B、C三台设备，其中A和B的功率相等，C的功率是A的两倍。这些设备由X和Y两台发电机供电，发电机X的最大输出功率等于A的功率，发电机Y的最大输出功率是X的三倍。要求设计一个逻辑电路，能够根据各台设备的运转和停止状态，以最节约能源的方式启、停发电机。

- 解
- (1) 逻辑抽象。
 - (2) 列出真值表；
 - (3) 画出卡诺图, 求输出L；
 - (4) 画出逻辑电路。

(2) 列出真值表

(3) 画出卡诺图, 求输出 L ;

卡诺图 for X :

BC	00	01	11	10
$A=0$	0	0	0	1
$A=1$	1	0	1	0

卡诺图 for Y :

BC	00	01	11	10
$A=0$	0	1	1	0
$A=1$	0	1	1	1

输 入			输 出	
A	B	C	X	Y
0	0	0	0	0
0	0	1	0	1
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

$$X = \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$Y = AB + C$$

(4) 画出逻辑图(略)。

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

$$Y = AB + C$$

例4 试设计一个码转换电路，将4位格雷码转换为自然二进制码。可以采用任何逻辑门电路来实现。

解：(1) 明确逻辑功能，列出真值表。

设输入变量为 G_3 、 G_2 、 G_1 、 G_0 为格雷码，

输出变量 B_3 、 B_2 、 B_1 和 B_0 为自然二进制码。

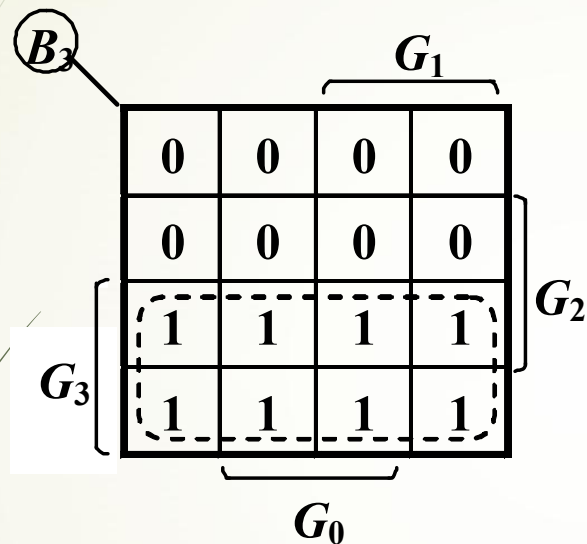
当输入格雷码按照从0到15递增排序时，

可列出逻辑电路真值表

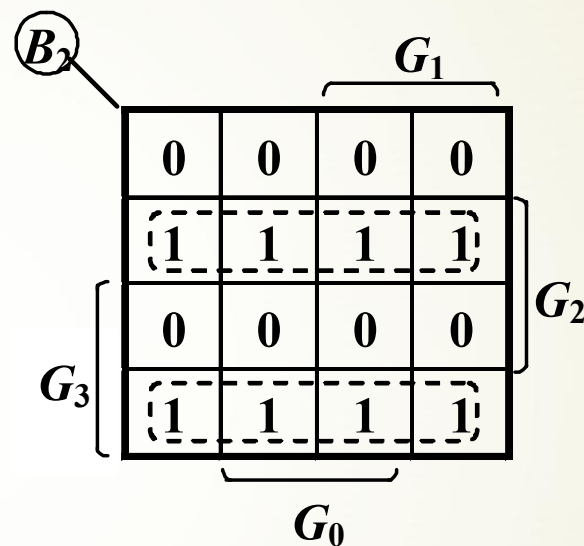
逻辑电路真值表

输 入	输 出	输 入	输 出
$G_3 G_2 G_1 G_0$	$B_3 B_2 B_1 B_0$	$G_3 G_2 G_1 G_0$	$B_3 B_2 B_1 B_0$
0 0 0 0	0 0 0 0	1 1 0 0	1 0 0 0
0 0 0 1	0 0 0 1	1 1 0 1	1 0 0 1
0 0 1 1	0 0 1 0	1 1 1 1	1 0 1 0
0 0 1 0	0 0 1 1	1 1 1 0	1 0 1 1
0 1 1 0	0 1 0 0	1 0 1 0	1 1 0 0
0 1 1 1	0 1 0 1	1 0 1 1	1 1 0 1
0 1 0 1	0 1 1 0	1 0 0 1	1 1 1 0
0 1 0 0	0 1 1 1	1 0 0 0	1 1 1 1

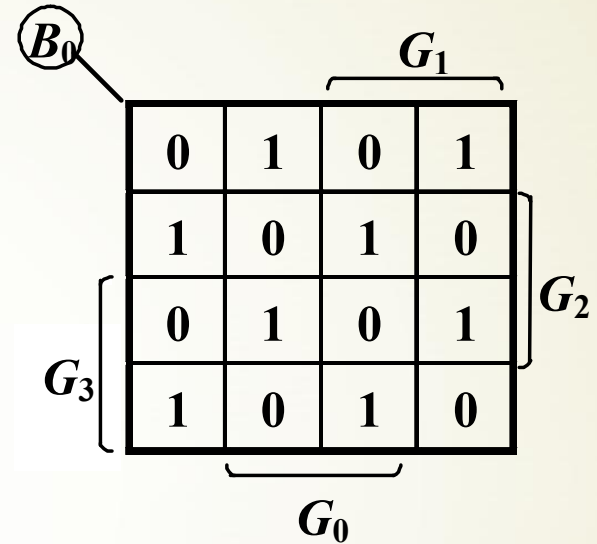
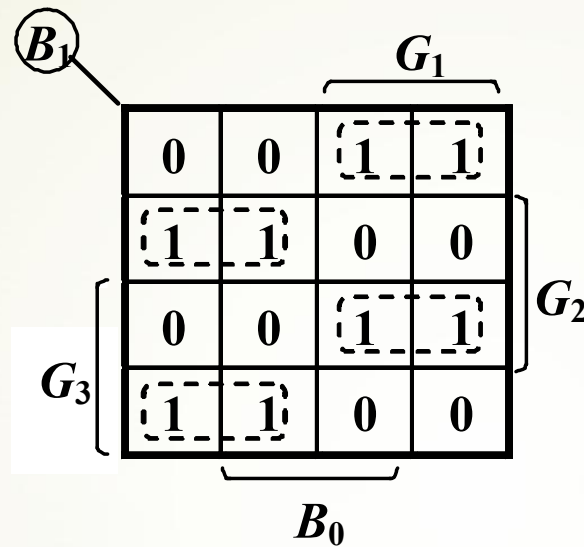
(2) 画出各输出函数的卡诺图，并化简和变换。



$$B_3 = G_3$$



$$B_2 = \bar{G}_3 G_2 + G_3 \bar{G}_2$$



$$B_1 = G_3 \bar{G}_2 \bar{G}_1 + \bar{G}_3 G_2 \bar{G}_1 + G_3 G_2 G_1 + \bar{G}_3 \bar{G}_2 G_1$$

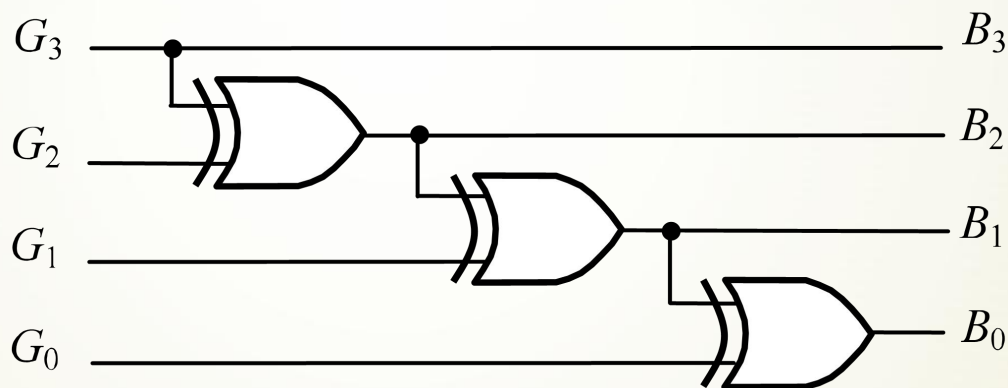
$$= (G_3 \bar{G}_2 + \bar{G}_3 G_2) \bar{G}_1 + \overline{G_3 \bar{G}_2 + \bar{G}_3 G_2} G_1$$

$$= G_3 \oplus G_2 \oplus G_1$$

$$B_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0$$

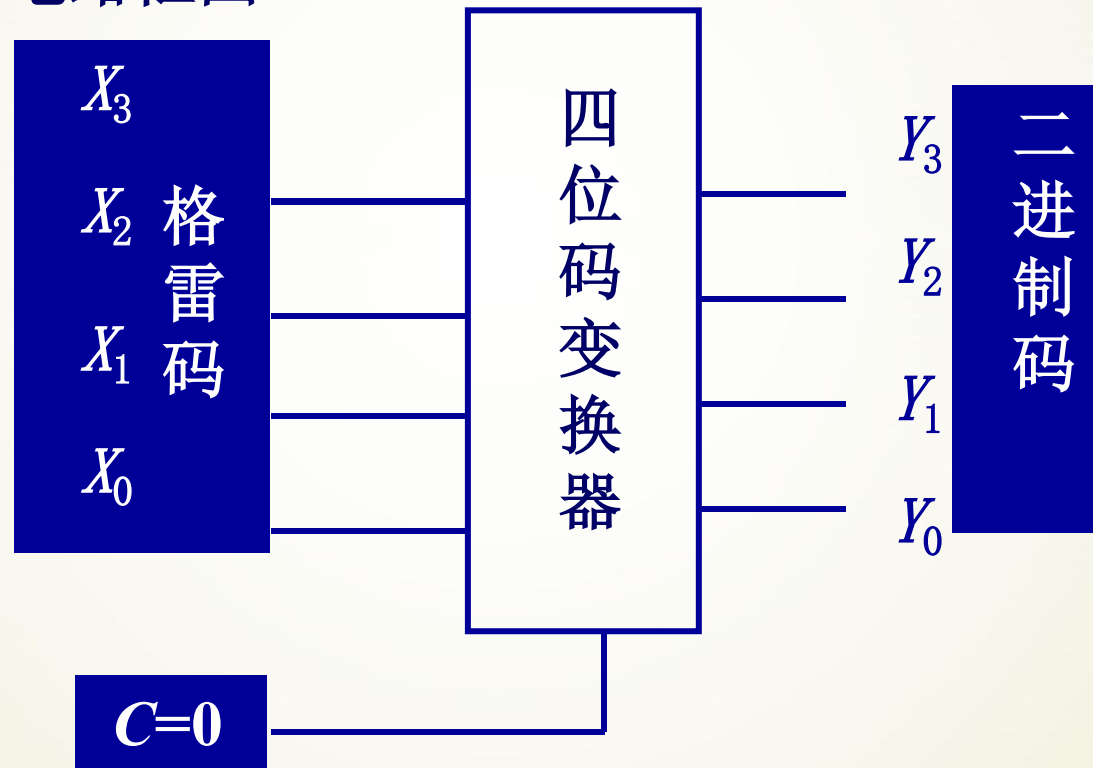
(3) 根据逻辑表达式，画出逻辑图

用异或门代替与门和或门能使逻辑电路比较简单。考虑相同乘积项可以减少门电路数目，降低实现电路的成本。



例5 试设计一可逆的四位码变换器。在控制信号 $C=1$ 时，它将二进制码转换为格雷码； $C=0$ 时，它格雷码将转换为二进制码。

电路框图



$C = 1$

二进制码 $X_3X_2X_1X_0$	格雷码 $g_3g_2g_1g_0$
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

$C = 0$

格雷码 $X_3X_2X_1X_0$	二进制码 $b_3b_2b_1b_0$
0000	0000
0001	0001
0011	0010
0010	0011
0110	0100
0111	0101
0101	0110
0100	0111
1100	1000
1101	1001
1111	1010
1110	1011
1010	1100
1011	1101
1001	1110
1000	1111

C = 1

2、简化和变换逻辑表达式(以 g_3 、 g_2 为例)

二进制码 $X_3X_2X_1X_0$	格雷码 $g_3g_2g_1g_0$
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

$$g_3 = X_3 C$$

g_3 $X_1 X_0$

$X_3 X_2$

	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$$g_2 = (X_3 \oplus X_2) C$$

g_2 $X_1 X_0$

$X_3 X_2$

	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$C = 0$ (以 b_3 、 b_2 为例)

格雷码 $X_3X_2X_1X_0$	二进制码 $b_3b_2b_1b_0$
0000	0000
0001	0001
0011	0010
0010	0011
0110	0100
0111	0101
0101	0110
0100	0111
1100	1000
1101	1001
1111	1010
1110	1011
1010	1100
1011	1101
1001	1110
1000	1111

b_3 $X_1 X_0$

$X_3 X_2$

	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$$b_3 = X_3 \bar{C}$$

b_2 $X_1 X_0$

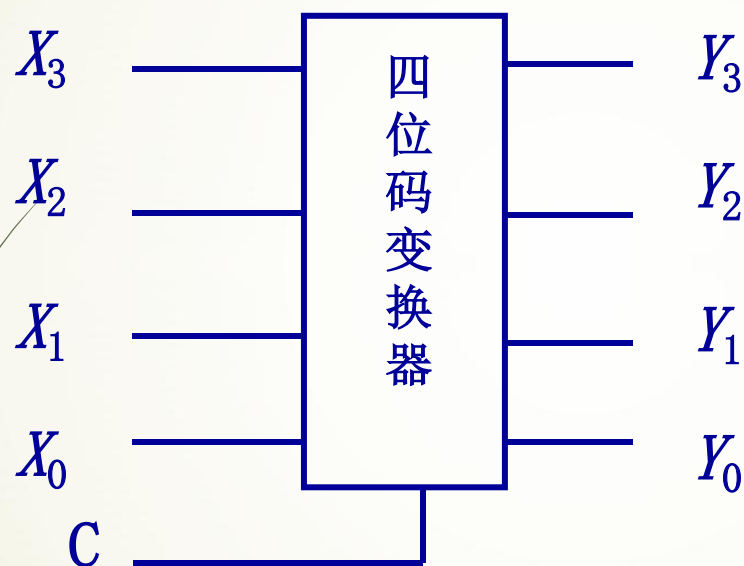
$X_3 X_2$

	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$$b_2 = (X_3 \oplus X_2) \bar{C}$$

$Y_3 = ?$

$Y_2 = ?$



$$\begin{aligned} Y_3 &= g_3 + b_3 \\ &= X_3 C + X_3 \bar{C} \end{aligned}$$

$$\begin{aligned} Y_2 &= g_2 + b_2 \\ &= X_2 C + X_2 \bar{C} \end{aligned}$$

画出逻辑电路图 (略)

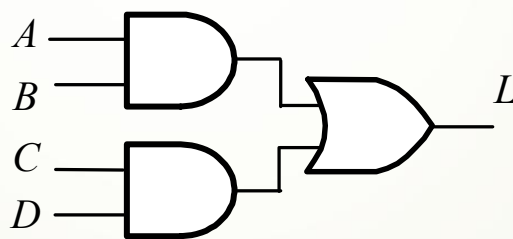
4.2.2 组合逻辑电路的优化实现

用指定芯片中特定资源实现逻辑函数，使电路的成本低并且工作速度快。因此需要对逻辑表达式进行变换，以减少芯片资源的数目和连线。

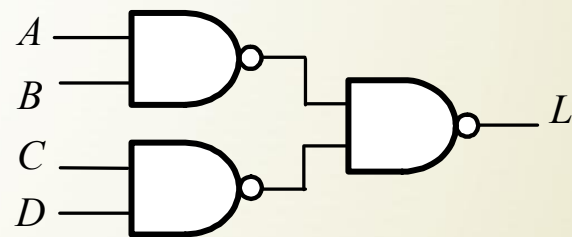
逻辑电路的成本是由电路中总的**逻辑门数量**加上所有逻辑门的**输入端数**来表示。

1、单输出电路

$$\begin{aligned} L &= AB + CD \\ &= \overline{\overline{AB} \cdot \overline{CD}} \end{aligned}$$



(a)与-或结构



(b)与非门结构

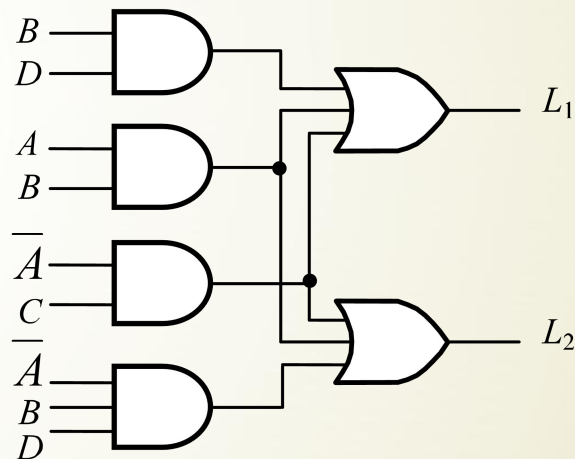
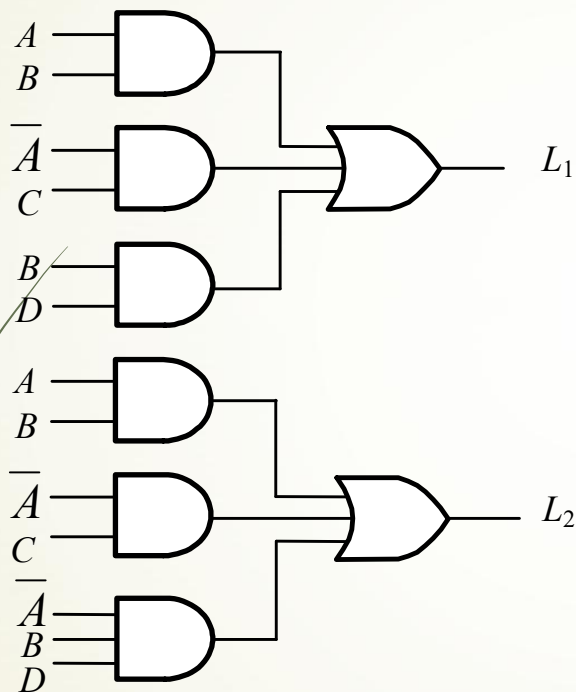
相同输入端的与非门比与门或者或门所用晶体管少，速度快。图(b)电路最优

2、多输出电路

输出多个逻辑函数时需要考虑共享相同乘积项，减少逻辑门数目。

$$L_1 = AB + \overline{A}C + BD$$

$$L_2 = AB + \overline{A}C + \overline{A}BD$$



(a) 如果分别实现两个逻辑函数，需要6个与门和两个或门。**成本为27**。(b) 如果考虑相同乘积项，需要4个与门两个或门，**成本为21**。

3、多级逻辑电路

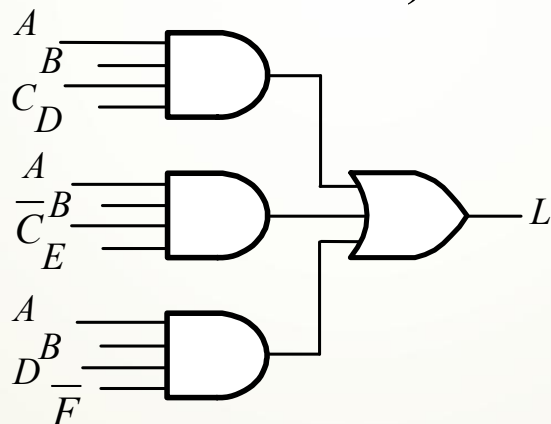
当限定逻辑门输入端数目，则需要进行逻辑变换。

(1) 提取公因子

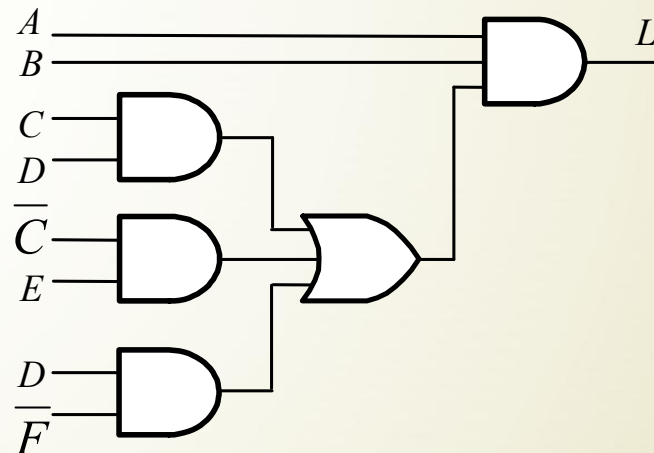
$$L = ABCD + AB\bar{C}E + ABD\bar{F}$$

用与门、或门实现时，限定逻辑门的扇入数为3，需要变换成：

$$L = AB(CD + \bar{C}E + D\bar{F})$$



(a)



(b)

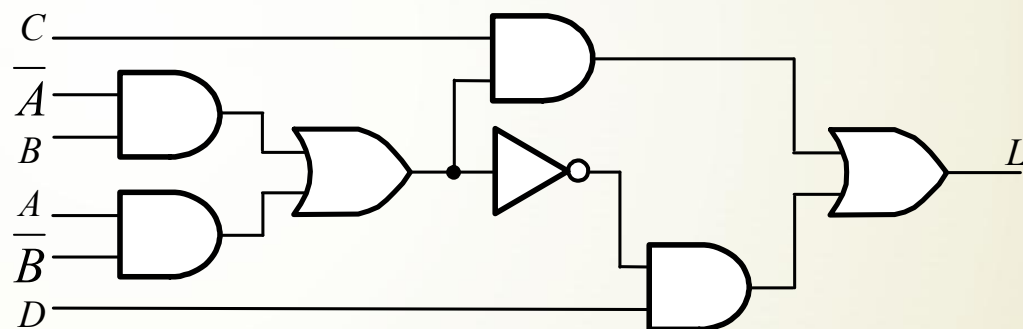
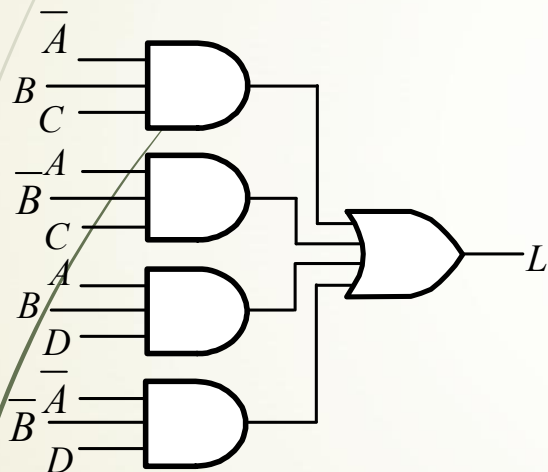
图(a) 电路为2级，成本为19。图(b) 为3级，但成本为17。

(2) 函数分解

$$L = \overline{A}BC + A\overline{B}C + ABD + \overline{A}\overline{B}D$$

用与门、或门实现时，限定逻辑门的扇入数为3，需要变换成：

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



图(a)电路为2级，图(b)为5级。

上述变换方法只适合手工化简，当变量数很多时，优化策略写入程序由计算机完成。

4.3 组合逻辑电路中的竞争冒险

4.3.1 产生竞争冒险的原因

4.3.2 消去竞争冒险的方法

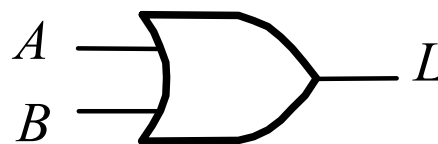
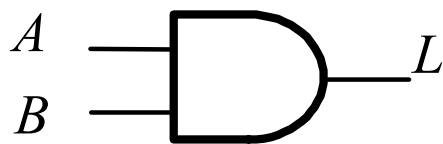
4.3 组合逻辑电路中的竞争冒险

4.3.1 产生竞争冒险的原因

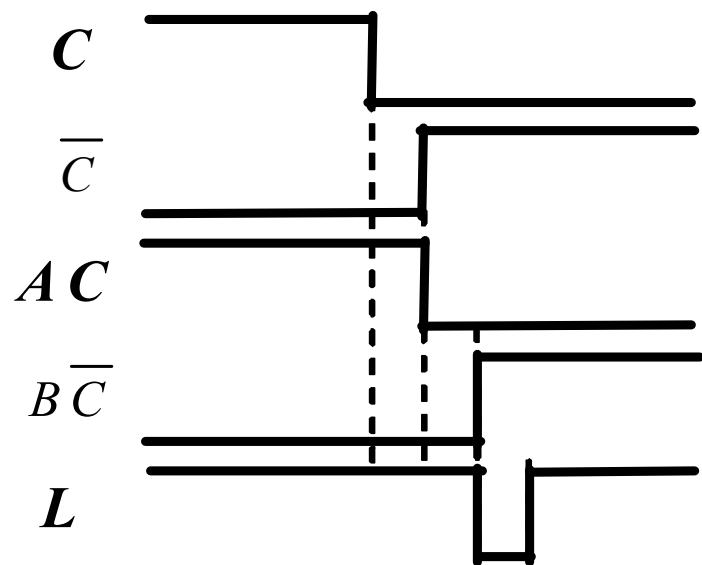
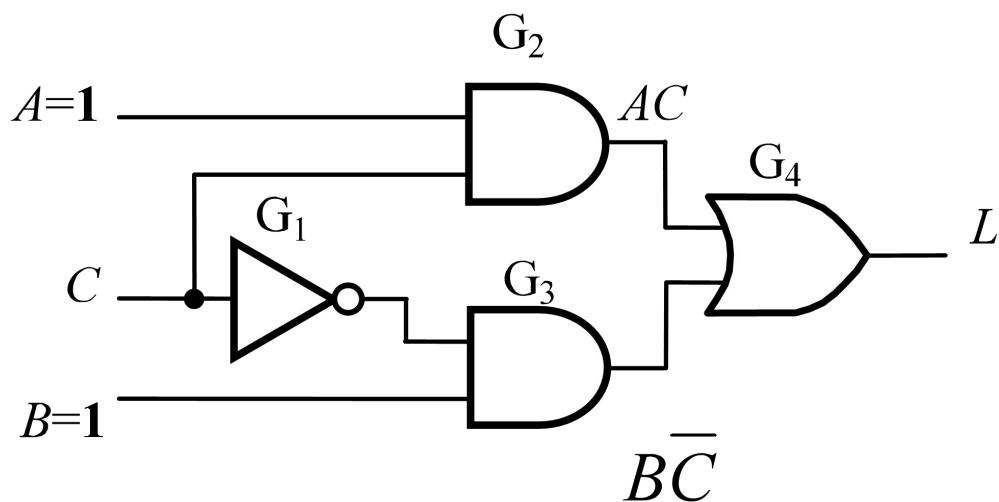
不考虑门的延时时间，且 $B=\bar{A}$

$$L = AB = 0$$

$$L = A + B = 1$$



考虑门的延时时间，且用非门实现 $B=\bar{A}$ 时



竞争: 当一个逻辑门的两个输入端的信号同时向相反方向变化，而变化的时间有差异的现象。

冒险: 两个输入端的信号取值的变化方向是相反时，如门电路输出端的逻辑表达式简化成两个互补信号相乘或者相加，由竞争而可能产生输出干扰脉冲的现象。

4.3.2 消去竞争冒险的方法

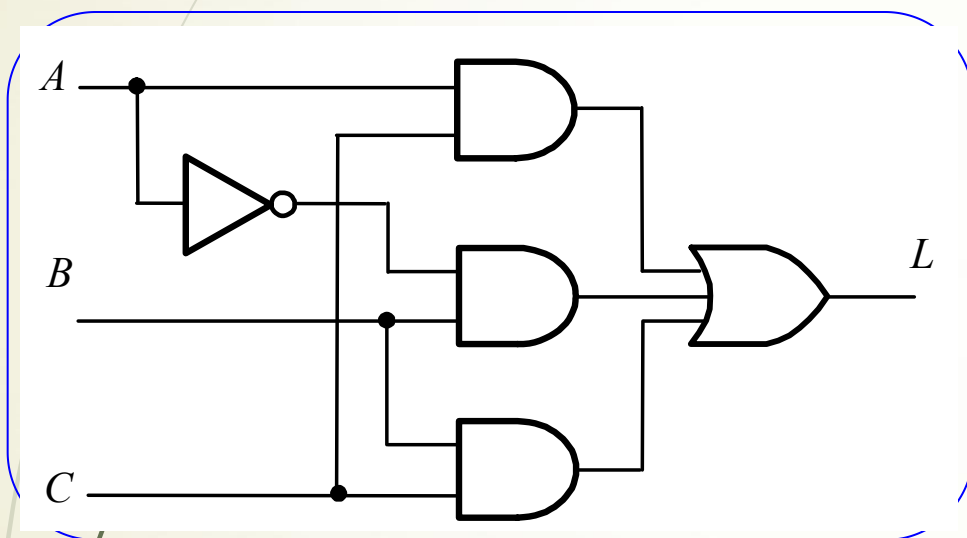
1. 发现并消除互补变量

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

$B = C = 0$ 时

$$F = A\overline{A}$$

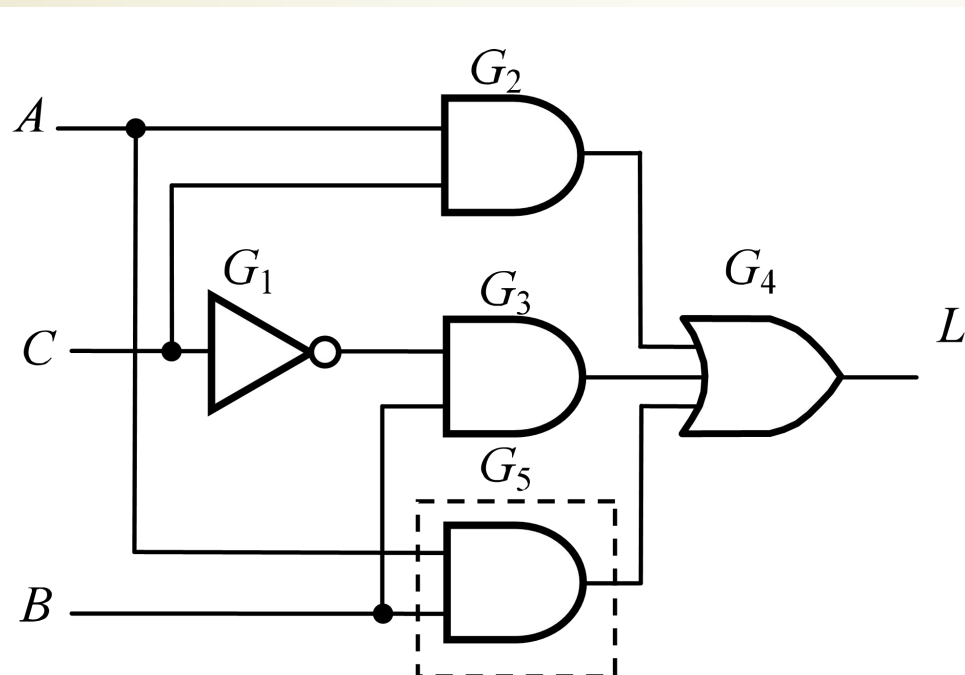
可能出现竞争冒险。



为消掉 $A\overline{A}$ ，变换逻辑函数式为

$$F = AC + \overline{A}B + BC$$

2. 增加乘积项, 避免互补项相加



$$L = AC + B\bar{C}$$

当 $A=B=1$ 时

$$L = C + \bar{C}$$

$$L = AC + B\bar{C}$$

$$L = AC + B\bar{C} + AB$$

当 $A=B=1$ 时, 根据逻辑表达式有

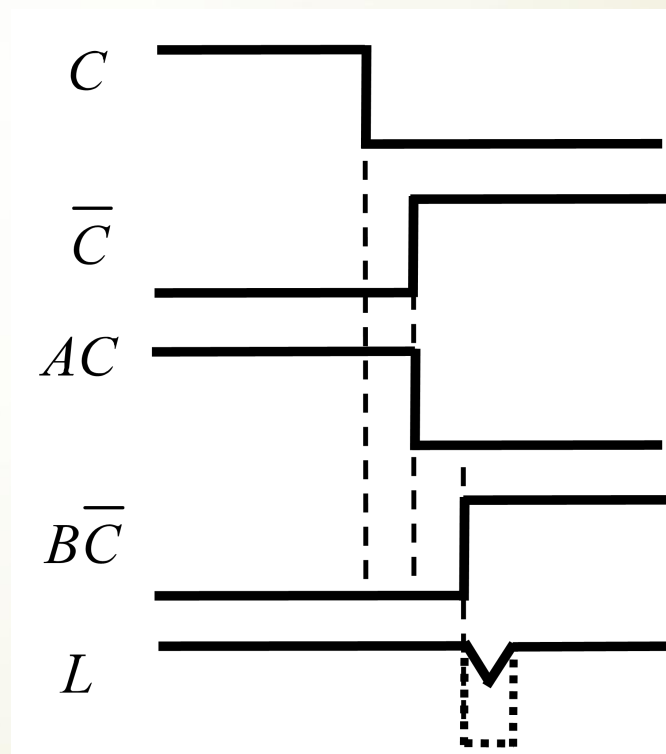
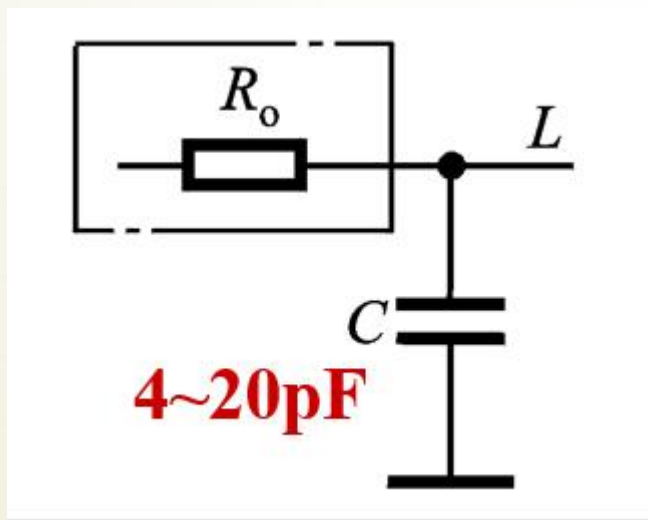
$$L = C + \bar{C} + 1$$

L	B	C				
			00	01	11	10
A	0	0	0	0	0	1
	1	0	1	1	1	1

AB

3. 输出端并联电容器

如果逻辑电路在较慢速度下工作，为了消去竞争冒险，可以在输出端并联一电容器，致使输出波形上升沿和下降沿变化比较缓慢，可对于很窄的脉冲起到平波的作用。



4.4 常用组合逻辑电路模块

4.4.1 编码器

4.4.2 译码器/数据分配器

4.4.3 数据选择器

4.4.4 数值比较器

4.4.5 算术运算电路

4.4 常用组合逻辑电路模块

4.4.1 编码器

1、编码器 (Encoder)的定义与分类

编码：赋予二进制代码特定含义的过程称为编码。

如：8421BCD码中，用1000表示数字8

如：ASCII码中，用1000001表示字母A等

编码器：具有编码功能的逻辑电路。

1、编码器 (Encoder)的定义与分类

编码器的逻辑功能:

能将每一个编码输入信号变换为不同的二进制的代码输出。

- ◆ 如BCD编码器：将10个编码输入信号分别编成10个4位码输出。
- ◆ 如8线-3线编码器：将8个输入的信号分别编成 8个3位二进制数码输出。

1、编码器 (Encoder)的定义与分类

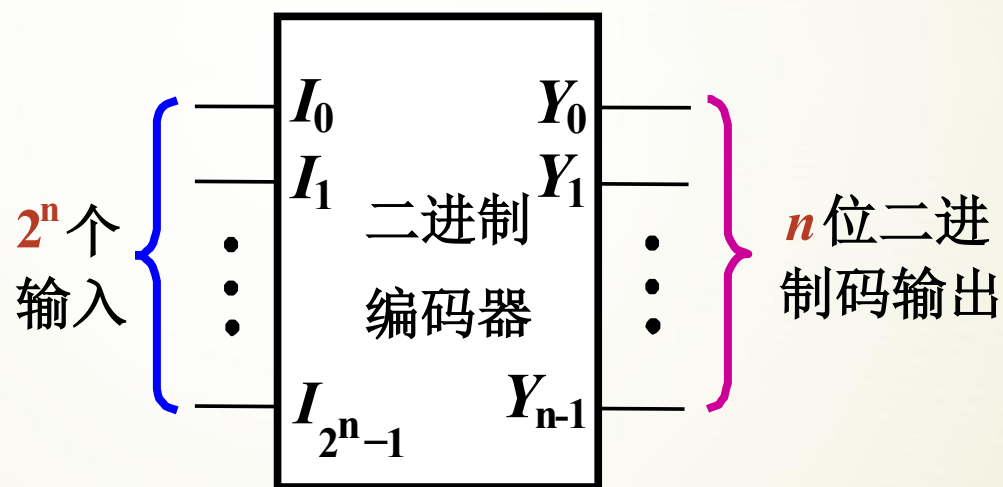
编码器的分类：普通编码器和优先编码器。

- ◆ 普通编码器：任何时候只允许输入一个有效编码信号，否则输出就会发生混乱。
- ◆ 优先编码器：允许同时输入两个以上的有效编码信号。当同时输入几个有效编码信号时，优先编码器能按预先设定的优先级别，只对其中优先权最高的一个进行编码。

2、编码器的工作原理

普通二进制编码器

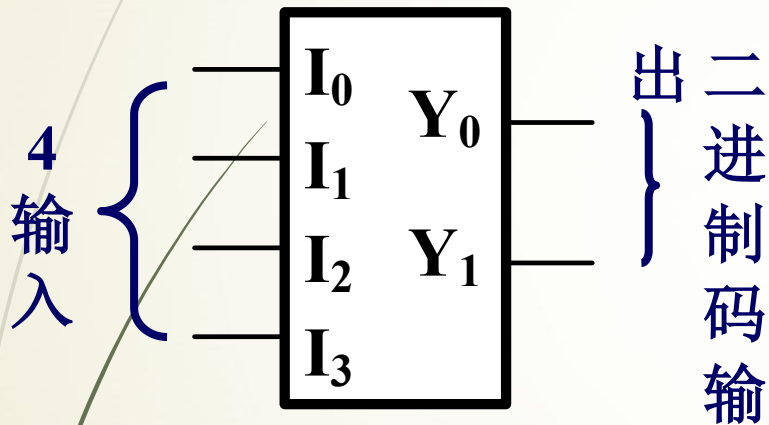
二进制编码器的结构框图



2、普通编码器

(1) 4线—2线普通二进制编码器 (设计)

(a) 逻辑框图



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

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

该表达式是否可以再简化？

(2) 逻辑功能表

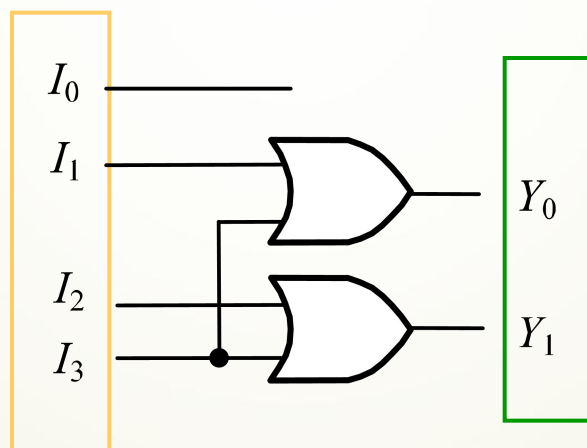
I_0	I_1	I_2	I_3	Y_1	Y_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

编码器的输入为高电平有效。

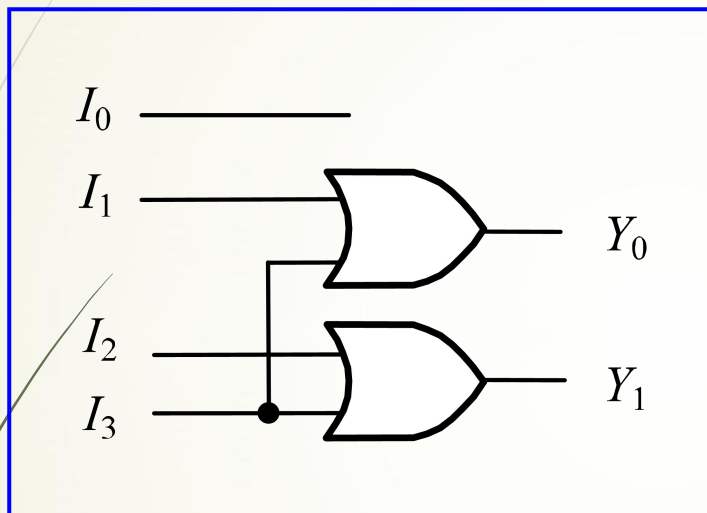
上述是将输入的其它12种组合对应的输出看做0。如果看做无关项，则表达式为

$$Y_1 = I_2 + I_3$$

$$Y_0 = I_1 + I_3$$



若有2个以上的输入为有效信号？



当只有 I_3 为1时，

$$Y_1 Y_0 = ? \quad Y_1 Y_0 = 11$$

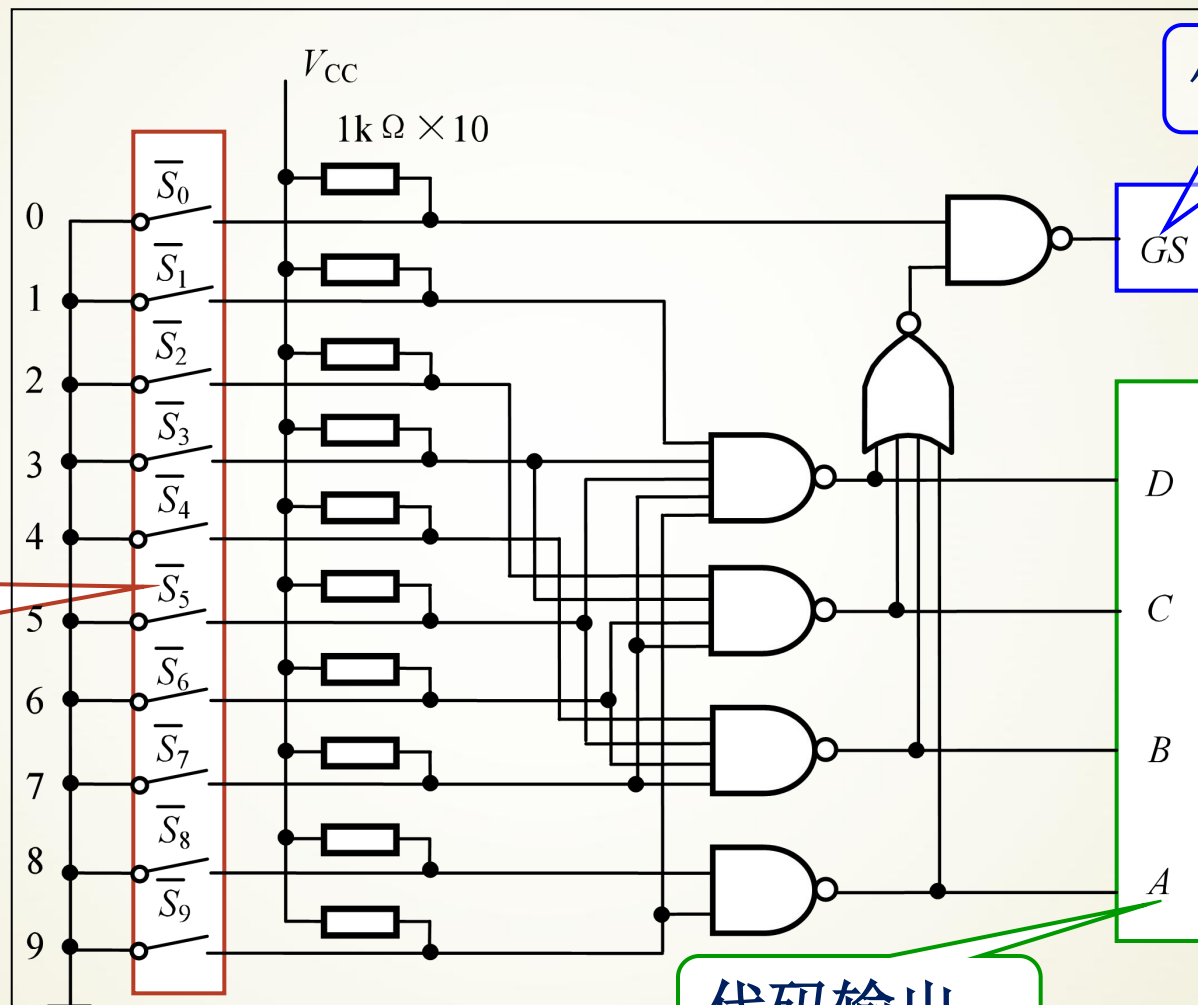
$I_1 = I_2 = 1$, $I_0 = I_1 = 0$ 时，

$$Y_1 Y_0 = ? \quad Y_1 Y_0 = 11$$

无法输出有效编码。

结论：普通编码器不能同时输入两个以上的有效编码信号

(2) 键盘输入8421BCD码编码器（分析）



键盘输入8421BCD码编码器功能表

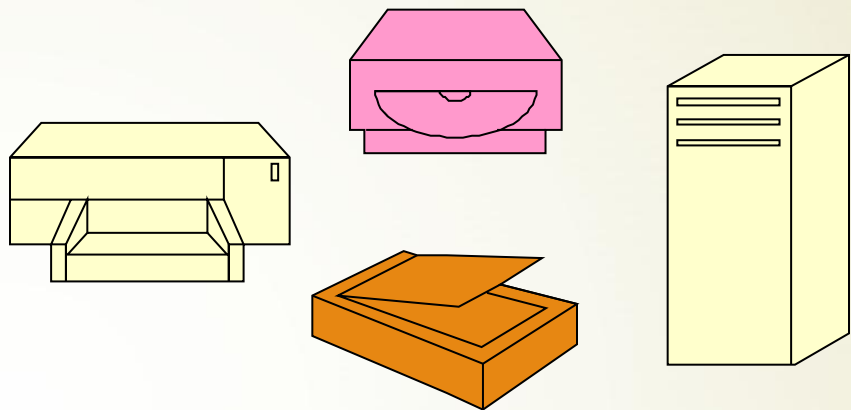
输 入										输 出				
$\overline{S_0}$	$\overline{S_1}$	$\overline{S_2}$	$\overline{S_3}$	$\overline{S_4}$	$\overline{S_5}$	$\overline{S_6}$	$\overline{S_7}$	$\overline{S_8}$	$\overline{S_9}$	A	B	C	D	GS
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	1	0	0	1	1
1	1	1	1	1	1	1	1	0	1	1	0	0	0	1
1	1	1	1	1	1	1	0	1	1	0	1	1	1	1
1	1	1	1	1	1	0	1	1	1	0	1	1	0	1
1	1	1	1	1	0	1	1	1	1	0	1	0	1	1
1	1	1	1	0	1	1	1	1	1	0	1	0	0	1
1	1	1	0	1	1	1	1	1	1	0	0	1	1	1
1	1	0	1	1	1	1	1	1	1	0	0	1	0	1
1	0	1	1	1	1	1	1	1	1	0	0	0	1	1
0	1	1	1	1	1	1	1	1	1	0	0	0	0	1

该编码器输入低电平有效，输出高电平有效，**GS**为标志位。

3. 优先编码器

优先编码器的提出：

实际应用中，经常有两个或更多输入编码信号同时有效。



必须根据轻重缓急，规定好这些外设允许操作的先后次序，即优先级别。

识别多个编码请求信号的优先级别，并进行相应编码的逻辑部件称为优先编码器。

8—3 线优先编码器功能表

输 入									输 出				
<i>EI</i>	<i>I</i> ₇	<i>I</i> ₆	<i>I</i> ₅	<i>I</i> ₄	<i>I</i> ₃	<i>I</i> ₂	<i>I</i> ₁	<i>I</i> ₀	<i>Y</i> ₂	<i>Y</i> ₁	<i>Y</i> ₀	<i>GS</i>	<i>EO</i>
0	×	×	×	×	×	×	×	×	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1
1	1	×	×	×	×	×	×	×	1	1	1	1	0
1	0	1	×	×	×	×	×	×	1	1	0	1	0
1	0	0	1	×	×	×	×	×	1	0	1	1	0
1	0	0	0	1	×	×	×	×	1	0	0	1	0
1	0	0	0	0	1	×	×	×	0	1	1	1	0
1	0	0	0	0	0	1	×	×	0	1	0	1	0
1	0	0	0	0	0	0	1	×	0	0	1	1	0
1	0	0	0	0	0	0	0	1	0	0	0	1	0

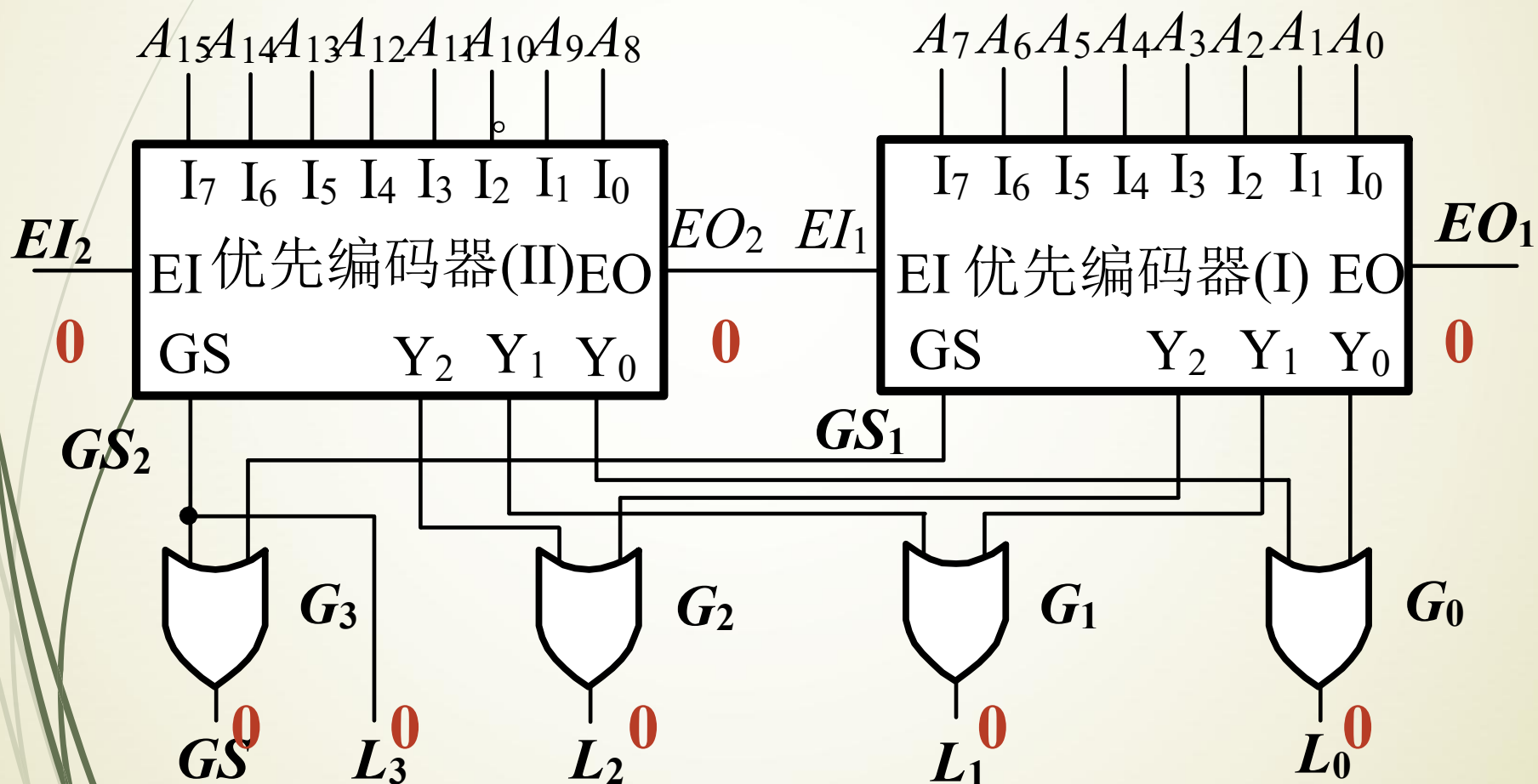
输入高电平有效，输出 $Y_2Y_1Y_0$ 为二进制代码

输入编码信号优先级从高到低为 $I_7 \sim I_0$

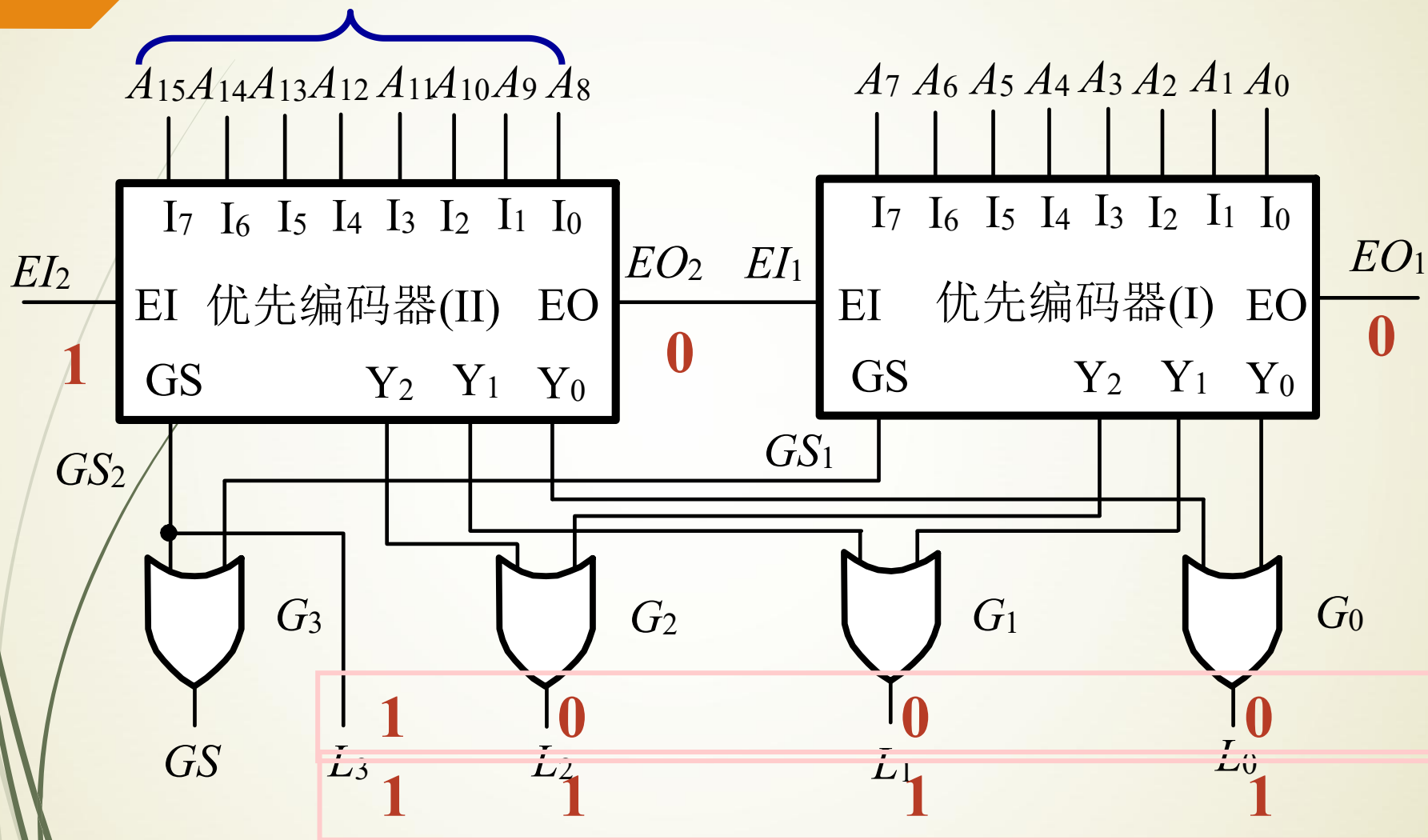
为什么要设计GS、EO输出信号？

用2个8线-3线优先编码器构成16线-4线优先编码器, 其逻辑图如下图所示, 试分析其工作原理。

当使能端 $EI=0$ 时, 无编码输出。



若有效电平输入



4.4.2 译码器/数据分配器

1. 译码器的定义与分类

译码：译码是编码的逆过程，它可将二进制码翻译成代表某一特定含义的信号。（即电路的某种状态）

译码器：具有译码功能的逻辑电路称为译码器。

译码器的分类：

唯一地址译码器

将一系列代码转换成与之一一对应的有效信号。

常见的唯一地址译码器：

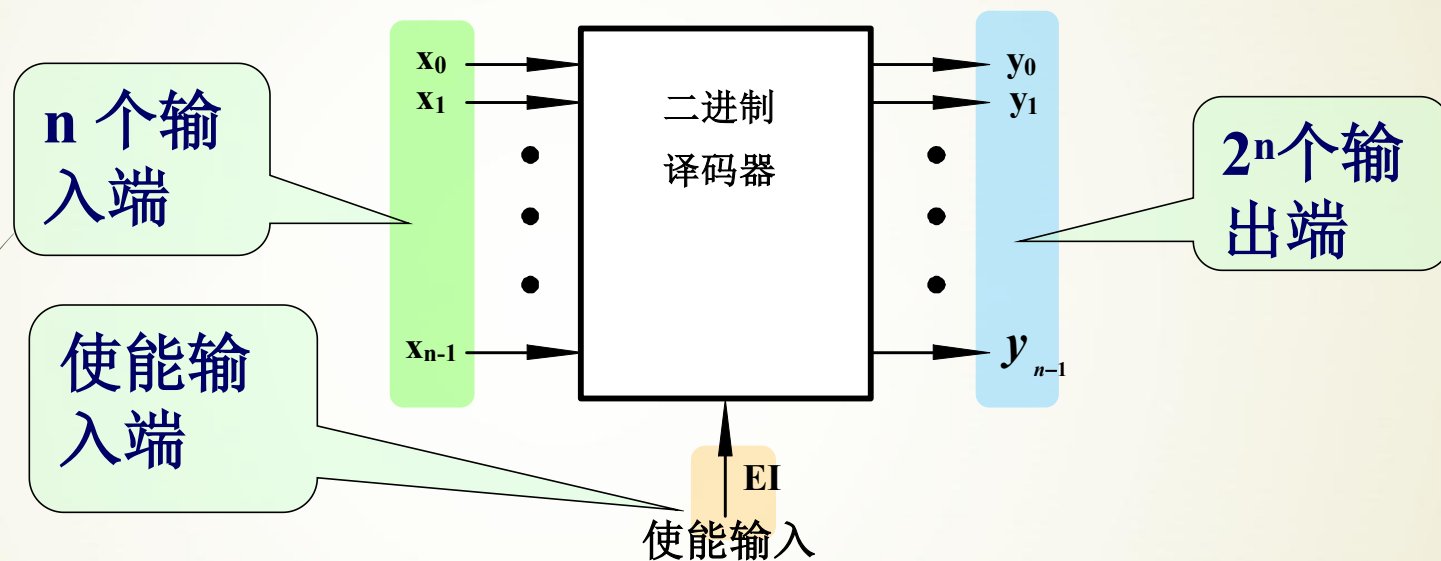
- 二进制译码器
- 二—十进制译码器
- 显示译码器

代码变换器

将一种代码转换成另一种代码。

2. 译码器电路及应用

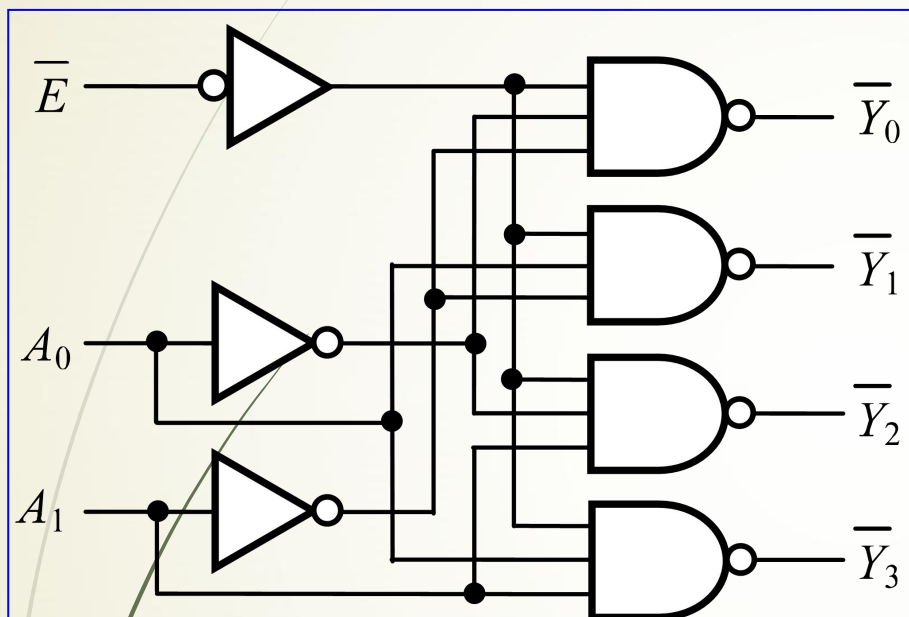
(1) 二进制译码器



设输入端的个数为 n ，输出端的个数为 M
则有

$$M=2^n$$

(a) 2线-4线译码器 (分析)



功能表

输入			输出			
$\overline{E}$	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$
1	×	×	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

$$\overline{Y}_0 = \overline{\overline{E} \overline{A_1} \overline{A_0}}$$

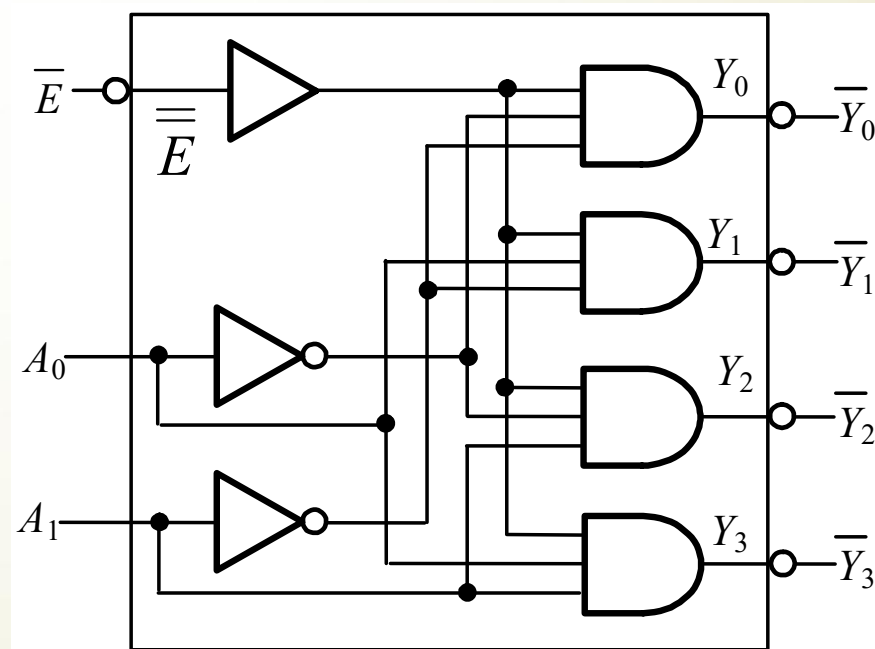
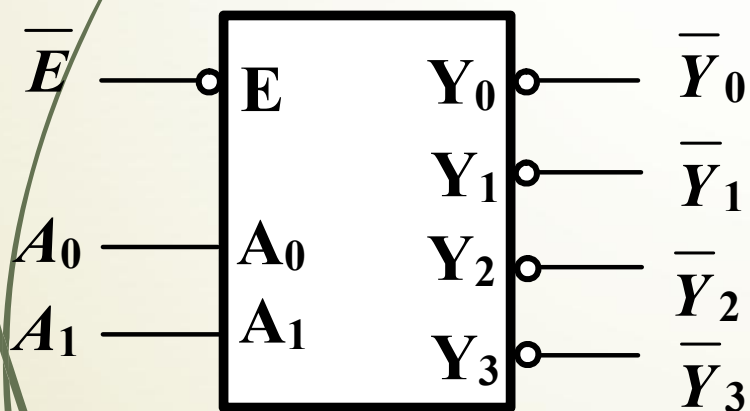
$$\overline{Y}_1 = \overline{\overline{E} \overline{A_1} A_0}$$

$$\overline{Y}_2 = \overline{\overline{E} A_1 \overline{A_0}}$$

$$\overline{Y}_3 = \overline{\overline{E} A_1 A_0}$$

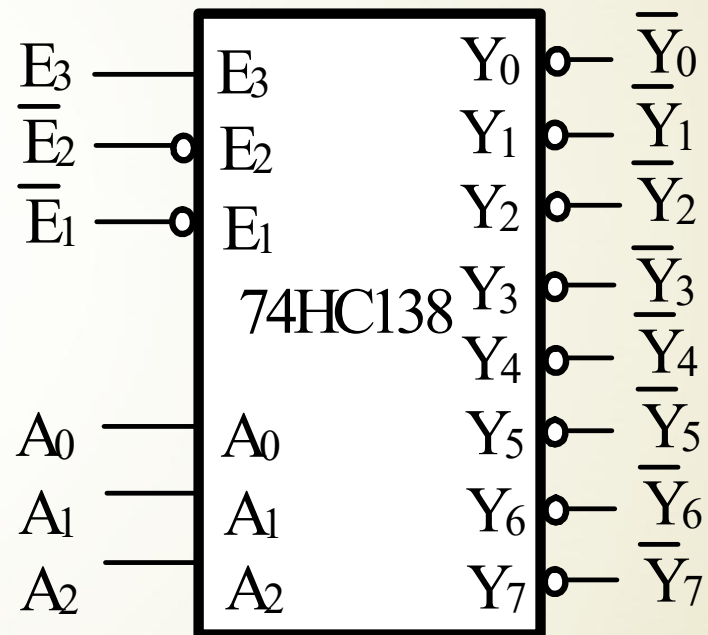
----逻辑符号说明

1/2 74x139



(b) 3线-8线译码器 (74HC138)

- ◆ 使能输入 E_3 高有效, $\overline{E_2}$ 、 $\overline{E_1}$ 低有效。
- ◆ 输入 $A_2A_1A_0$ 高有效。
- ◆ 输出 $\overline{Y_7} \sim \overline{Y_0}$ 低有效。
- ◆ 低有效变量上面加“—”。



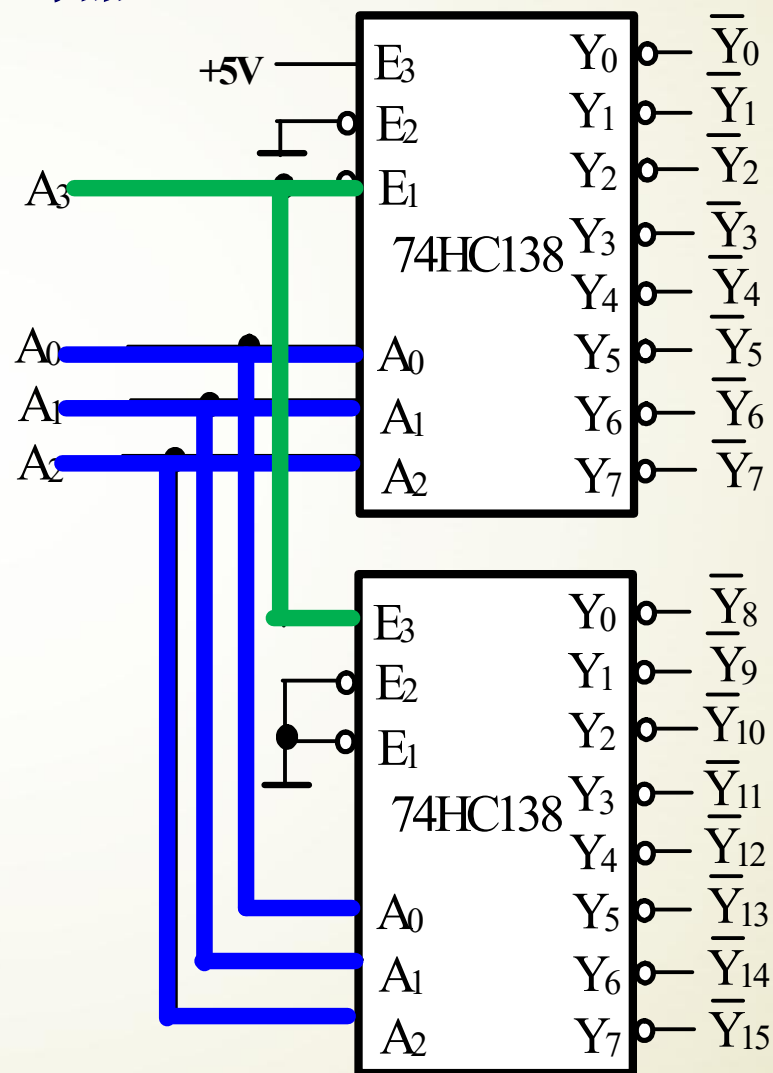
逻辑符号

3线-8线译码器（74HC138）功能表

[illegible]

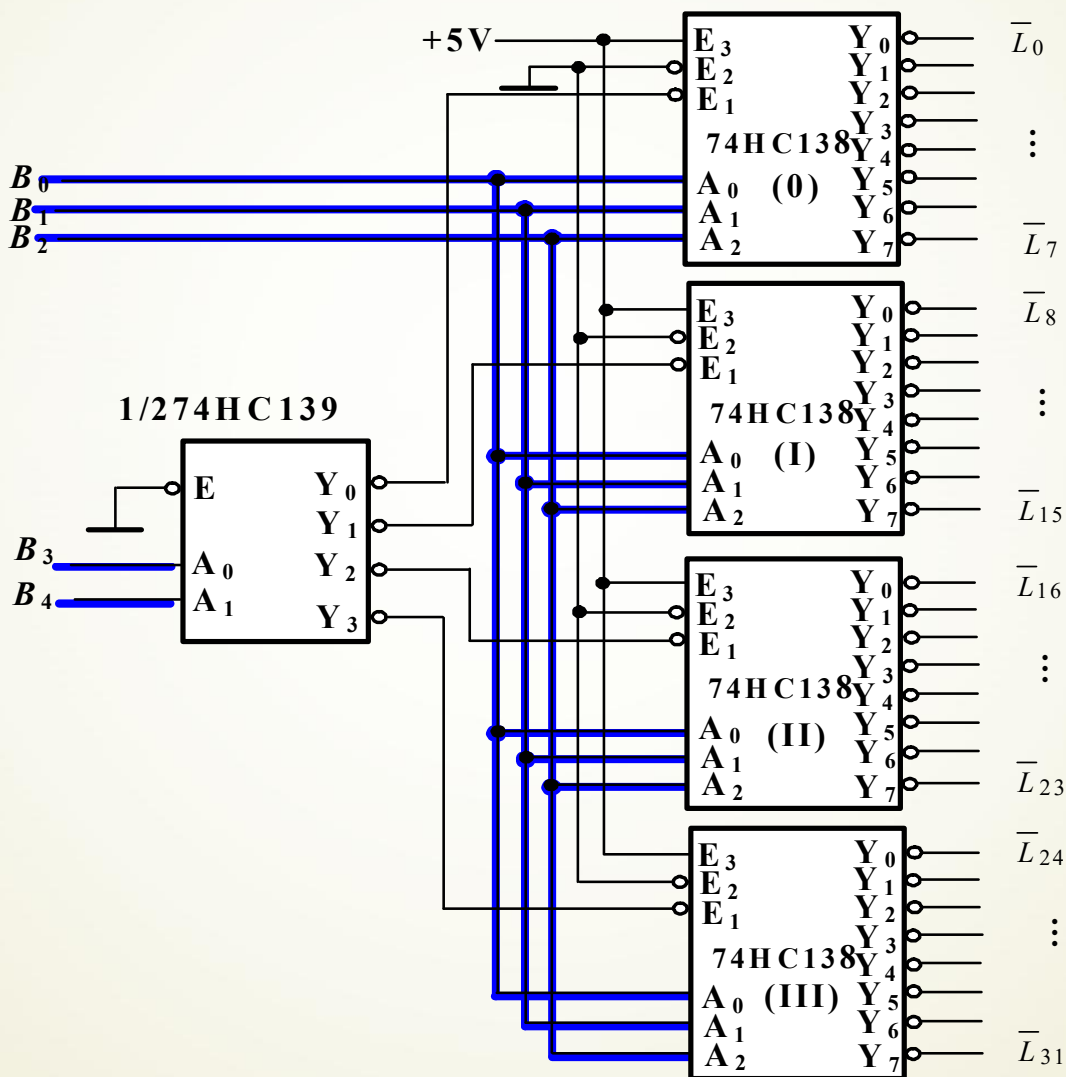
(c) 译码器的扩展

用74X138构成4线-16线译码器



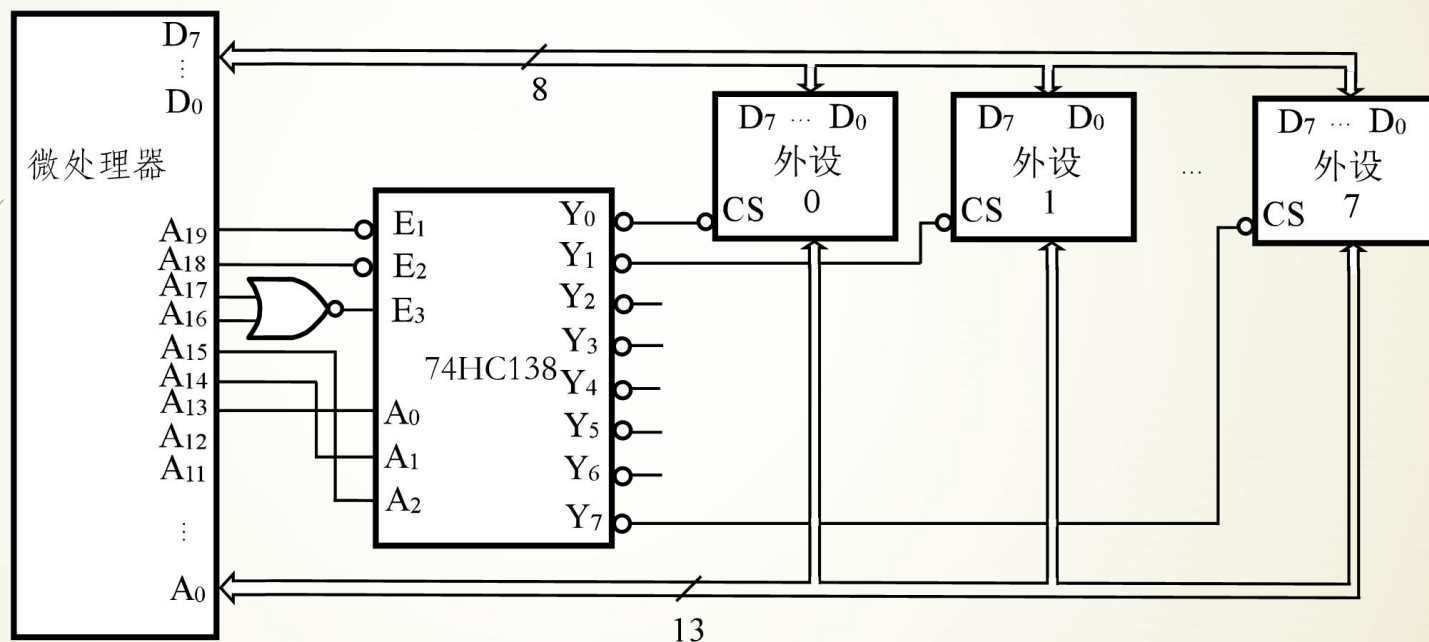
(c) 译码器的扩展

用74X139和74X138构成5线-32线译码器



(d) 译码器应用

译码器常用于识别不同设备。当 $A_{19}A_{18}A_{17}A_{16}=0000$,
 $A_{15}A_{14}A_{13}=000$, 选中外设0。



$$\begin{aligned}\overline{CS}_0 &= \overline{Y}_0 = \overline{\overline{A}_{19} \overline{A}_{18} (A_{17} + A_{16}) \overline{A}_{15} \overline{A}_{14} \overline{A}_{13}} \\ &= A_{19} + A_{18} + A_{17} + A_{16} + A_{15} + A_{14} + A_{13}\end{aligned}$$

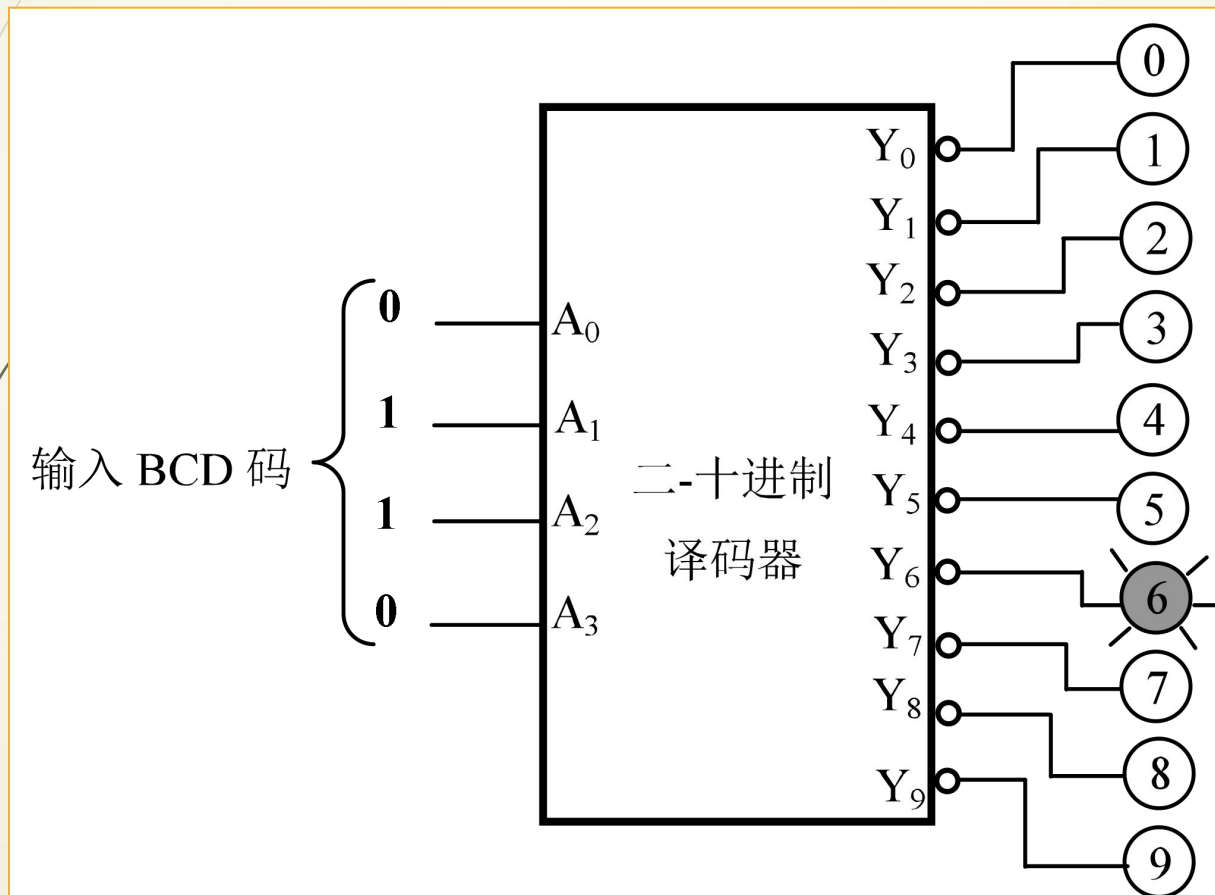
真值表

对于BCD代码以外的伪码（1010~1111这6个代码） $Y_0 \sim Y_9$ 均为高电平。

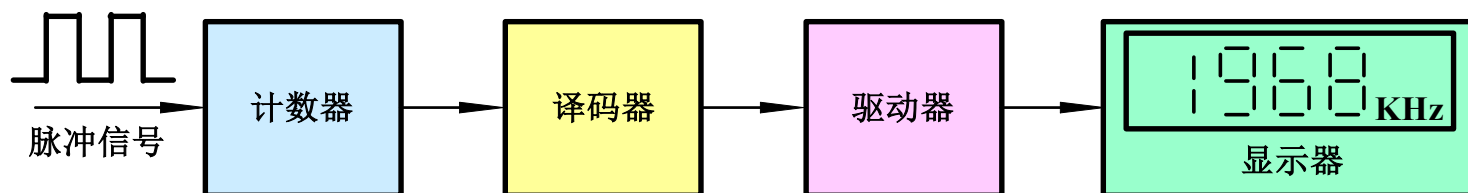
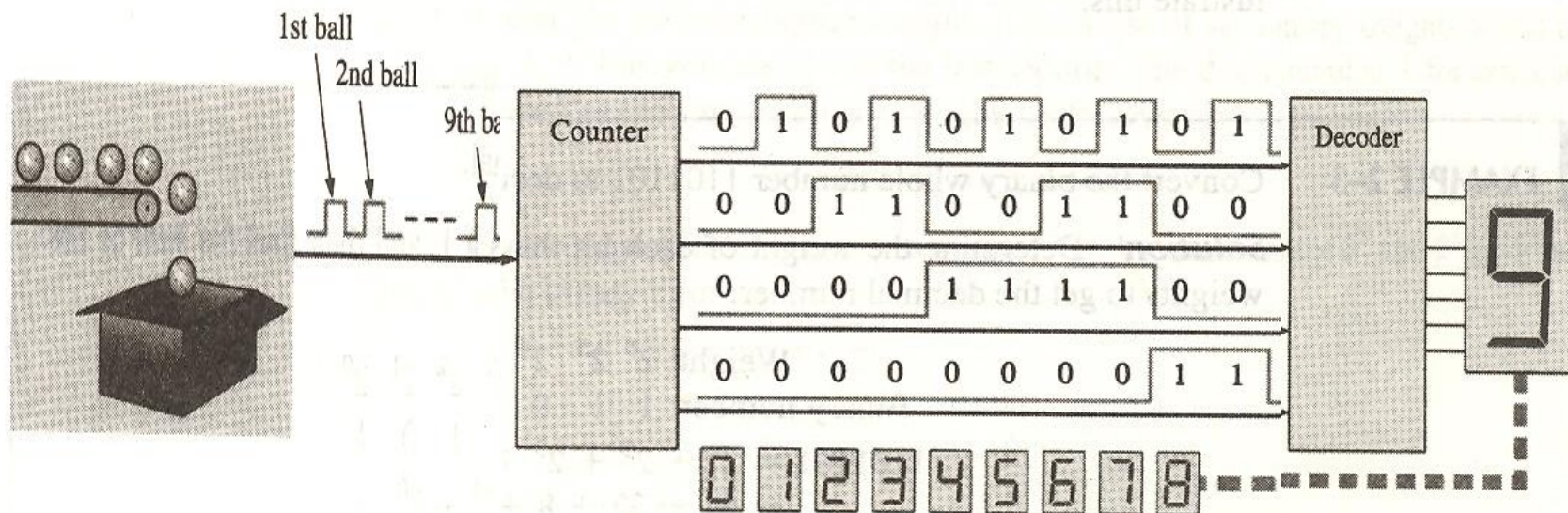
[illegible]

二-十进制译码器功能：

将8421BCD码译成为10个状态输出。

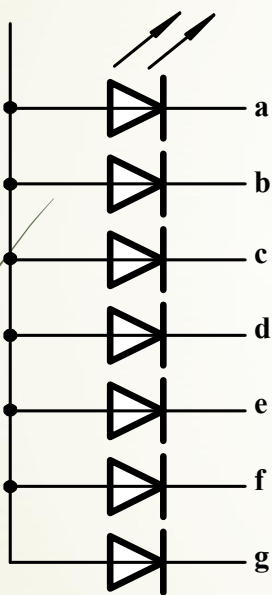


(3) 显示译码器

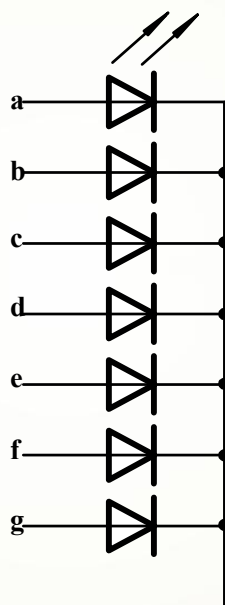


1. 七段显示译码器

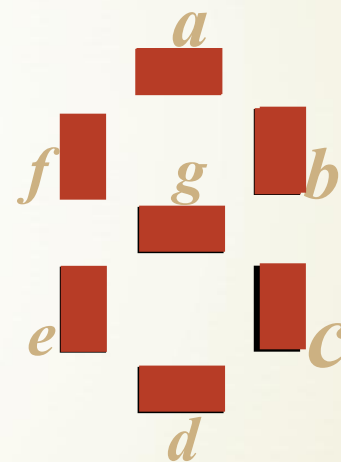
(1) 最常用的显示器有：半导体发光二极管和液晶显示器。



共阳极显示器



共阴极显示器

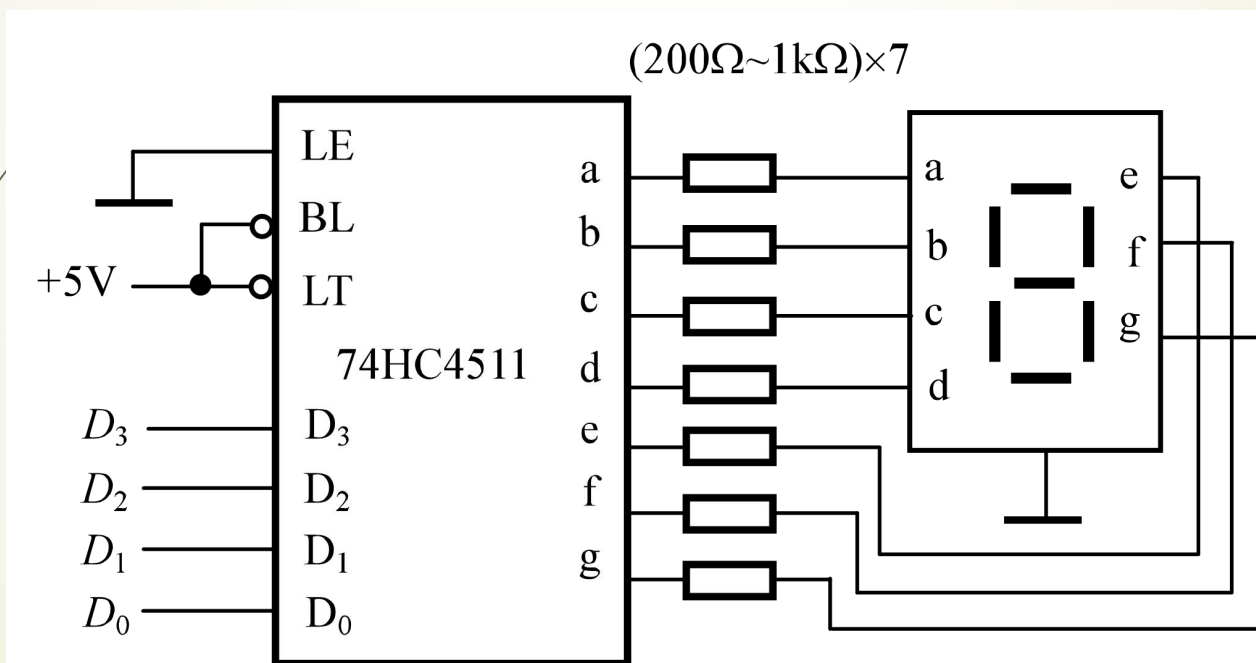


显示器分段布局图

常用的集成七段显示译码器

-----CMOS七段显示译码器74HC4511

显示译码器与显示器的连接方式



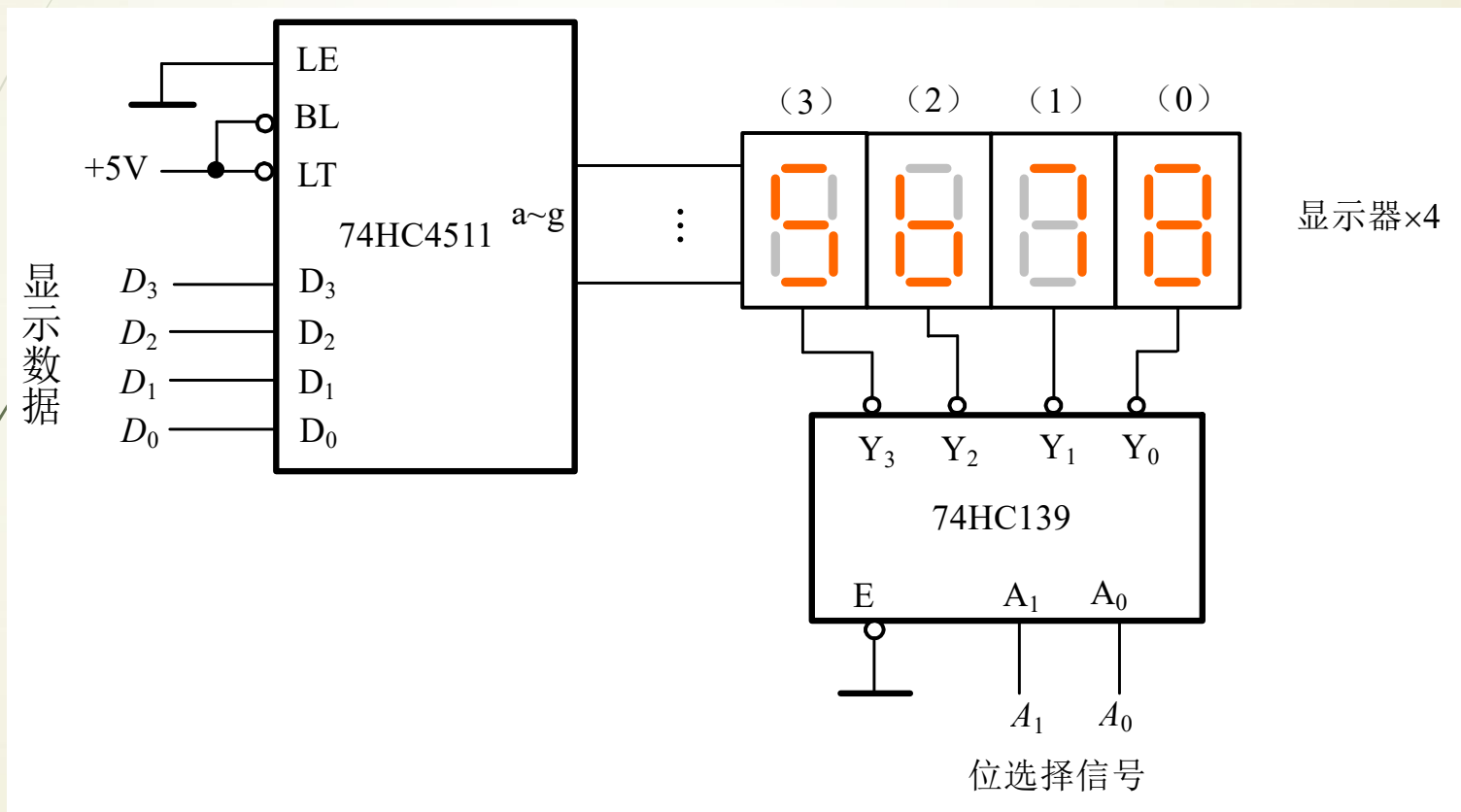
CMOS七段显示译码器74HC4511功能表

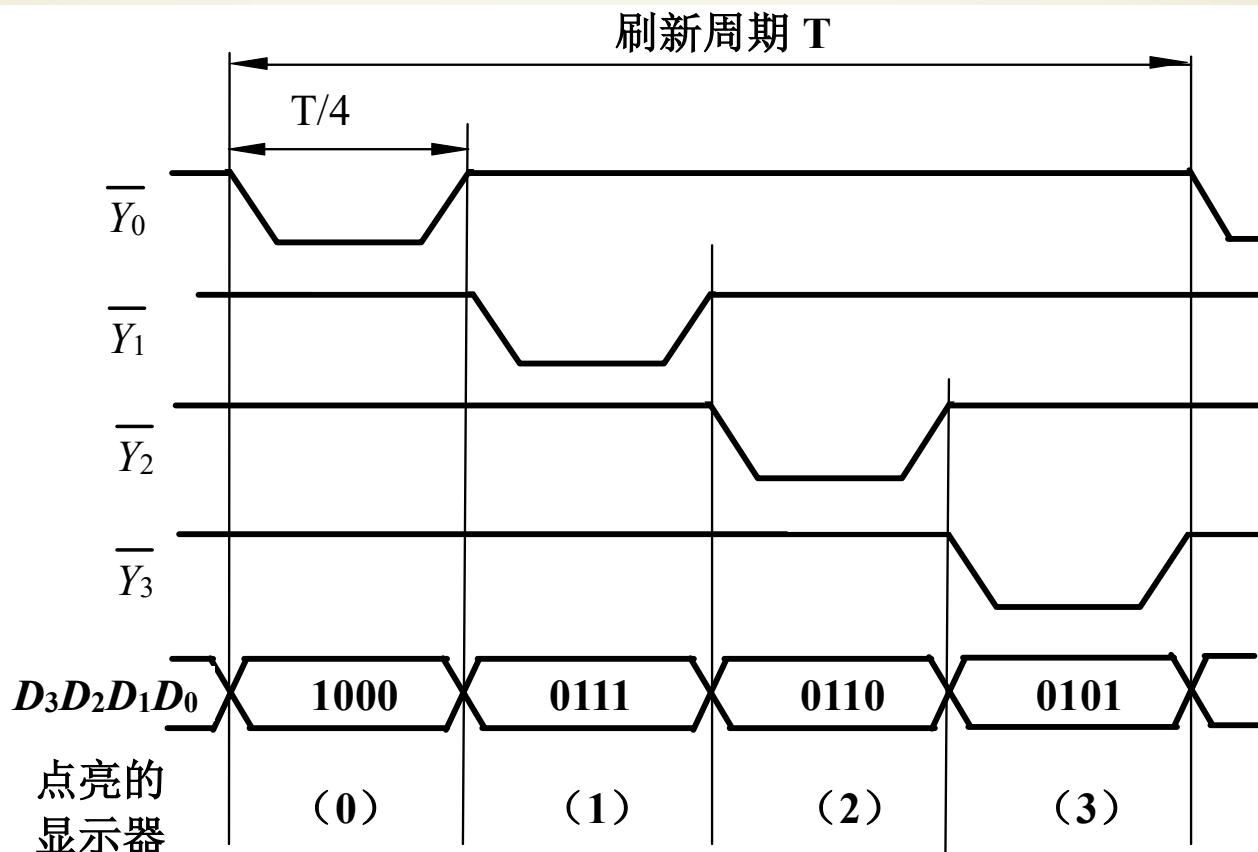
十进制或功能	输 入							输 出							字形
	LE	$\overline{BL}$	$\overline{LT}$	D_3	D_2	D_1	D_0	a	b	c	d	e	f	g	
0	0	1	1	0	0	0	0	1	1	1	1	1	1	0	0
1	0	1	1	0	0	0	1	0	1	1	0	0	0	0	1
2	0	1	1	0	0	1	0	1	1	0	1	1	0	1	2
3	0	1	1	0	0	1	1	1	1	1	1	0	0	1	3
4	0	1	1	0	1	0	0	0	1	1	0	0	1	1	4
5	0	1	1	0	1	0	1	1	0	1	1	0	1	1	5
6	0	1	1	0	1	1	0	0	0	1	1	1	1	1	6
7	0	1	1	0	1	1	1	1	1	1	0	0	0	0	7
8	0	1	1	1	0	0	0	1	1	1	1	1	1	1	8
9	0	1	1	1	0	0	1	1	1	1	1	0	1	1	9



十进制 或功能	输 入							输 出							字形
	LE	$\overline{BL}$	$\overline{LT}$	D_3	D_2	D_1	D_0	a	b	c	d	e	f	g	
10	0	1	1	1	0	1	0	0	0	0	0	0	0	0	熄灭
11	0	1	1	1	0	1	1	0	0	0	0	0	0	0	熄灭
12	0	1	1	1	1	0	0	0	0	0	0	0	0	0	熄灭
13	0	1	1	1	1	0	1	0	0	0	0	0	0	0	熄灭
14	0	1	1	1	1	1	0	0	0	0	0	0	0	0	熄灭
15	0	1	1	1	1	1	1	0	0	0	0	0	0	0	熄灭
灯 测 试	×	×	0	×	×	×	×	1	1	1	1	1	1	1	8
灭 灯	×	0	1	×	×	×	×	0	0	0	0	0	0	0	熄灭
锁 存	1	1	1	×	×	×	×	*							*

例 由译码器、显示译码及4个七段显示器构成的4位动态显示电路如图所示，试分析工作原理。



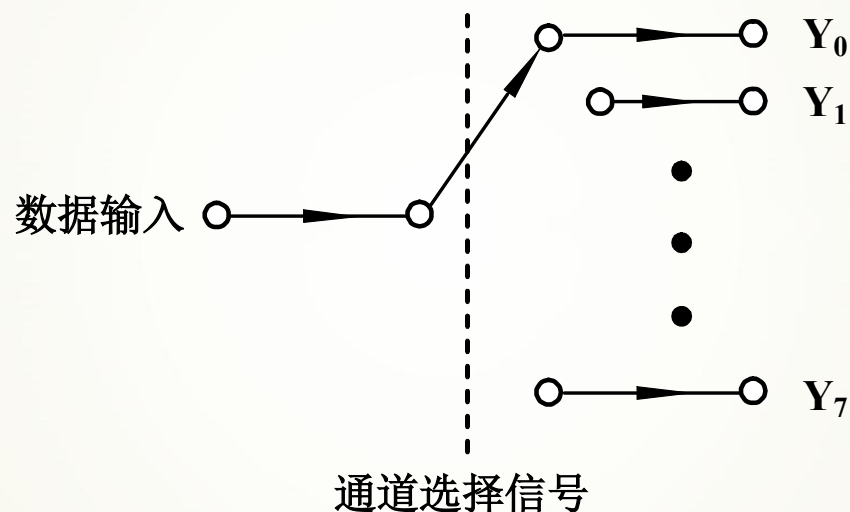


位选择信号A1、A0控制 $\overline{Y}_3 \sim \overline{Y}_0$ 依次产生低电平，使4个显示器轮流显示。要显示的数据组依次送到 $D_3D_2D_1D_0$ 分别在4个显示器上显示。利用人的视觉暂留时间，当刷新频率达到一定数值（如40Hz，即 $T=1/40\text{Hz}=0.025\text{s}$ ）可看到稳定的数字。

3. 数据分配器

—用74HC138组成数据分配器

数据分配器示意图

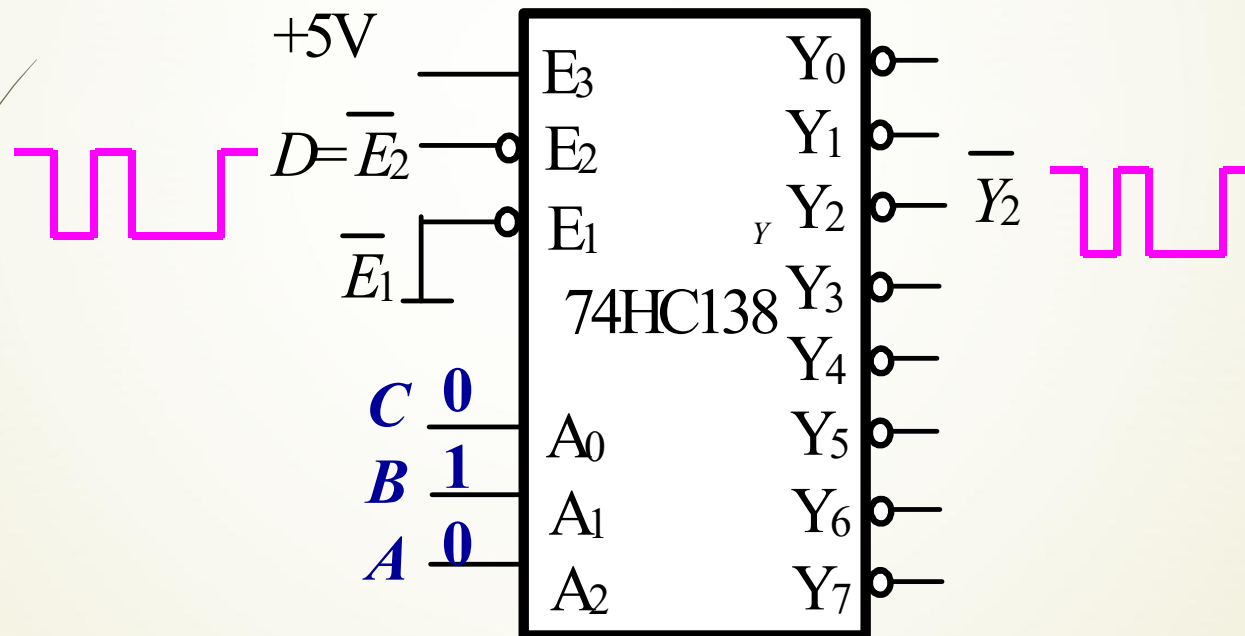


数据分配器：相当于多输出的单刀多掷开关，是将公共数据线上的数据按需要送到不同的通道上去的逻辑电路。

用译码器实现数据分配器

$$\overline{Y_2} = \overline{E_3} \overline{E_2} D \overline{A} B \overline{C}$$

当 $ABC = 010$ 时, $\overline{Y_2} = D$





例: 试用门电路设计一个具有低电平使能控制的1线—4线数据分配器, 使能信号无效时, 电路所有的输出为高阻态。当通道选择信号将1路输入信号连接到其中1路输出端时, 其他输出端为高阻状态。

1. 列真值表

输出端有3种状态 (0、1、z), 输出级是4个三态门组成。其控制信号由 $\bar{E}$ 、S1、S0共同作用产生。

输 入			三 态 门 控 制 信 号				输 出			
$\bar{E}$	S ₁	S ₀	C ₃	C ₂	C ₁	C ₀	Y ₃	Y ₂	Y ₁	Y ₀
0	0	0	0	0	0	1	z	z	z	In
0	0	1	0	0	1	0	z	z	In	z
0	1	0	0	1	0	0	z	In	z	z
0	1	1	1	0	0	0	In	z	z	z
1	x	x	0	0	0	0	z	z	z	z

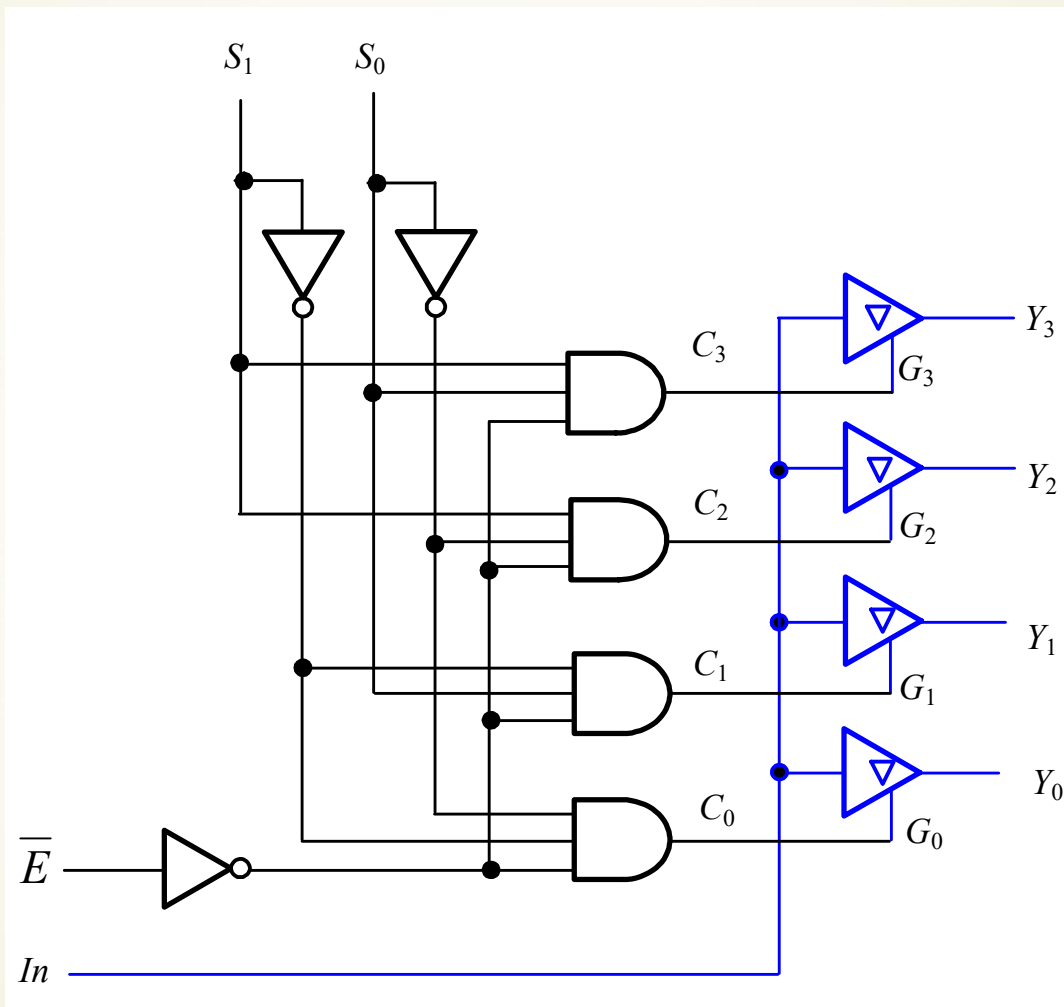
2. 写出4个三态门控制端的逻辑表达式

$$C_0 = \overline{\overline{E}} \cdot \overline{S_1} \cdot \overline{S_0} \quad C_1 = \overline{\overline{E}} \cdot \overline{S_1} \cdot S_0$$

$$C_2 = \overline{\overline{E}} \cdot S_1 \cdot \overline{S_0} \quad C_3 = \overline{\overline{E}} \cdot S_1 \cdot S_0$$

输 入			三 态 门 控 制 信 号				输 出			
$\overline{E}$	S_1	S_0	C_3	C_2	C_1	C_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	1	z	z	z	In
0	0	1	0	0	1	0	z	z	In	z
0	1	0	0	1	0	0	z	In	z	z
0	1	1	1	0	0	0	In	z	z	z
1	x	x	0	0	0	0	z	z	z	z

3. 画逻辑电路

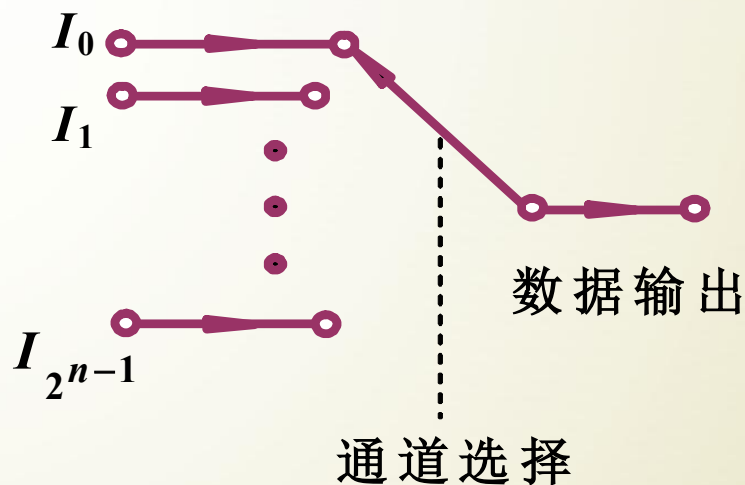


4.4.3 数据选择器

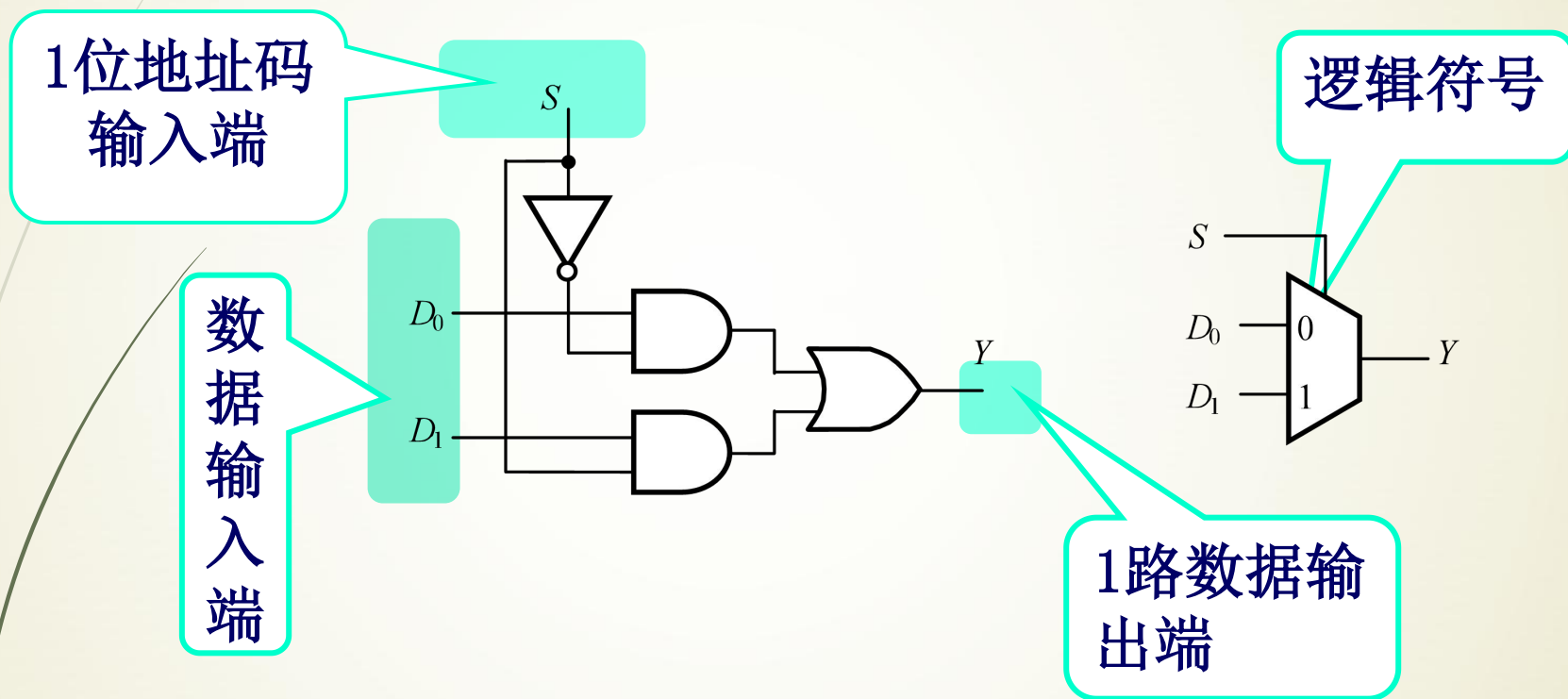
1、数据选择器的定义与功能

数据选择器： 能够实现数据选择功能的逻辑电路。它的作用相当于多个输入的单刀多掷开关，又称“多路开关”。

数据选择的功能： 在通道选择信号的作用下，将多个通道的数据分时传送到公共的数据通道上去的。



2选1数据选择器



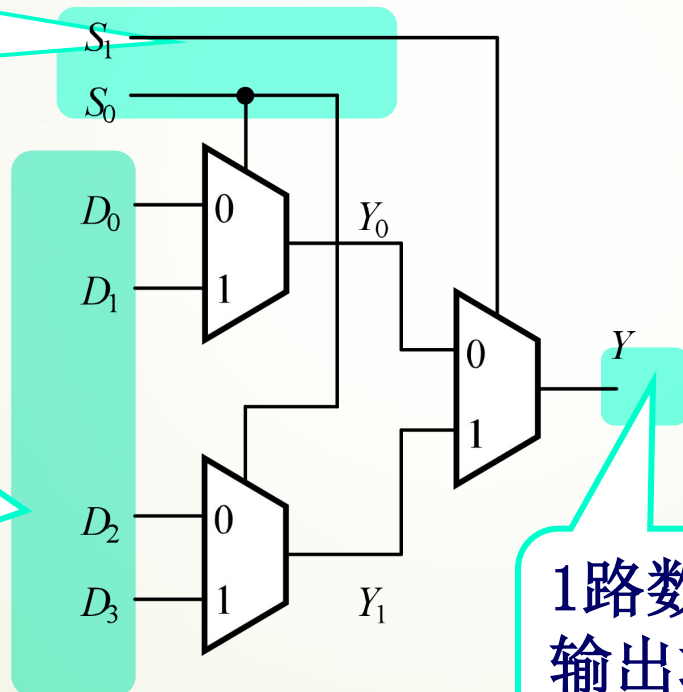
4选1数据选择器

(1) 逻辑电路

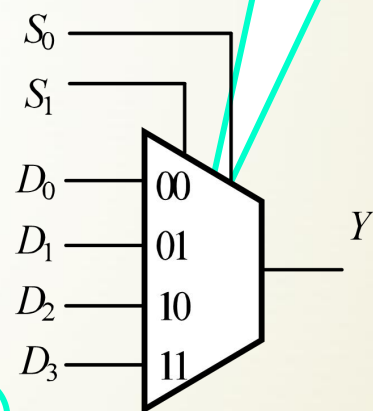
由3个2选1数据选择器构成4选1数据选择器。

2 位地址
码输入端

数据
输入端



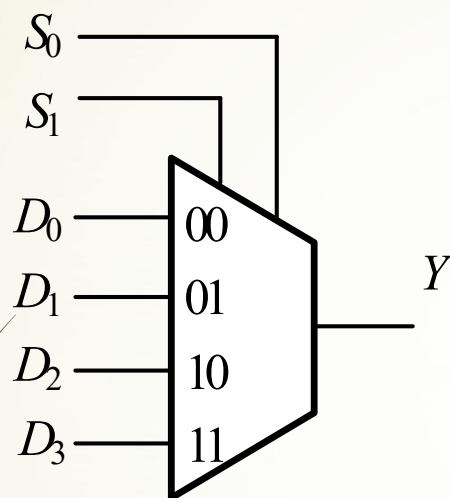
逻辑符号



1路数据
输出端

(2) 工作原理及逻辑功能

真值表



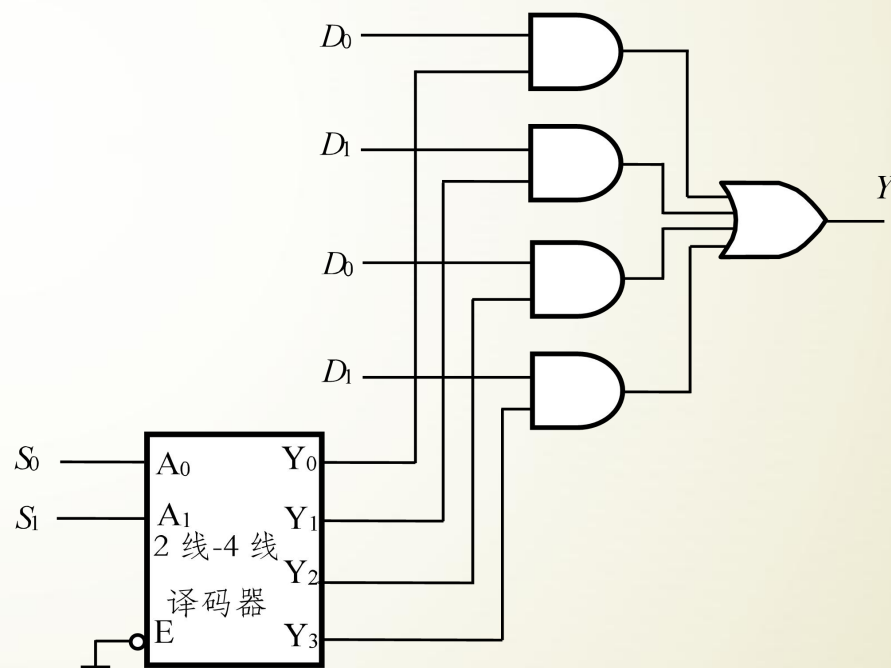
选择输入		输 出
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$$Y = \overline{S_1}\overline{S_0}D_0 + \overline{S_1}S_0D_1 + S_1\overline{S_0}D_2 + S_1S_0D_3$$

$$Y = D_0m_0 + D_1m_1 + D_2m_2 + D_3m_3$$

(3) 用译码器构成数据选择器

译码器与数据选择器都具有选择功能，因此可以用二进制译码器构成数据选择器。用 S_1S_0 选择 D_0 、 D_1 、 D_2 或 D_3 中的一个数据传送到输出端。



(4) 数据选择器实现逻辑函数

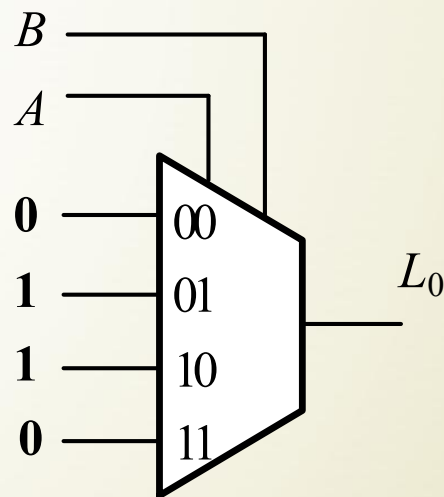
例4.4.8 试用数据选择器实现下列逻辑函数

① 用4选1数据选择器实现 $L_0 = \overline{A}B + A\overline{B}$

② 用2选1数据选择器和必要的逻辑门实现 $L_1 = \overline{A}\overline{C} + BC$

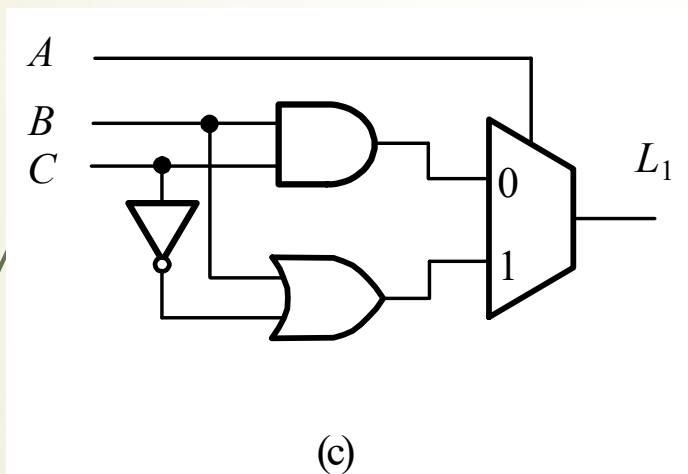
$$Y = \overline{S_1}\overline{S_0}D_0 + \overline{S_1}S_0D_1 + S_1\overline{S_0}D_2 + S_1S_0D_3$$

① 当 $S_1 = A, S_2 = B,$
 $D_0 = D_3 = 0, D_1 = D_2 = 1$



② 用2选1数据选择器和必要的逻辑门实现 $L_1 = \overline{A}C + BC$

(a) 从真值表考察 A 、 B 、 C 与 L_1 的关系。2选1数据选择器只有1个选通端接输入 A ，表达式有3个变量。因此数据端需要输入2个变量。



输 入			输 出	
A	B	C	L_1	
0	0	0	0	$L_1 = BC$
0	0	1	0	
0	1	0	0	
0	1	1	1	
1	0	0	1	$L_1 = B + \overline{C}$
1	0	1	0	
1	1	0	1	
1	1	1	1	

② 用2选1数据选择器和必要的逻辑门实现 $L_1 = \overline{A}\overline{C} + BC$

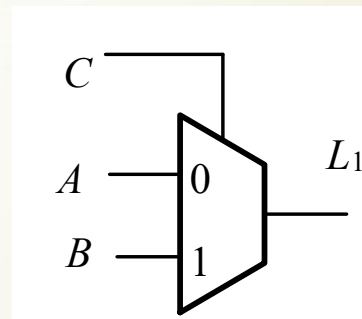
(b) 用香农展开定理: $f(X_1, X_2, \dots, X_n) = \overline{X_1} \cdot f(0, X_2, \dots, X_n) + X_1 \cdot f(1, X_2, \dots, X_n)$

对A展开, 结果与真值表的相同。

$$\begin{aligned} L_1 &= \overline{A}(0 \cdot \overline{C} + BC) + A(1 \cdot \overline{C} + BC) \\ &= \overline{A}BC + A(\overline{C} + BC) \\ &= \overline{A}BC + A(B + \overline{C}) \end{aligned}$$

对C展开, 成本更低。

$$\begin{aligned} L_1 &= \overline{C}(A \cdot \overline{0} + B \cdot 0) + C(A \cdot \overline{1} + B \cdot 1) \\ &= \overline{C}(A) + C(B) \end{aligned}$$



总结：

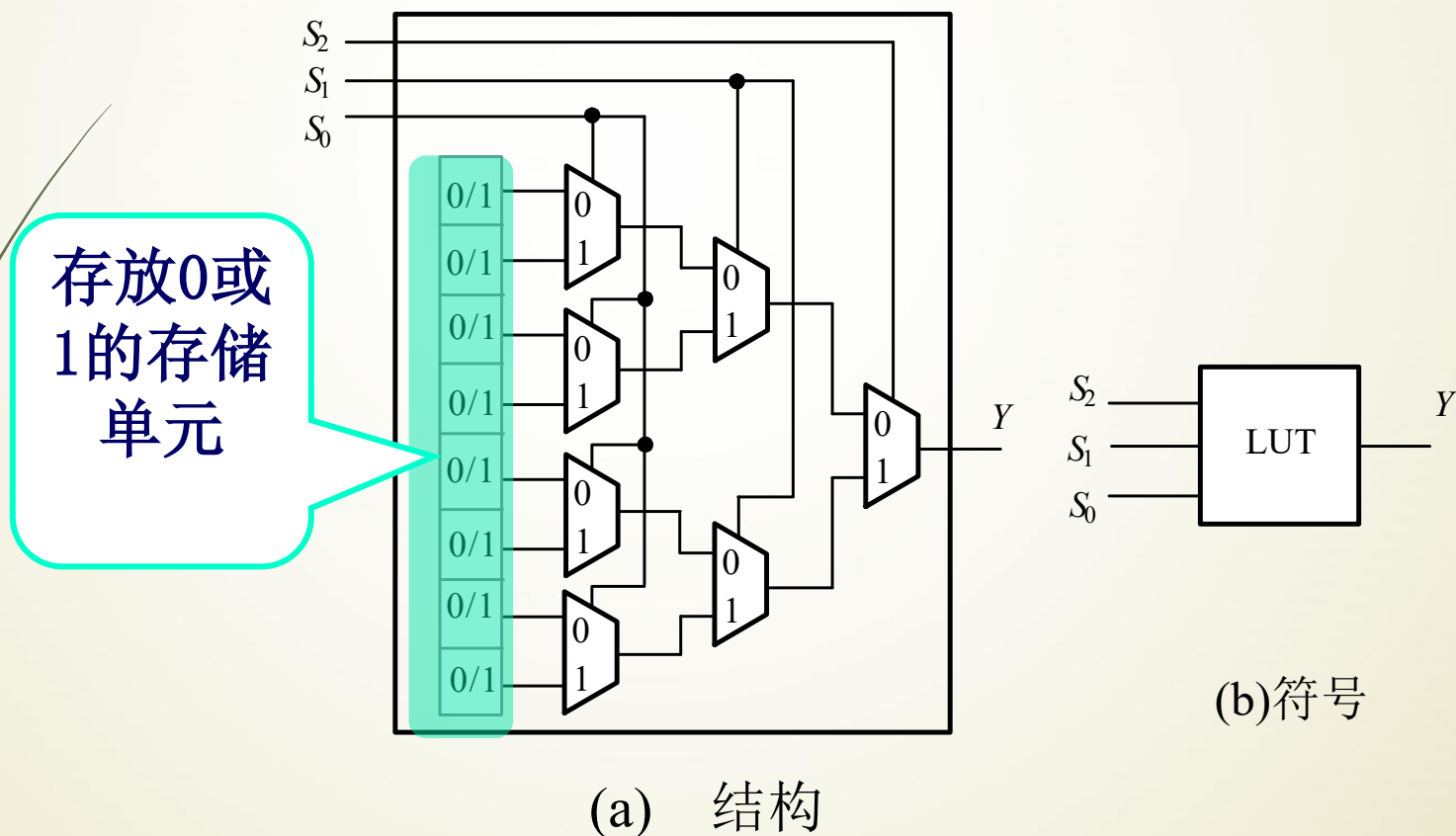
利用数据选择器实现函数的一般步骤：（变量数=选通端数）

- a、将函数变换成最小项表达式
- b、地址信号 S_2 、 S_1 、 S_0 作为函数的输入变量
- c、处理数据输入 $D_0 \sim D_7$ 信号电平。逻辑表达式中有 m_i ,则相应 $D_i=1$ ，其他的数据输入端均为0。

当变量数>选通端数，考虑如何将某些变量接入数据端。

(5) 数据选择器构成查找表LUT

构成FPGA基本单元的逻辑块主要是查找表LUT。LUT实质是一个小规模的存储器，以真值表的形式实现给定的逻辑函数。3输入LUT的结构及逻辑符号如图。



用查找表LUT实现逻辑函数

$$L_1 = \overline{A}\overline{C} + BC$$

用LUT实现逻辑函数，变量A、B、C接选择输入端，对存储单元进行编程。

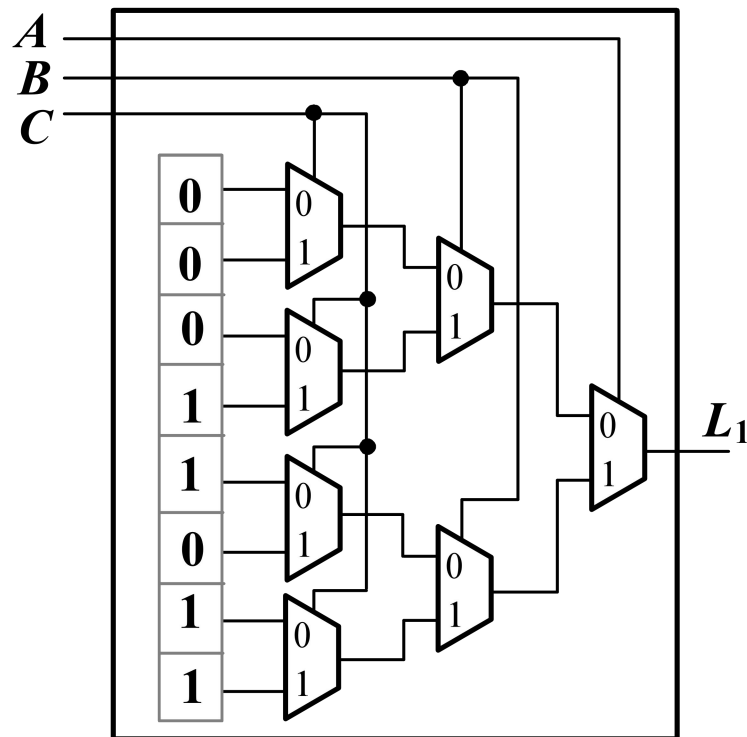
根据前面例题已知

$$L_1 = \overline{A}\overline{C} + BC$$

$$= \sum m(3, 4, 6, 7)$$

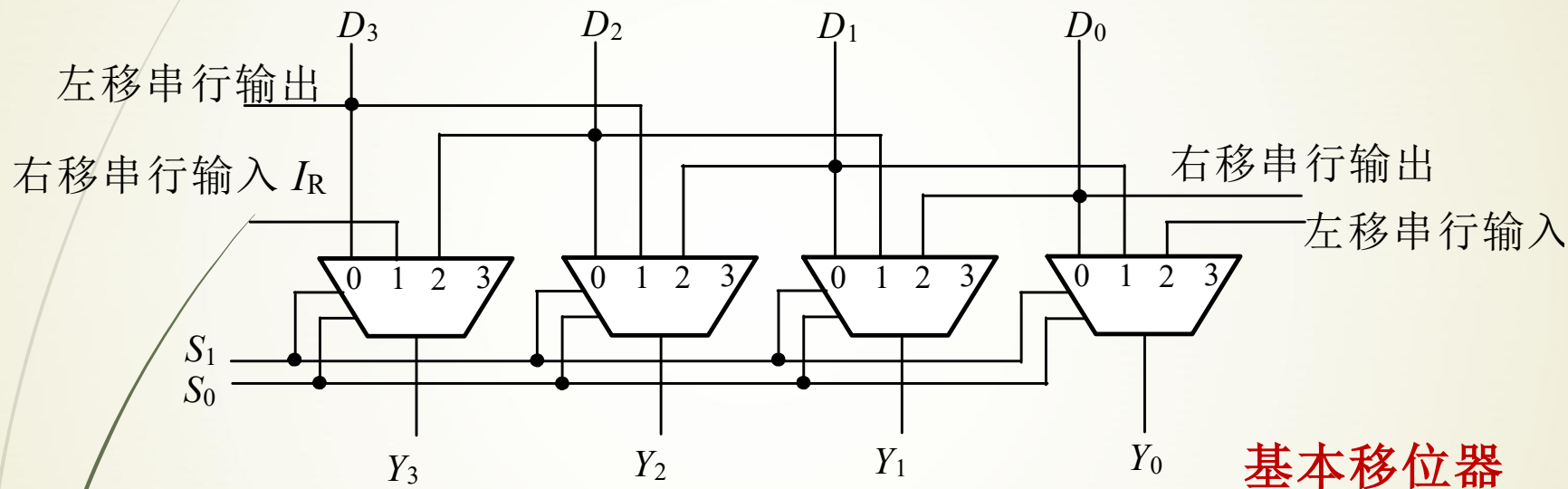
$$D_0 = D_1 = D_2 = D_5 = 0$$

$$D_3 = D_4 = D_6 = D_7 = 1$$



(6) 数据选择器构成移位器

第6章介绍的移位寄存器的移位是在时钟脉冲控制下进行的，一个时钟脉冲只能移动一位。但在运算电路中常需要高速移位，这时便可采用组合逻辑电路构成的移位器。

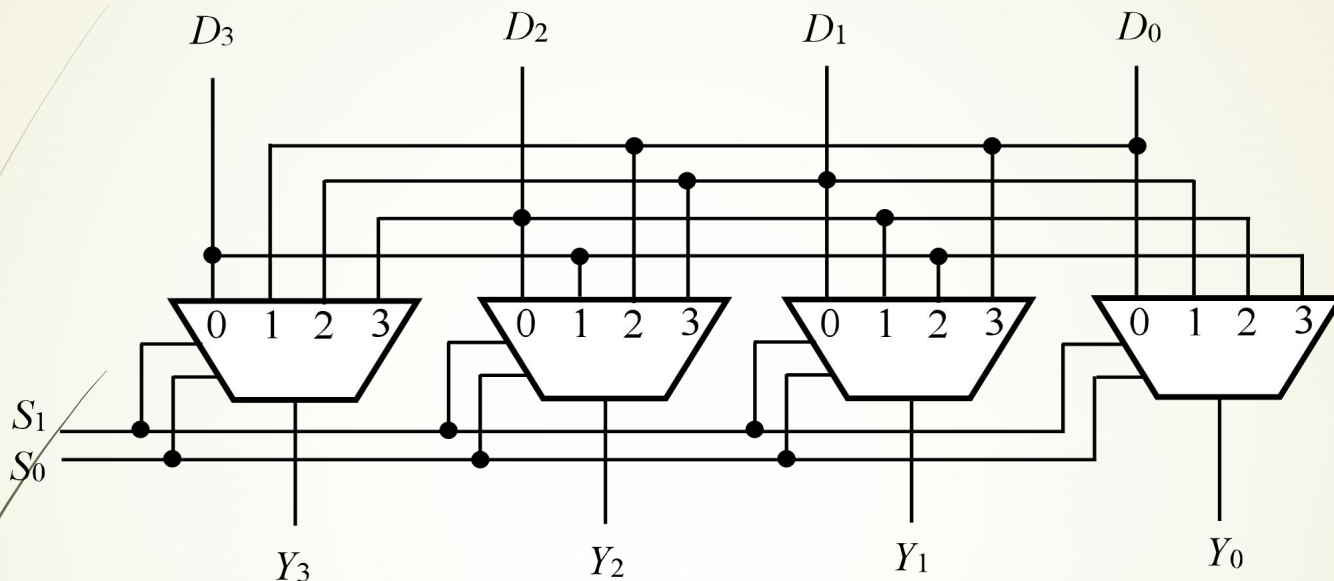


基本移位器

选 择	输 出	功 能
$S_1 S_0$	$Y_3 Y_2 Y_1 Y_0$	
0 0	$D_3 D_2 D_1 D_0$	直接输出
0 1	$I_R D_3 D_2 D_1$	右移一位
1 0	$D_2 D_1 D_0 I_L$	左移一位

(6) 数据选择器构成移位器

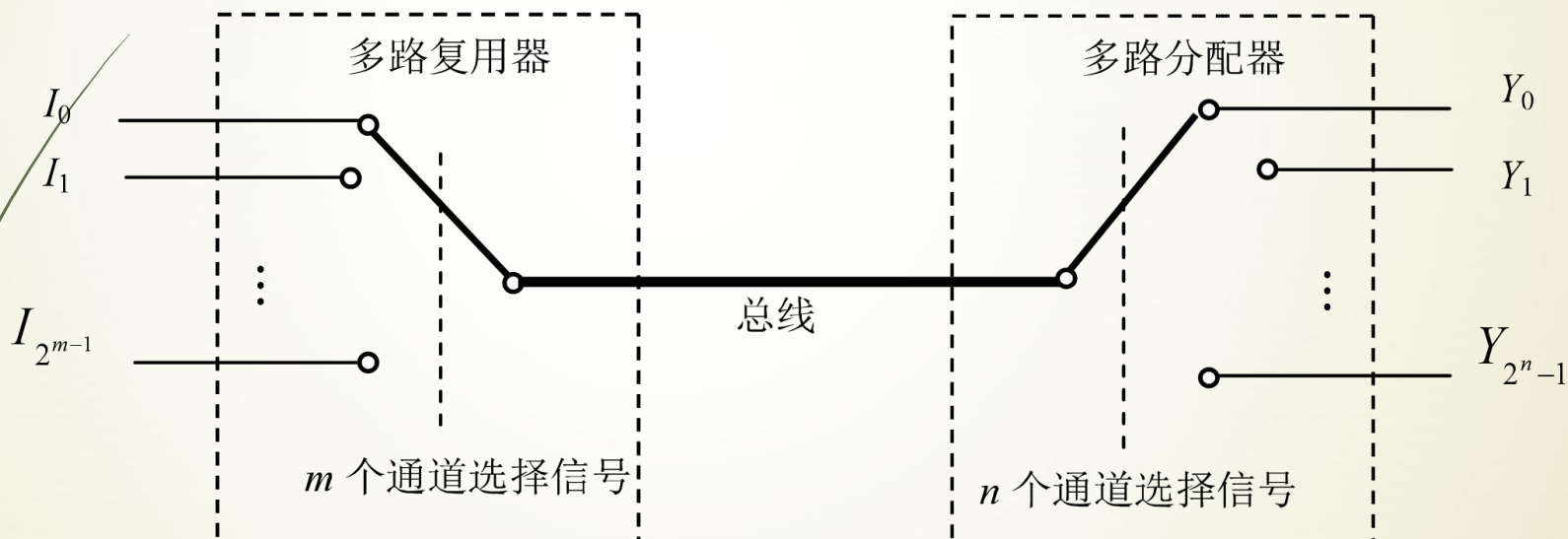
桶形移位器



选 择	输 出	功 能
$S_1 \ S_0$	$Y_3 \ Y_2 \ Y_1 \ Y_0$	
0 0	$D_3 \ D_2 \ D_1 \ D_0$	直接输出
0 1	$D_0 \ D_3 \ D_2 \ D_1$	循环右移一位（循环左移三位）
1 0	$D_1 \ D_0 \ D_3 \ D_2$	循环右移二位（循环左移二位）
1 1	$D_2 \ D_1 \ D_0 \ D_3$	循环右移三位（循环左移一位）

(7) 数据选择器、数据分配器与总线的连接

这种信息传输的基本原理在通信系统、计算机网络系统、以及计算机内部各功能部件之间的信息转送等等都有广泛的应用。



4.4.4 数值比较器

数值比较器：对两个1位数字进行比较（ A 、 B ），以判断其大小的逻辑电路。

1、 1位数值比较器(设计)

输入：两个一位二进制数 A 、 B 。

输出： $F_{A>B} = 1$ ，表示 A 大于 B

$F_{A<B} = 1$ ，表示 A 小于 B

$F_{A=B} = 1$ ，表示 A 等于 B

1位数值比较器

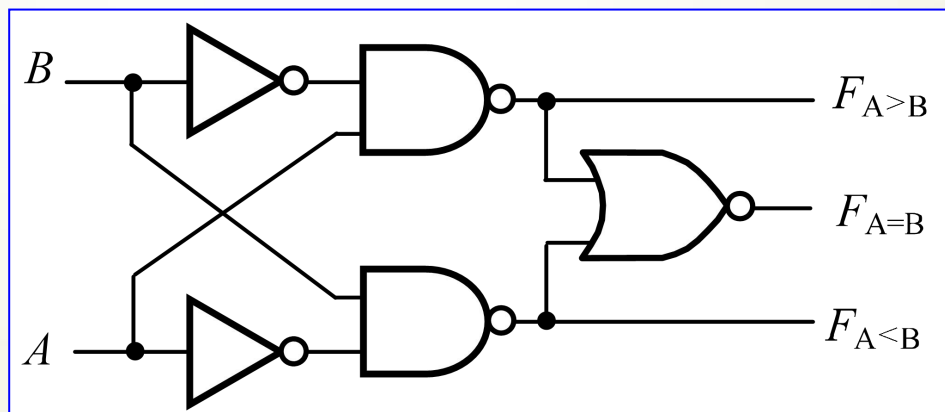
一位数值比较器真值表

输 入		输 出		
A	B	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

$$F_{A>B} = A \overline{B}$$

$$F_{A<B} = \overline{A} B$$

$$F_{A=B} = \overline{A} \overline{B} + AB$$



2、2 位数值比较器：

比较两个2 位二进制数的大小的电路

输入：两个2位二进制数 $A=A_1 A_0$ 、 $B=B_1 B_0$

能否用1位数值比较器设计两位数值比较器？

用一位数值比较器设计多位数值比较器的原则

当高位 (A_1 、 B_1) 不相等时，无需比较低位 (A_0 、 B_0)，高位比较的结果就是两个数的比较结果。

当高位相等时，两数的比较结果由低位比较的结果决定。

真值表

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

$$F_{A>B} = (A_1 > B_1) + (A_1 = B_1)(A_0 > B_0)$$

$$F_{A<B} = (A_1 < B_1) + (A_1 = B_1)(A_0 < B_0)$$

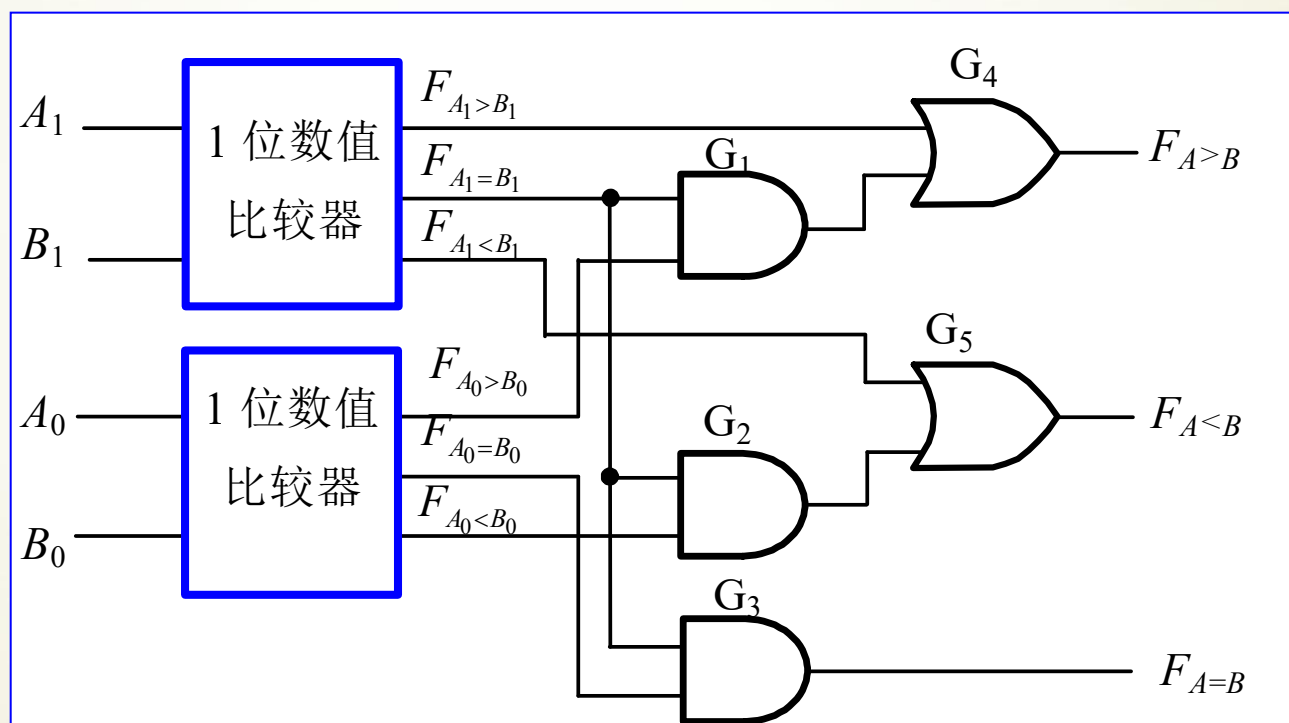
$$F_{A=B} = (A_1 = B_1)(A_0 = B_0)$$

注意：上述不是真正的逻辑函数表达式，只示意逻辑关系。

$$F_{A>B} = (A_1 > B_1) + (A_1 = B_1)(A_0 > B_0) \quad F_{A=B} = (A_1 = B_1)(A_0 = B_0)$$

$$F_{A<B} = (A_1 < B_1) + (A_1 = B_1)(A_0 < B_0)$$

两位数值比较器逻辑图



3、4 位数值比较器：

4位数值比较器功能表

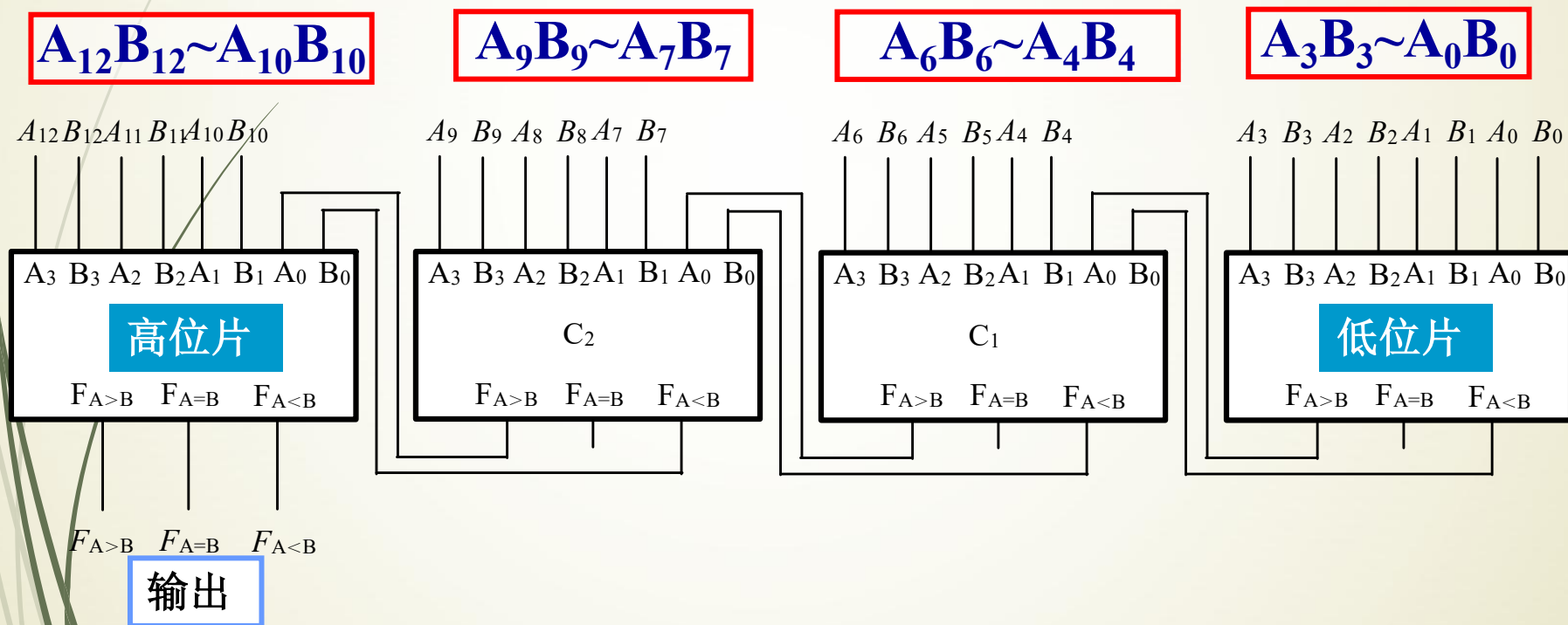
输 入				输 出		
$A_3 B_3$	$A_2 B_2$	$A_1 B_1$	$A_0 B_0$	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
$A_3 > B_3$	$\times$	$\times$	$\times$	1	0	0
$A_3 < B_3$	$\times$	$\times$	$\times$	0	1	0
$A_3 = B_3$	$A_2 > B_2$	$\times$	$\times$	1	0	0
$A_3 = B_3$	$A_2 < B_2$	$\times$	$\times$	0	1	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	$\times$	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	$\times$	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
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	0	0	1

4、数值比较器的扩展

用四个4位数值比较器组成13位数值比较器（串联扩展方式）。

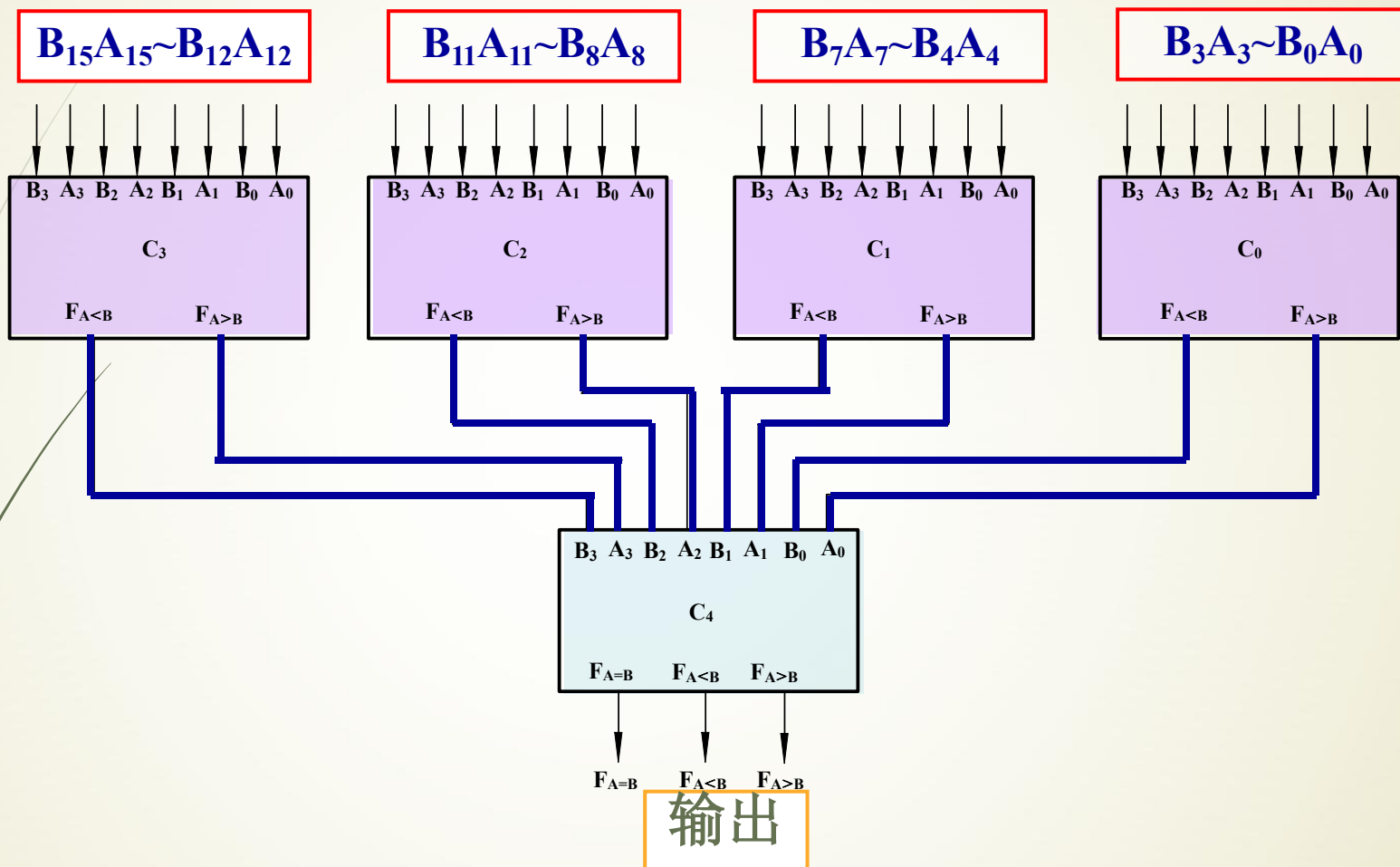
输入： $A=A_{12}A_{11}A_{10}\cdots A_3A_2A_1A_0$ $B=B_{12}B_{11}B_{10}\cdots B_3B_2B_1B_0$

输出： $F_{A>B}$ $F_{A<B}$ $F_{A=B}$



问题：如果每一片延迟时间为10ns，四片串行比较器延迟时间？

用4位数值比较器组成16位数值比较器的**并联扩展方式**。



问题：如果每一片延迟时间为10ns，16位并行比较器延迟时间？

4.4.5 算术运算电路

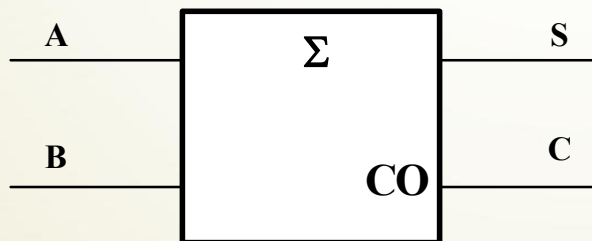
1、半加器和全加器

两个1位二进制数相加时，不考虑低位来的进位的加法
---半加

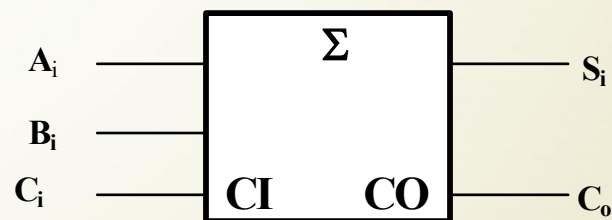
在两个1位二进制数相加时，考虑低位进位的加法
---全加

加法器分为半加器和全加器两种。

半加器



全加器



(1) 1位半加器 (Half Adder)

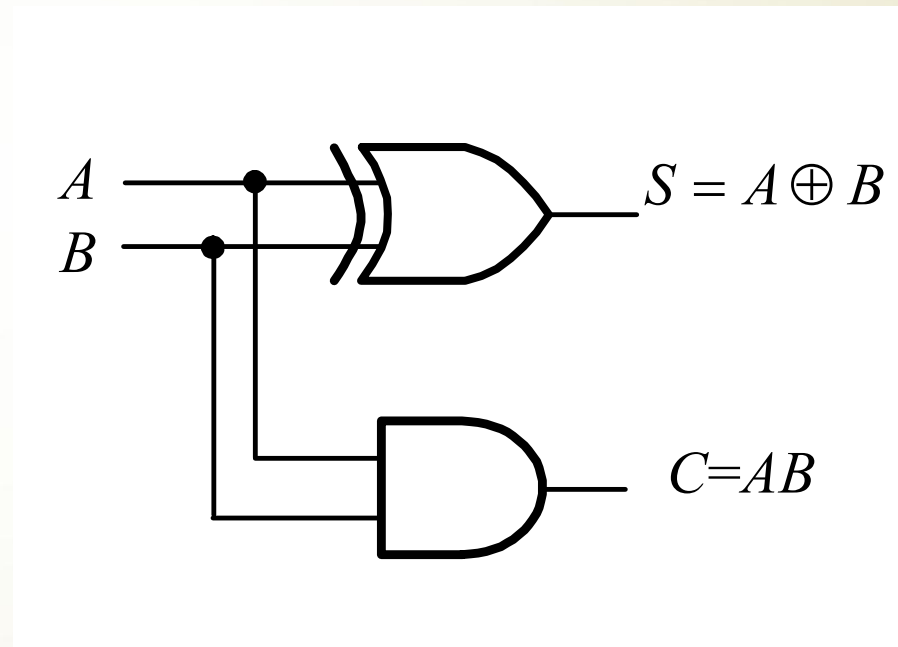
不考虑低位进位，将两个1位二进制数A、B相加的器件。

半加器的真值表

逻辑表达式

$$S = \bar{A}B + A\bar{B}$$

$$C = AB$$



如用与非门实现最少要几个门？

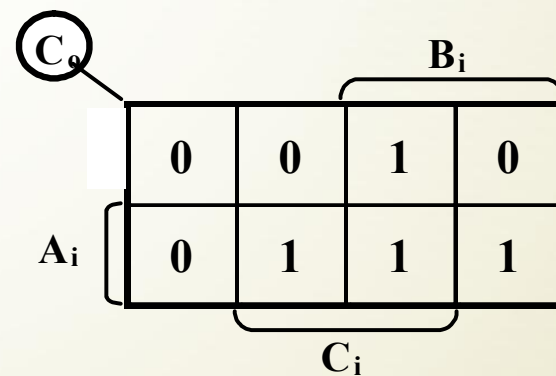
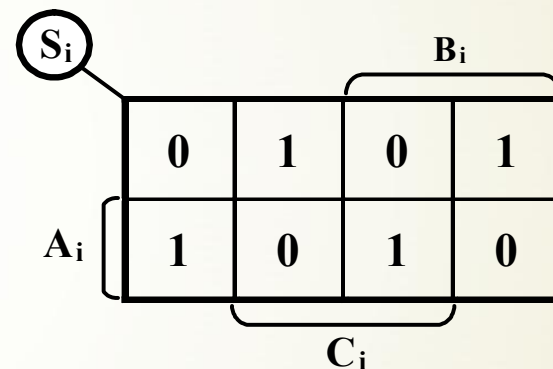
逻辑图

(2) 全加器 (Full Adder)

全加器能进行加数、被加数和低位来的进位信号相加，并根据求和结果给出该位的进位信号。

全加器真值表

A	B	C _i	S	C _o
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



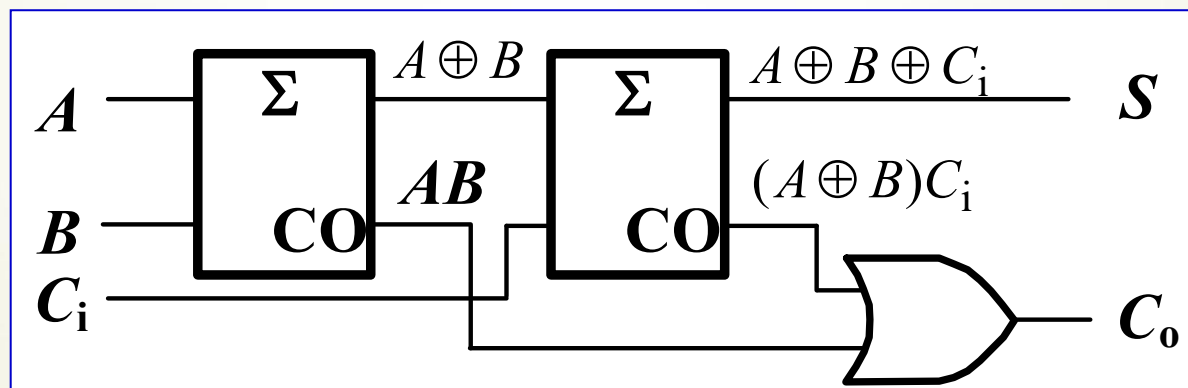
于是可得全加器的逻辑表达式为

$$S = \overline{A}\overline{B}C_i + \overline{A}B\overline{C}_i + A\overline{B}\overline{C}_i + ABC_i$$

$$= A \oplus B \oplus C_i$$

$$C_o = AB + \overline{A}\overline{B}C_i + \overline{A}BC_i$$

$$= AB + (A \oplus B)C_i$$



加法器的应用

全加器真值表

A	B	C_i	S	C_o
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

ABC_i 有奇数个1时 S 为1;

ABC_i 有偶数个1和全为0时
 S 为0。

----用全加器组成三位二进制
代码奇偶校验器

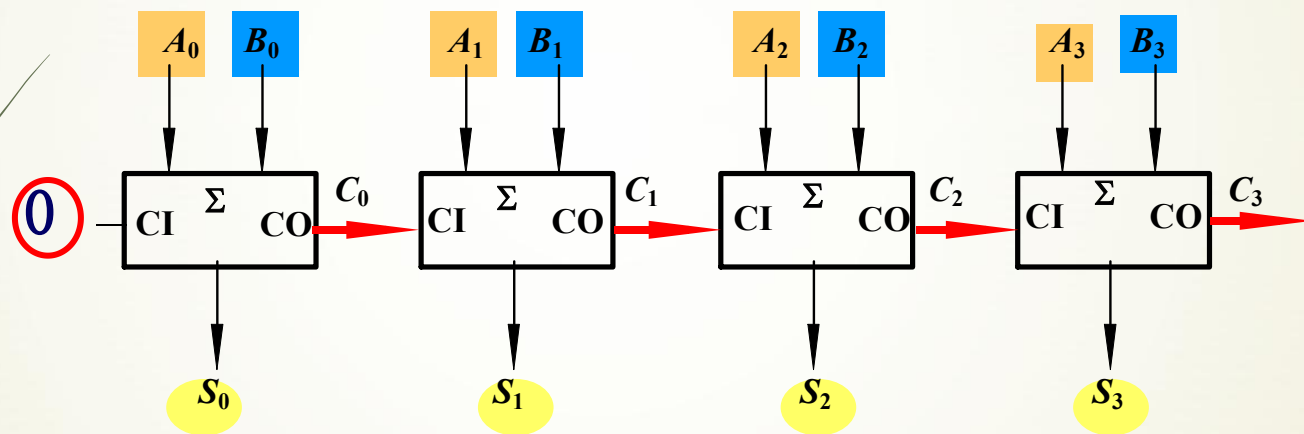
用全加器组成八位二进制代码
奇偶校验器，电路应如何连接？

2、多位数加法器

◆ 如何用1位全加器实现两个四位二进制数相加？

$$A_3 A_2 A_1 A_0 + B_3 B_2 B_1 B_0 = ?$$

(1) 串行进位加法器



◆ 低位的进位信号送给邻近高位作为输入信号，采用串行进位加法器运算速度不高。

(2) 超前进位加法器

提高运算速度的基本思想：设计进位信号产生电路，在输入每位的加数和被加数时，同时获得该位全加的进位信号，而无需等待最低位的进位信号。

定义第 i 位的进位信号 (C_i)：

$$C_i = A_i B_i + (A_i \oplus B_i) C_{i-1}$$

定义两个中间变量 G_i 和 P_i ：

$$G_i = A_i B_i$$

$$P_i = (A_i \oplus B_i)$$

$$C_i = G_i + P_i C_{i-1}$$

$$S_i = A_i \oplus B_i \oplus C_{i-1}$$

4位全加器进位信号的产生:

由于 $C_i = G_i + P_i C_{i-1}$ $[G_i = A_i B_i \quad p_i = (A_i \oplus B_i)]$

$$C_0 = G_0 + P_0 C_{-1}$$

$$C_1 = G_1 + P_1 C_0$$

$$C_1 = G_1 + P_1 G_0 + P_1 P_0 C_{-1}$$

$$C_2 = G_2 + P_2 C_1$$

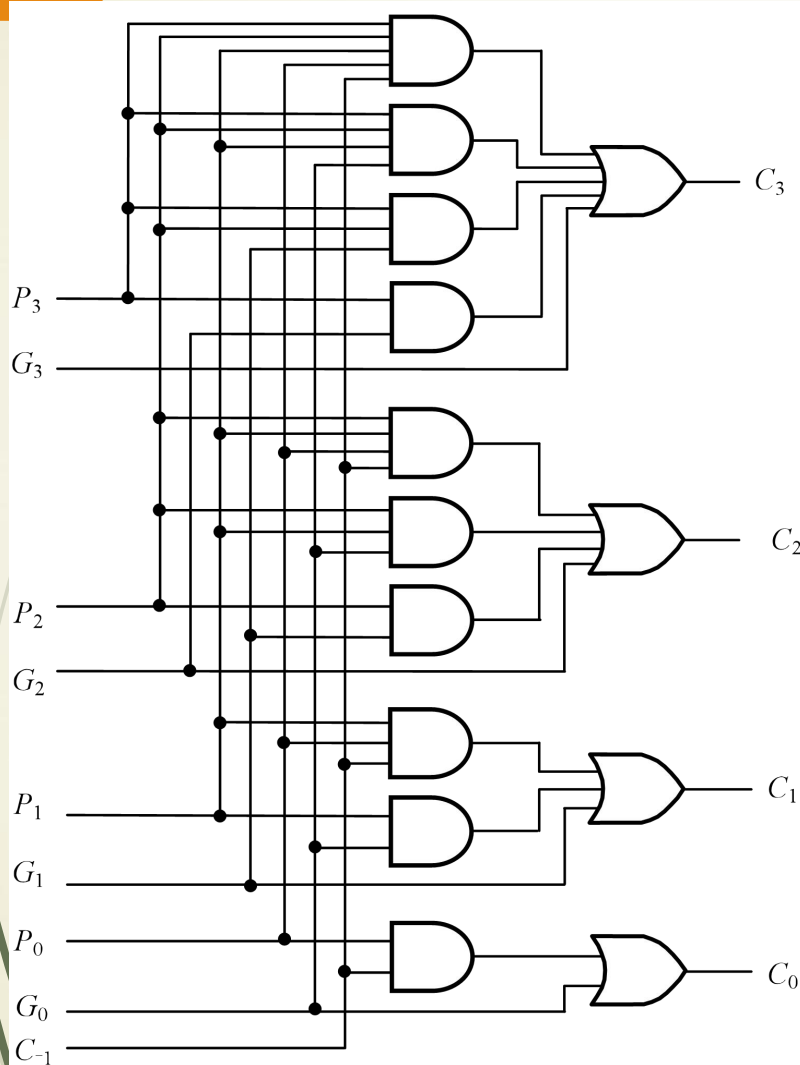
$$C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{-1}$$

$$\begin{aligned} C_3 &= G_3 + P_3 C_2 = G_3 + P_3 (G_2 + P_2 C_1) = G_3 + P_3 G_2 + P_3 P_2 C_1 \\ &= G_3 + P_3 G_2 + P_3 P_2 (G_1 + P_1 C_0) \end{aligned}$$

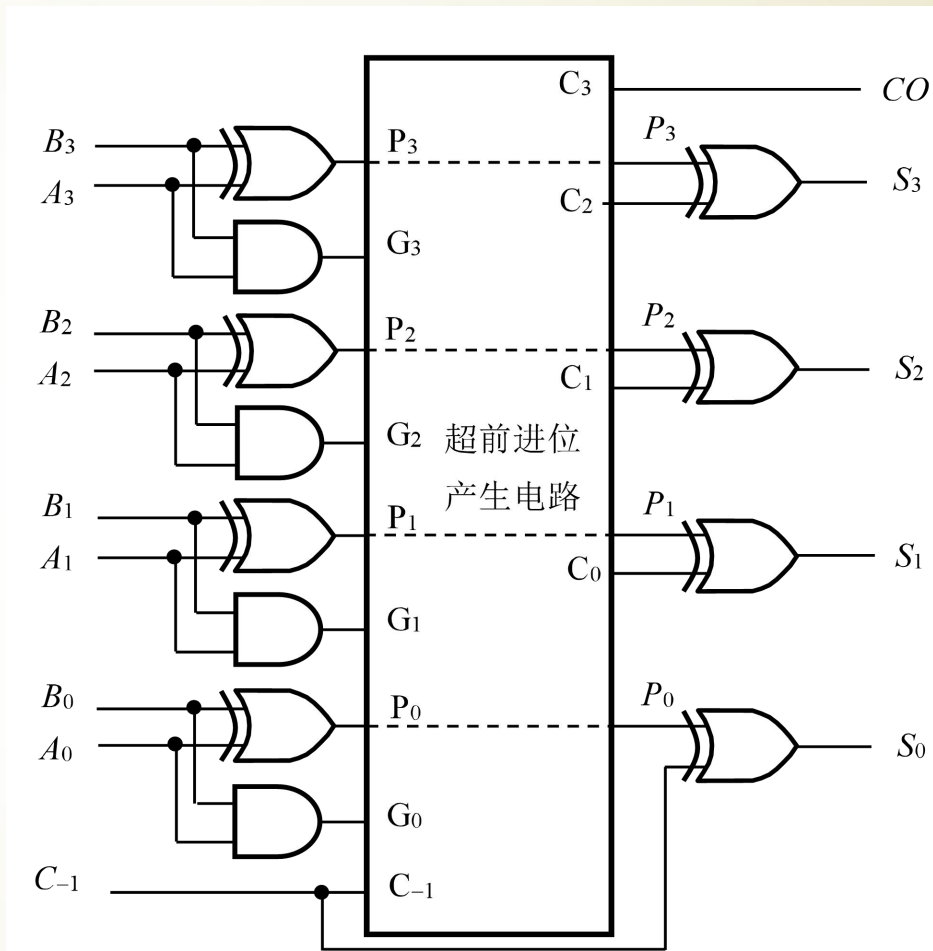
$$C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 (G_0 + P_0 C_{-1})$$

进位信号只由被加数、加数和C-1决定，而与其它低位的进位无关。提高了速度，但位数增加时，进位电路复杂度增加。

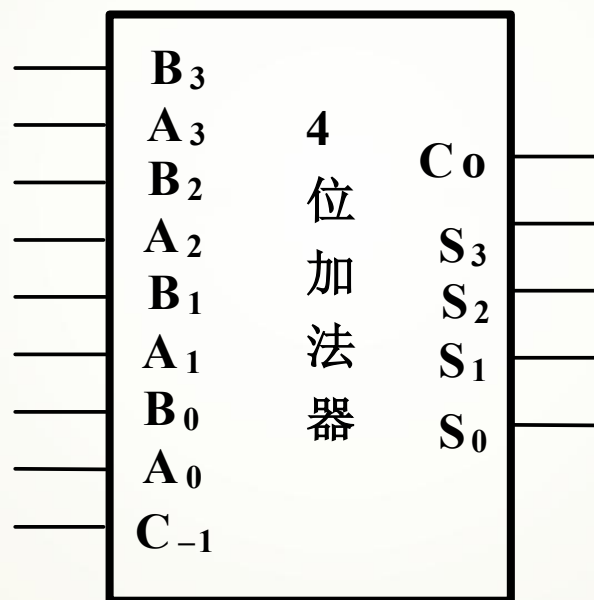
超前进位产生电路



4位超前进位加法器

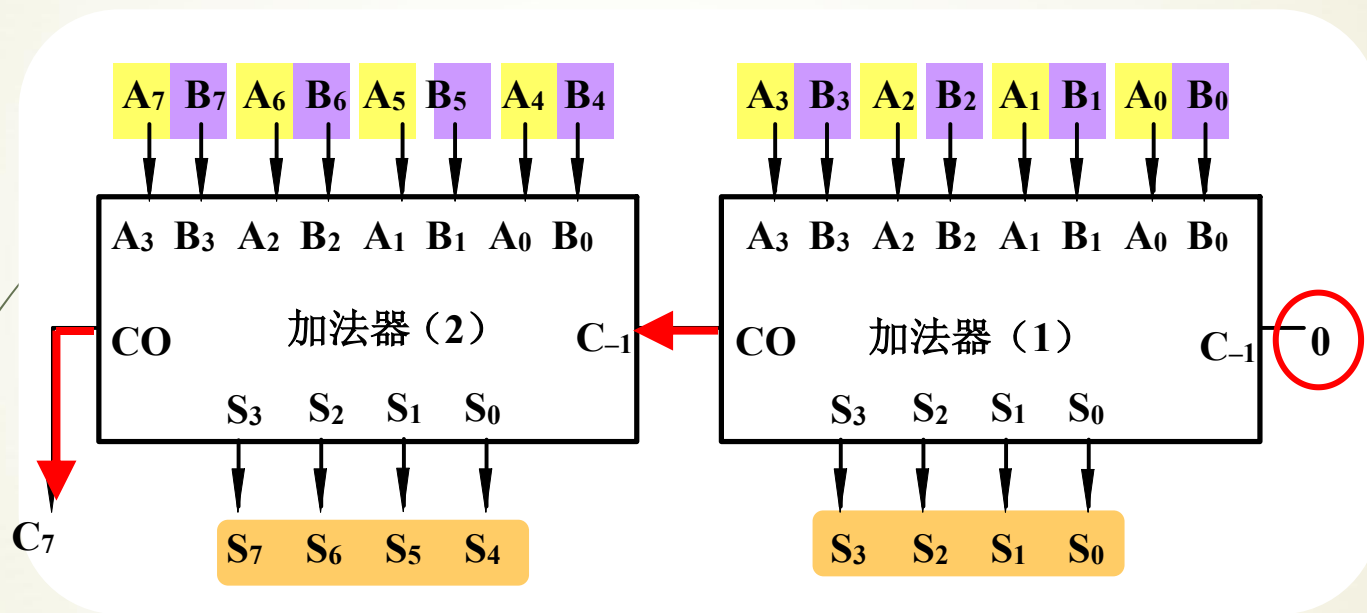


4位加法器框图



(3) 4位超前进位加法器的应用

例1. 用两片4位加法器构成一个8位二进制数加法器。

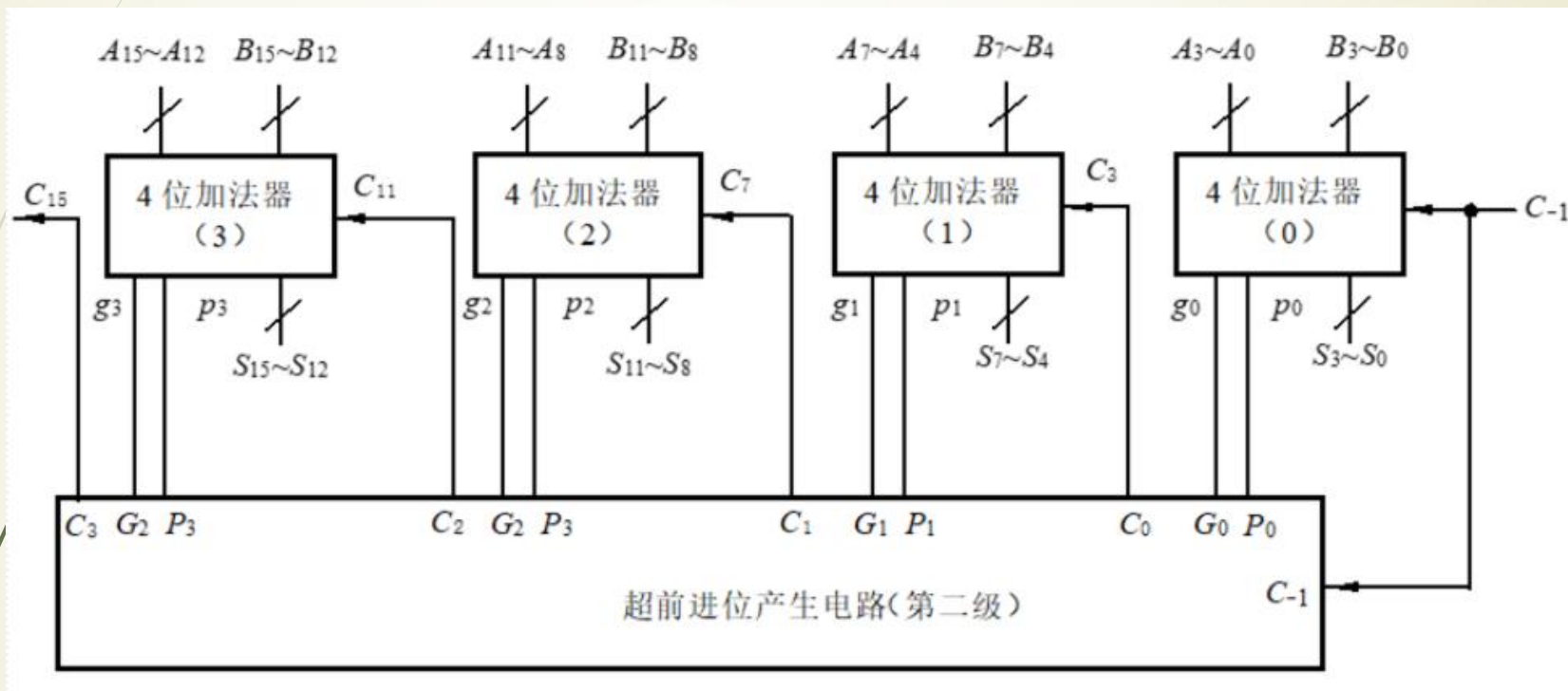


在片内是超前进位，而片与片之间是串行进位。

问题：如果每一片延迟时间为 $4t_{pd}$ ，4片串行进位延迟时间？

(3) 4位超前进位加法器的应用

例2. 用四片4位加法器构成一个16位二进制数加法器。



在片内是超前进位，而片与片之间是超前进位。

延迟问题：共8级门延迟(书P193) $8 t_{pd}$ 。当级数增加，延迟不增加。

例. 用4位加法器构成将8421BCD码转换为余3码的码制转换电路。

8421码

0000

0001

0010

⋮

+0011

+0011

+0011

余3码

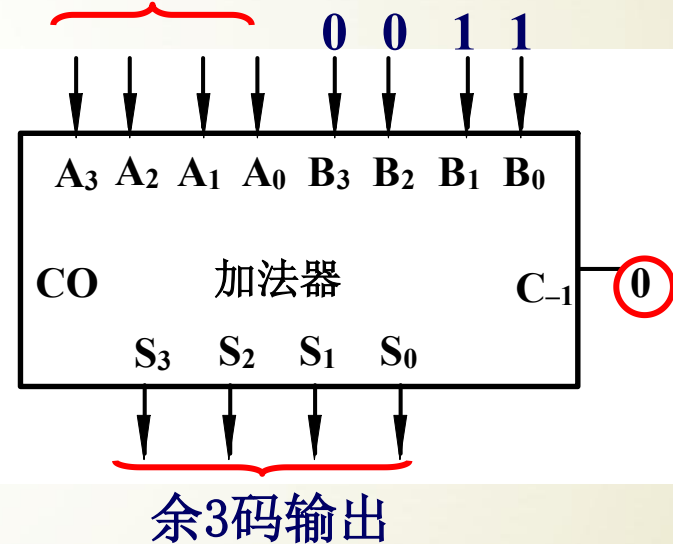
0011

0100

0101

⋮

8421码输入



3、 减法运算

若 n 位二进制的原码为 $N_{\text{原}}$ ，则与它相对应的2 的补码为

$$N_{\text{补}} = 2^N - N_{\text{原}}$$

补码与反码的关系式

$$N_{\text{补}} = N_{\text{反}} + 1$$

设两个数 A 、 B 相减，利用以上两式
可得

$$A - B = A + B_{\text{补}} - 2^n = A + B_{\text{反}} + 1 - 2^n$$

在实际应用中，通常是将减法运算变为加法运算来处理，即采用加补码的方法完成减法运算。

1) $A-B \geq 0$ 的情况。

2) $A-B < 0$ 的情况。

$A=0101$, $B=0010$

$$\begin{array}{rcccccl} & \boxed{0} & 1 & 0 & 1 & A \\ & \boxed{1} & 1 & 0 & 1 & B_{\text{反}} \\ + & & & & 1 & \\ \hline 1 & \boxed{0} & 0 & 1 & 1 & \end{array}$$

舍弃

当 $A-B \geq 0$ 时，舍弃的进位为1，所得结果就是差的原码，不需再求反补。

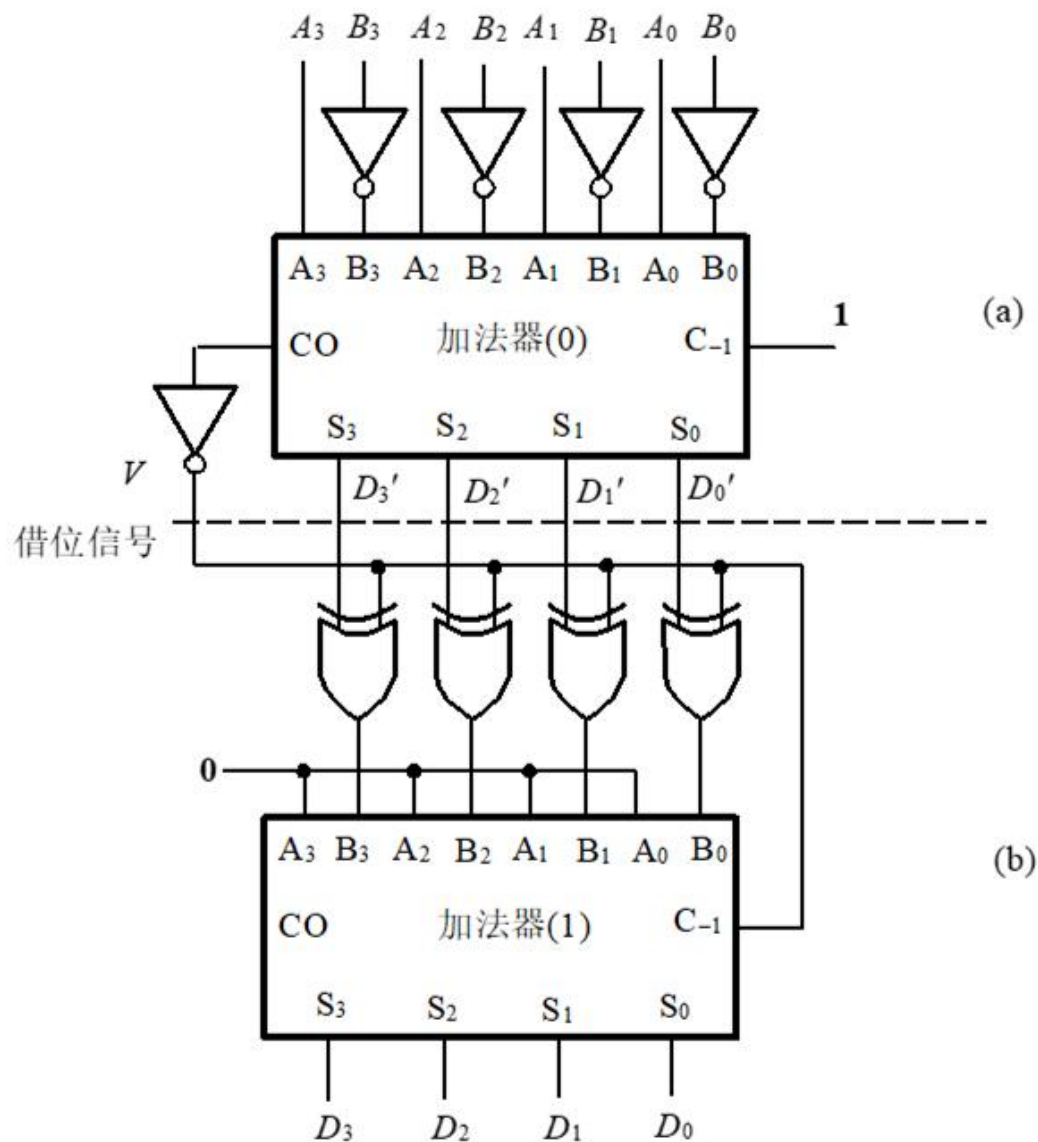
$A=0010$, $B=0101$

$$\begin{array}{rcccccl} & \boxed{0} & 0 & 1 & 0 & A \\ & \boxed{1} & 0 & 1 & 0 & B_{\text{反}} \\ + & & & & 1 & \\ \hline 0 & \boxed{1} & 1 & 0 & 1 & \end{array}$$

舍弃

当 $A-B < 0$ 时，舍弃的进位为0，所得结果是补码，要得到原码需再求补。

输出为原码的4位减法运算逻辑图



4.5 组合逻辑的可编程电路实现

4.5.1 PLD的电路表示、编程技术及分类

4.5.2 组合逻辑电路的PLD实现

4.5 组合逻辑的可编程电路实现

可编程逻辑器件是一种可以由用户定义和设置逻辑功能的器件。该类器件具有逻辑功能实现灵活、集成度高、处理速度快和可靠性高等特点。

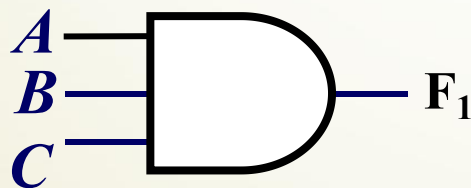
4.5.1 PLD的电路表示、编程技术及分类

1、基本门电路的表示方式

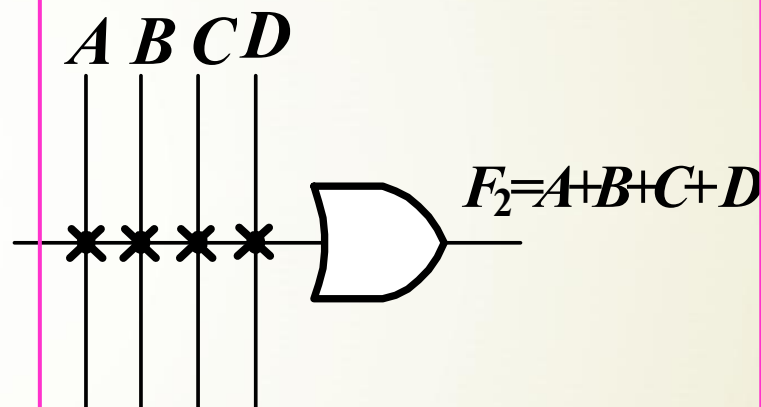
与门



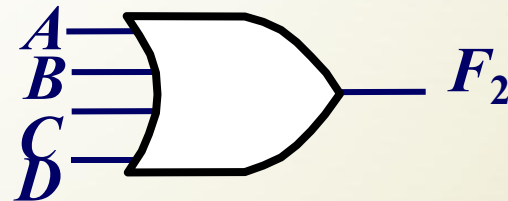
$$F_1 = A \cdot B \cdot C$$



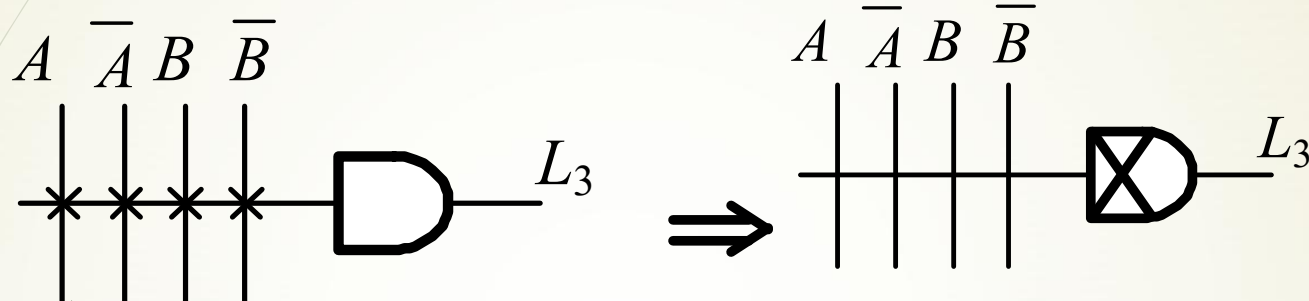
或门



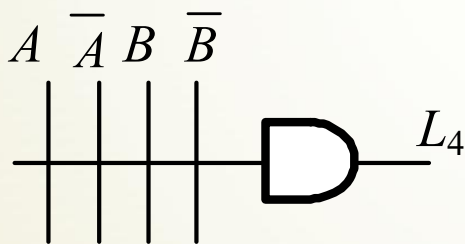
$$F_2 = A + B + C + D$$



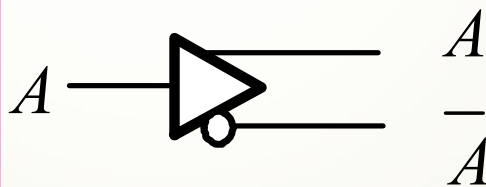
1、基本门电路的表示方式



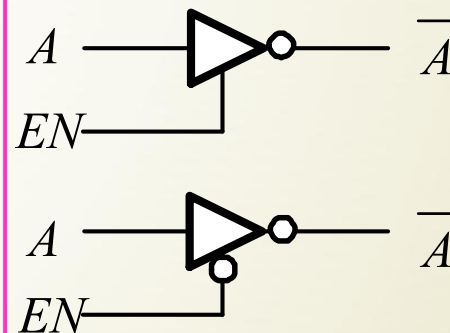
输出恒等于0的与门



输出为1的与门

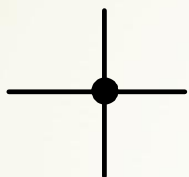


输入缓冲器

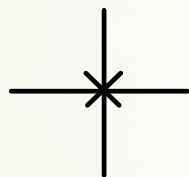


三态输出缓冲器

2. 连接方式



硬线连接单元，不可以编程改变。



被编程接通单元

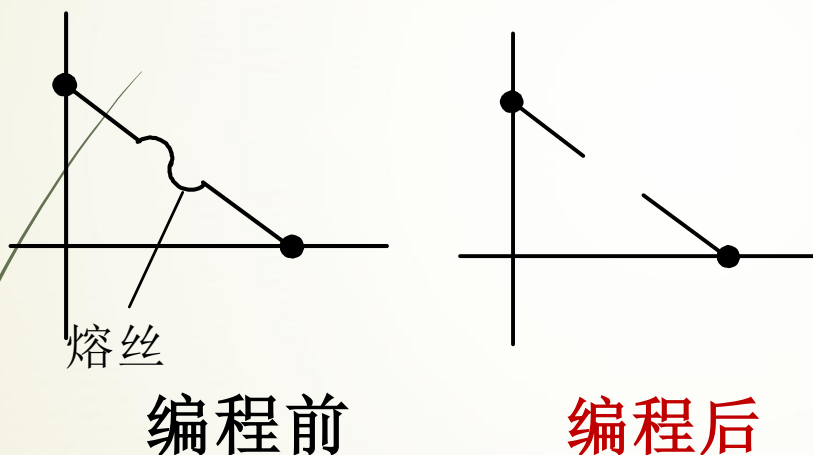


被编程擦除单元

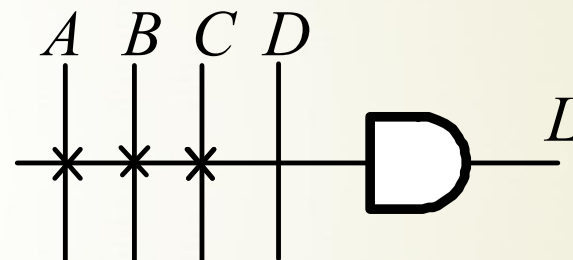
3. 编程连接技术

在PLD中的可编程连接单元主要有熔丝型、反熔丝型和浮栅MOS型

(1) 熔丝型



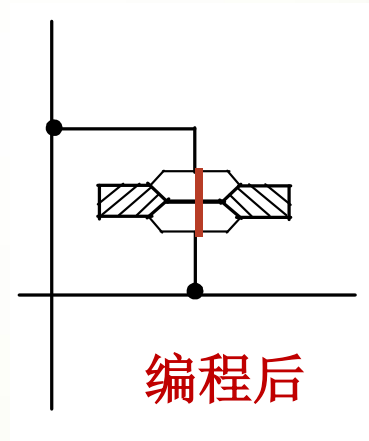
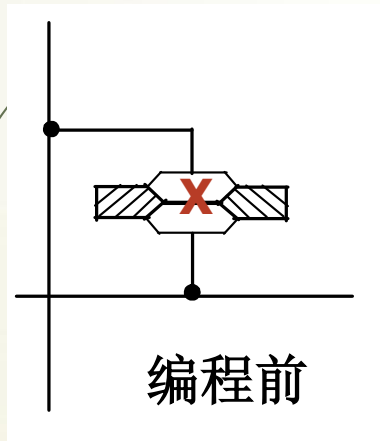
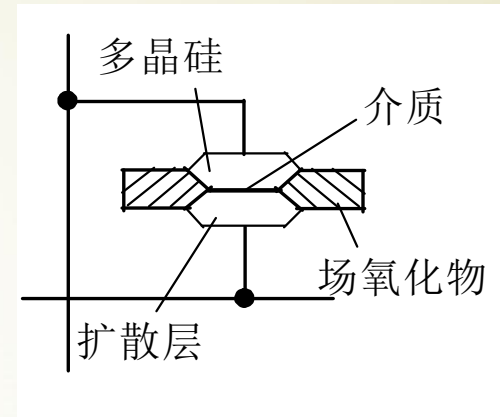
PLD表示的与门



没有编程时，单元由金属熔丝连通。在编程时，用比工作电流大许多的电流，将**不需要连通**的熔丝烧断。

(2) 反熔丝型

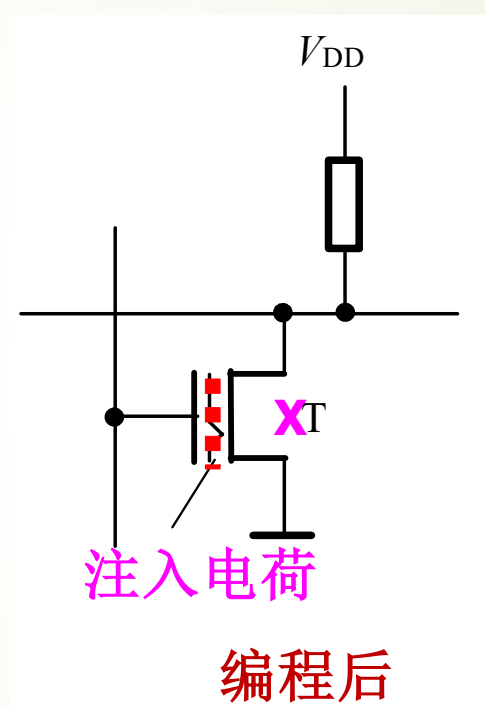
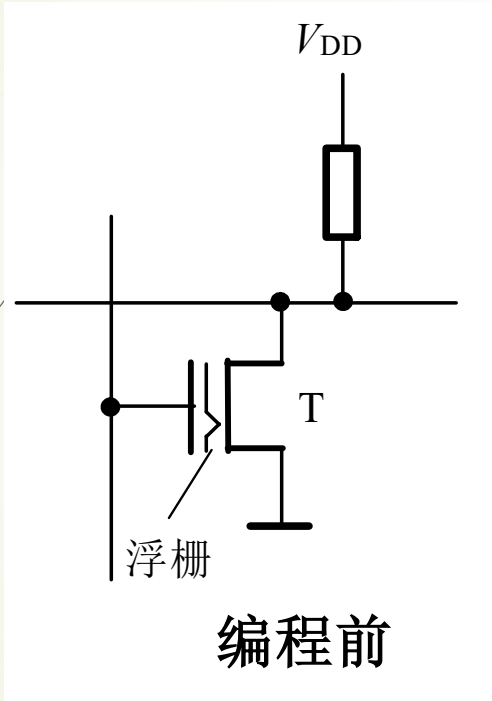
多晶硅层和扩散层两个导体之间，有一层很薄的绝缘介质层，未编程时呈现高阻抗，相当于**断开**状态。



编程时利用**高压**将介质层击穿，呈现**导通**状态。

(3) 浮栅MOS管型

编程前，浮栅上没有电荷的MOS管，与普通增强型NMOS管一样,当栅极加正常逻辑高电平时，MOS管导通,否则截止。



经编程向浮栅上注入电子后，栅极加正常高电平或低电平，MOS管始终相当于断开。如果将浮栅上的电子释放，可以使MOS管恢复正常。即可多次编程。

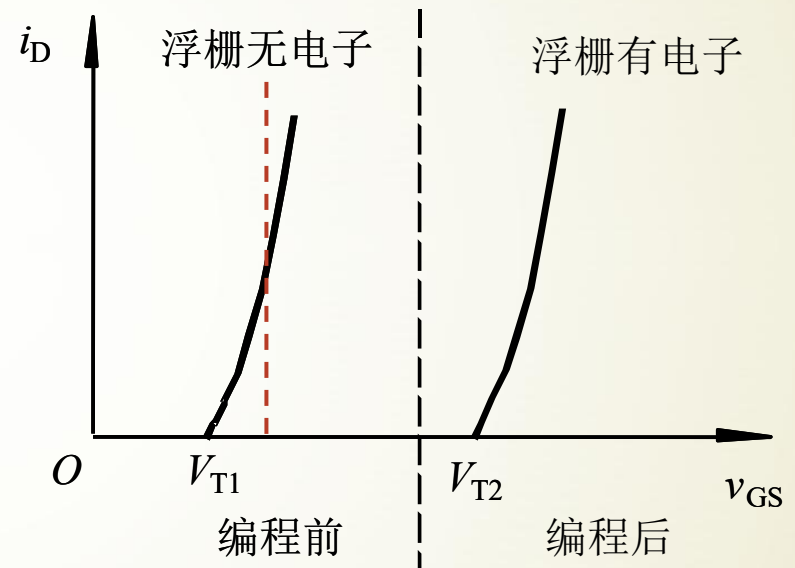
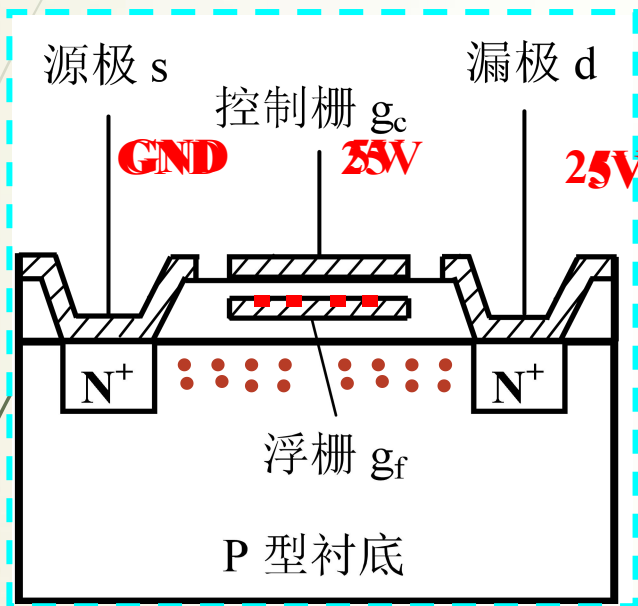
(4) 浮栅MOS管开关



用不同的浮栅MOS管连接的PLD，编程信息的擦除方法也不同。SIMOS管连接的PLD，采用紫外光照射擦除；Flotox MOS管和快闪叠栅MOS管，采用电擦除方法。

a. 叠栅注入MOS(SIMOS)管

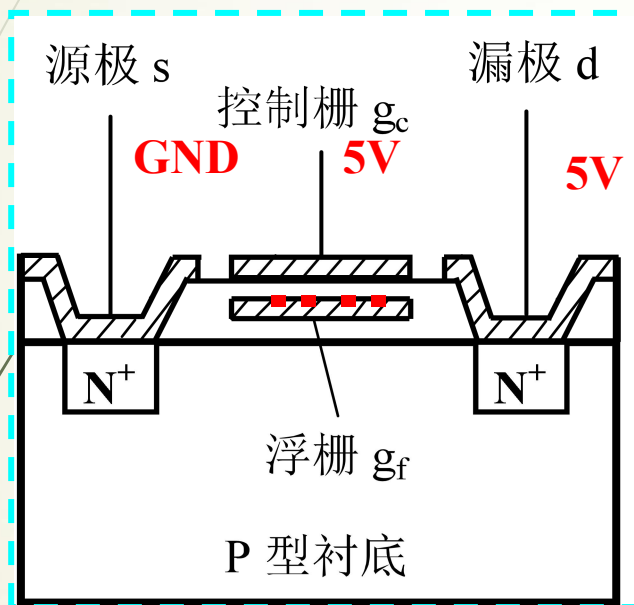
当浮栅上没有电荷时，给控制栅加上大于 V_{T1} 的控制电压，MOS管导通。



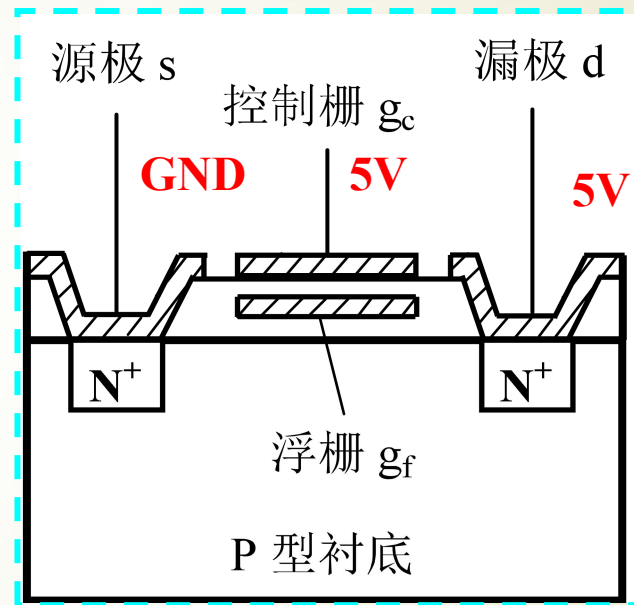
若要擦除，可用紫外线或X射线，距管子2厘米处照射15-20分钟。

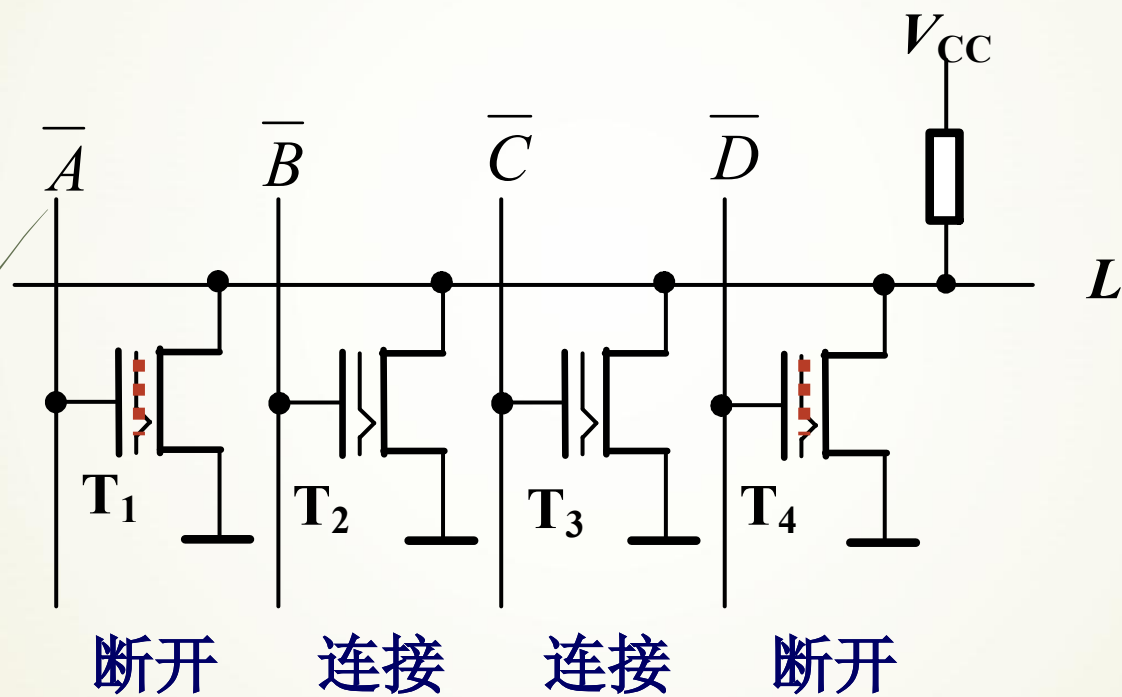


截止



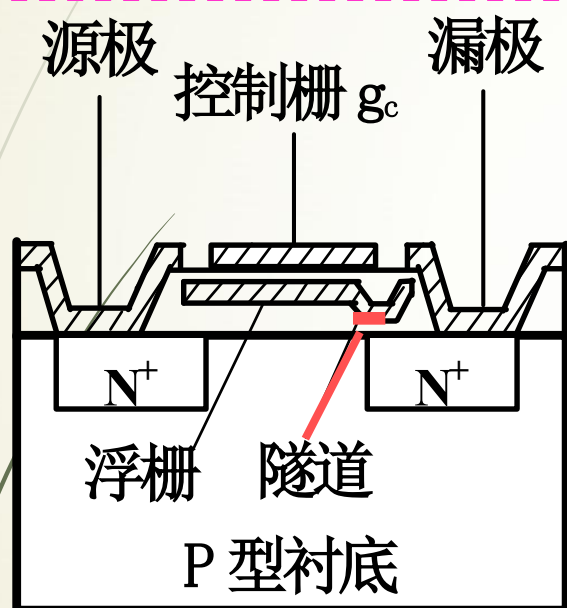
导通





$$L=B \cdot C$$

b.浮栅隧道氧化层MOS(Flotox MOS)管



浮栅延长区与漏区 N^+ 之间的交叠处有一个厚度约为80Å (埃)的薄绝缘层——隧道区。

当隧道区的电场强度大到一定程度，使漏区与浮栅间出现导电隧道，形成电流将浮栅电荷泄放掉。

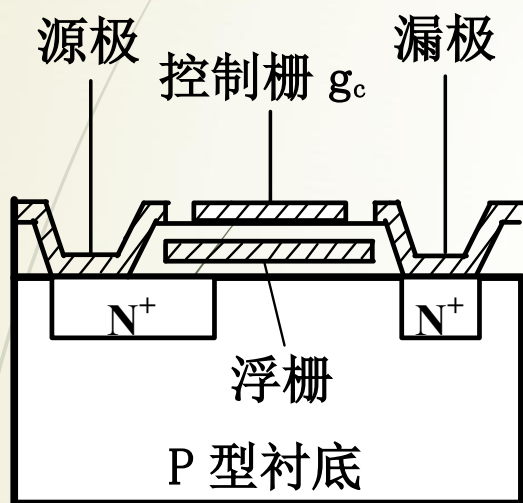
隧道MOS管是用电擦除的，擦除速度快。

c. 快闪叠栅MOS管开关 (Flash Memory) (自学)

结构特点:

1. 闪速存储器存储单元MOS管的源极N+区大于漏极N+区, 而SIMOS管的源极N+区和漏极N+区是对称的;

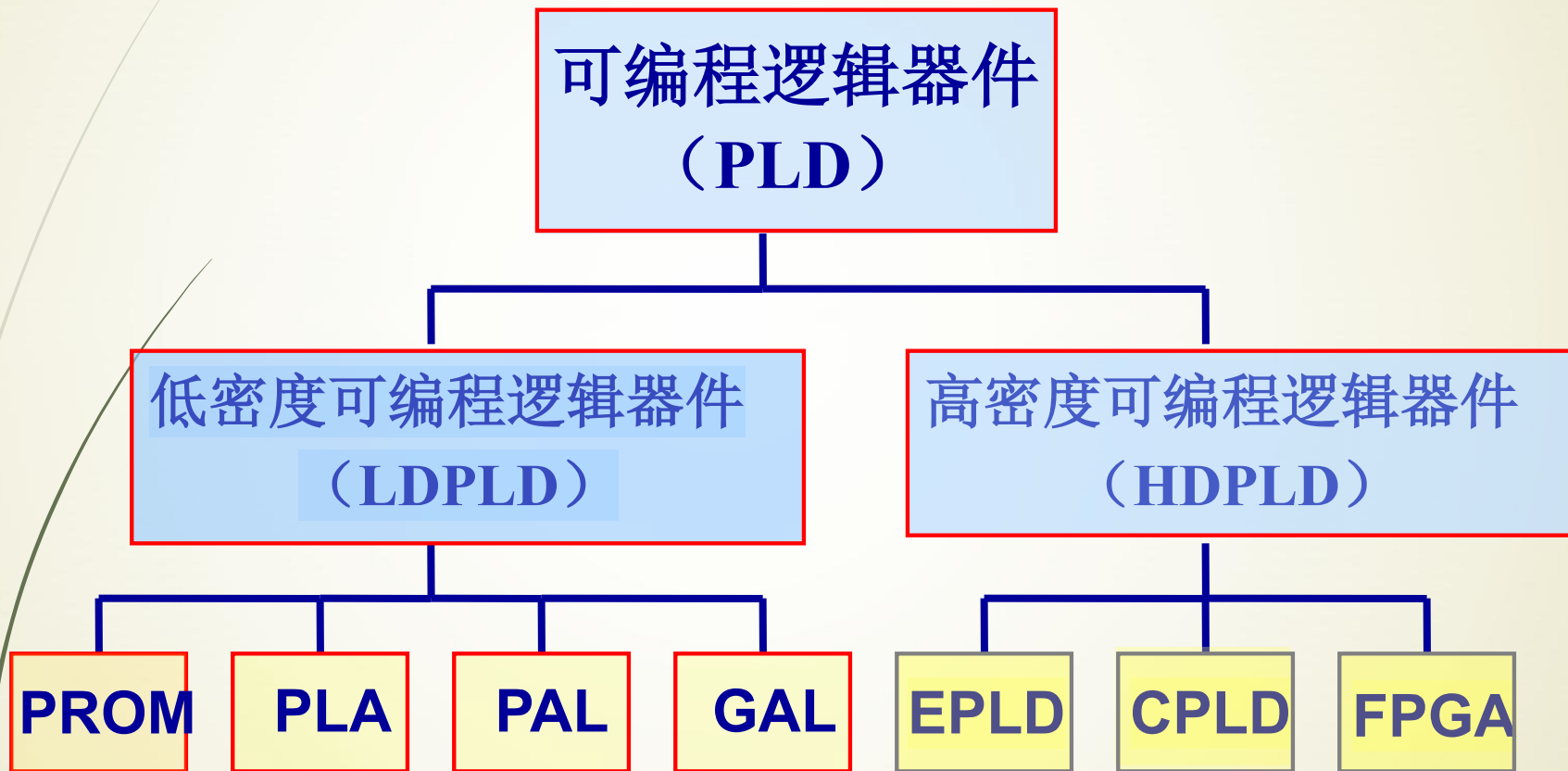
2. 浮栅到P型衬底间的氧化绝缘层比SIMOS管的更薄。



特点: 结构简单、集成度高、编程可靠、擦除快捷。

4. PLD的分类

(1) 按集成密度划分为





(2) 按结构特点划分

- 简单PLD (PAL, GAL)

- 复杂的可编程器件(CPLD) :

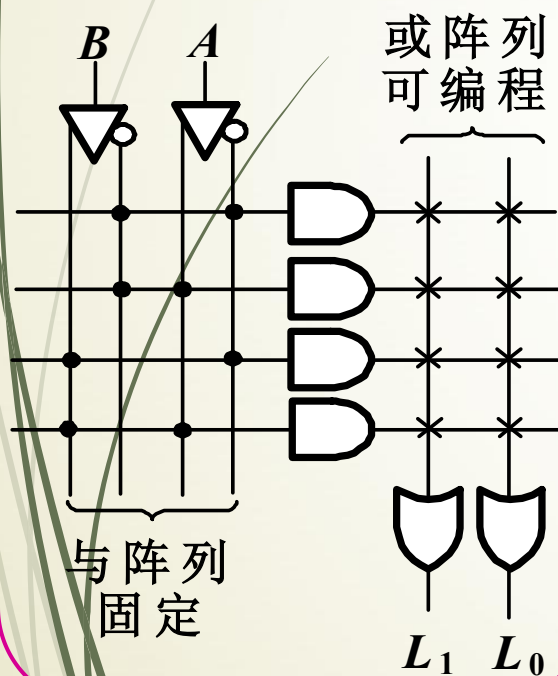
CPLD的代表芯片如: Xilinx的XC9500系列

- 现场可编程门阵列(FPGA)

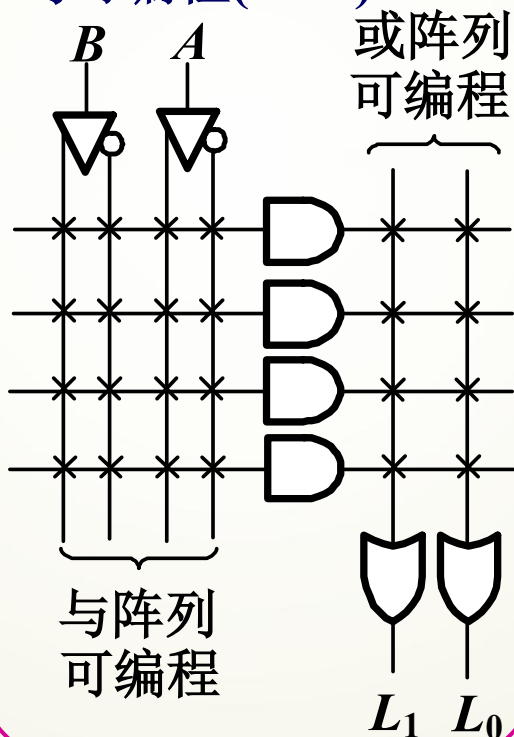
(3) 按PLD中的与、或阵列是否编程分

PLD中的三种与、或阵列

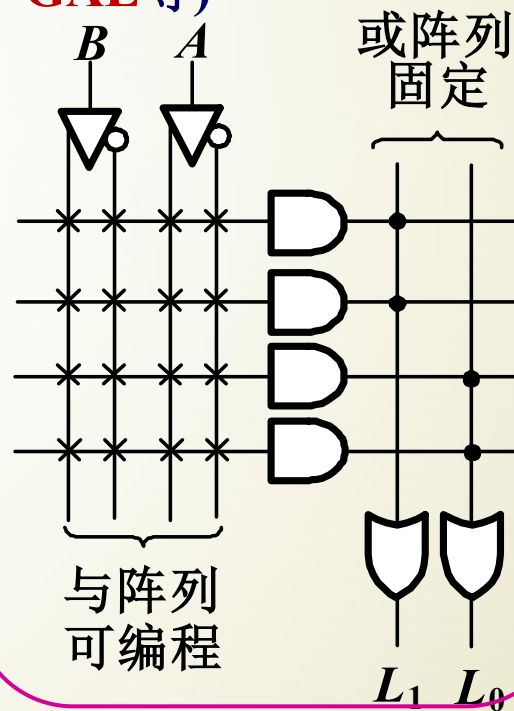
与阵列固定，或阵列可编程(**PROM**)



与阵列、或阵列均可编程(**PLA**)



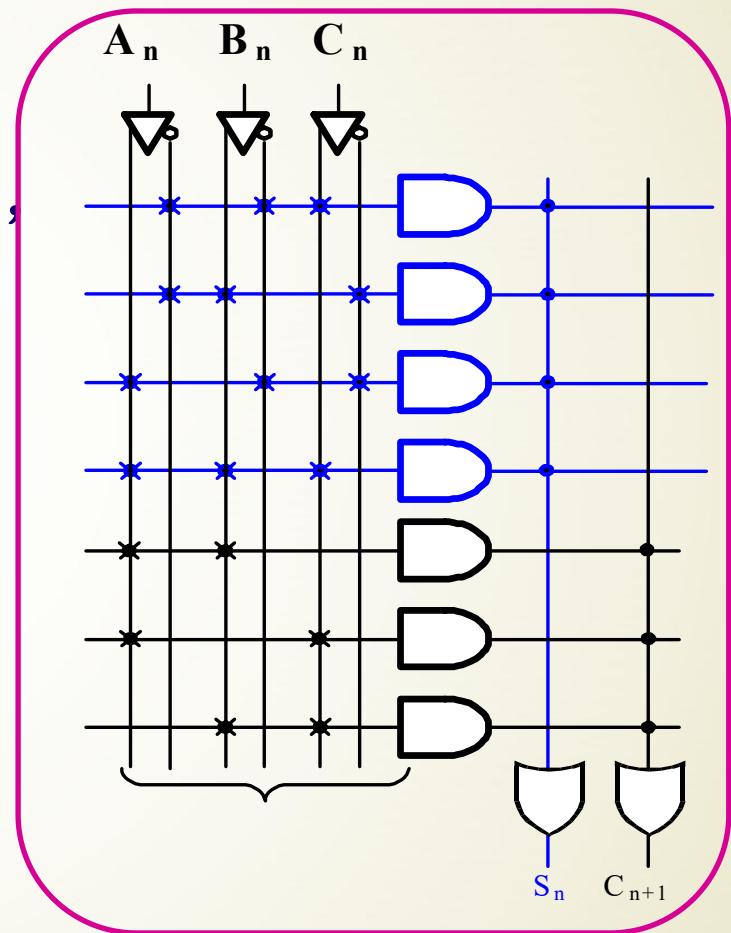
与阵列可编程，或阵列固定(**PAL**和**GAL**等)



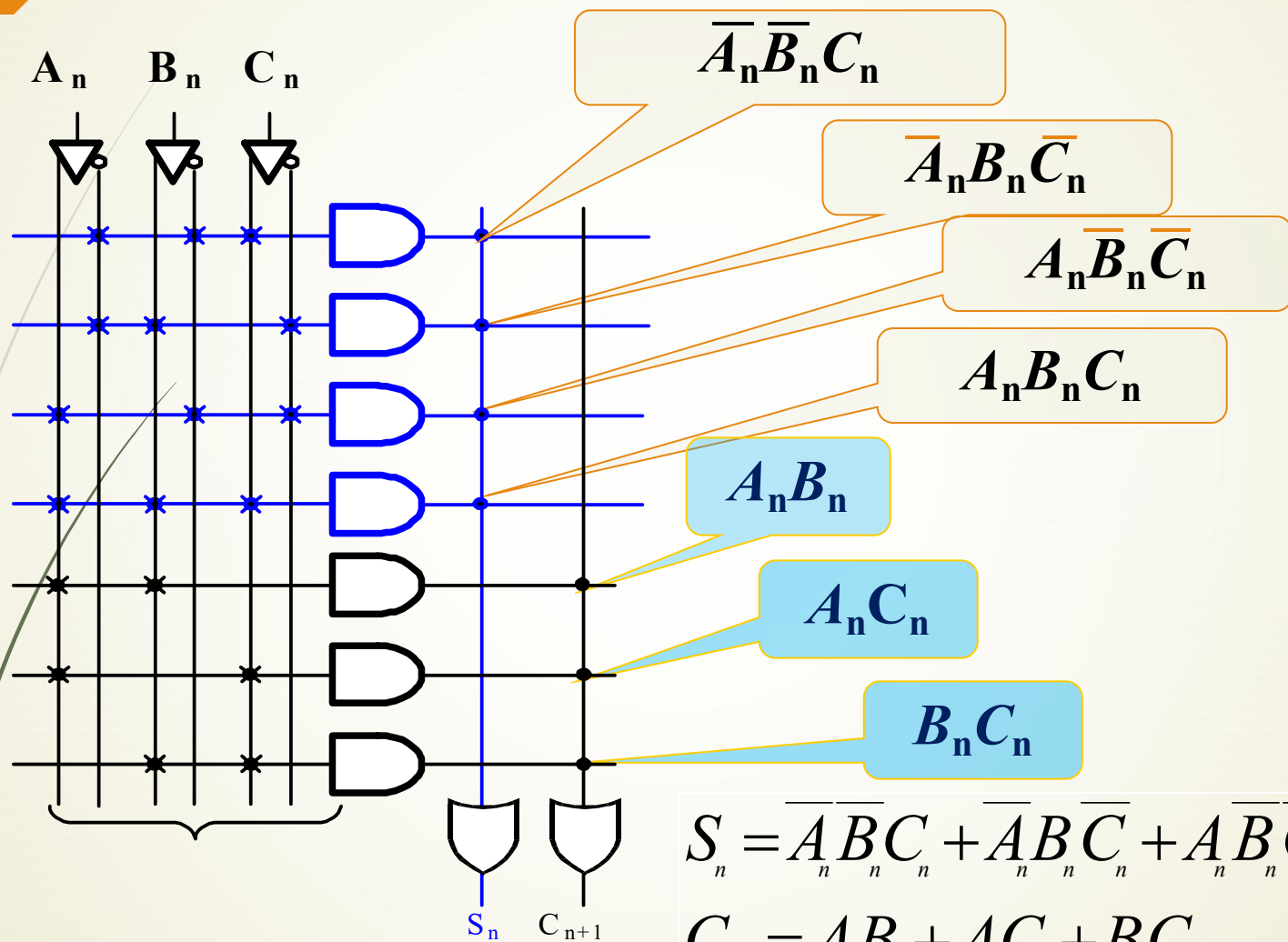
4.5.2 组合逻辑电路的 PLD 实现

例1 由PLA构成的逻辑电路如图所示，试写出该电路的逻辑表达式，并确定其逻辑功能。

写出该电路的逻辑表达式：



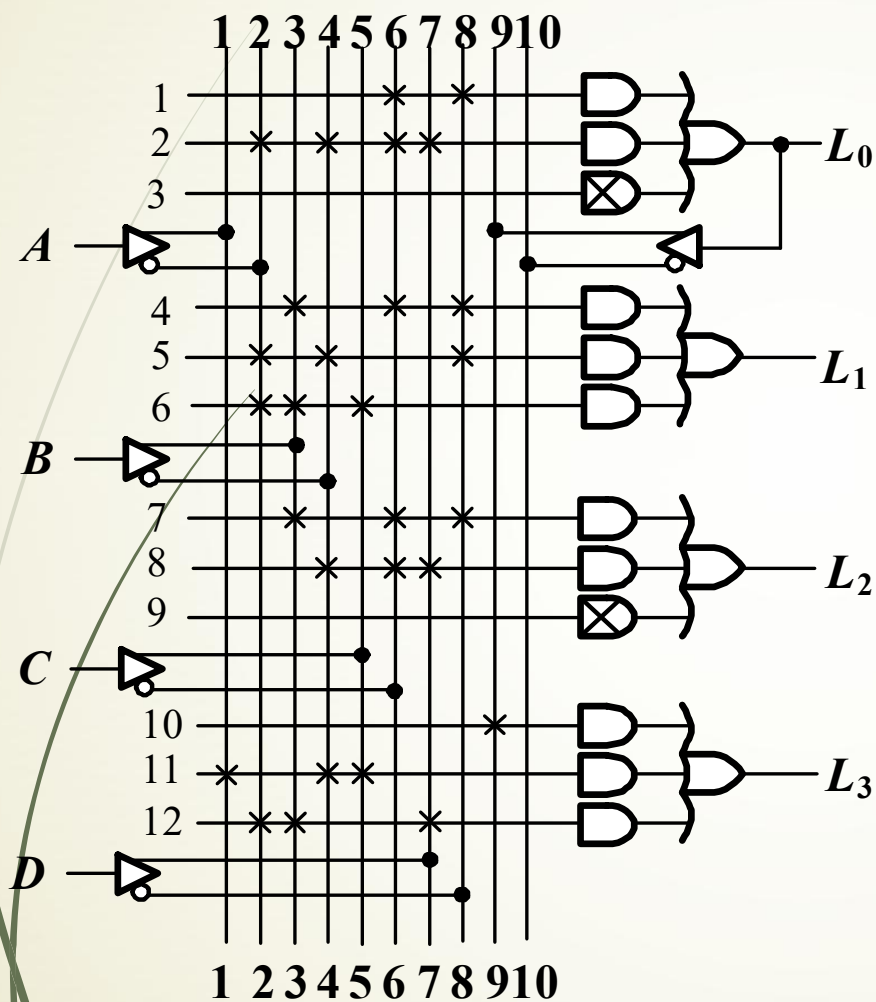
全加器



$$S_n = \bar{A}_n \bar{B}_n C_n + \bar{A}_n B_n \bar{C}_n + A_n \bar{B}_n \bar{C}_n + A_n B_n C_n$$

$$C_{n+1} = A_n B_n + A_n C_n + B_n C_n$$

试写出该电路的逻辑表达式。



$$L_0 = \overline{\overline{C}}\overline{\overline{D}} + \overline{\overline{A}}\overline{\overline{B}}\overline{\overline{C}}\overline{\overline{D}}$$

$$L_1 = \overline{\overline{B}}\overline{\overline{C}}\overline{\overline{D}} + \overline{\overline{A}}\overline{\overline{B}}\overline{\overline{D}} + \overline{\overline{A}}\overline{\overline{B}}\overline{\overline{C}}$$

$$L_2 = \overline{\overline{B}}\overline{\overline{C}}\overline{\overline{D}} + \overline{\overline{B}}\overline{\overline{C}}\overline{\overline{D}}$$

$$L_3 = L_0 + \overline{\overline{A}}\overline{\overline{B}}\overline{\overline{C}} + \overline{\overline{A}}\overline{\overline{B}}\overline{\overline{D}}$$

4.6 用VerilogHDL描述组合逻辑电路

4.6.1 组合逻辑电路的行为级建模

4.6.2 分模块、分层次的电路设计

4.6.1 组合逻辑电路的行为级建模

组合逻辑电路的行为级描述一般使用**assign**结构和过程赋值语句、条件语句（**if-else**）、多路分支语句（**case-endcase**）和**for**循环语句等。

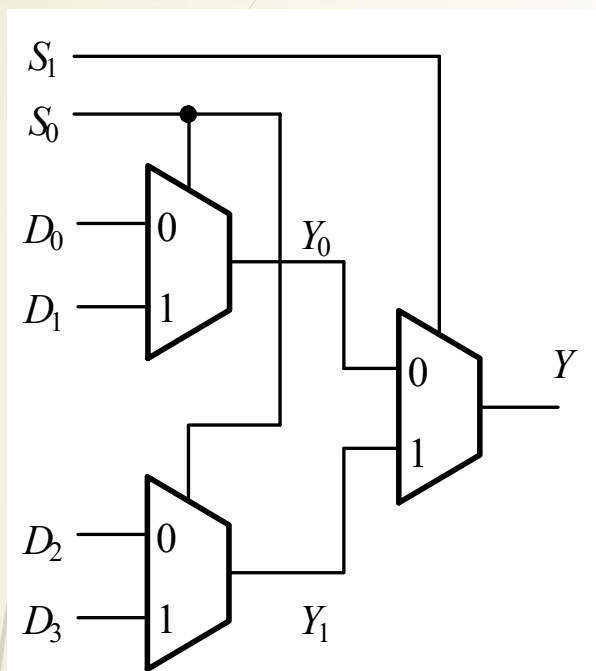
1、条件语句（if语句）

条件语句就是根据判断条件是否成立，确定下一步的运算。
Verilog语言中有3种形式的if语句：

- (1) `if (condition_expr) true_statement;`
- (2) `if (condition_expr) true_statement;`
`else false_statement;`
- (3) `if (condition_expr1) true_statement1;`
`else if (condition_expr2) true_statement2;`
`else if (condition_expr3) true_statement3;`
`.....`
`else default_statement;`

if后面的条件表达式一般为逻辑表达式或关系表达式。执行if语句时，首先计算表达式的值，若结果为0、x或z，按“假”处理；若结果为1，按“真”处理，并执行相应的语句。

例：使用if-else语句对4选1数据选择器的行为进行描述



```
module mux4to1_bh(  
    input [3:0] D,  
    input [1:0] S,  
    output reg Y  
); //输入输出端口及变量数据类型  
always @(D, S) //电路功能描述  
    if (S == 2'b00)    Y = D[0];  
    else if (S == 2'b01) Y = D[1];  
    else if (S == 2'b10) Y = D[2];  
    else                Y = D[3];  
endmodule
```

注意，过程赋值语句只能给寄存器型变量赋值，因此，输出变量Y的数据类型定义为**reg**。

2、多路分支语句（case语句）

是一种多分支条件选择语句，一般形式如下

```
case (case_expr)
    item_expr1: statement1;
    item_expr2: statement2;
    .....
    default: default_statement; //default语句可以省略
endcase
```

注意：当分支项中的语句是多条语句，必须在最前面写上关键词begin，在最后写上关键词end，成为顺序语句块。

另外，用关键词casex和casez表示含有无关项x和高阻z的情况。

例：对具有使能端En 的4选1数据选择器行为进行Verilog描述。
当En=0时，数据选择器工作，En=1时，禁止工作，输出为0。

```
module mux4to1_bh (  
    input [3:0] D, [1:0] S, En,  
    output reg Y  
);  
    always @(D, S, En) //2001, 2005 syntax  
begin  
    if (En==1) Y = 0; //En=1时，输出为0  
    else          //En=0时，选择器工作  
        case (S)  
            2'd0: Y = D[0];  
            2'd1: Y = D[1];  
            2'd2: Y = D[2];  
            2'd3: Y = D[3];  
        endcase  
    end  
endmodule
```

例：对基本的4线-2线优先编码器的行为进行Verilog描述。

```
module priority(  
    input [3:0] W,  
    output reg [1:0] Y  
);  
    always @(W)  
        case (W)  
            4'b1xxx: Y = 3;  
            4'b01xx: Y = 2;  
            4'b001x: Y = 1;  
            4'b0001: Y = 0;  
            default: Y = 2'bxx; //W无效时,Y为高阻  
        endcase  
    endendmodule
```

3、for循环语句

一般形式如下

for (initial_assignment; condition; step_assignment) statement;

initial_assignment为循环变量的初始值。

Condition为循环的条件，若为真，执行过程赋值语句
statement，若不成立，循环结束，执行**for**后面的语句。

step_assignment为循环变量的步长，每次迭代后，循环变量
将增加或减少一个步长。

试用Verilog语言描述具有高电平使能的3线-8线译码器.

```
module ecoder3to8_bh(  
    input [2:0] A,  
    input En,  
    output reg [7:0]Y  
);  
    integer k;    //声明一个整型变量k  
    always @(A, En) //  
        begin  
            Y = 8'b1111_1111; //设译码器输出的默认值  
            for(k = 0; k <= 7; k = k+1) //下面的if-else语句循环8次  
            {  
                if ((En==1) && (A== k) )  
                    Y[k] = 0; //当En=1时，根据A进行译码  
                else  
                    Y[k] = 1; //处理使能无效或输入无效的情况  
            }  
        end  
endmodule
```

循环8次

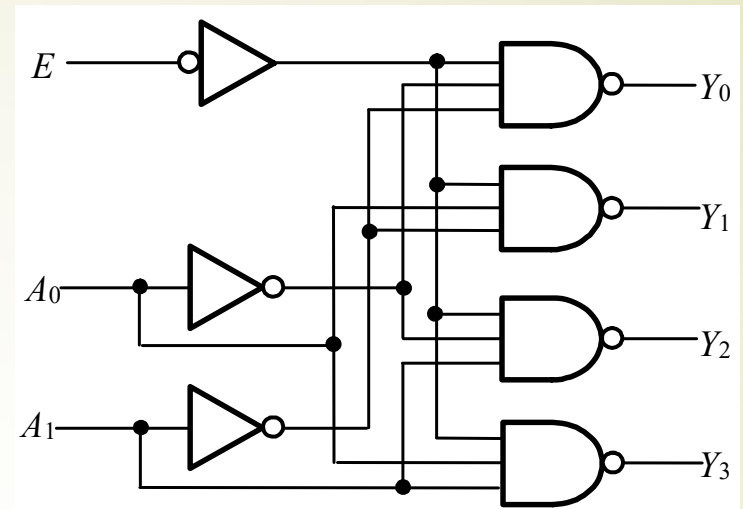
例：用条件运算符描述了一个2选1的数据选择器。

```
module mux2x1_df (  
    input A,B,SEL;  
    output L  
);  
    assign L = SEL ? A : B;  
endmodule
```

在连续赋值语句中，如果SEL=1，则输出L=A；否则L=B。

```
module mux2x1_df (  
    input A,B,SEL,  
    output reg Y  
);  
    always @( D1,D0,S ) //用always语句和条件运算符建模  
        L = S ? D1 : D0;  
endmodule
```

例：用数据流建模方法对2线-4线译码器的行为进行描述。



```
module decoder_df (
    input A1,A0,E,
    output [3:0] Y
) ;

    assign Y[0] = ~(~A1 & ~A0 & ~E);
    assign Y[1] = ~(~A1 & A0 & ~E);
    assign Y[2] = ~(A1 & ~A0 & ~E);
    assign Y[3] = ~(A1 & A0 & ~E);
endmodule
```

$$Y_0 = \overline{\overline{A_1}} \overline{\overline{A_0}} \overline{\overline{E}}$$

$$Y_1 = \overline{\overline{A_1}} A_0 \overline{\overline{E}}$$

$$Y_2 = \overline{A_1} \overline{\overline{A_0}} \overline{\overline{E}}$$

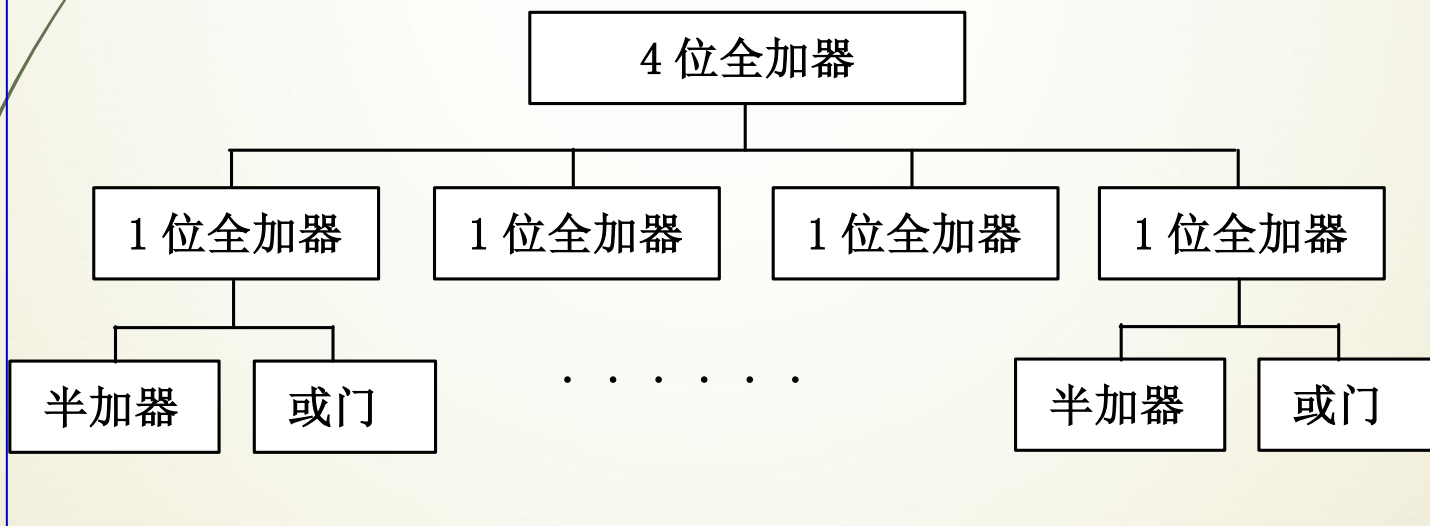
$$Y_3 = \overline{A_1} A_0 \overline{\overline{E}}$$

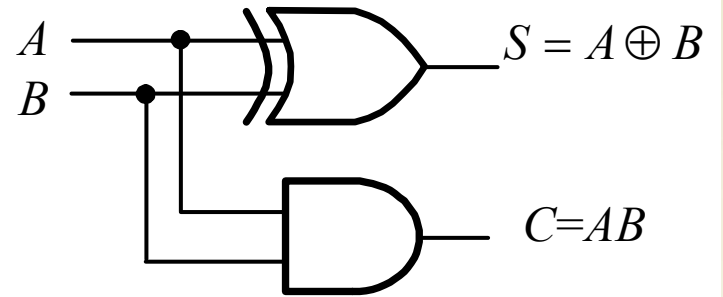
4.6.2 分模块、分层次的电路设计

分层次的电路设计:在电路设计中,将两个或多个模块组合起来描述电路逻辑功能的设计方法。

设计方法: 自顶向下和自底向上两种常用的设计方法

4位全加器的层次结构框图





```
module halfadder (  
    input A,B,  
    output S,C  
);  
    xor (S,A,B);  
    and (C,A,B);  
endmodule
```

} 半加器的门级描述

```
module fulladder (
```

```
  input A,B,Ci,
```

```
  output S,CO
```

```
);
```

```
  wire S1,D1,D2; //内部节点信号
```

```
  halfadder HA1 (.B(B),.S(S1),.C(D1),.A(A));
```

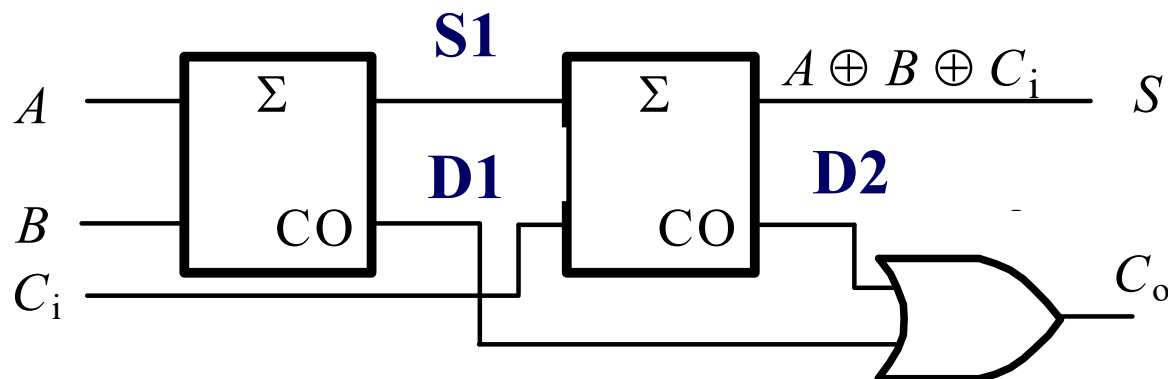
```
  halfadder HA2 (.A(S1),.B(Ci),.S(Sum),.C(D2));
```

```
  or (Co,D2,D1);
```

```
  or g1(CO,D2,D1);
```

```
endmodule
```

全加器的描述
-名称关联调用
半加器,括号外
是原名称,括
号里是现在名
称,位置时序
可改变。



```
module _4bit_adder (S,C3,A,B,C_1);
```

```
input [3:0] A,B;
```

```
input C_1;
```

```
output [3:0] S;
```

```
output C3;
```

```
wire C0,C1,C2; //内部进位信号
```

```
fulladder FA0 (S[0],C0,A[0],B[0],C_1),
```

```
FA1 (S[1],C1,A[1],B[1],C0),
```

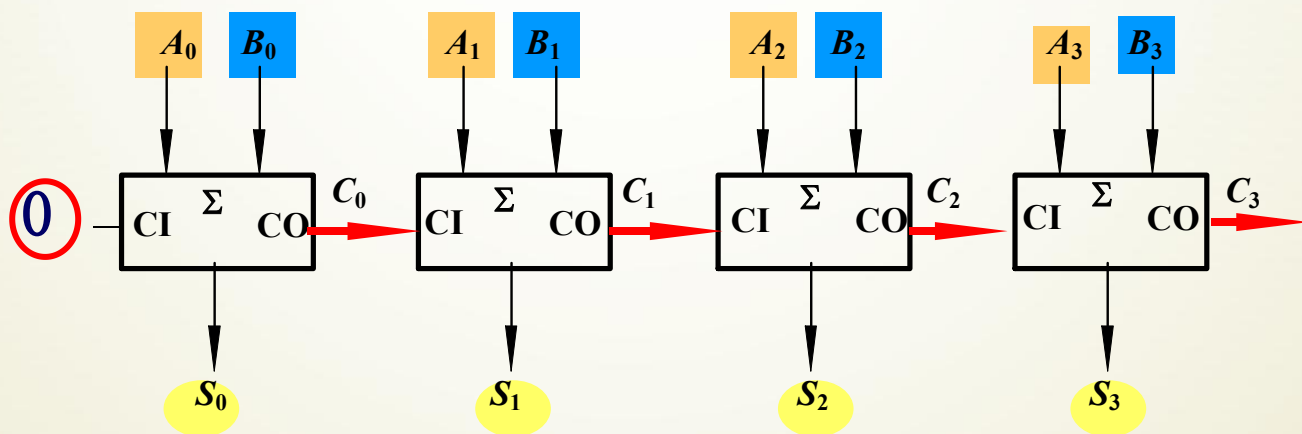
```
FA2 (S[2],C2,A[2],B[2],C1),
```

```
FA3 (S[3],C3,A[3],B[3],C2);
```

```
endmodule
```

4位全加器的描述

--位置关联调用1位全加器，注意端口排列顺序



//带参数的4位加法器子模块

module adder #(parameter n=4**)**

(

input [n-1:0] A,B,

input Ci,

output [n-1:0] S,

output Co

);

assign {Co,S} = A + B + Ci;

endmodule

} 4位全加器的描述

//按名称关联端口及参数

```
module adder_hier (  
    input [7:0] A1,B1,  
    input Cin1,  
    output [7:0] Sum1,  
    output Cout1  
);  
    adder #( .n(8)) U1  
(  
    .A(A1),  
    .B(B1),  
    .Ci(Cin1),  
    .S( Sum1),  
    .Co( Cout1)  
);
```

8位全加器