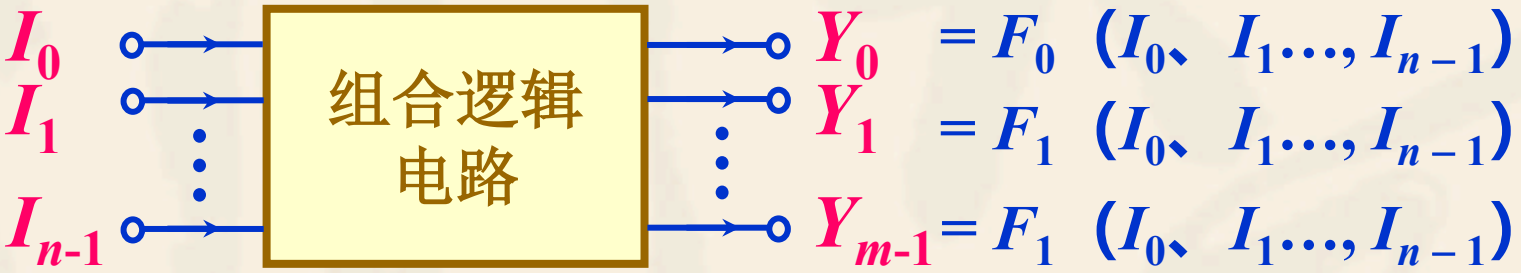
第三章 组合逻辑电路

- 3.1 组合电路的分析方法和设计方法
  - 3.1.1 组合电路的基本分析方法
  - 3.1.2 组合电路的基本设计方法
- 3.2 加法器和数值比较器
  - 3.2.1 加法器
  - 3.2.2 数值比较器
- 3.3 编码器和译码器
  - 3.3.1 编码器
  - 3.3.2 译码器
- 3.4 数据选择器和分配器
  - 3.4.1 数据选择器
  - 3.4.2 数据分配器
- 3.5 用中规模集成电路构成的组合电路的设计
  - 3.5.1 用数据选择器实现组合逻辑函数
  - 3.5.2 用二进制译码器实现组合逻辑函数

概　述

一、组合逻辑电路的特点

$$Y(t_n) = F\,[I(t_n)]$$



1. 逻辑功能特点

电路在任何时刻的输出状态只取决于该时刻的输入状态，而与原来的状态无关。

2. 电路结构特点

(1) 输出、输入之间没有反馈延迟电路

(2) 不包含记忆性元件(触发器)，仅由门电路构成

二、组合电路逻辑功能的表示方法

真值表，卡诺图，逻辑表达式，时间图(波形图)

三、组合电路分类

① 按逻辑功能不同：

加法器　　比较器　　编码器　　译码器
数据选择器和分配器　　只读存储器

② 按开关元件不同：　　CMOS　　TTL

③ 按集成度不同：　　SSI　MSI　LSI　VLSI

3.1 组合电路的分析方法和设计方法

3.1.1 组合电路的基本分析方法

一、分析方法

逻辑图 → 逻辑表达式 → 化简 → 真值表 → 说明功能

分析目的：

① 确定输入变量不同取值时功能是否满足要求；

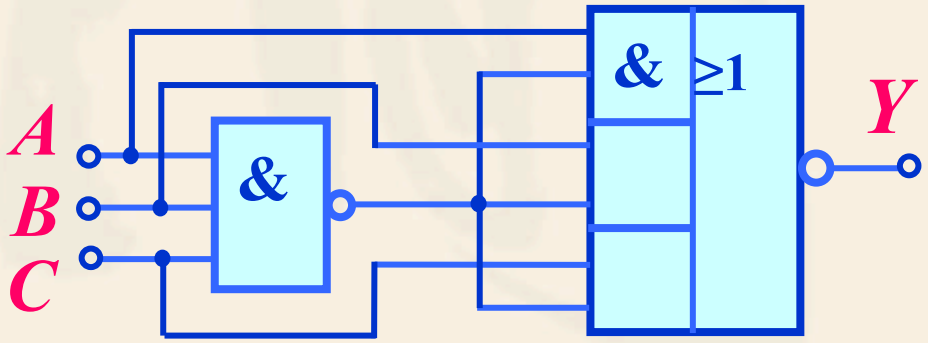
② 变换电路的结构形式(如：与或 ⟶与非-与非)；

③ 得到输出函数的标准与或表达式，以便用 MSI、LSI 实现；

④ 得到其功能的逻辑描述，以便用于包括该电路的系统分析。

二、分析举例

［例］分析图中所示电路的逻辑功能



真值表

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

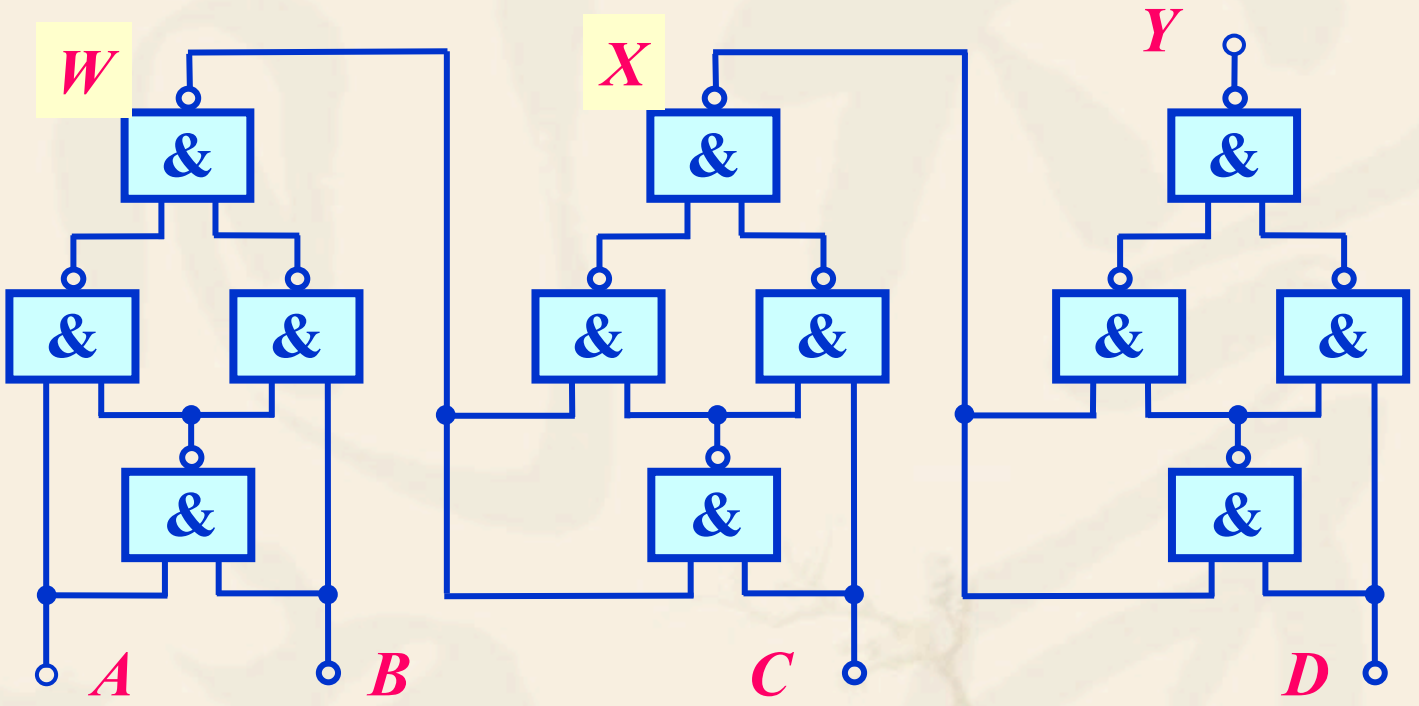
［解］ 表达式

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

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

功能 判断输入信号极性是否相同的电路 — 符合电路

［例 3.1.1］分析图中所示电路的逻辑功能，输入信号*A*、*B*、*C*、*D*是一组二进制代码。

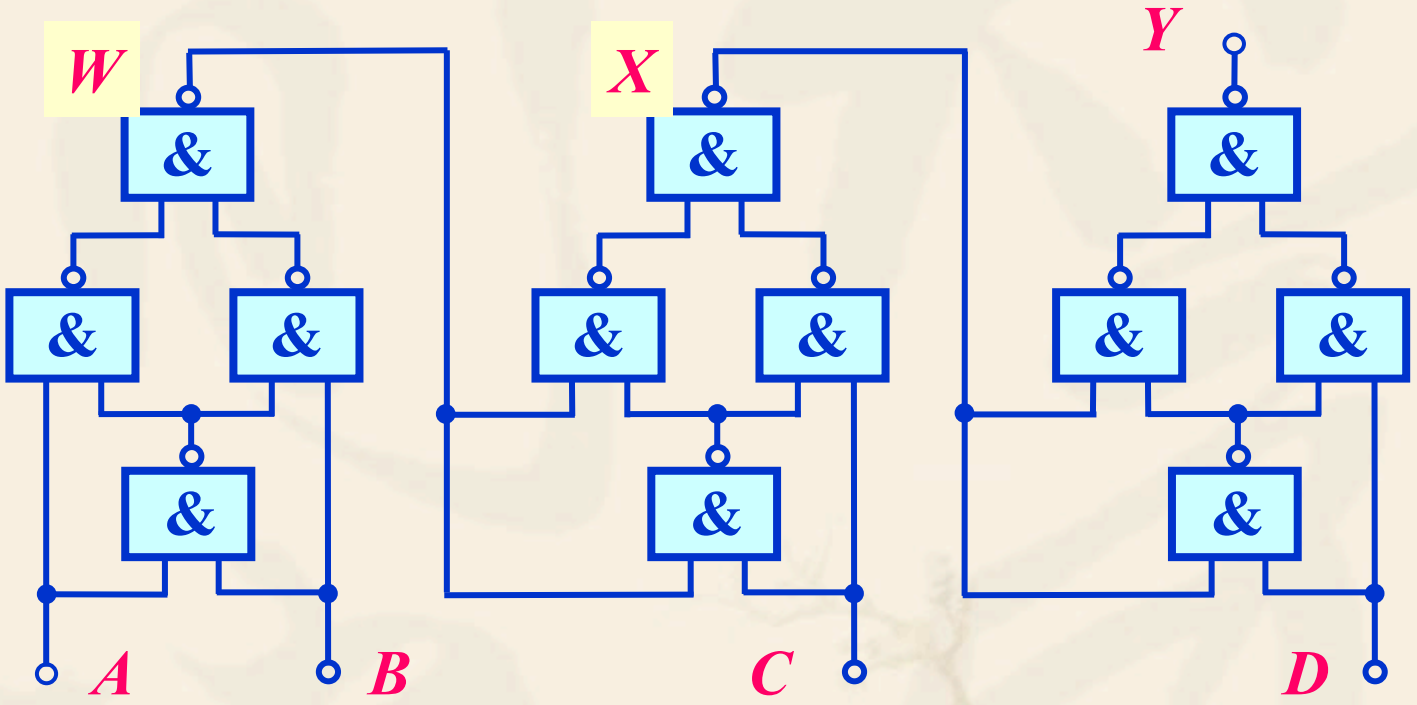


［解］(1) 逐级写输出函数的逻辑表达式

$$W = \overline{A\,\overline{AB}}\;\overline{\overline{AB}\,B} \qquad Y = \overline{X\,\overline{XD}}\;\overline{\overline{XD}\,D}$$

$$X = \overline{W\,\overline{WC}}\;\overline{\overline{WC}\,C}$$

［例 3.1.1］分析图中所示电路的逻辑功能，输入信号*A*、*B*、*C*、*D*是一组二进制代码。



［解］(2) 化简 $W = A\overline{AB}\ \overline{AB}B = A\overline{B} + \overline{A}B$

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

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

$$+ \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}BCD + A\overline{B}CD + AB\overline{C}D$$

［例 3.1.1］分析图中所示电路的逻辑功能，输入信号*A*、*B*、*C*、*D*是一组二进制代码。

［解］(3) 列真值表

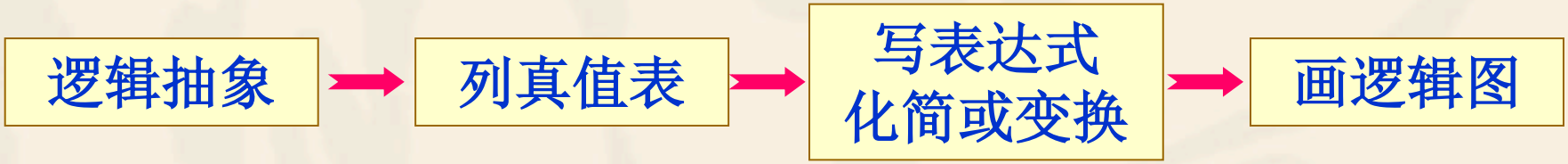
$$Y = A\overline{B}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}\,\overline{D} + \overline{A}\,\overline{B}C\overline{D} + ABC\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}BCD + A\overline{B}CD + AB\overline{C}D$$

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

(4) 功能说明：当输入四位代码中 1 的个数为奇数时输出为 1，为偶数时输出为 0 — 检奇电路。

3.1.2 组合电路的基本设计方法

一、设计方法



逻辑抽象：

① 根据因果关系确定输入、输出变量

② 状态赋值 — 用 0 和 1 表示信号的不同状态

③ 根据功能要求列出真值表

化简或变换：

根据所用元器件(分立元件 或 集成芯片)的情况将函数式进行化简或变换。

二、 设计举例

[例 3.1.2] 设计一个表决电路，要求输出信号的电平与三个输入信号中的多数电平一致。

[解] (1) 逻辑抽象

① 设定变量： 输入 A、B、C ， 输出 Y

② 状态赋值：

A、B、$C = 0$ 表示 输入信号为低电平

A、B、$C = 1$ 表示 输入信号为高电平

$Y = 0$ 表示 输入信号中多数为低电平

$Y = 1$ 表示 输入信号中多数为高电平

二、 设计举例

[例 3.1.2] 设计一个表决电路，要求输出信号的电平与三个输入信号中的多数电平一致。

[解] ③ 列真值表

(2)写输出表达式并化简

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

$$= BC + A\square C + AB\square$$

$$= BC + AC + AB$$

最简与或式 ➡ 最简与非-与非式

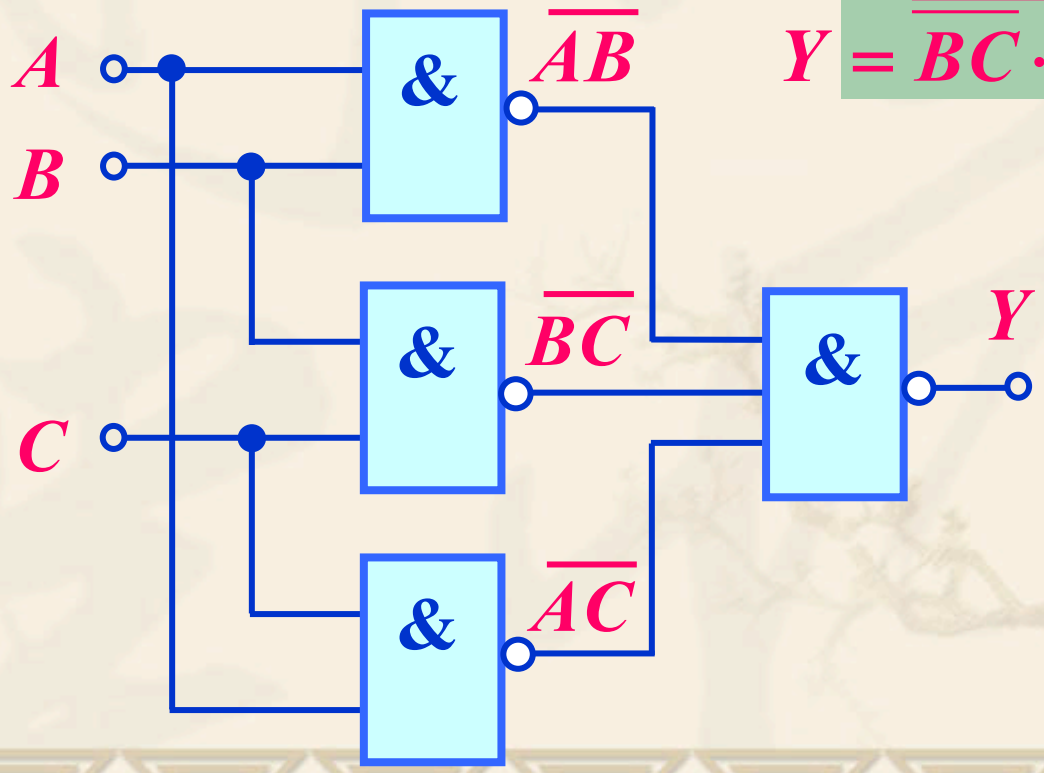
$$Y = \overline{\overline{BC + AC + AB}} = \overline{\overline{BC} \cdot \overline{AC} \cdot \overline{AB}}$$

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

二、设计举例

[例 3. 1. 2] 设计一个表决电路，要求输出信号的电平与三个输入信号中的多数电平一致。

[解] (3) 画逻辑图 —用与非门实现



$$Y = \overline{\overline{BC} \cdot \overline{AC} \cdot \overline{AB}}$$

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

［解］ (1)逻辑抽象

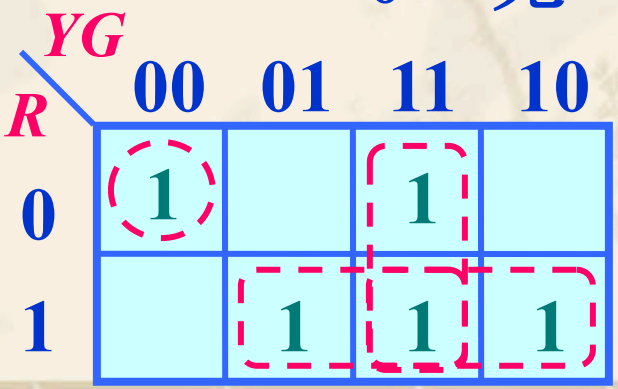
输入变量：R（红）
Y（黄）
G（绿）
} 1 -- 亮，0 -- 灭

输出变量：Z（有无故障） { 1 -- 有，0 -- 无

列真值表

R	Y	G	Z
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(2)卡诺图化简

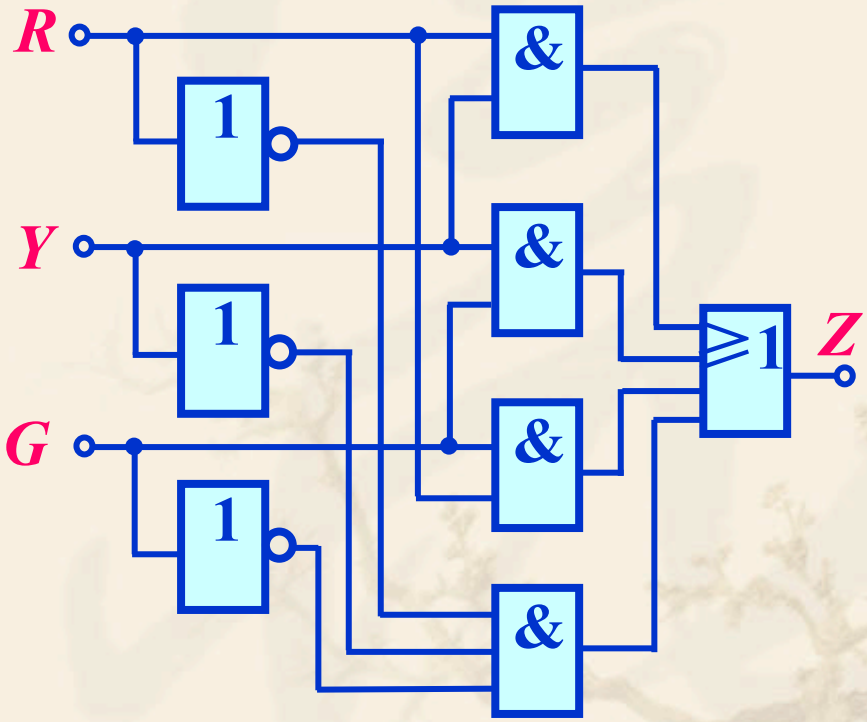


$$Z = \overline{R}\,\overline{Y}\,\overline{G} + RY + RG + YG$$

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

［解］ (3) 画逻辑图

$$Z = \overline{R}\,\overline{Y}\,\overline{G} + RY + RG + YG$$



3.2 加法器和数值比较器

3.2.1 加法器

一、半加器和全加器

1. 半加器（Half Adder）

两个 1 位二进制数相加不考虑低位进位。

$$A_i + B_i = S_i(\text{和}) \rightarrow C_i(\text{进位})$$

真值表

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

函数式

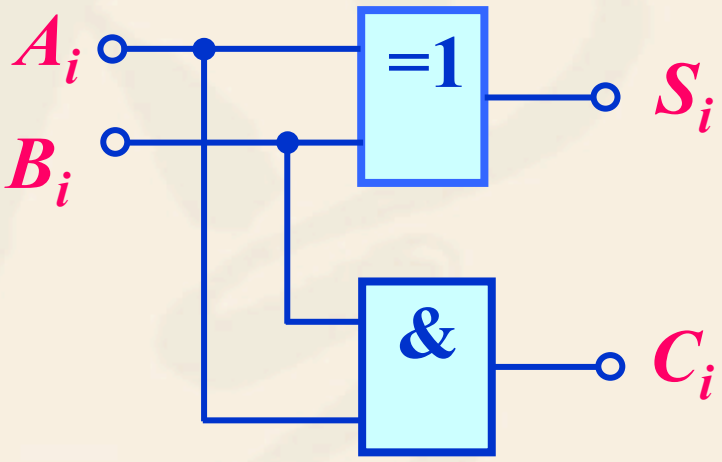
$$S_i = \overline{A}_i B_i + A_i \overline{B}_i$$
$$= A \oplus B$$
$$C_i = A_i B_i$$

半加器（Half Adder）

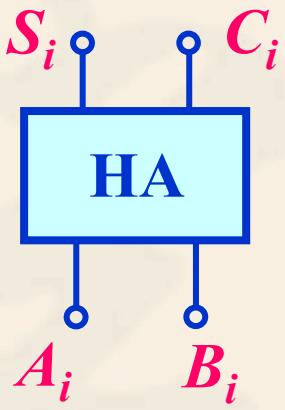
函数式

$$S_i = \overline{A}_i B_i + A_i \overline{B}_i$$
$$= A \oplus B$$
$$C_i = A_i B_i$$

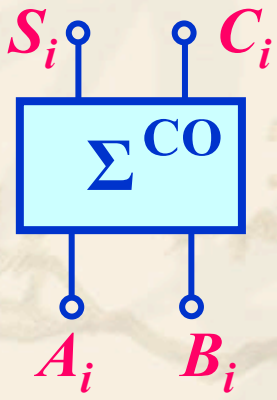
逻辑图



曾用符号



国标符号



2. 全加器（Full Adder）

两个 1 位二进制数相加，考虑低位进位。

$A_i + B_i + C_{i-1}$（低位进位）

$= S_i$（和）$\rightarrow C_i$（向高位进位）

```
  1 0 1 1  --- A
  1 1 1 0  --- B
+ 1 1 1 0 0  --- 低位进位
-----------
高位进位← 1 1 0 0 1  --- S
```

真值表

A	B	C_{i-1}	S_i	C_i	A	B	C_{i-1}	S_i	C_i
0	0	0	0	0	1	0	0	1	0
0	0	1	1	0	1	0	1	0	1
0	1	0	1	0	1	1	0	0	1
0	1	1	0	1	1	1	1	1	1

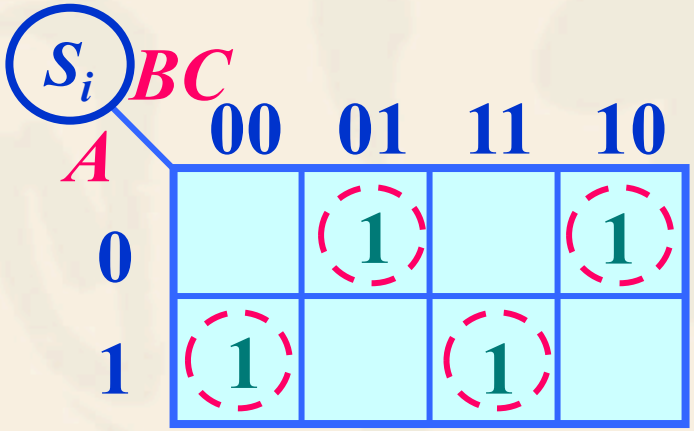
标准与或式

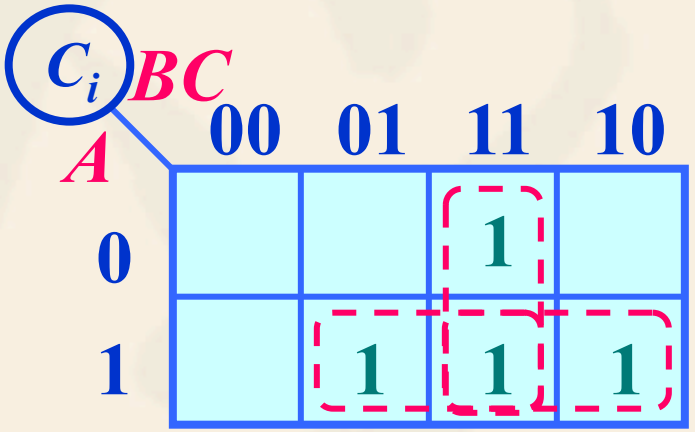
$$S_i = \overline{A}_i\overline{B}_iC_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + A_i\overline{B}_i\overline{C}_{i-1} + A_iB_iC_{i-1}$$

$$C_i = \overline{A}_iB_iC_{i-1} + A_i\overline{B}_iC_{i-1} + A_iB_i\overline{C}_{i-1} + A_iB_iC_{i-1}$$

全加器（Full Adder）

卡诺图





最简与或式

圈“1”

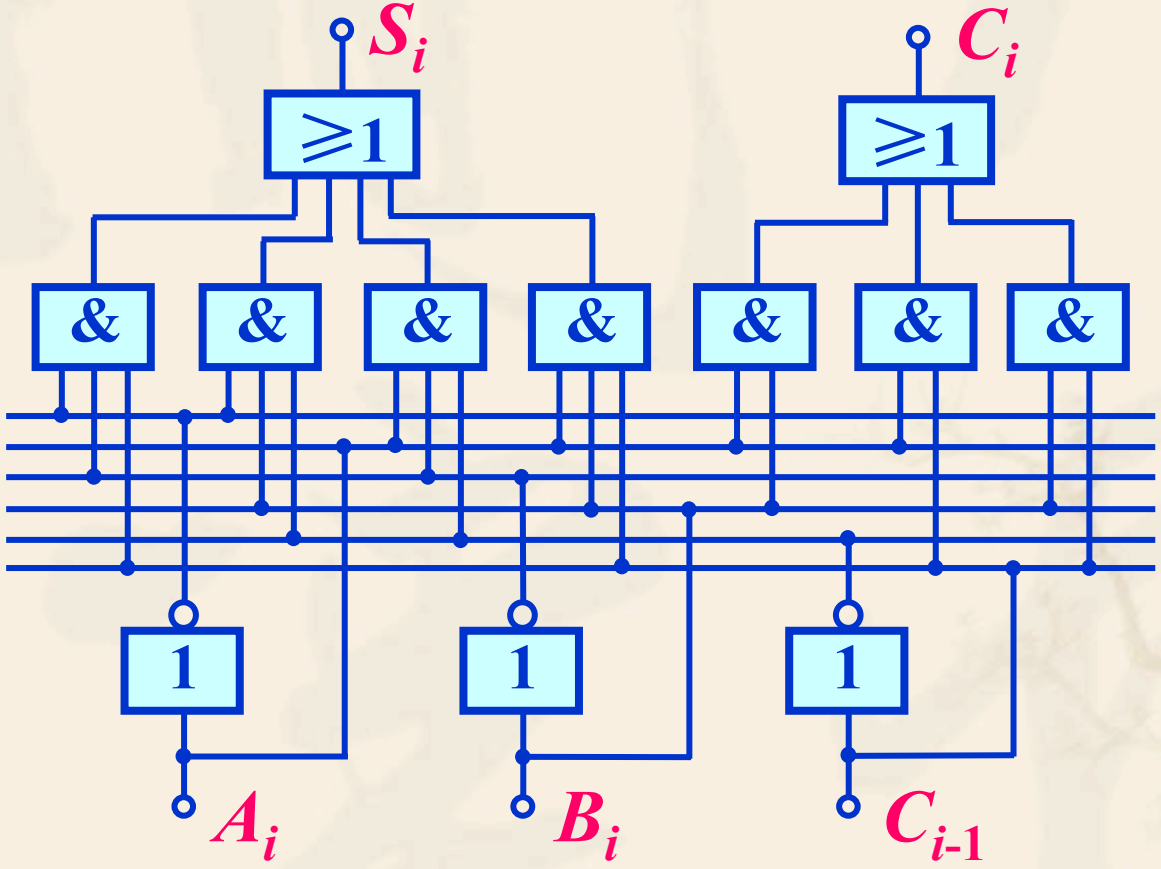
$$\begin{cases} S_i = \overline{A}_i\overline{B}_iC_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + A_i\overline{B}_i\overline{C}_{i-1} + A_iB_iC_{i-1} \\ C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1} \end{cases}$$

圈“0”

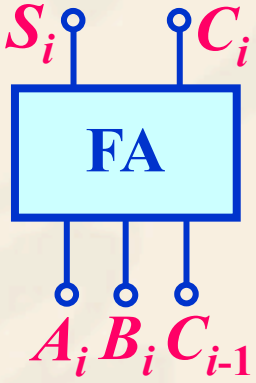
$$\begin{cases} \overline{S}_i = \overline{A}_i\overline{B}_i\overline{C}_{i-1} + \overline{A}_iB_iC_{i-1} + A_i\overline{B}_iC_{i-1} + A_iB_i\overline{C}_{i-1} \\ \overline{C}_i = \overline{A}_i\overline{B}_i + \overline{A}_i\overline{C}_{i-1} + \overline{B}_i\overline{C}_{i-1} \end{cases}$$

逻辑图

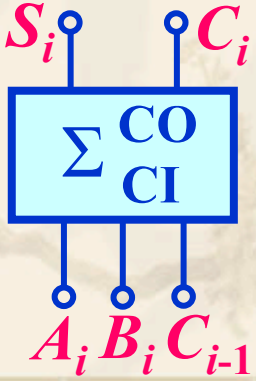
(a) 用与门、或门和非门实现



曾用符号



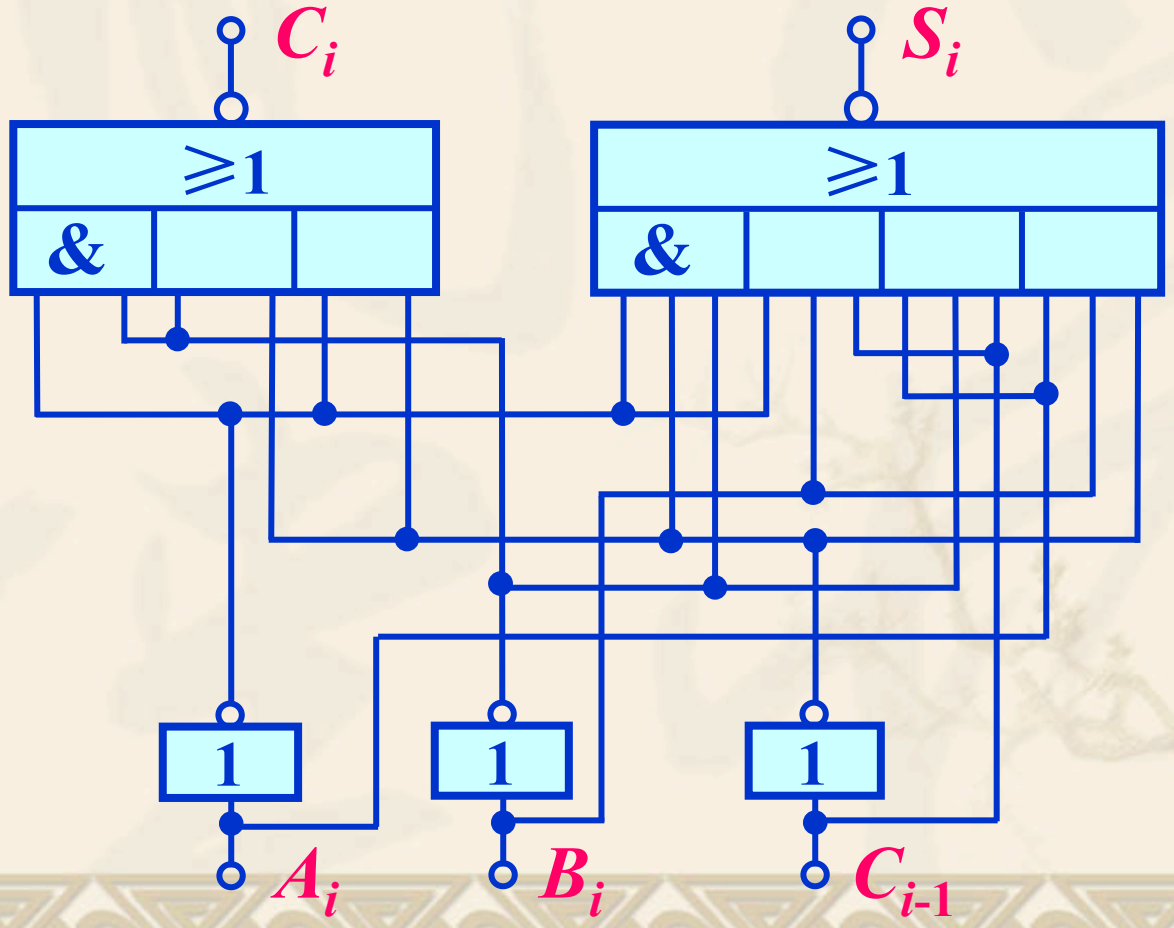
国标符号



(b) 用与或非门和非门实现

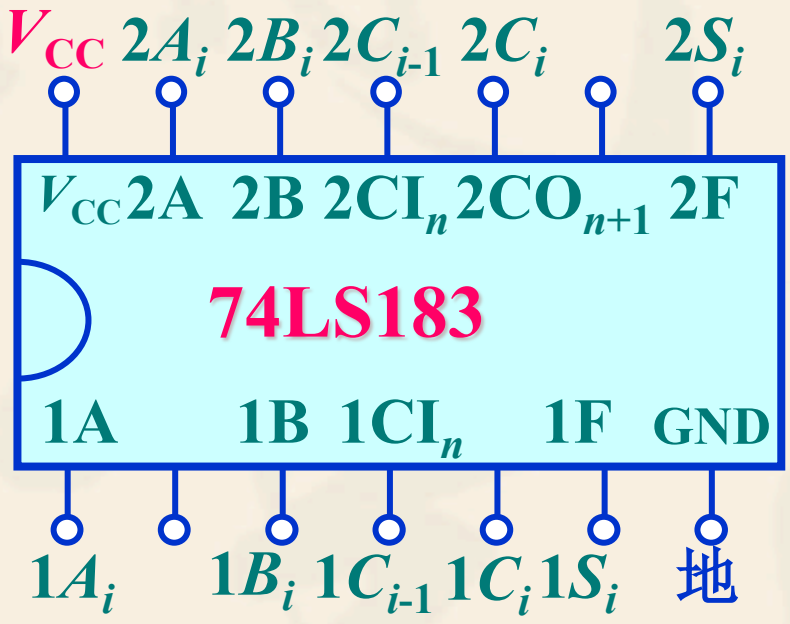
$$S_i = \overline{\overline{A}_i\overline{B}_i\overline{C}_{i-1} + \overline{A}_iB_iC_{i-1} + A_i\overline{B}_iC_{i-1} + A_iB_i\overline{C}_{i-1}}$$

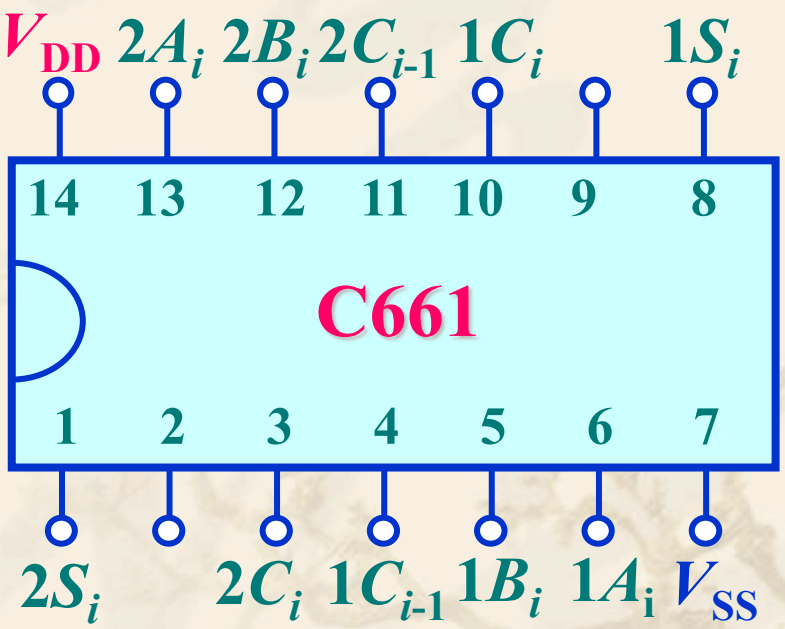
$$C_i = \overline{\overline{A}_i\overline{B}_i + \overline{A}_i\overline{C}_{i-1} + \overline{B}_i\overline{C}_{i-1}}$$



3. 集成全加器

双全加器 { TTL：74LS183
CMOS：C661



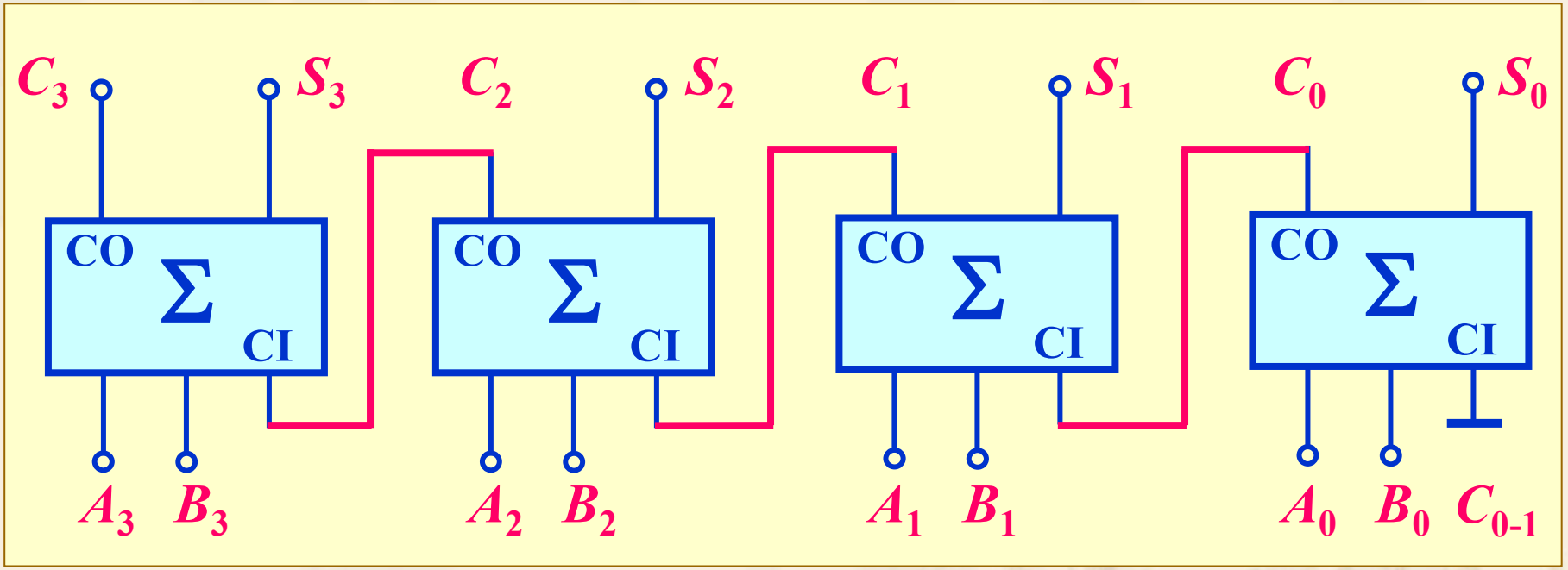


二、加法器（Adder）

实现多位二进制数相加的电路

1. 4 位串行进位加法器

$A = A_3A_2A_1A_0$　　$B = B_3B_2B_1B_0$



特点：
- 电路简单，连接方便
- 速度低 = 4 t_{pd}

t_{pd} — 1位全加器的平均传输延迟时间

2. 超前进位加法器

作加法运算时，总进位信号由输入二进制数直接产生。

$$C_0 = A_0B_0 + (A_0 + B_0)C_{0-1}$$

$$C_1 = A_1B_1 + (A_1 + B_1)C_0$$

$$= A_1B_1 + (A_1 + B_1)\left[A_0B_0 + (A_0 + B_0)C_{0-1}\right]$$

$$\vdots$$

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

特点

优点：速度快

缺点：电路比较复杂

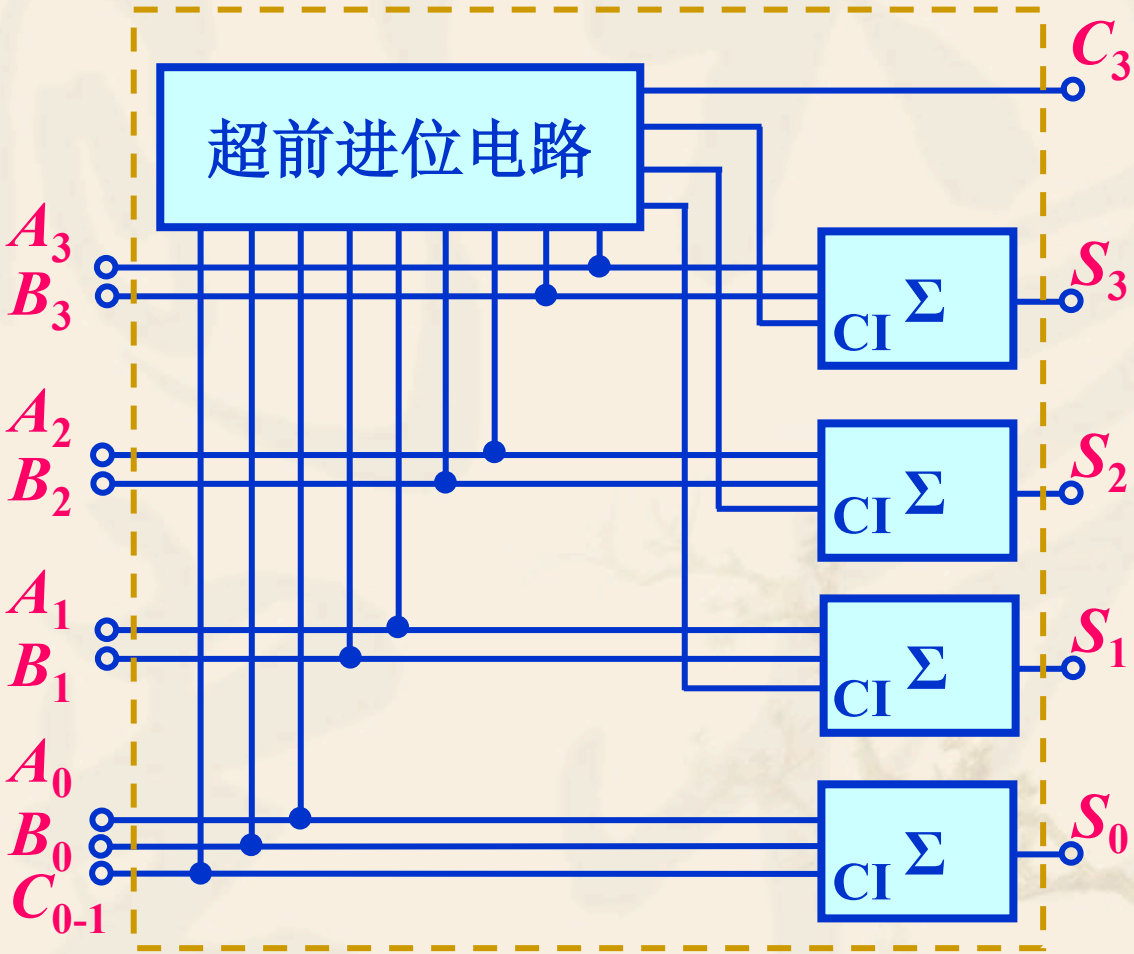
应用举例

8421 BCD 码→ 余 3 码

集成芯片

CMOS：CC4008

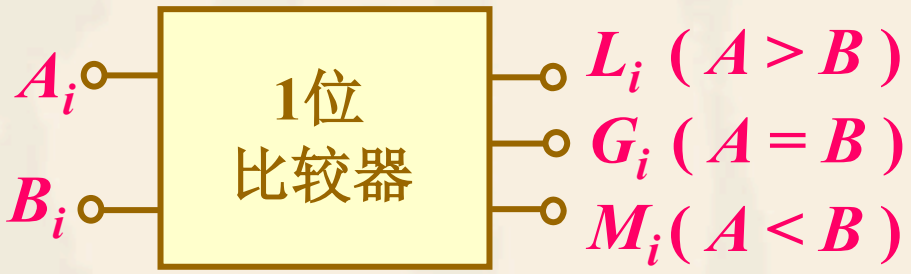
TTL：74283 74LS283



逻辑结构示意图

3.2.2 数值比较器（Digital Comparator）

一、1 位数值比较器



真值表

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

函数式

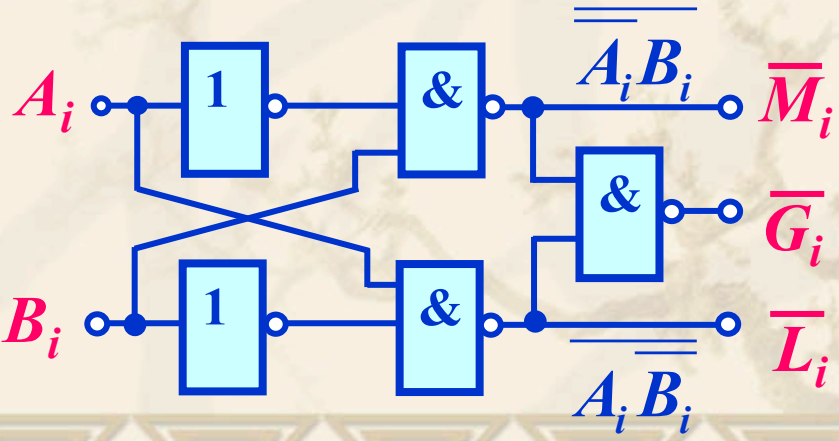
$$L_i = A_i\overline{B}_i \qquad G_i = \overline{A}_i\overline{B}_i + A_iB_i$$

$$M_i = \overline{A}_iB_i \qquad = A_i \odot B_i$$

$$\overline{L}_i = \overline{A_i\overline{B}_i} \qquad \overline{G}_i = \overline{\overline{A_i\overline{B}_i} \cdot \overline{\overline{A}_iB_i}}$$

$$\overline{M}_i = \overline{\overline{A}_iB_i}$$

逻辑图 —用与非门和非门实现



多位数值比较器

比较两个多位数,应首先从高位开始,逐位比较。

例如：　$A=A_3A_2A_1A_0$　　$B=B_3B_2B_1B_0$

比较方法为：

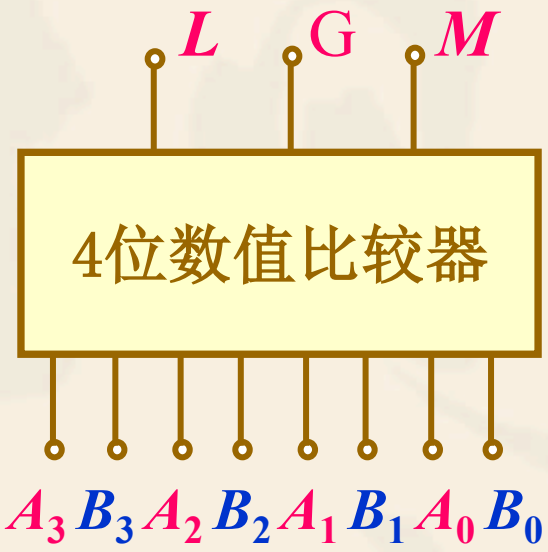
① 首先比较A_3和B_3，如$A_3B_3=10$，则A>B,如$A_3B_3=01$，则A<B; 如$A_3B_3=00$或11(相等)，则比较A_2和B_2;

② 比较A_2和B_2，如$A_2B_2=10$，则A>B,如 $A_2B_2=01$，则A<B;如$A_2B_2=00$或11 (相等)，则比较A_1和B_1;

③ 比较A_1和B_1，如$A_1B_1=10$，则A>B,如 $A_1B_1=01$，则A<B;如$A_1B_1=00$或11 (相等)，则比较A_0和B_0;

④ 比较A_0和B_0，如$A_0B_0=10$，则A>B,如 $A_0B_0=01$，则A<B;如$A_0B_0=00$或11 (相等)，则比较A=B.

二、4 位数值比较器　$A = A_3A_2A_1A_0$　$B = B_3B_2B_1B_0$



$A > B$　$L = 1$

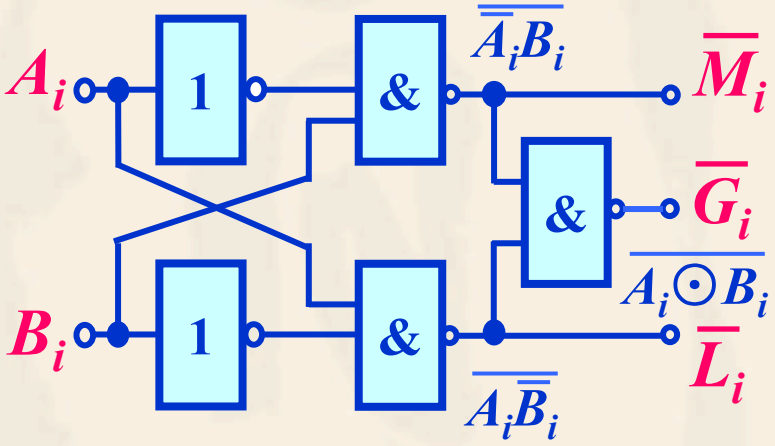
$A = B$　$G = 1$

$A < B$　$M = 1$

真值表

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

1 位数值比较器

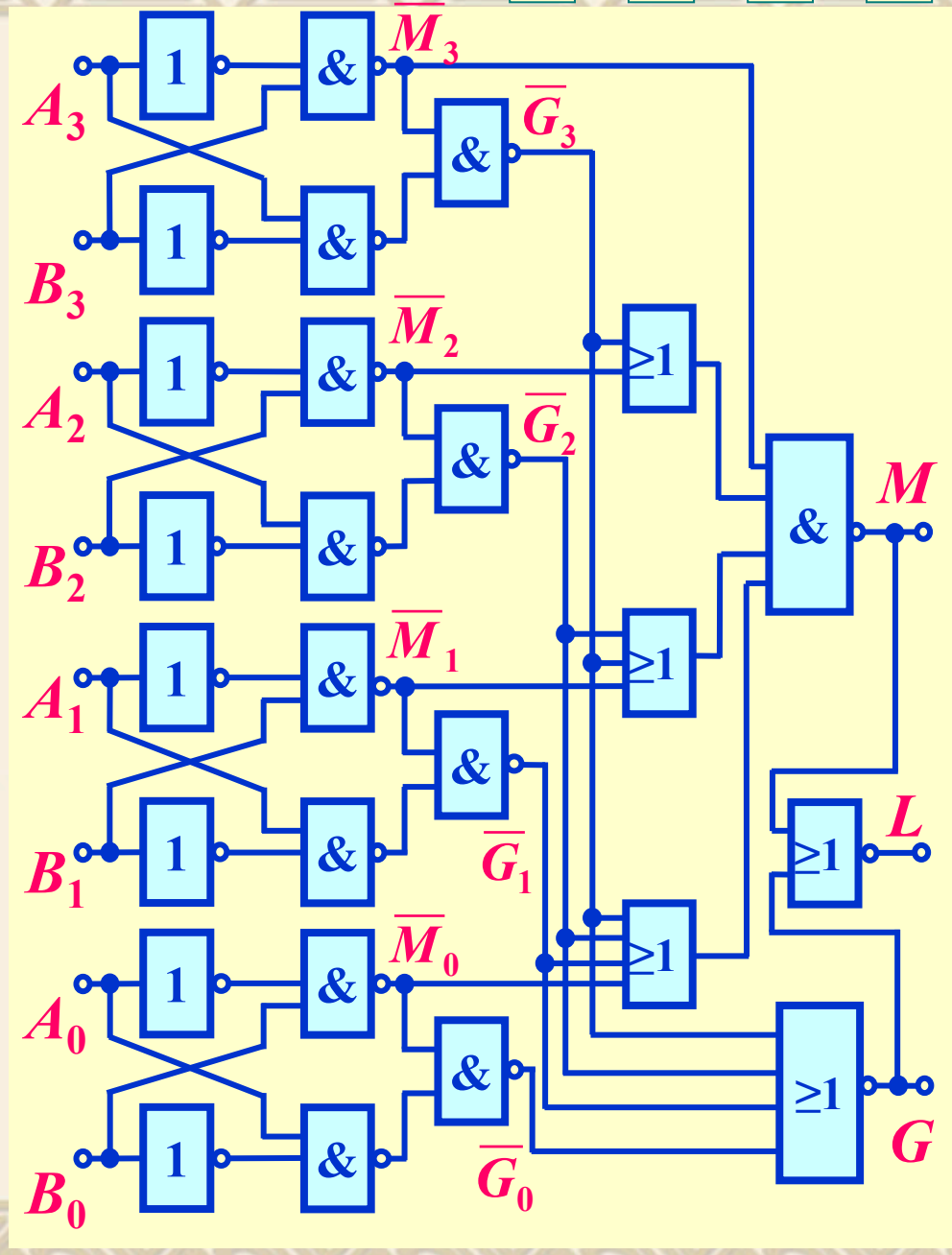


4 位数值比较器

$$M = \overline{A}_3B_3 + (A_3 \odot B_3)\,\overline{A}_2B_2 + (A_3 \odot B_3)(A_2 \odot B_2)\,\overline{A}_1\,B_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)\,\overline{A}_0B_0$$

$$G = (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$$

$$L = \overline{M+G}$$



4 位集成数值比较器的真值表

比较输入				级联输入			输出		
A_3B_3	A_2B_2	A_1B_1	A_0B_0	A<B	A=B	A>B	$F_{A<B}$	$F_{A=B}$	$F_{A>B}$
>	×	×	×	×	×	×	0	0	1
=	>	×	×	×	×	×	0	0	1
=	=	>	×	×	×	×	0	0	1
=	=	=	>	×	×	×	0	0	1
=	=	=	=	0	0	1	0	0	1
=	=	=	=	0	1	0	0	1	0
=	=	=	=	1	0	0	1	0	0
<	×	×	×	×	×	×	1	0	0
=	<	×	×	×	×	×	1	0	0

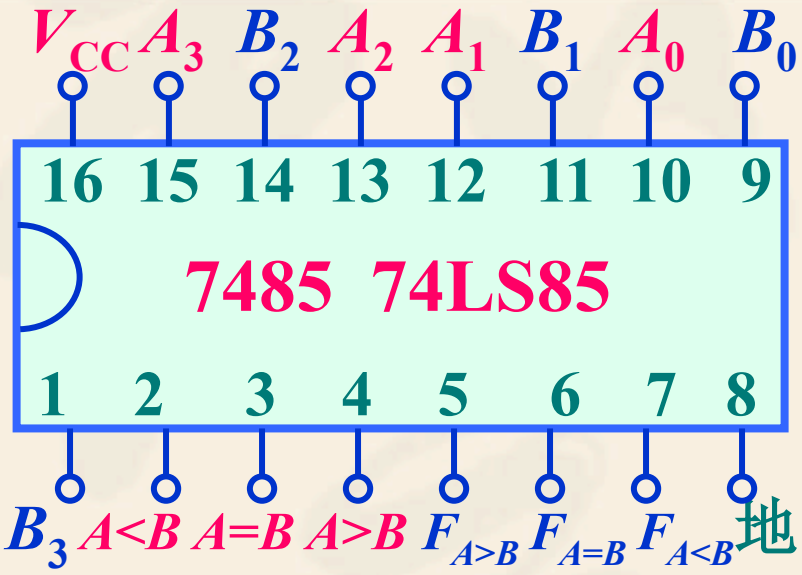
级联输入：供扩展使用，一般接低位芯片的比较输出，即接低位芯片的 $F_{A<B}$、$F_{A=B}$、$F_{A>B}$。

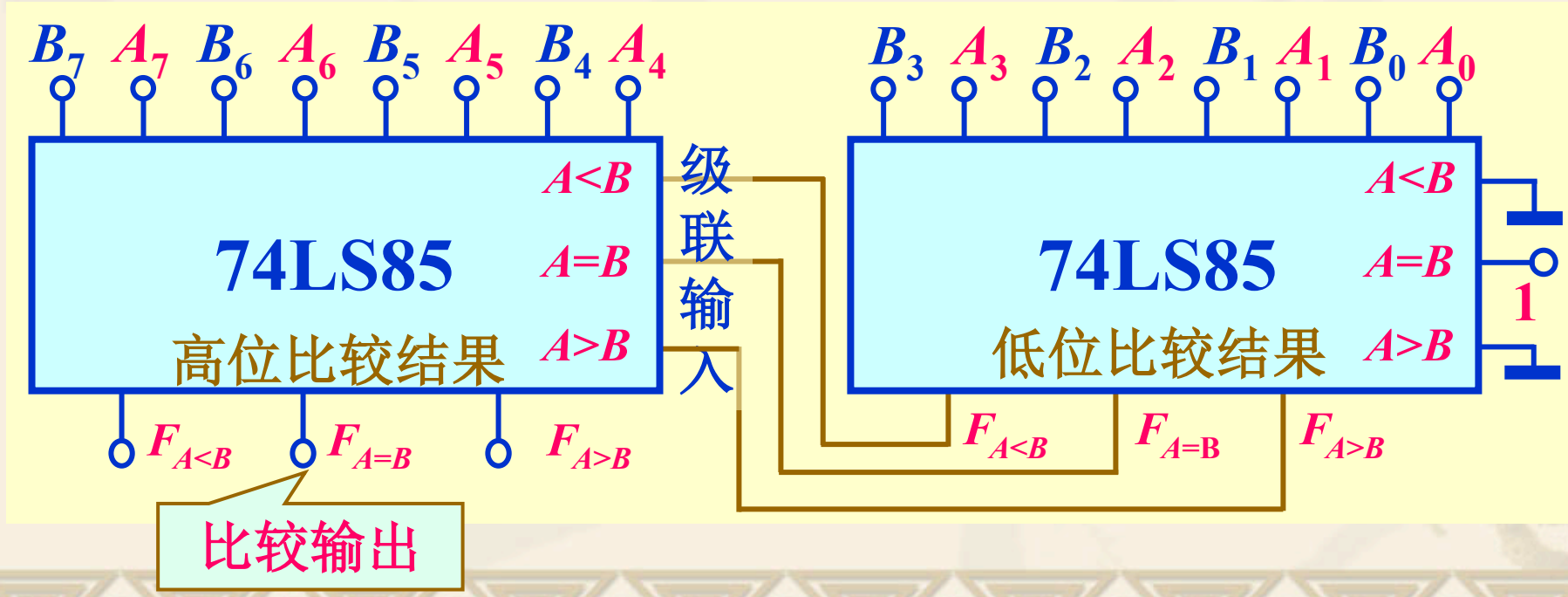
集成数值比较器

74LS85 (TTL)

扩展：两片 4 位数值比较器

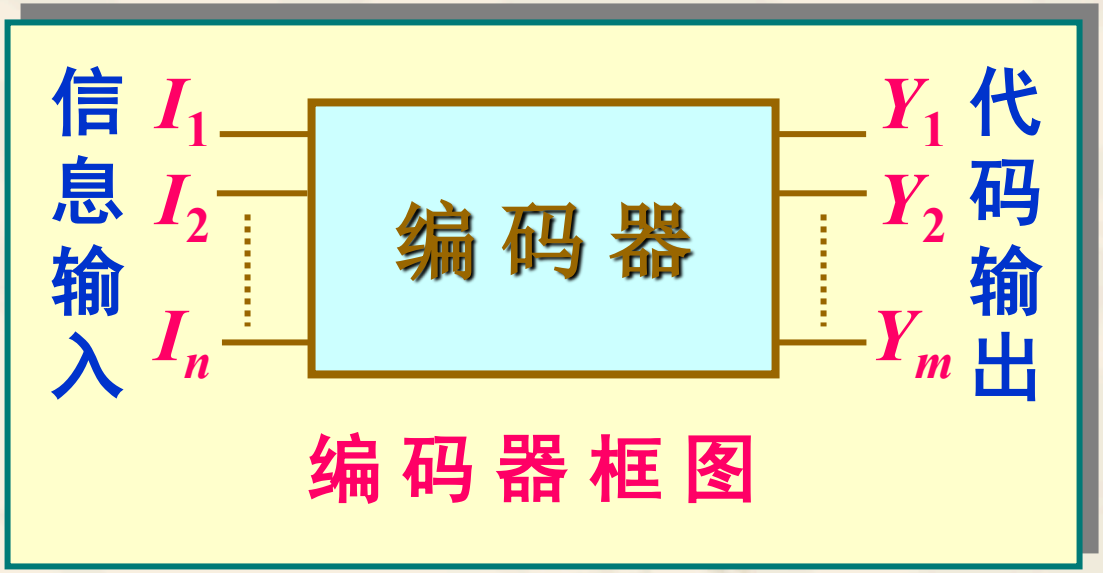
→ 8 位数值比较器





3.3 编码器和译码器

3.3.1 编码器（Encoder）



编码器框图

编码：用文字、符号或者数字表示特定对象的过程
（用二进制代码表示不同事物）

分类：
- 二进制编码器 $2^n \rightarrow n$
- 二—十进制编码器 $10 \rightarrow 4$

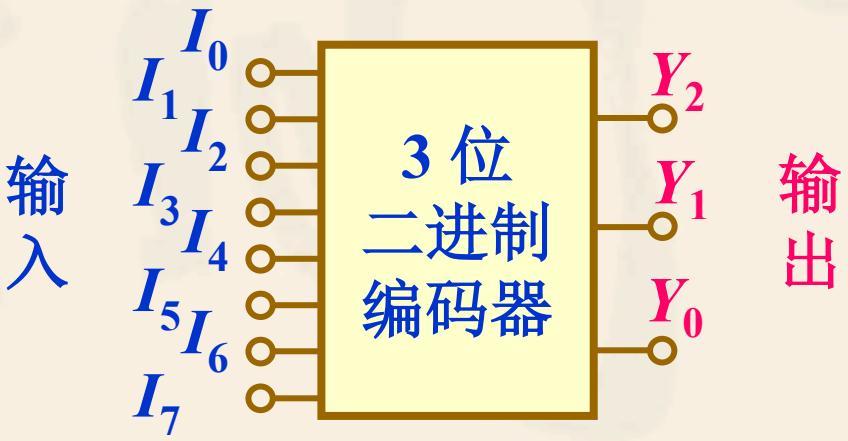
或

- 普通编码器
- 优先编码器

一、二进制编码器

用 n 位二进制代码对 $N=2^n$ 个信号进行编码的电路

1. 3 位二进制编码器(8 线- 3 线)



$I_0 \sim I_7$ 是一组互相排斥的输入变量，任何时刻只能有一个端输入有效信号。

函数式

$$Y_2 = I_4 + I_5 + I_6 + I_7$$
$$Y_1 = I_2 + I_3 + I_6 + I_7$$
$$Y_0 = I_1 + I_3 + I_5 + I_7$$

编码表

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

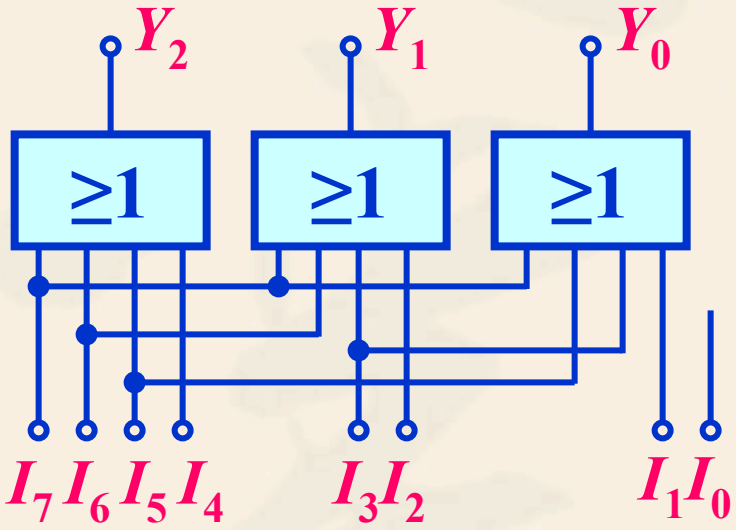
函数式

$$Y_2 = I_4 + I_5 + I_6 + I_7 = \overline{\bar{I}_4 \cdot \bar{I}_5 \cdot \bar{I}_6 \cdot \bar{I}_7}$$

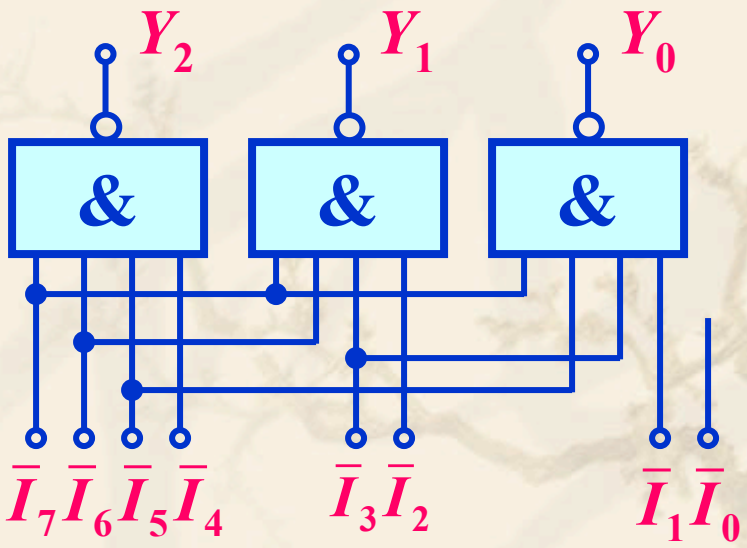
$$Y_1 = I_2 + I_3 + I_6 + I_7 = \overline{\bar{I}_2 \cdot \bar{I}_3 \cdot \bar{I}_6 \cdot \bar{I}_7}$$

$$Y_0 = I_1 + I_3 + I_5 + I_7 = \overline{\bar{I}_1 \cdot \bar{I}_3 \cdot \bar{I}_5 \cdot \bar{I}_7}$$

逻辑图 — 用或门实现



— 用与非门实现



2. 3 位二进制优先编码器

优先编码：允许几个信号同时输入，但只对优先级别最高的进行编码。优先顺序：$I_7 \rightarrow I_0$

编码表

输入								输出		
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

函数式

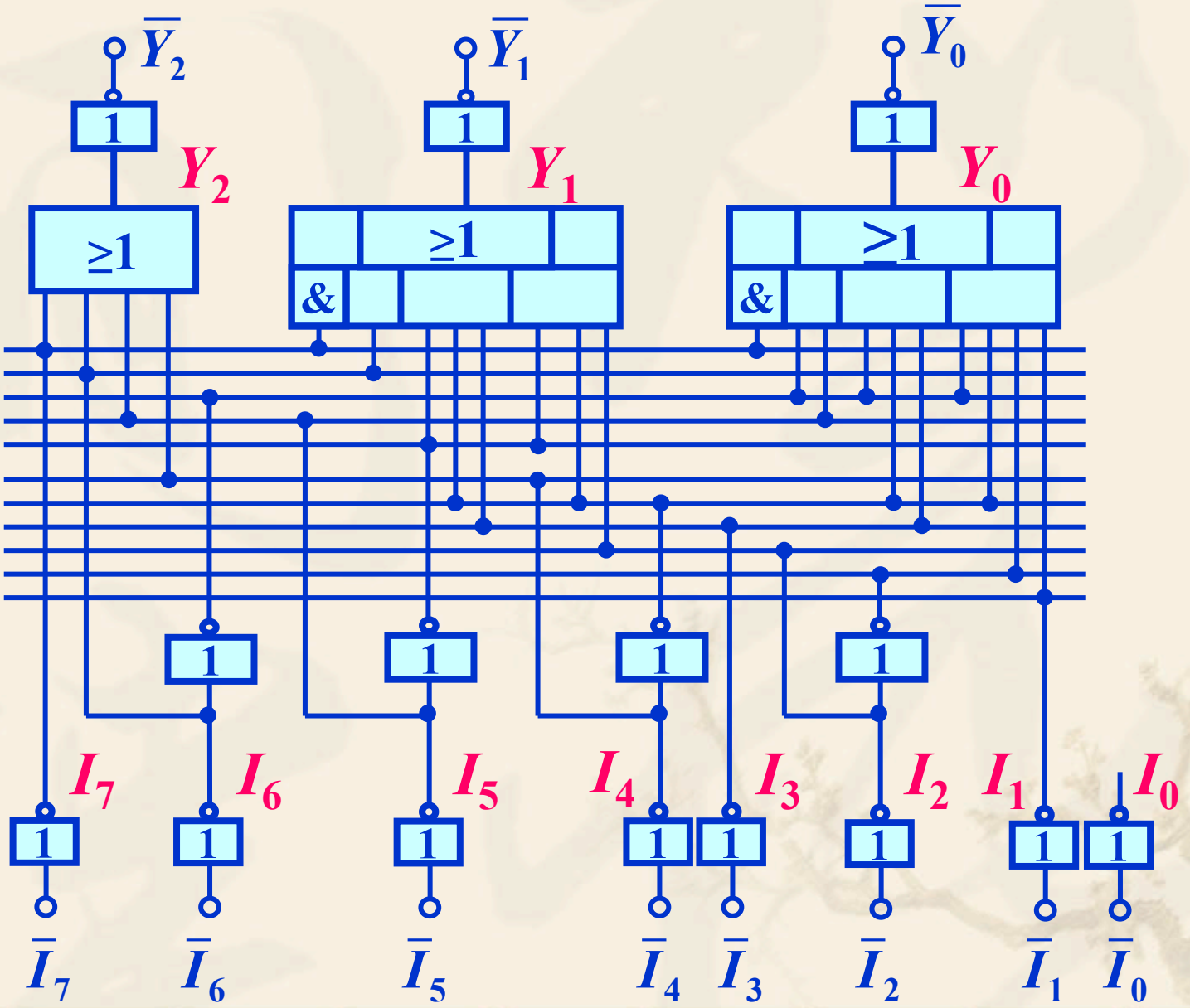
$$Y_2 = I_7 + I_6 + I_5 + I_4$$

$$Y_1 = I_7 + I_6 + \overline{I}_5 \overline{I}_4 I_3 + \overline{I}_5 \overline{I}_4 I_2$$

$$Y_0 = I_7 + \overline{I}_6 I_5 + \overline{I}_6 \overline{I}_4 I_3 + \overline{I}_6 \overline{I}_4 \overline{I}_2 I_1$$

逻辑图

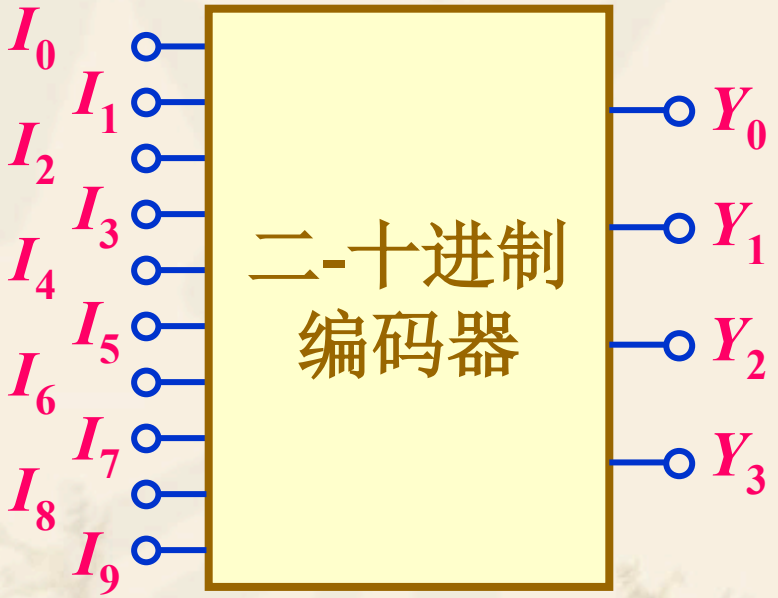
输入输出为反变量



二、二-十进制编码器

用 4 位二进制代码对 0 ~ 9 十个信号进行编码的电路。

1. 8421 BCD 编码器
2. 8421 BCD 优先编码器
3. 集成 10线 -4线优先编码器
   (74147　74LS147)



三、几种常用编码

1. 二-十进制编码

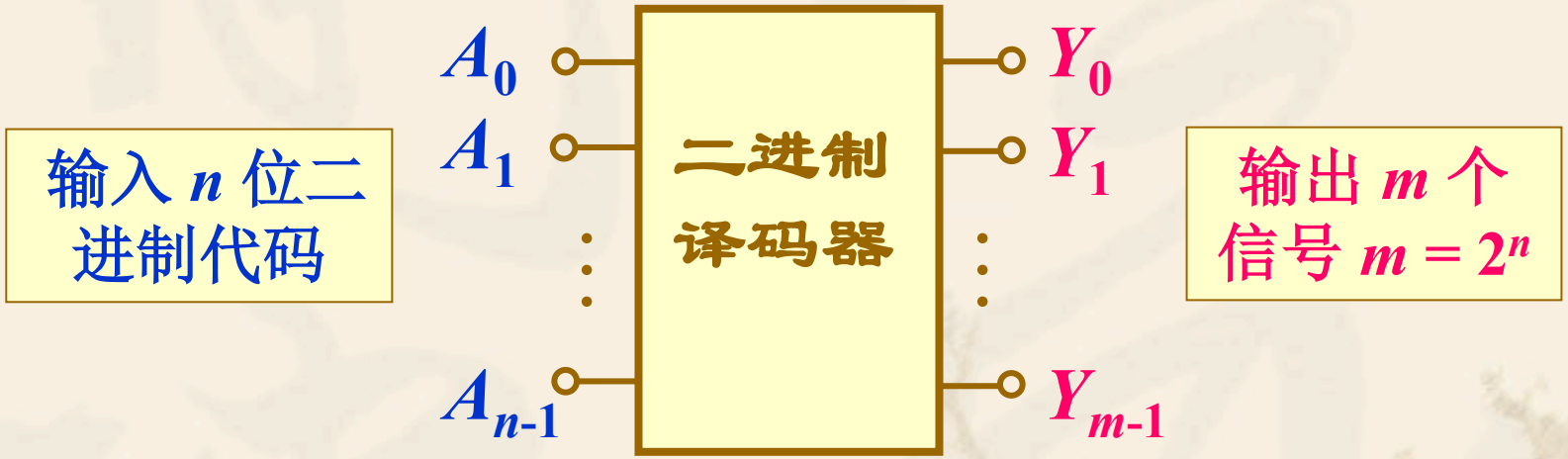
8421 码　余 3 码　　2421 码
5211 码　余 3 循环码　右移循环码

2. 其他　　循环码（反射码或格雷码）
ISO码　　ANSCII（ASCII）码

3.3.2 译码器（Decoder）

编码的逆过程，将二进制代码翻译为原来的含义

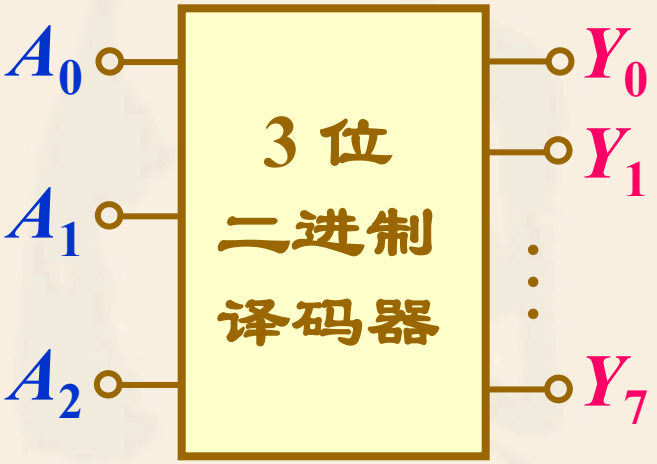
一、二进制译码器(Binary Decoder)



如： 2 线 — 4 线译码器 3 线 — 8 线译码器

4 线 — 16 线译码器

1. 3位二进制译码器 (3 线 – 8 线)



真值表

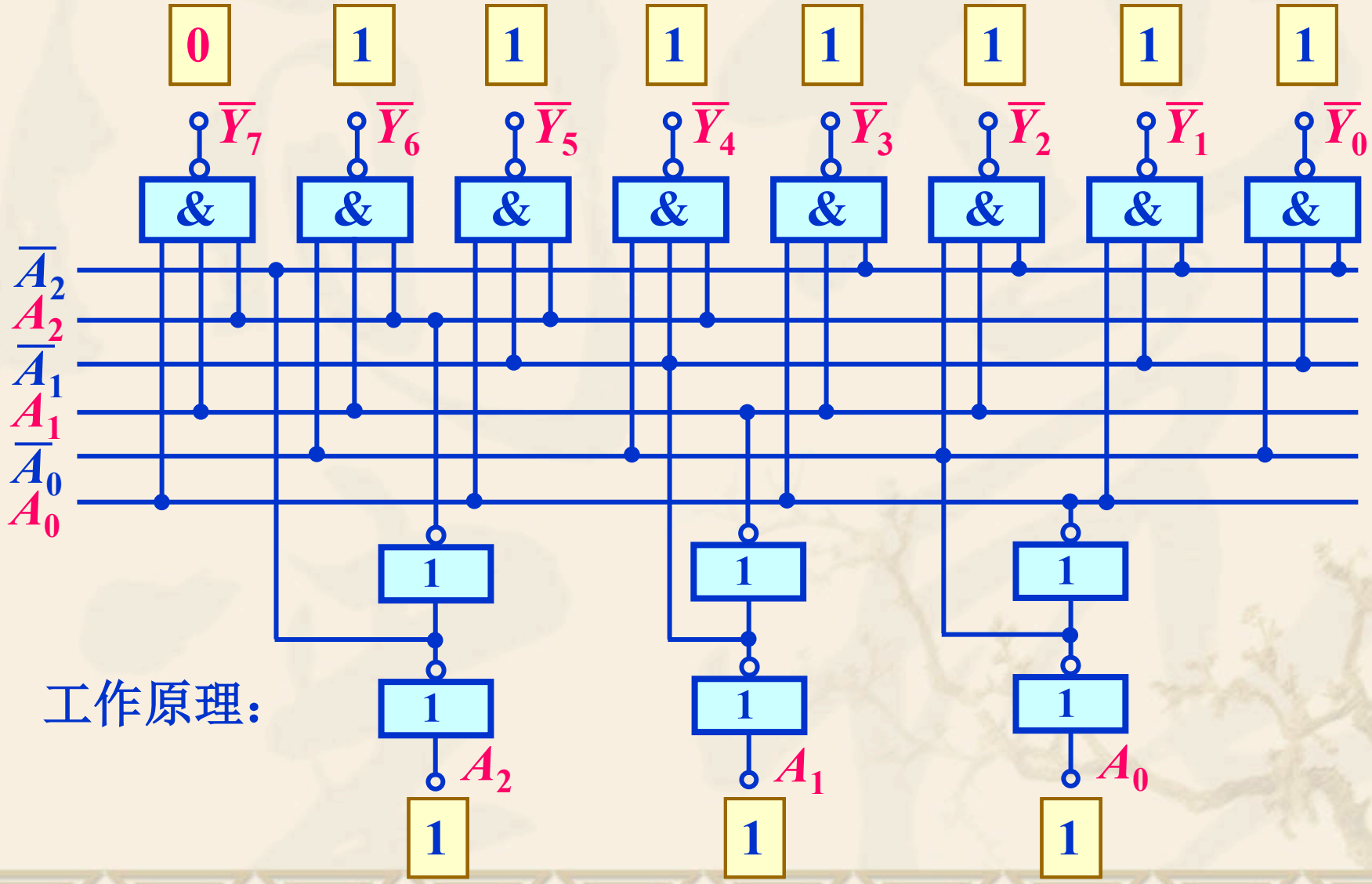
函数式

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

$$Y_0 = \overline{A}_2\overline{A}_1\overline{A}_0 \quad Y_2 = \overline{A}_2 A_1\overline{A}_0 \quad Y_4 = A_2\overline{A}_1\overline{A}_0 \quad Y_6 = A_2 A_1\overline{A}_0$$

$$Y_1 = \overline{A}_2\overline{A}_1 A_0 \quad Y_3 = \overline{A}_2 A_1 A_0 \quad Y_5 = A_2\overline{A}_1 A_0 \quad Y_7 = A_2 A_1 A_0$$

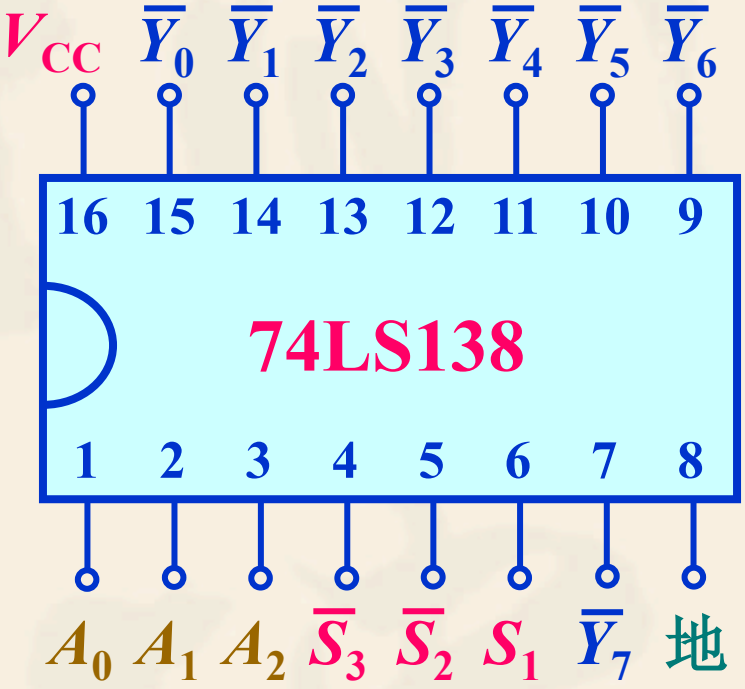
3 线 - 8 线译码器逻辑图 —输出低电平有效



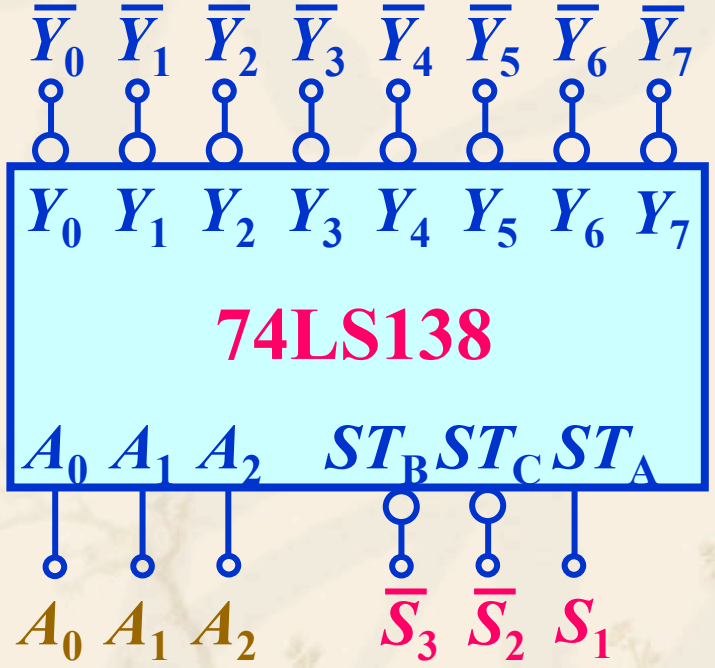
工作原理：

2. 集成 3 线 – 8 线译码器 -- 74LS138

引脚排列图



功能示意图



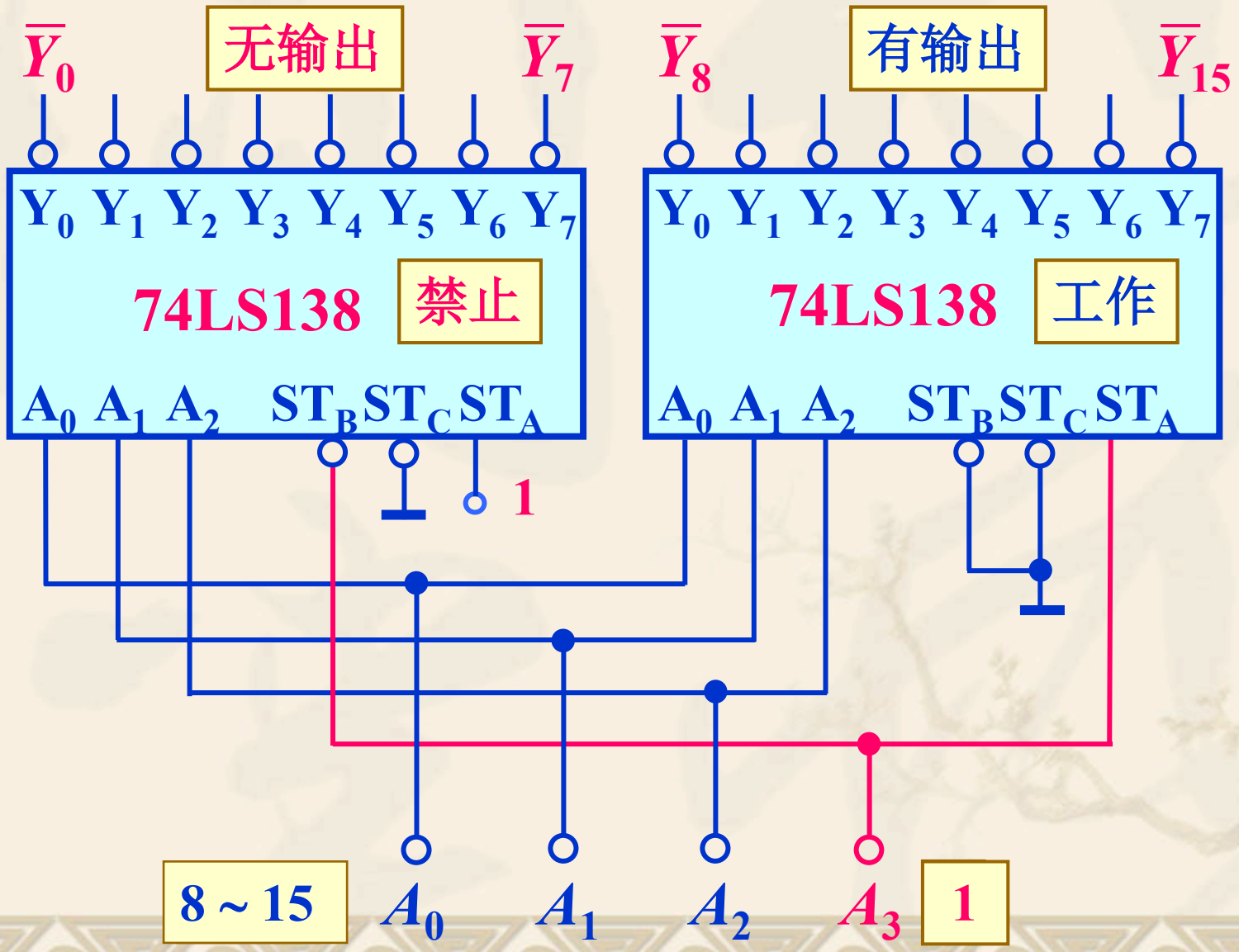
输入选通控制端

S_1、$\overline{S}_2$、$\overline{S}_3$

$S_1 = 0$ 或 $\overline{S}_2 + \overline{S}_3 = 1$　芯片禁止工作

$S_1 = 1$ 且 $\overline{S}_2 + \overline{S}_3 = 0$　芯片正常工作

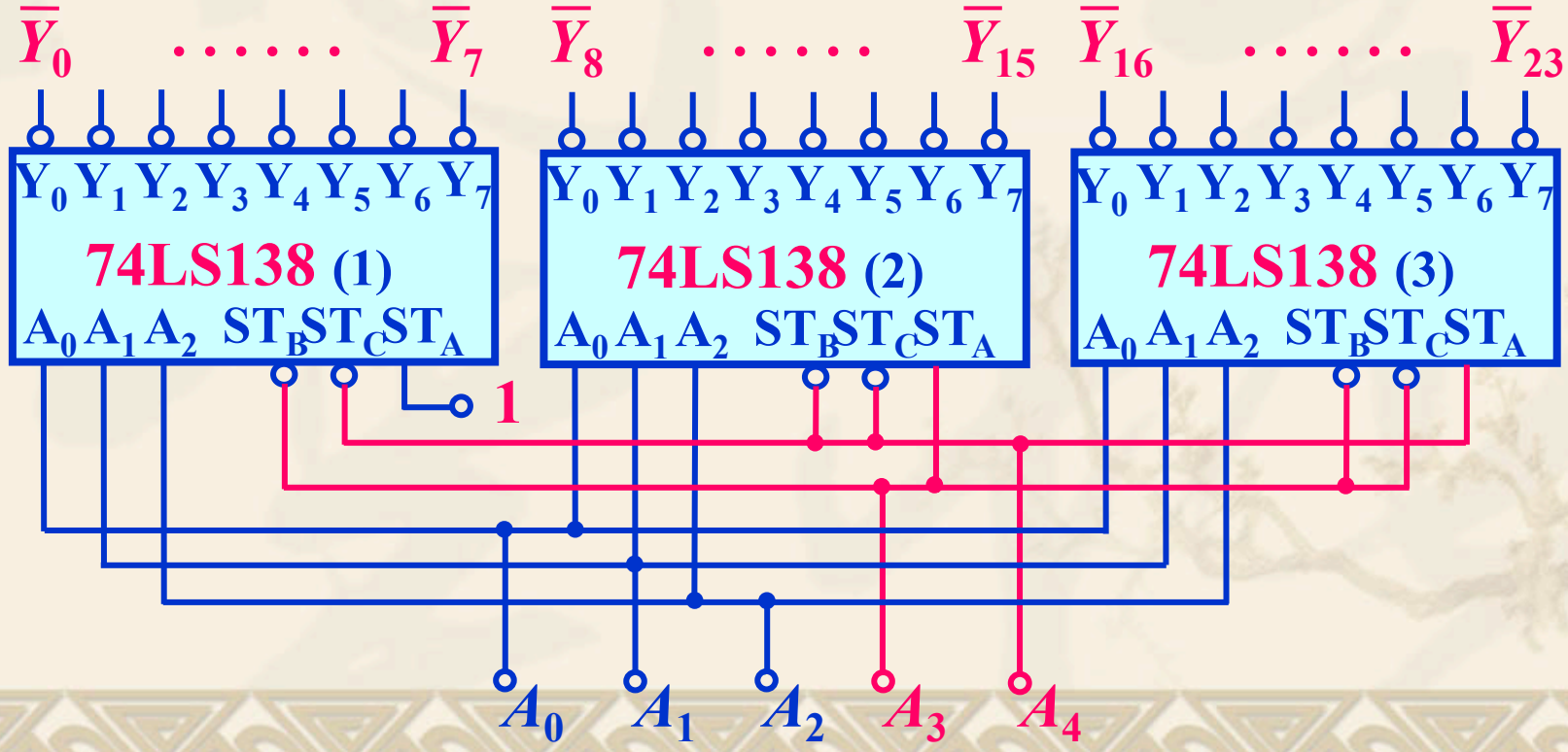
3. 二进制译码器的级联 两片3 线 – 8 线 → 4 线-16 线



三片 3 线- 8 线

5 线 - 24 线

A_4	A_3	(1)	(2)	(3)	输 出
0	0	工	禁	禁	$\overline{Y}_0 \sim \overline{Y}_7$
0	1	禁	工	禁	$\overline{Y}_8 \sim \overline{Y}_{15}$
1	0	禁	禁	工	$\overline{Y}_{16} \sim \overline{Y}_{23}$
1	1	禁	禁	禁	全为 1



4. 二进制译码器的主要特点

功能特点： 输出端提供全部最小项

电路特点： 与门(原变量输出)

与非门(反变量输出)

二、二-十进制译码器

(Binary-Coded Decimal Decoder)

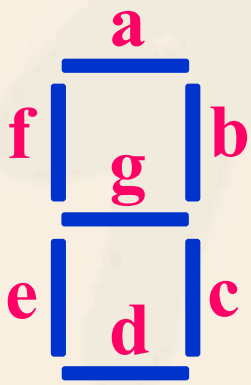
将 BCD 码翻译成对应的十个输出信号

集成 4 线 –10 线译码器： 7442 74LS42

三、显示译码器

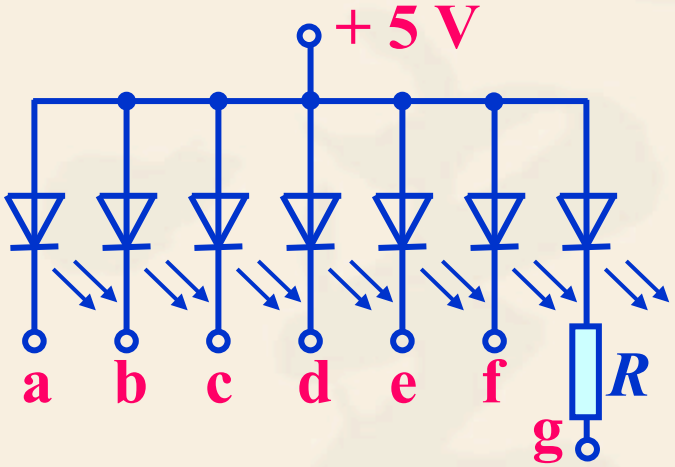
数码显示器

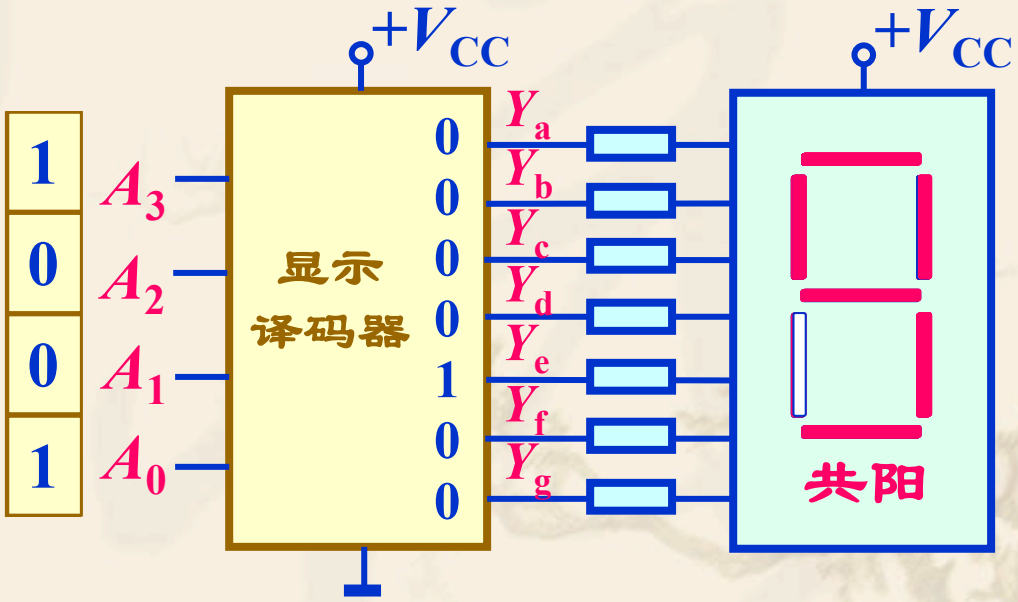
- 半导体显示(LED)
- 液晶显示(LCD)



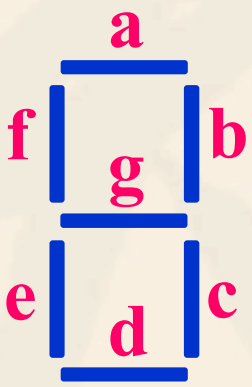
每字段是一只发光二极管

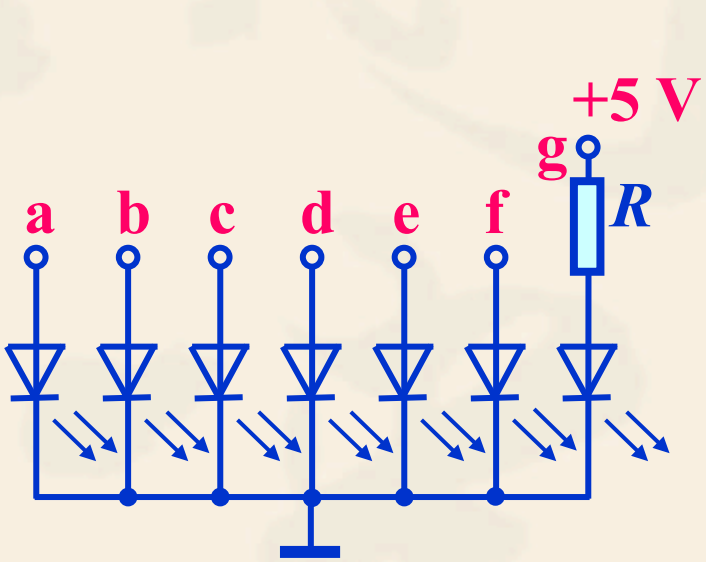
共阳极 — 低电平驱动

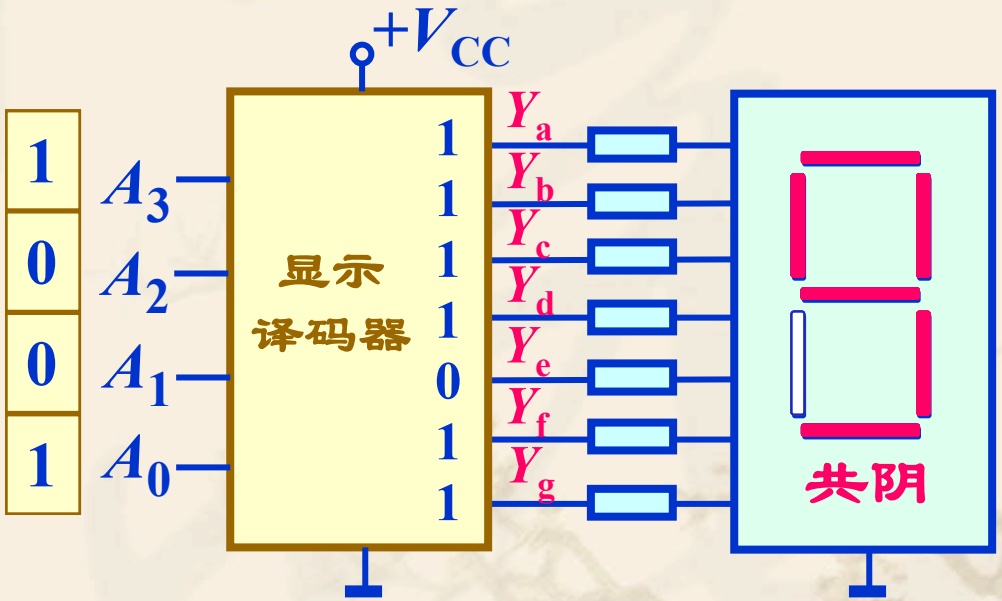




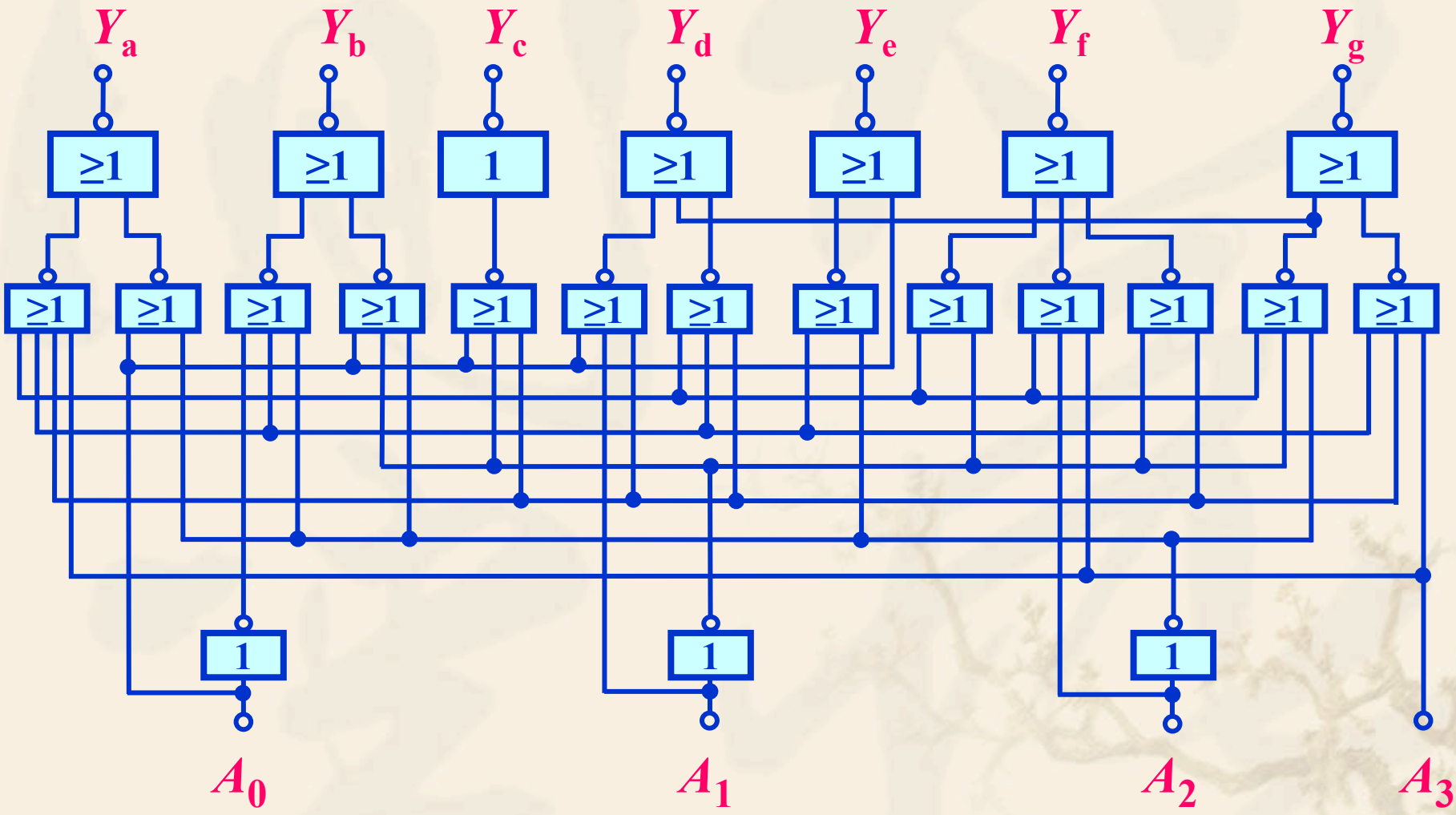
共阴极 — 高电平驱动


a
f
b
g
e
d
c

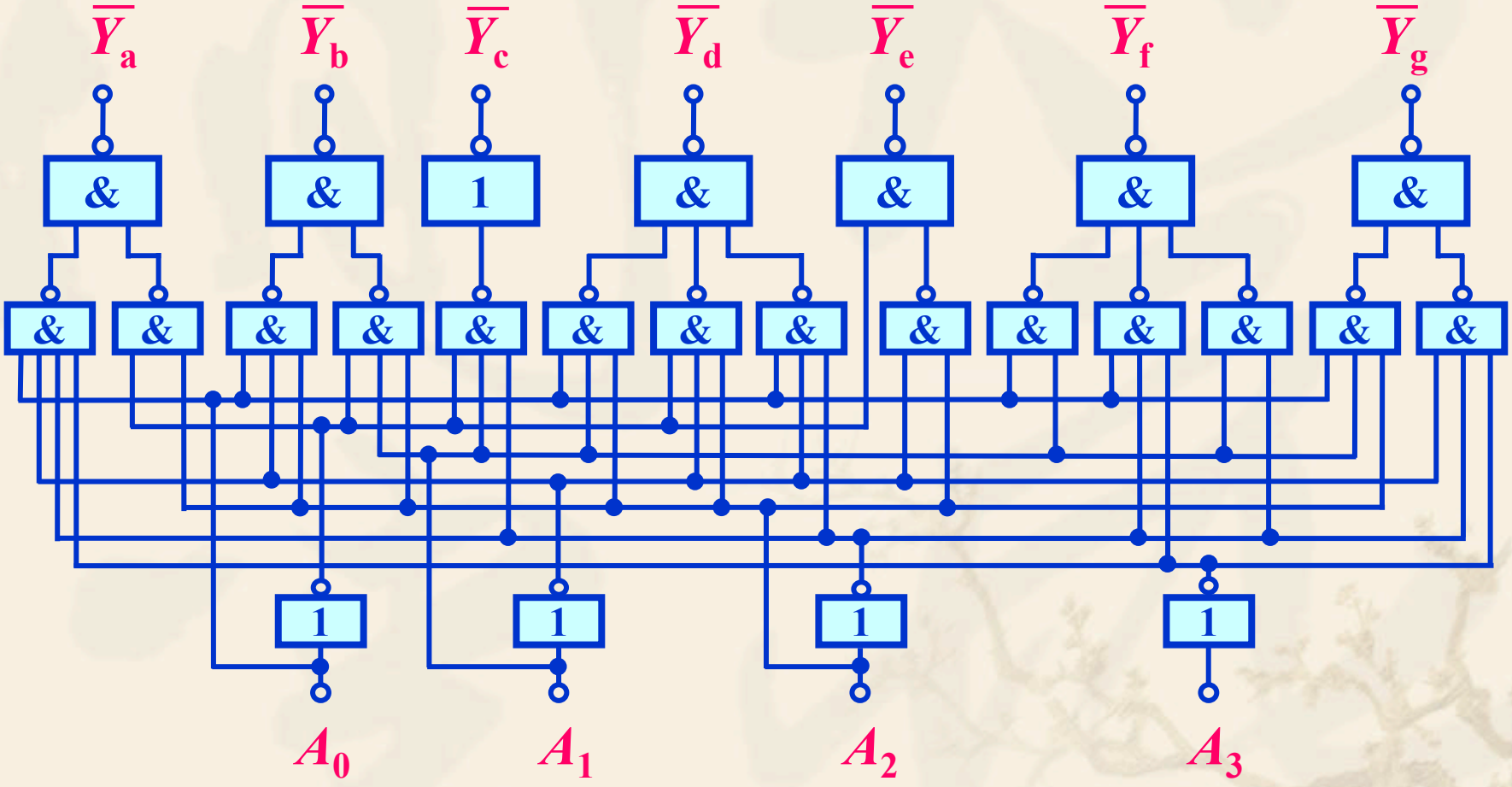

+5 V
g
R
a
b
c
d
e
f


+VCC
1
0
0
1
A3
A2
A1
A0
显示
译码器
1
1
1
1
0
1
1
Ya
Yb
Yc
Yd
Ye
Yf
Yg
共阴

驱动共阴极数码管的电路 — 输出高电平有效

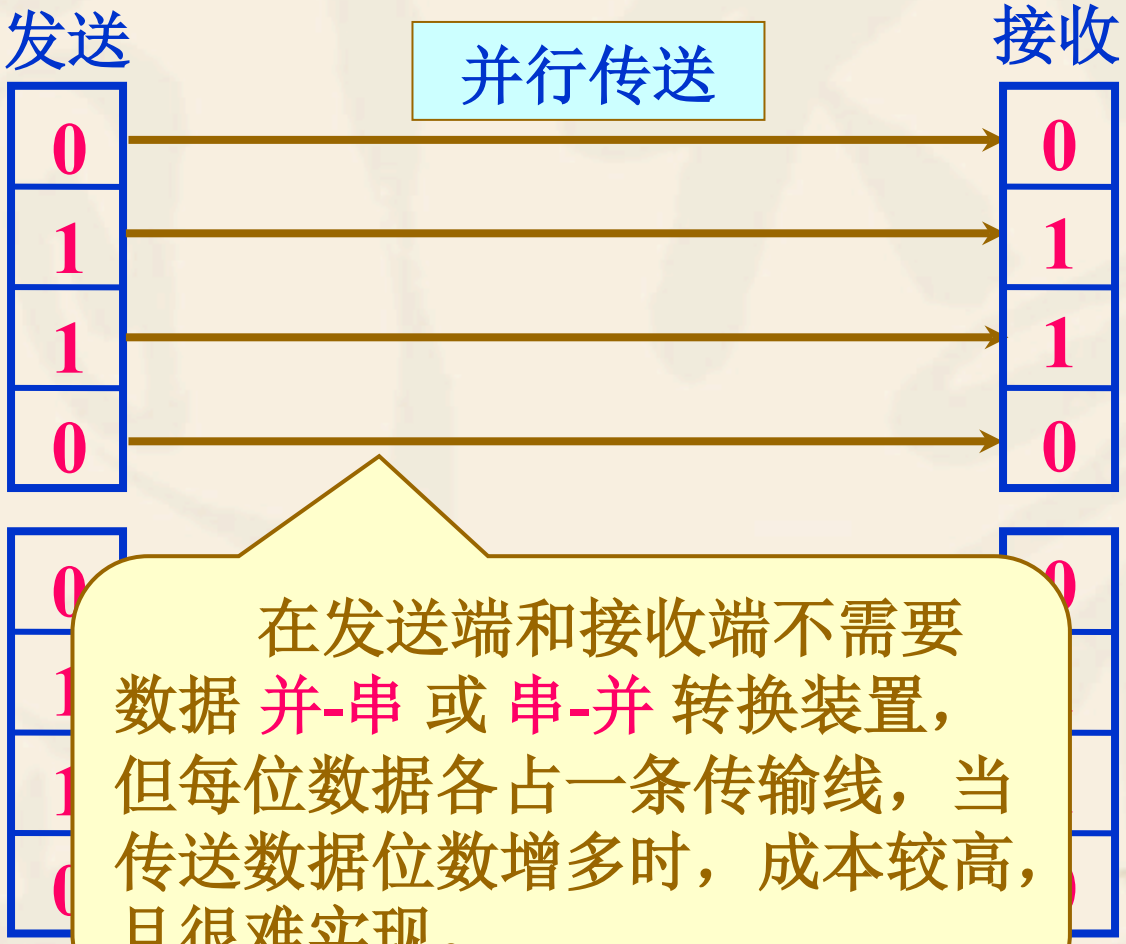


驱动共阳极数码管的电路 — 输出低电平有效



3.4 数据选择器和分配器

数据传输方式



在发送端和接收端不需要数据 并-串 或 串-并 转换装置，但每位数据各占一条传输线，当传送数据位数增多时，成本较高，且很难实现。

并-串转换：数据选择器　　串-并转换：数据分配器

3.4.1 数据选择器（Data Selector）

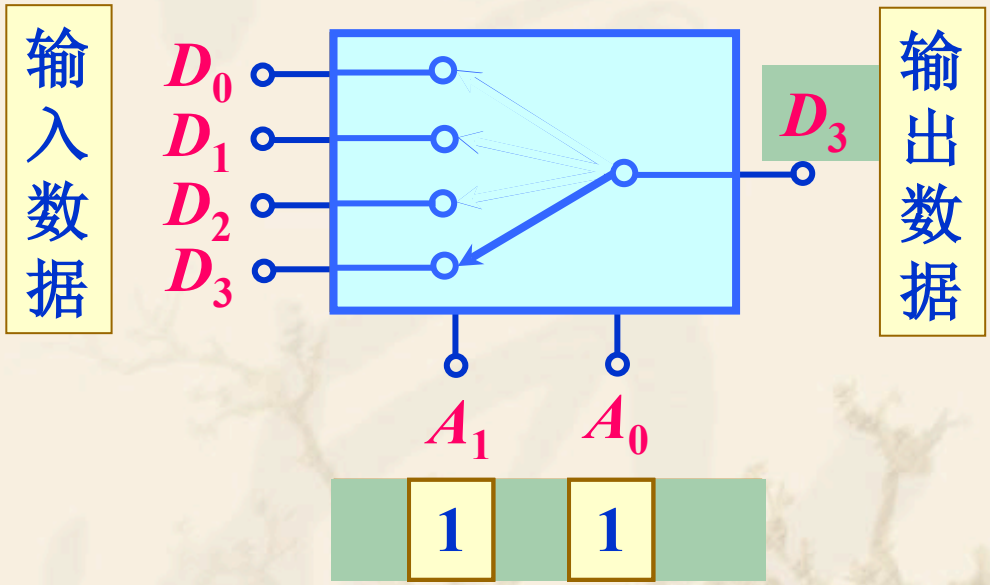
能够从多路数据输入中选择一路作为输出的电路

一、4 选 1 数据选择器

1. 逻辑抽象

真值表

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



2. 逻辑表达式 $Y = D_0\overline{A}_1\overline{A}_0 + D_1\overline{A}_1A_0 + D_2A_1\overline{A}_0 + D_3A_1A_0$

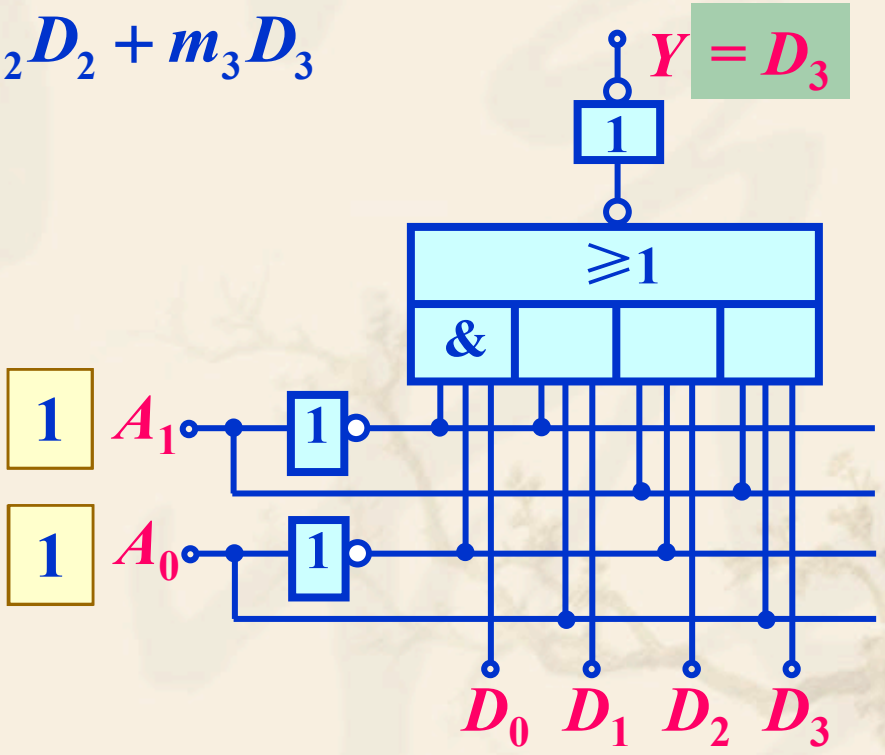
一、4 选 1 数据选择器

2. 逻辑表达式

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

$$= m_0D_0 + m_1D_1 + m_2D_2 + m_3D_3$$

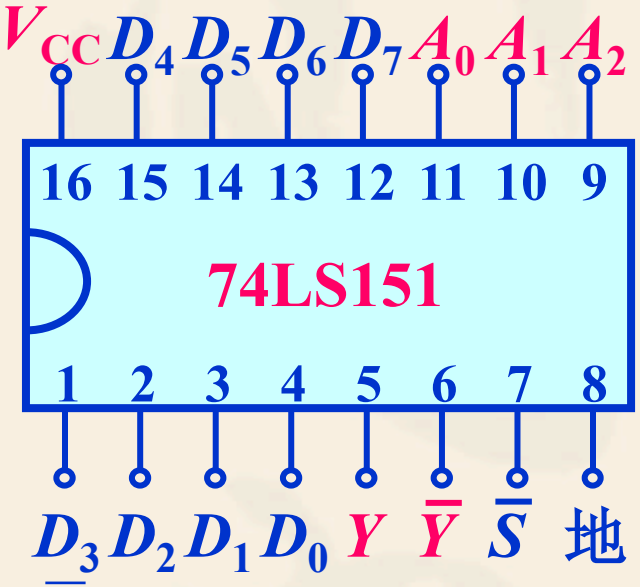
3. 逻辑图



二、集成数据选择器

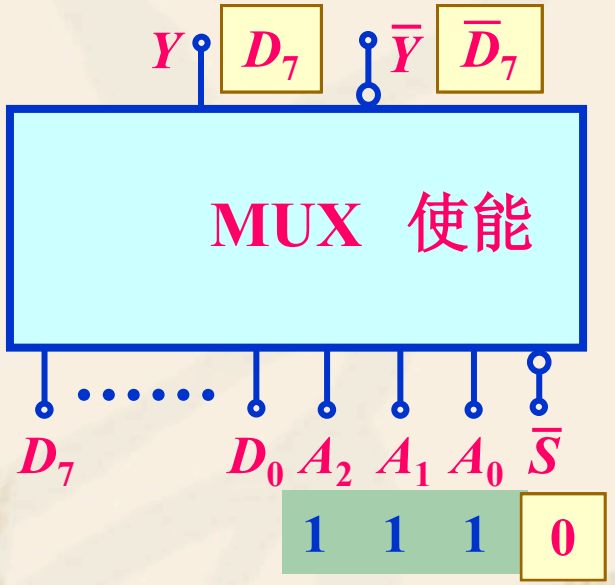
1. 8选1数据选择器　74151　74LS151　74251　74LS251

引脚排列图



$\overline{S}$ — 选通控制端

功能示意图

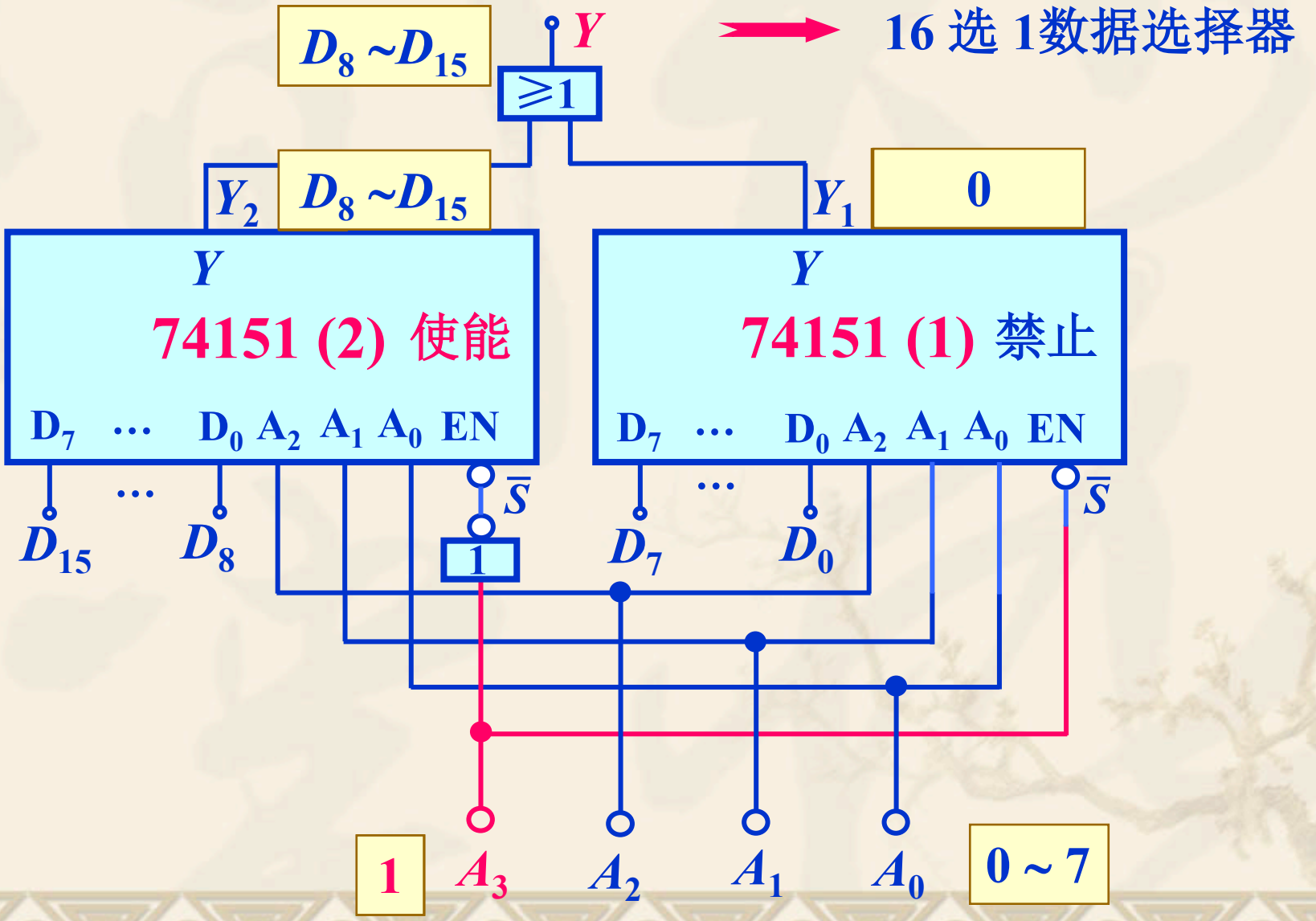


当$\overline{S}=1$时，选择器被禁止　$Y=0$　$\overline{Y}=1$

当$\overline{S}=0$时，选择器被选中（使能）

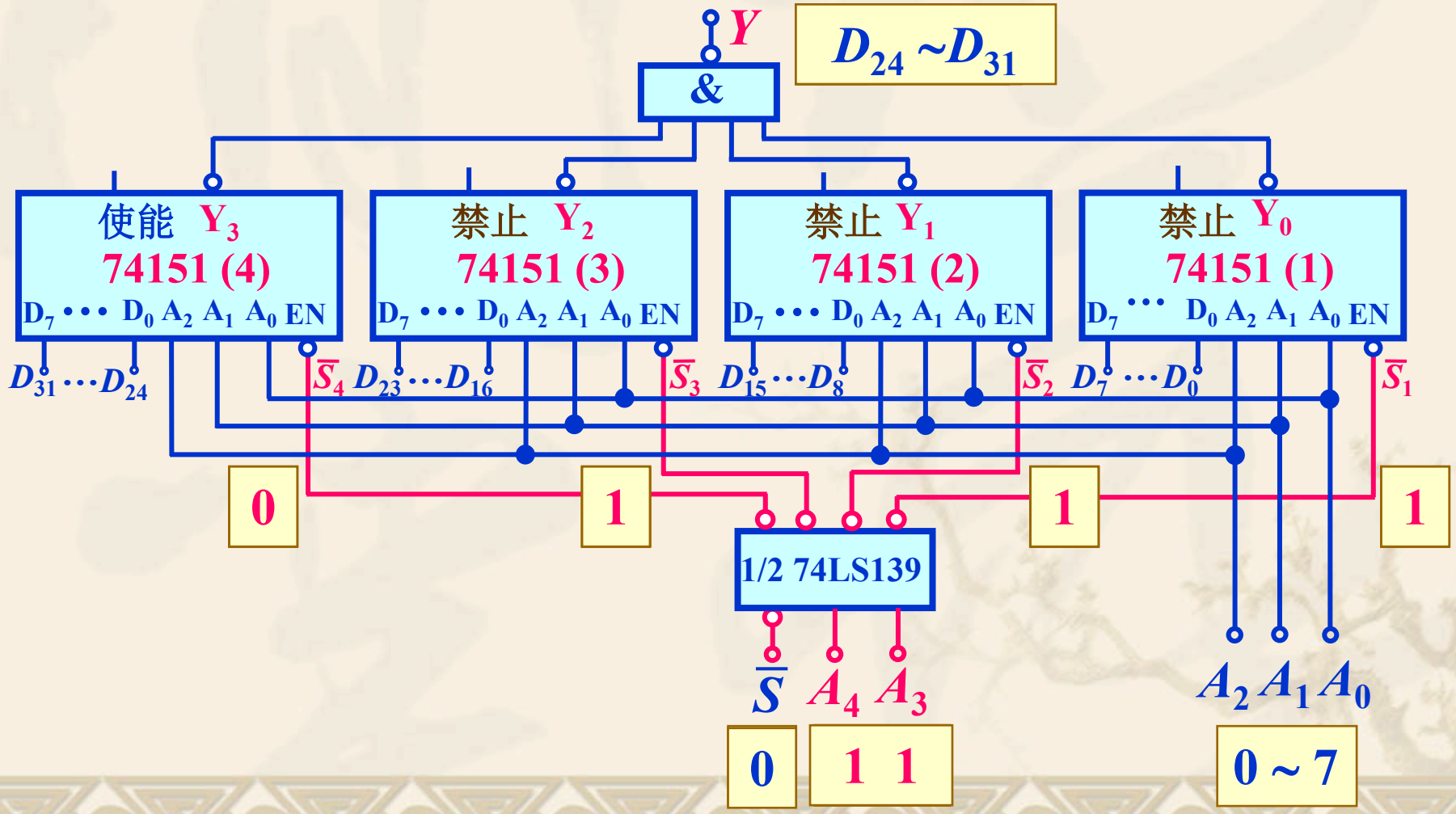
$$Y=D_0\overline{A}_2\overline{A}_1\overline{A}_0+D_1\overline{A}_2\overline{A}_1A_0+\cdots+D_7A_2A_1A_0$$

2. 集成数据选择器的扩展　两片 8 选 1（74151）



四片 8 选 1（74151）⟶ 32 选 1 数据选择器

方法 1： 74LS139 双 2 线 - 4 线译码器



四片 8 选 1（74151）→ 32 选 1 数据选择器

方法 1：真值表（使用 74LS139 双 2 线 - 4 线译码器）

A_4	A_3	译码器输出	(1)	(2)	(3)	(4)	输出信号
0	0	$\overline{Y}_0 = 0$	工	禁	禁	禁	$D_0 \sim D_7$
0	1	$\overline{Y}_1 = 0$	禁	工	禁	禁	$D_8 \sim D_{15}$
1	0	$\overline{Y}_2 = 0$	禁	禁	工	禁	$D_{16} \sim D_{23}$
1	1	$\overline{Y}_3 = 0$	禁	禁	禁	工	$D_{24} \sim D_{31}$

方法 2：74LS153 双 4 选 1 数据选择器（电路略）

四路 8 位并行数据 → 四片8选1 → 四路 1 位串行数据 → 一片4选1 → 一路 1 位串行数据

3.4.2 数据分配器 (Data Demultiplexer)

将 1 路输入数据，根据需要分别传送到 m 个输出端

一、1 路- 4 路数据分配器

数据输出

数据输入 D

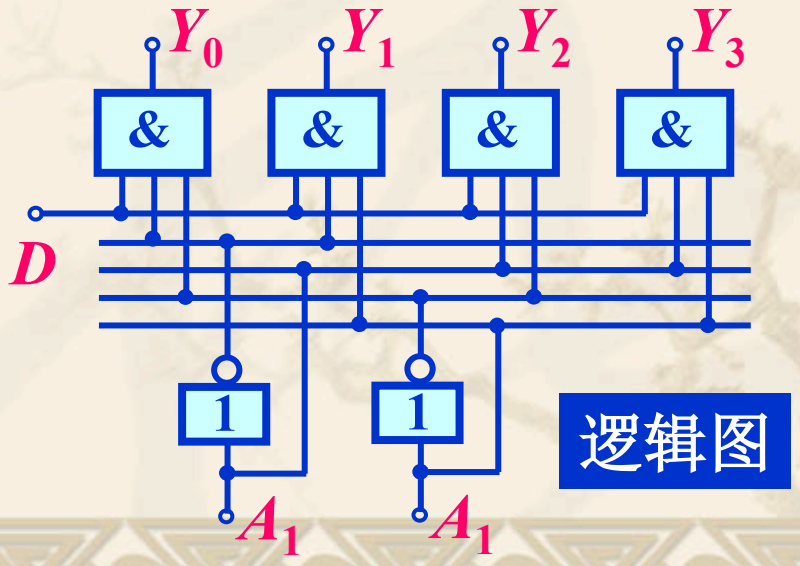
选择控制 A_1 A_0

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

函数式

真值表

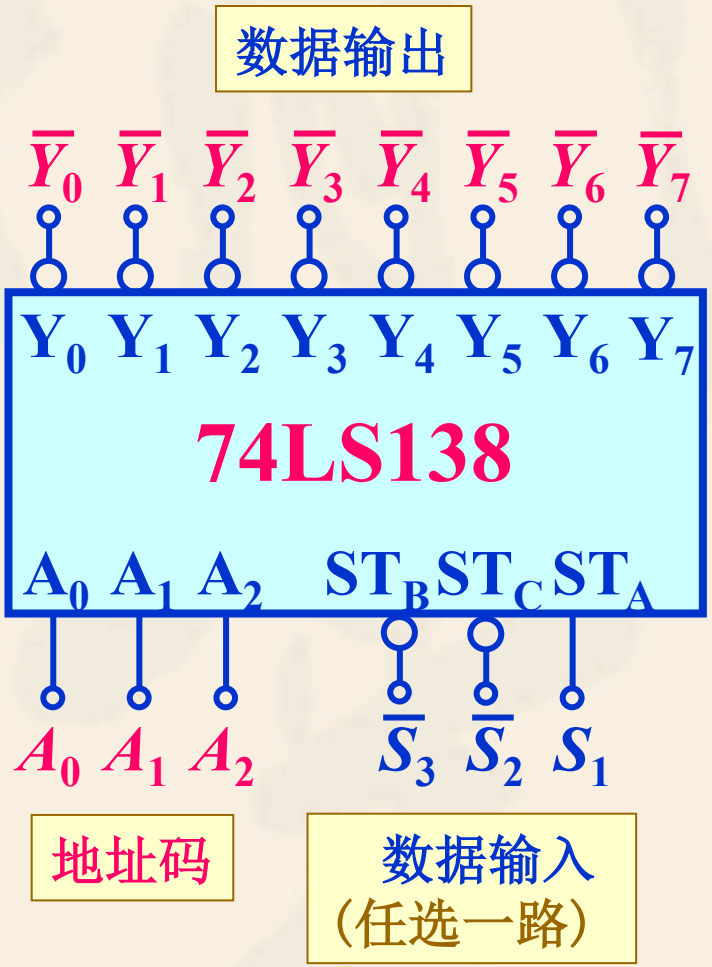
A_1	A_0	Y_0	Y_1	Y_2	Y_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D



逻辑图

二、集成数据分配器

用 3 线-8 线译码器可实现 1 路-8 路数据分配器



S_1 — 数据输入（D）

$\overline{Y}_0 \sim \overline{Y}_7$ — 数据输出（$\overline{D}$）

$\overline{S}_2$、$\overline{S}_3$ — 使能控制端

$\overline{S}_2 = \overline{S}_3 = 0$时，

实现数据分配器的功能 。

$\overline{S}_3$ — 数据输入（D）

$\overline{Y}_0 \sim \overline{Y}_7$ — 数据输出（D）

S_1、$\overline{S}_2$ — 使能控制端

$S_1 = 1$，$\overline{S}_2 = 0$时，

实现数据分配器的功能 。

3.5 用 MSI 实现组合逻辑函数

3.5.1 用数据选择器实现组合逻辑函数

一、基本原理和步骤

1. 原理：选择器输出为标准与或式，含地址变量的全部最小项。例如

4 选 1 $Y = D_0\overline{A}_1\overline{A}_0 + D_1\overline{A}_1A_0 + D_2A_1\overline{A}_0 + D_3A_1A_0$

8 选 1 $Y = D_0\overline{A}_2\overline{A}_1\overline{A}_0 + \cdots + D_7A_2A_1A_0$

而任何组合逻辑函数都可以表示成为最小项之和的形式，故可用数据选择器实现。

2. 基本步骤

(1) 根据 $n = k - 1$ 确定数据选择器的规模和型号

(n —选择器地址码，k —函数的变量个数)

(2) 写出函数的标准与或式和选择器输出信号表达式

(3) 对照比较确定选择器各个输入变量的表达式

(4) 根据采用的数据选择器和求出的表达式画出连线图。

二、应用举例

［例 3.5.1］用数据选择器实现函数 $F = AB + BC + AC$

［解］(1) $n = k - 1 = 3 - 1 = 2$ 可用 4 选 1 数据选择器 74LS153

(2) 标准与或式 $F = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$

数据选择器 $Y = D_0\overline{A}_1\overline{A}_0 + D_1\overline{A}_1A_0 + D_2A_1\overline{A}_0 + D_3A_1A_0$

(3) 确定输入变量和地址码的对应关系

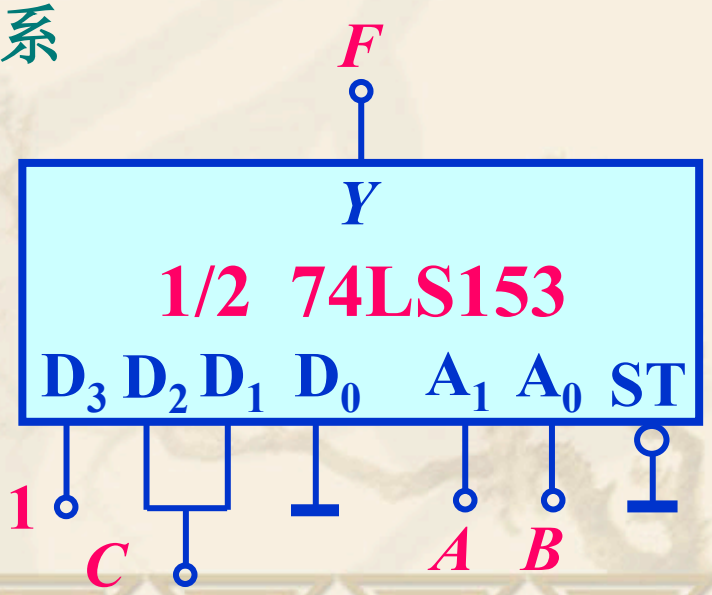
方法一：令 $A_1 = A,\ A_0 = B$

$Y = D_0\overline{A}\,\overline{B} + D_1\overline{A}B + D_2A\overline{B} + D_3AB$

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

则 $D_0 = 0$ $D_1 = D_2 = C$ $D_3 = 1$

(4) 画连线图



二、应用举例

［例 3.5.1］用数据选择器实现函数 $F = AB + BC + AC$

［解］ 方法二：令 $A_1 = B,\ A_0 = C$

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

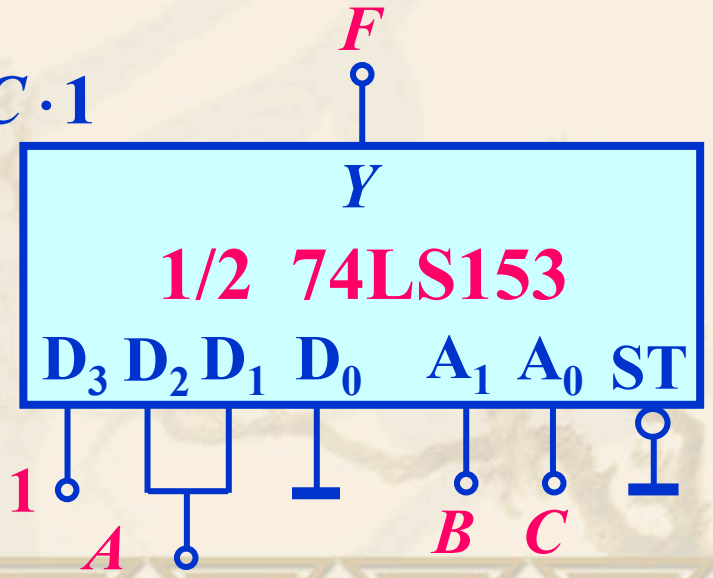
$$= D_0\overline{B}\overline{C} + D_1\overline{B}C + D_2B\overline{C} + D_3BC$$

$$F = \overline{B}CA + B\overline{C}A + BC\overline{A} + BCA$$

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

则 $D_0 = 0 \quad D_1 = D_2 = A \quad D_3 = 1$

画连线图



[例] 用数据选择器实现函数 $Z=\sum_m(3,4,5,6,7,8,9,10,12,14)$

[解] (1) $n=k$-1 = 4-1 = 3 用 8 选 1 数据选择器 74LS151

(2) 函数 Z 的标准与或式

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

8 选 1 $Y=D_0\overline{A}_2\overline{A}_1\overline{A}_0+D_1\overline{A}_2\overline{A}_1A_0+\cdots+D_7A_2A_1A_0$

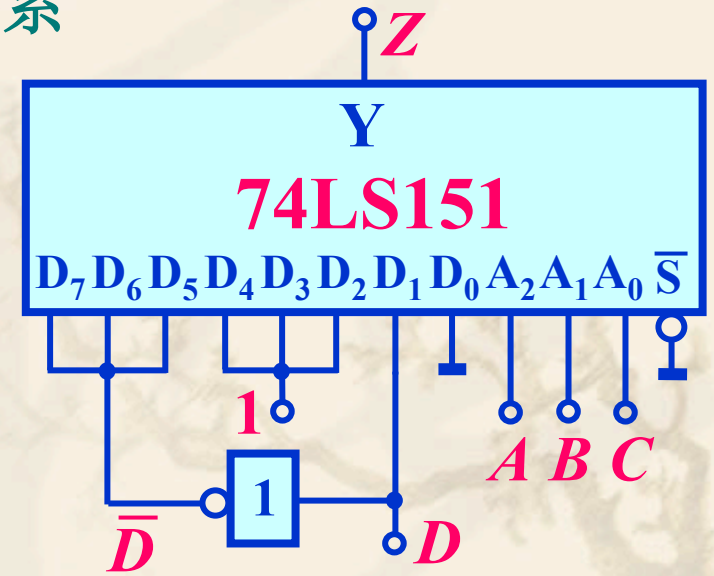
(3) 确定输入变量和地址码的对应关系

若令 $A_2=A,\ A_1=B,\ A_0=C$

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

则 $D_1=D$ $D_2=D_3=D_4=1$

$D_5=D_6=D_7=\overline{D}$ $D_0=0$

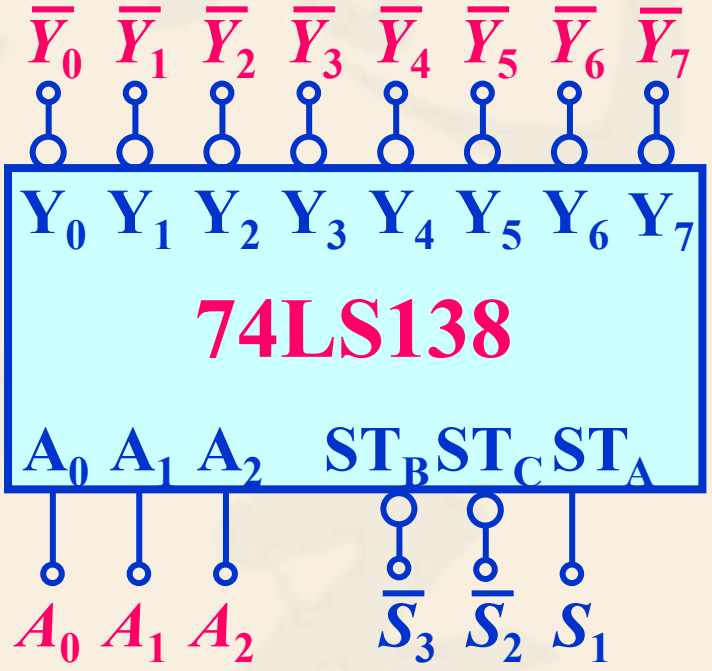


(4) 画连线图

3.5.2 用二进制译码器实现组合逻辑函数

一、基本原理与步骤

1. 基本原理： 二进制译码器又叫变量译码器或最小项译码器,它的输出端提供了其输入变量的全部最小项。



$$S_1 = 1, \overline{S}_2 = \overline{S}_3 = 0$$

$$\overline{Y}_0 = \overline{\overline{A}_2\overline{A}_1\overline{A}_0} = \overline{m}_0$$

$$\overline{Y}_1 = \overline{\overline{A}_2\overline{A}_1 A_0} = \overline{m}_1$$

$$\vdots$$

$$\overline{Y}_7 = \overline{A_2 A_1 A_0} = \overline{m}_7$$

任何一个函数都可以写成最小项之和的形式

2. 基本步骤

(1) 选择集成二进制译码器

(2) 写函数的标准与非-与非式

(3) 确认变量和输入关系

(4) 画连线图

二、应用举例

[例] 用集成译码器实现函数 $Z = AB + BC + AC$

[解] (1) 三个输入变量，选 3 线 – 8 线译码器 74LS138

(2) 函数的标准与非-与非式

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

$$= m_3 + m_5 + m_6 + m_7$$

$$= \overline{m_3 \cdot m_5 \cdot m_6 \cdot m_7}$$

[例] 用集成译码器实现函数 $Z = AB + BC + AC$

[解] 选 3 线 – 8 线译码器 74LS138

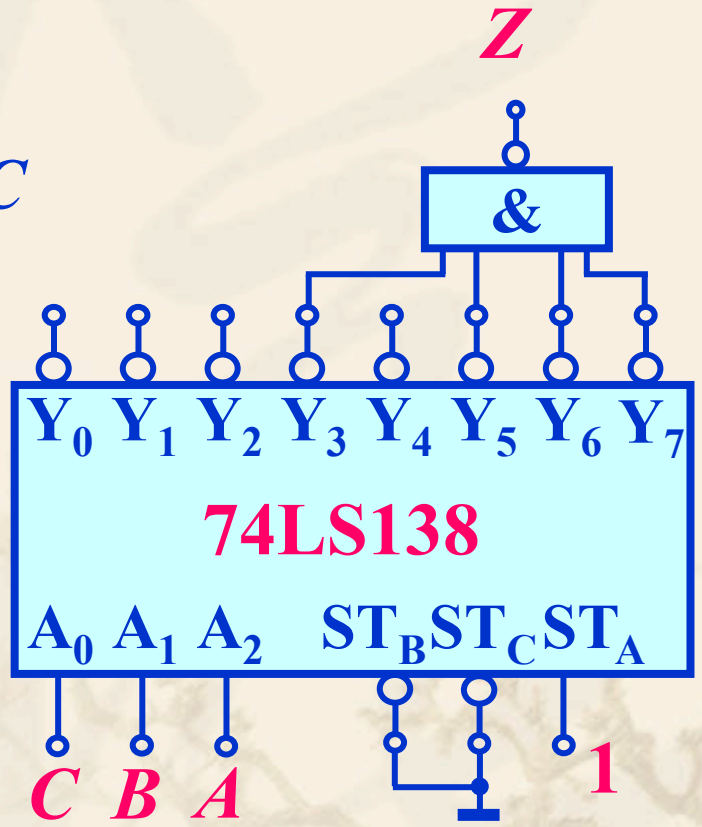
(3) 确认变量和输入关系

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

$$= \overline{\overline{m_3} \cdot \overline{m_5} \cdot \overline{m_6} \cdot \overline{m_7}}$$

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

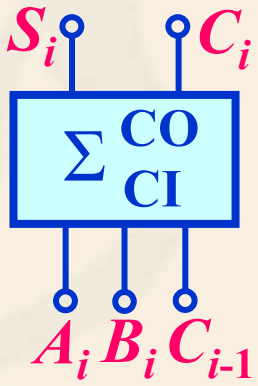
则 $Z = \overline{\overline{Y_3} \cdot \overline{Y_5} \cdot \overline{Y_6} \cdot \overline{Y_7}}$



(4) 画连线图

在输出端需增加一个与非门

[例 3. 5. 2] 试用集成译码器设计一个全加器。



[解] (1) 选择译码器：全加器的符号如图所示

选 3 线 – 8 线译码器 74LS138

(2) 写出函数的标准与非-与非式

$$S_i = \overline{A}_i\overline{B}_iC_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + A_i\overline{B}_i\overline{C}_{i-1} + A_iB_iC_{i-1}$$

$$= m_1 + m_2 + m_4 + m_7 = \overline{m_1 \cdot m_2 \cdot m_4 \cdot m_7}$$

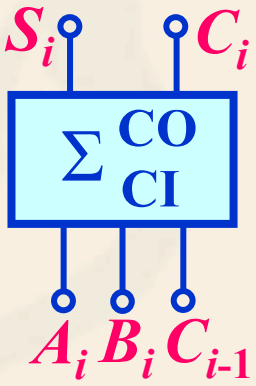
$$C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1}$$

$$= \overline{A}_iB_iC_{i-1} + A_i\overline{B}_iC_{i-1} + A_iB_i\overline{C}_{i-1} + A_iB_iC_{i-1}$$

$$= m_3 + m_5 + m_6 + m_7$$

$$= \overline{m_3 \cdot m_5 \cdot m_6 \cdot m_7}$$

[例 3. 5. 2] 试用集成译码器设计一个全加器。



[解] 选 3 线 – 8 线译码器 74LS138

(2) 函数的标准与非-与非式

$$S_i = \overline{\overline{m_1} \cdot \overline{m_2} \cdot \overline{m_4} \cdot \overline{m_7}} \qquad C_i = \overline{\overline{m_3} \cdot \overline{m_5} \cdot \overline{m_6} \cdot \overline{m_7}}$$

(3) 确认表达式

$$A_2 = A_i \quad A_1 = B_i \quad A_0 = C_{i-1}$$

$$S_i = \overline{\overline{Y_1} \cdot \overline{Y_2} \cdot \overline{Y_4} \cdot \overline{Y_7}}$$

$$C_i = \overline{\overline{Y_3} \cdot \overline{Y_5} \cdot \overline{Y_6} \cdot \overline{Y_7}}$$

(4) 画连线图