第四章 组合逻辑电路

主 要 内 容

- 组合逻辑电路的特点
- 组合逻辑电路的分析与设计方法
- 几种常用的组合逻辑电路
  - 编码器、译码器、数据选择器、加法器和数值比较器
- 层次化和模块化的设计方法
- 可编程逻辑器件
- 硬件描述语言
- 用可编程通用模块设计组合逻辑电路
- 竞争-冒险现象及其消除方法

4.1 概述

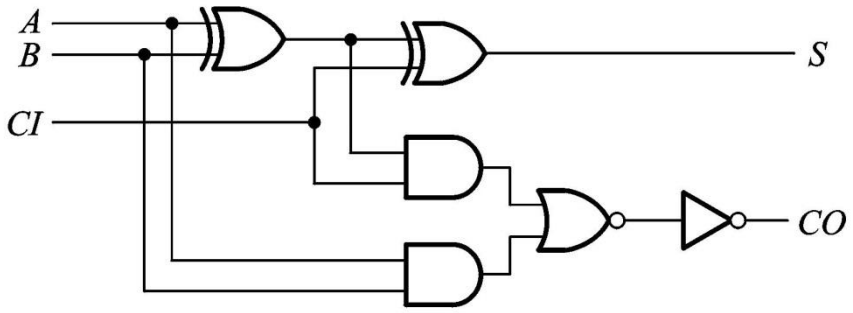
一、组合逻辑电路的特点

1. 从功能上

2. 从电路结构上

任意时刻的输出仅取决于该时刻的输入

不含记忆（存储）元件



4-1-1

二、逻辑功能的描述

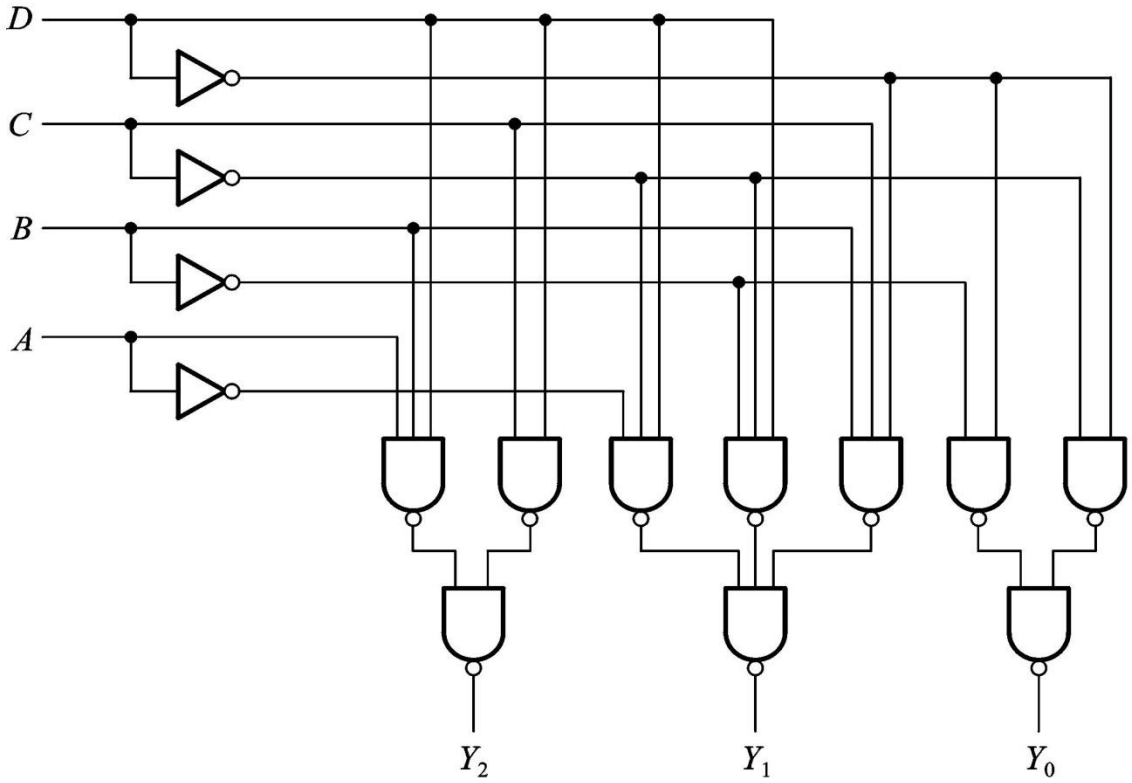
组合逻辑电路的框图

逻辑函数式

$$y_1 = f_1(a_1 a_2 \cdots a_n)$$
$$y_2 = f_2(a_1 a_2 \cdots a_n) \quad \Rightarrow \quad Y = F(A)$$
$$\vdots$$
$$y_m = f_m(a_1 a_2 \cdots a_n)$$

4.2.1 组合逻辑电路的分析方法

- 所谓分析一个给定的逻辑电路，就是要通过分析找出电路的逻辑功能来。
- 分析过程
  - 从电路的输入到输出逐级写出逻辑函数式，最后得到表示输出与输入关系的逻辑函数式
  - 用公式化简法或卡诺图化简法将得到的函数式化简或变换，使逻辑关系简单明了。
  - 为了使电路的逻辑功能更加直观，有时还可以将逻辑函数式转换为真值表的形式。



$$\begin{cases} Y_2 = ((DC)'(DBA)')' = DC + DBA \\ Y_1 = ((D'CB)'(DC'B')'(DC'A')')' = D'CB + DC'B' + DC'A' \\ Y_0 = ((D'C')'(D'B')')' = D'C' + D'B' \end{cases}$$

输入				输出		
D	C	B	A	Y_2	Y_1	Y_0
0	0	0	0	0	0	1
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	0	1
0	1	0	1	0	0	1
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	1	0
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	1	0	0

4.2.2 组合逻辑电路的设计方法

- 根据给出的实际逻辑问题，求出实现这一逻辑功能的最简单逻辑电路，这就是设计组合逻辑电路时要完成的工作。
- 所谓的“最简”
  - 电路所用的器件数量最少
  - 器件的种类最少
  - 器件之间的连线最少

设计步骤

一、逻辑抽象

- 分析因果关系，确定输入/输出变量
- 定义逻辑状态的含意（赋值）
- 列出真值表

二、写出函数式

三、选定器件类型

四、根据所选器件：对逻辑式化简（用门）

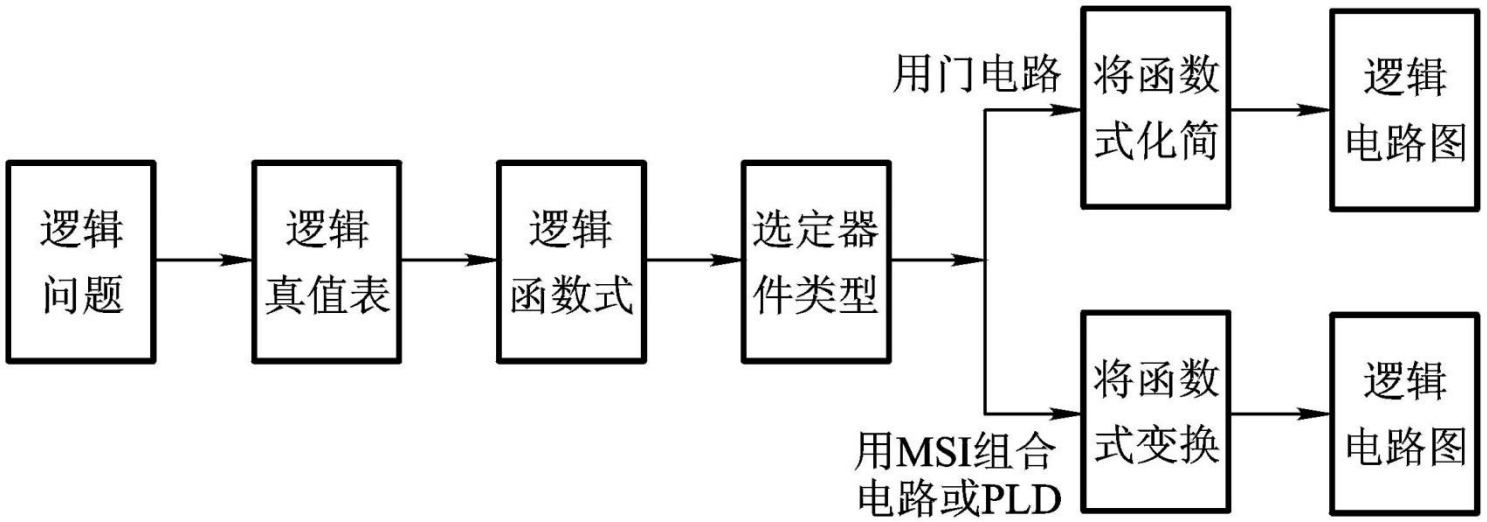
变换（用中规模集成电路MSI）

或进行相应的描述（可编程逻辑器件PLD）

五、画出逻辑电路图，或下载到PLD

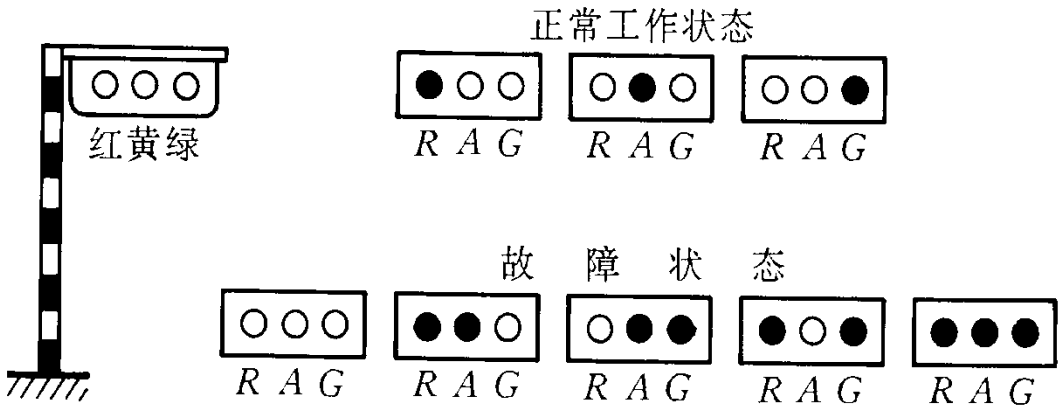
六、工艺设计

设计步骤


逻辑
问题
逻辑
真值表
逻辑
函数式
选定器
件类型
用门电路
将函数
式化简
逻辑
电路图
用MSI组合
电路或PLD
将函数
式变换
逻辑
电路图

设计举例：

- 设计一个监视交通信号灯状态的逻辑电路



设计举例：

1.抽象

- 输入变量：

  红（R）、黄（A）、绿（G）

- 输出变量：

  故障信号（Z）

2.写出逻辑表达式

$$Z = R'A'G' + R'AG + RA'G + RAG' + RAG$$

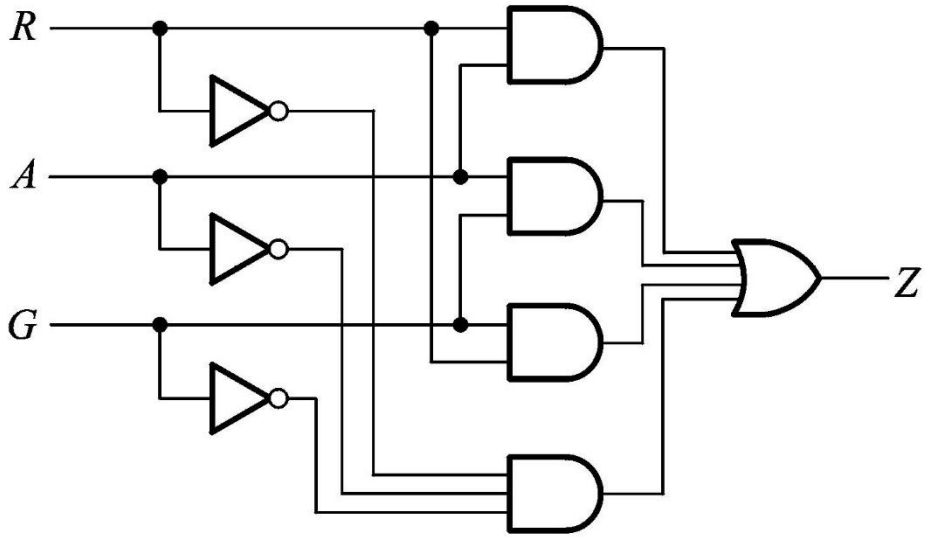
输入变量			输出
R	A	G	Z
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

设计举例：

3. 选用小规模SSI器件
4. 化简

$$Z = R'A'G' + RA + RG + AG$$

5. 画出逻辑图



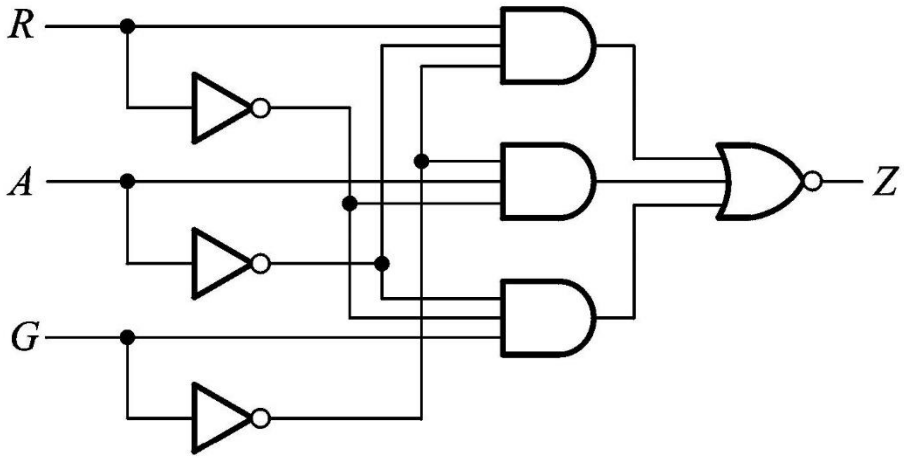
设计举例：

4. 化简（与或非）

$$Z=(RA'G'+R'AG'+R'A'G)'$$

R \ AG	00	01	11	10
0	1	0	1	0
1	0	1	1	1

5. 画出逻辑图（与或非）



4.3 若干常用组合逻辑电路

4.3.1 编码器

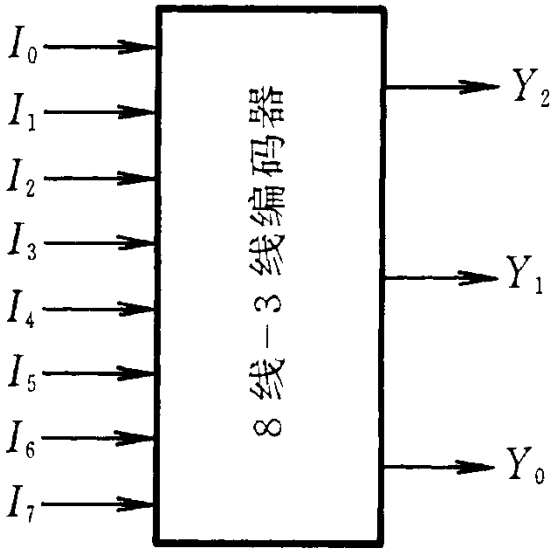
- 编码：将输入的每个高、低电平信号编成一个对应的二进制代码

- 普通编码器
- 优先编码器

一、普通编码器

- 特点：任何时刻只允许输入一个编码信号。
- 例：3位二进制普通编码器(8线-3线)

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



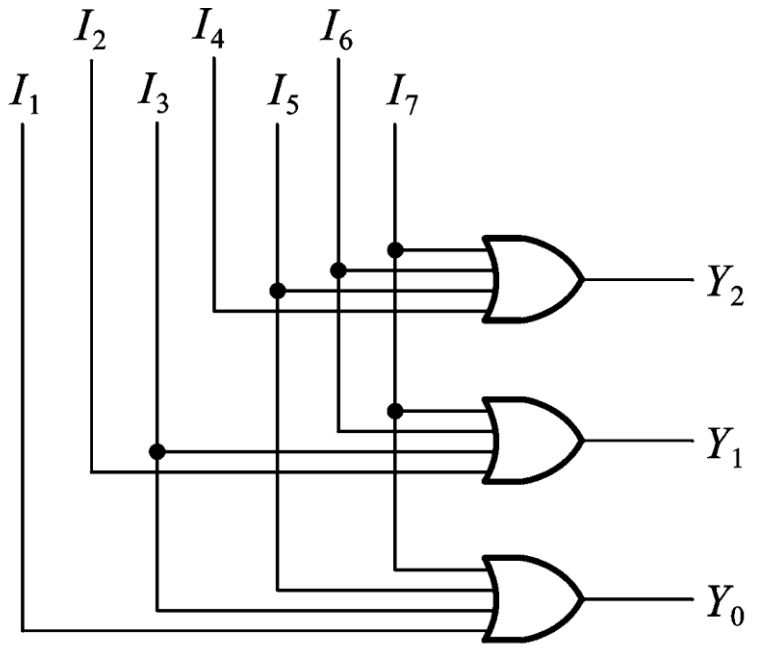
$$Y_2 = I_7'I_6'I_5'I_4I_3'I_2'I_1'I_0' + I_7'I_6'I_5I_4'I_3'I_2'I_1'I_0' + I_7'I_6I_5'I_4'I_3'I_2'I_1'I_0' + I_7I_6'I_5'I_4'I_3'I_2'I_1'I_0'$$

利用无关项化简，得：

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

$$Y_1 = I_2 + I_3 + I_6 + I_7$$

$$Y_0 = I_1 + I_3 + I_5 + I_7$$



二、优先编码器

- 特点：允许同时输入两个以上的编码信号，但只对其中优先权最高的一个进行编码。
- 例：8线-3线优先编码器
- （设I_7优先权最高…I_0优先权最低）

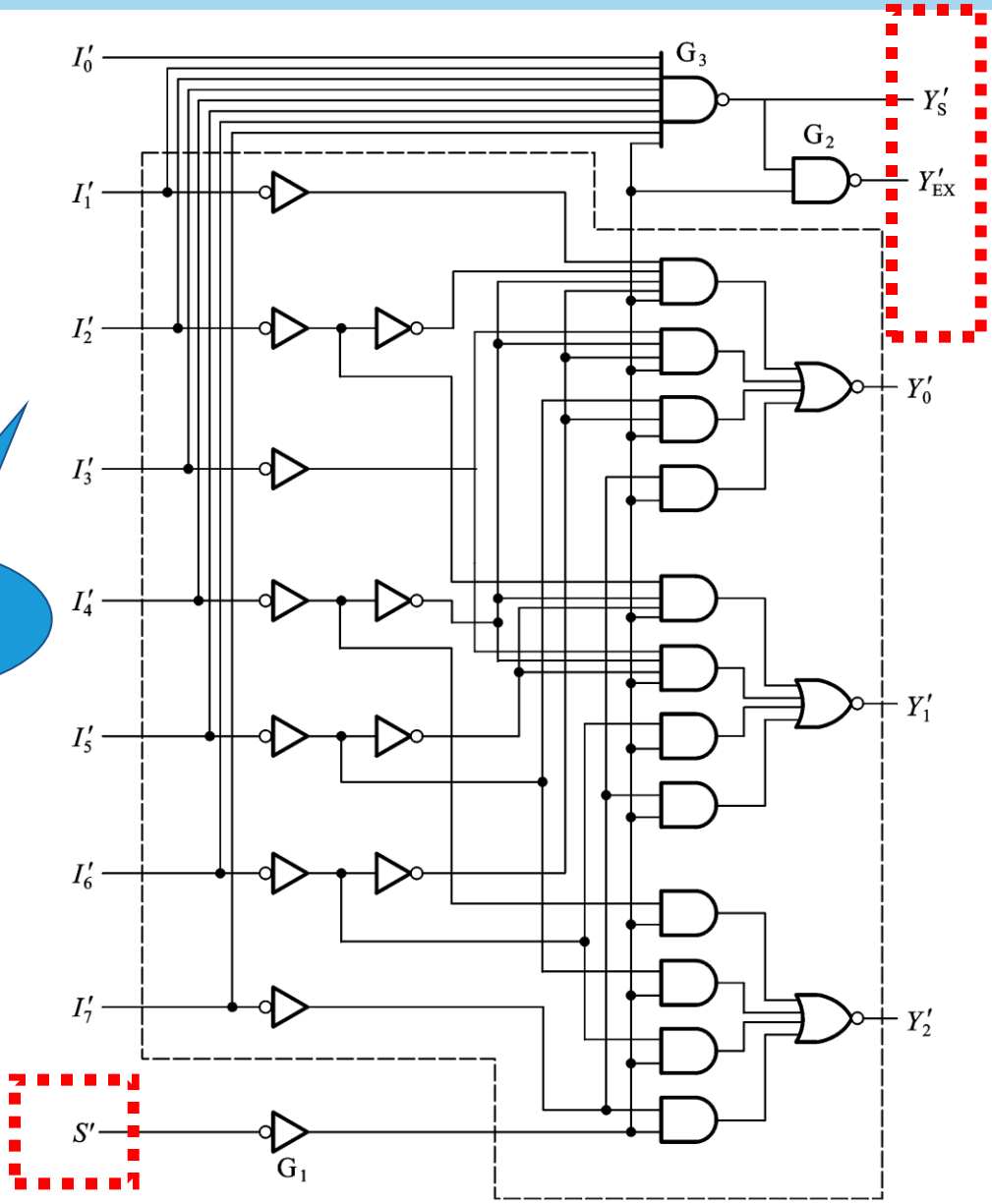
输入								输出		
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Y_2	Y_1	Y_0
X	X	X	X	X	X	X	1	1	1	1
X	X	X	X	X	X	1	0	1	1	0
X	X	X	X	X	1	0	0	1	0	1
X	X	X	X	1	0	0	0	1	0	0
X	X	X	1	0	0	0	0	0	1	1
X	X	1	0	0	0	0	0	0	1	0
X	1	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0

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

$$A + A'B = A + B$$

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

实例：74HC148



低电平

$$Y_2' = (I_7 + I_6 + I_5 + I_4)'$$

选通信号

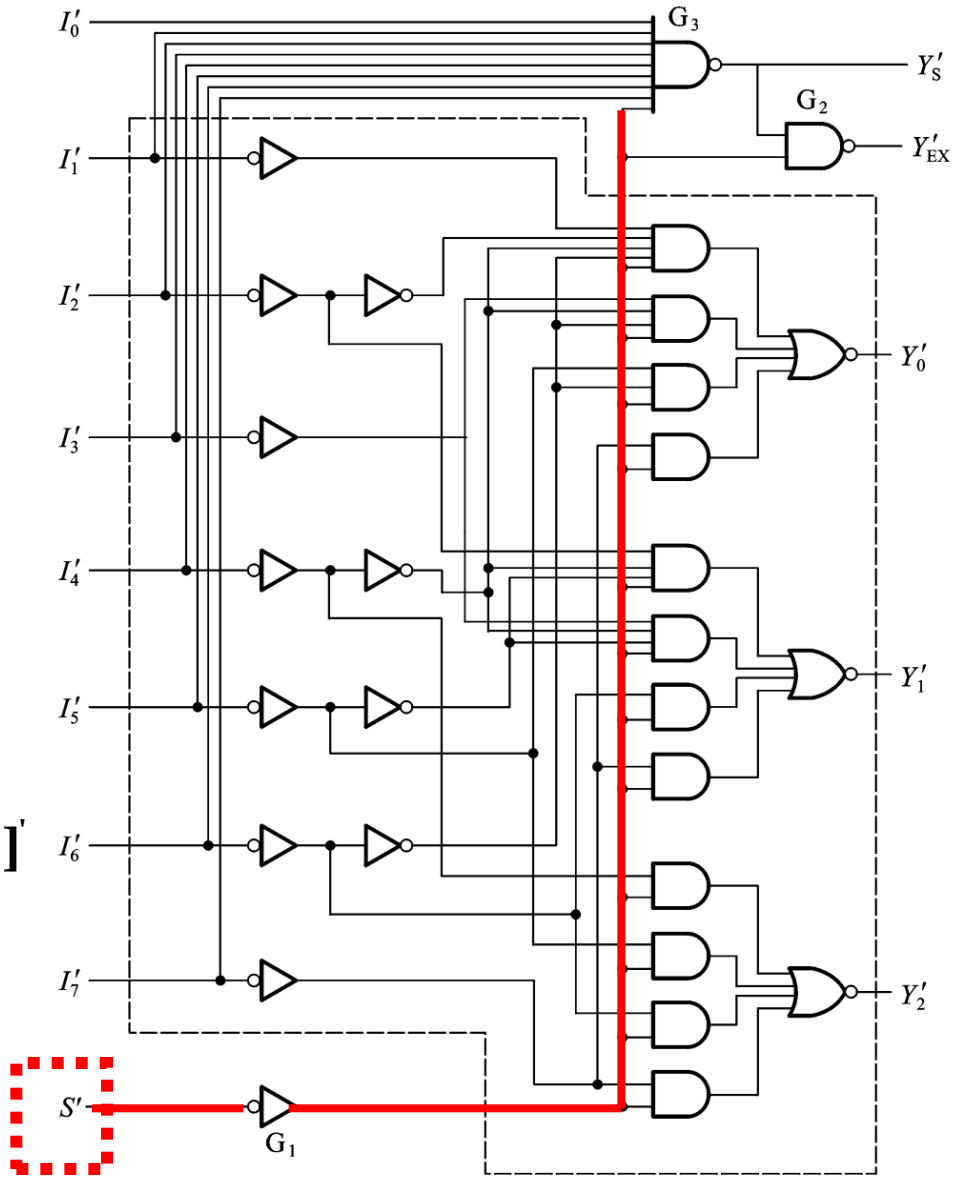
$$Y_2' = [(I_7 + I_6 + I_5 + I_4)S]'$$

$$Y_2' = [(I_7 + I_6 + I_5 + I_4)S]'$$

$$Y_1' = [(I_7 + I_6 + I_5 I_4' I_3' + I_2 I_4' I_5')S]'$$

$$Y_0' = [(I_7 + I_6' I_5 + I_3 I_4' I_6' + I_1 I_2 I_4' I_6')S]'$$

选通信号



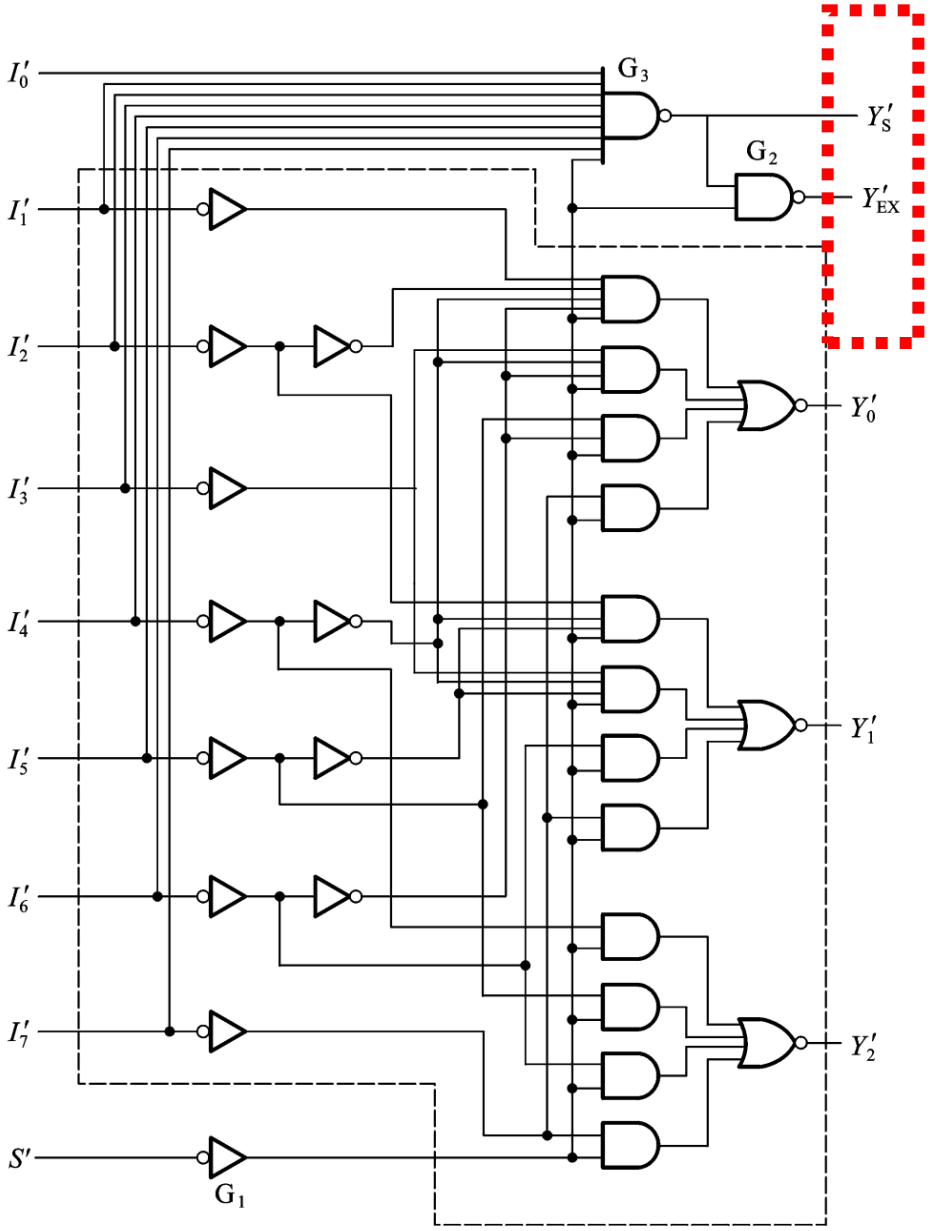
为0时，电路工作
无编码输入

$$Y_S' = (I_7'I_6'I_5'I_4'I_3'I_2'I_1'I_0'S)'$$

$$Y_{EX}' = [(I_7'I_6'I_5'I_4'I_3'I_2'I_1'I_0'S)' \quad S]'$$

$$= [(I_7 + I_6 + I_5 + I_4 + I_3 + I_2 + I_1 + I_0)S]'$$

为0时，电路工作
有编码输入



附加输出信号

输入									输出				
S'	I_0'	I_1'	I_2'	I_3'	I_4'	I_5'	I_6'	I_7'	Y_2'	Y_1'	Y_0'	Y_S'	Y_{EX}'
1	X	X	X	X	X	X	X	X	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	0	1
0	X	X	X	X	X	X	X	0	0	0	0	1	0
0	X	X	X	X	X	X	0	1	0	0	1	1	0
0	X	X	X	X	X	0	1	1	0	1	0	1	0
0	X	X	X	X	0	1	1	1	0	1	1	1	0
0	X	X	X	0	1	1	1	1	1	0	0	1	0
0	X	X	0	1	1	1	1	1	1	0	1	1	0
0	X	0	1	1	1	1	1	1	1	1	0	1	0
0	0	1	1	1	1	1	1	1	1	1	1	1	0

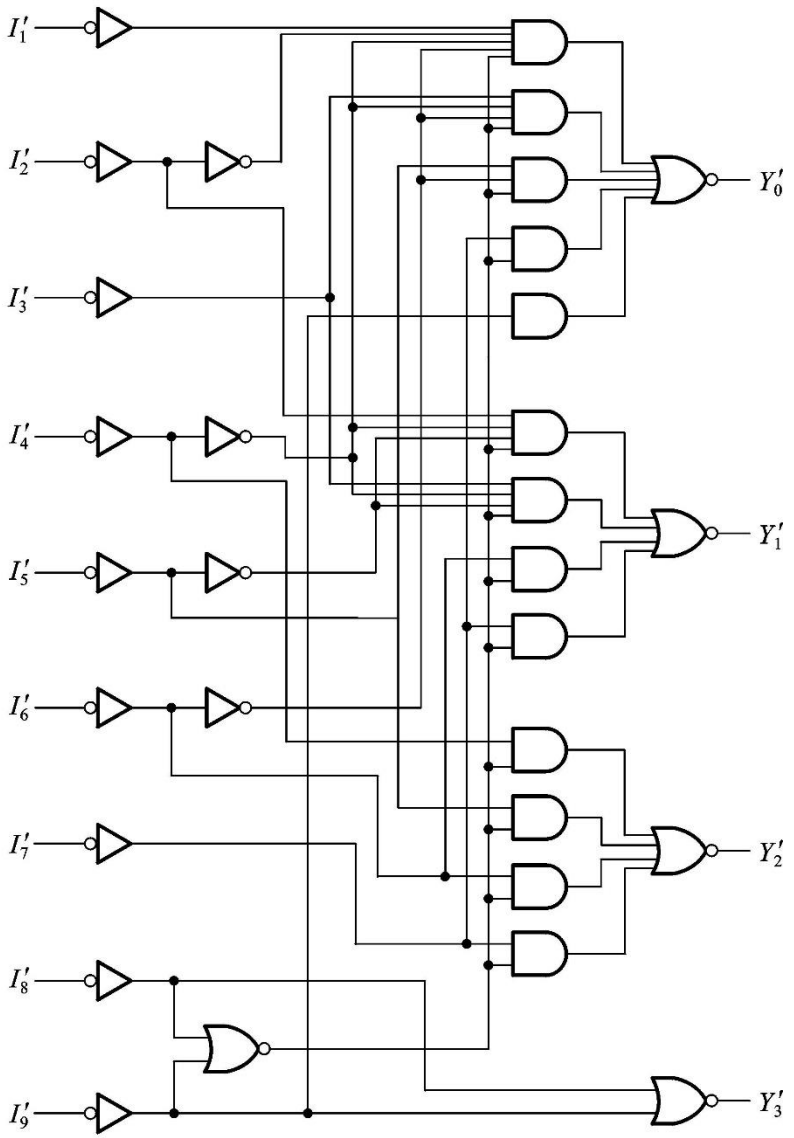
附加输出信号的状态及含意

$Y_S^{'}$	$Y_{EX}^{'}$	状态
1	1	不工作
0	1	工作，但无输入
1	0	工作，且有输入
0	0	不可能出现

三、二-十进制优先编码器

- 将 $I_9' \sim I_1'$ 编成0110 ~ 1110（BCD码的反码形式）
- I_9'的优先权最高，I_0' 最低
- 输入的低电平信号变成一个对应的十进制的编码

二-十进制优先编码器74LS147



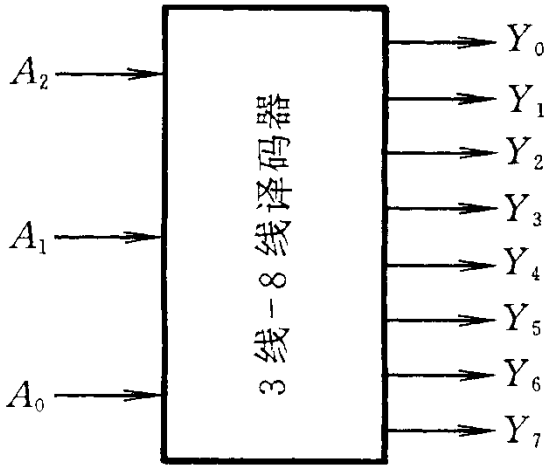
输入									输出			
$\overline{I}_1$	$\overline{I}_2$	$\overline{I}_3$	$\overline{I}_4$	$\overline{I}_5$	$\overline{I}_6$	$\overline{I}_7$	$\overline{I}_8$	$\overline{I}_9$	$\overline{Y}_3$	$\overline{Y}_2$	$\overline{Y}_1$	$\overline{Y}_0$
1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0

4.3.2 译码器

- 译码：将每个输入的二进制代码译成对应的输出高、低电平信号或另外一个代码。（译码是编码的反操作）
- 常用的有：二进制译码器，二-十进制译码器，显示译码器等。

一、二进制译码器

例：3线—8线译码器



输入			输出							
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

真值表 → 逻辑表达式：

$$Y_0 = A_2' A_1' A_0' = m_0$$

$$Y_1 = A_2' A_1' A_0 = m_1$$

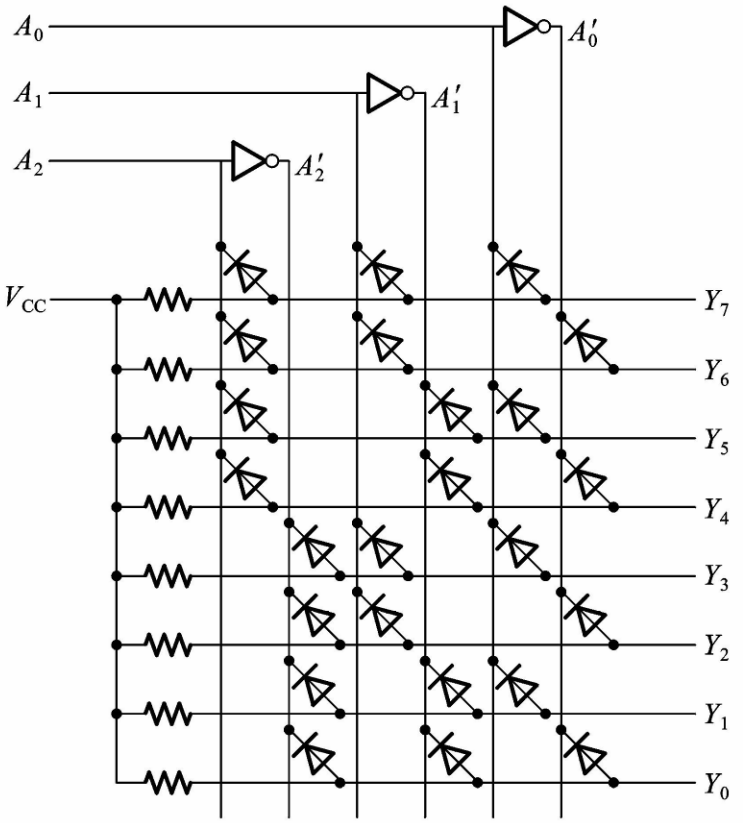
$$Y_2 = A_2' A_1 A_0' = m_2$$

…

$$Y_7 = A_2 A_1 A_0 = m_7$$

→ 用电路进行实现

用二极管与门阵列组成的3线—8线译码器

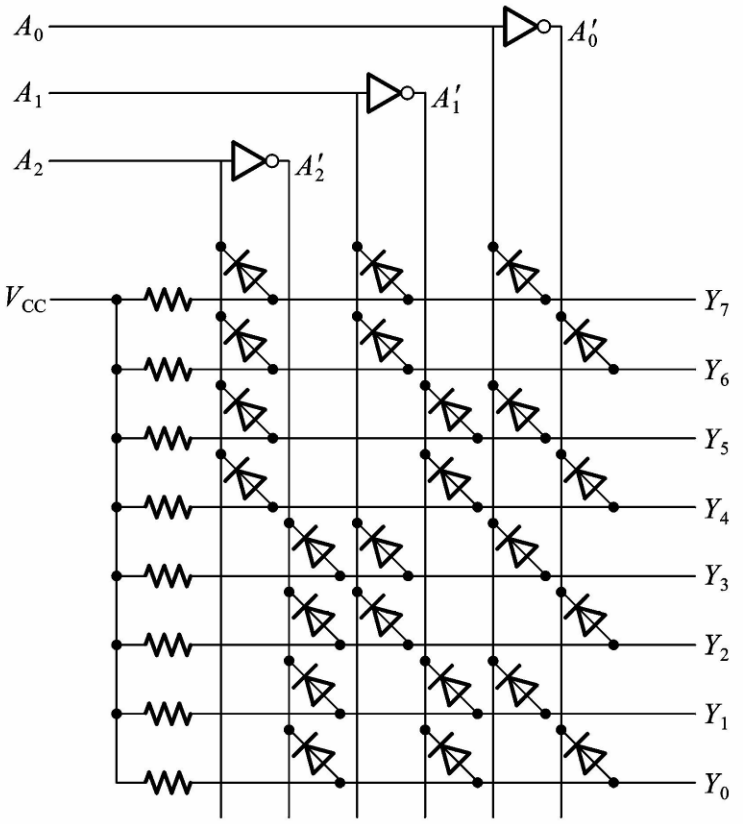


用二极管与门阵列组成的3线－8线译码器

●缺点：

1. 电路的输入电阻较低而输出电阻较高；

2. 输出的高、低电平信号发生偏移（偏离输入信号的高、低电平）

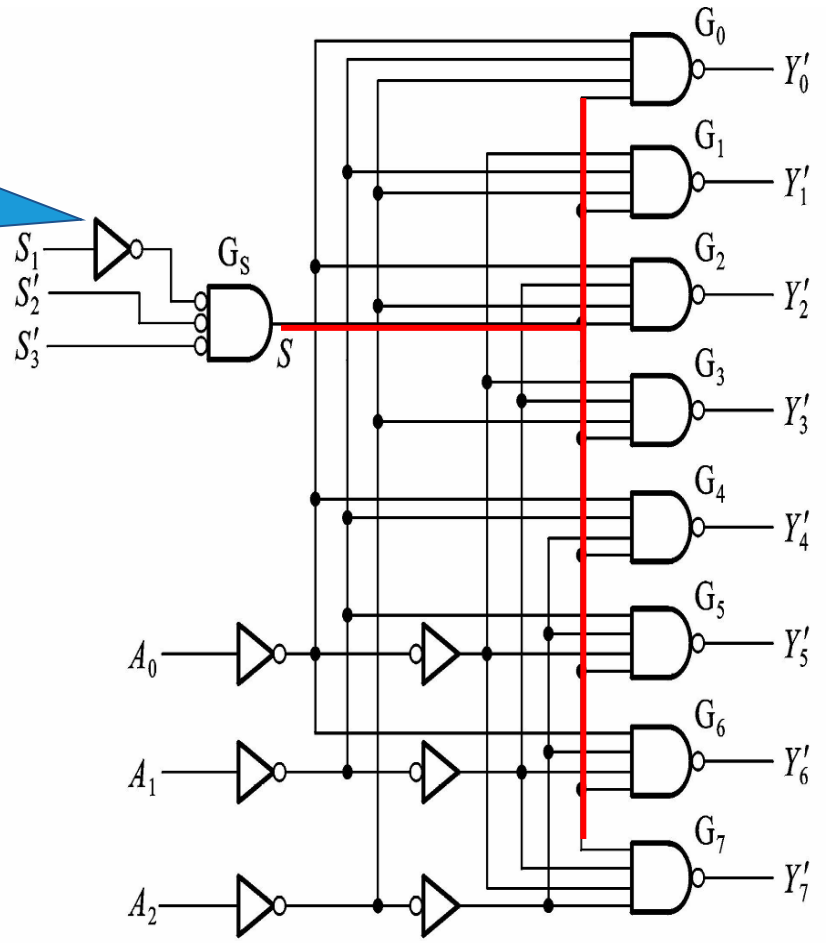


集成译码器实例：74HC138

附加
控制端

$$S = S_3 S_2 S_1$$

$$Y_i' = (S \quad m_i)'$$

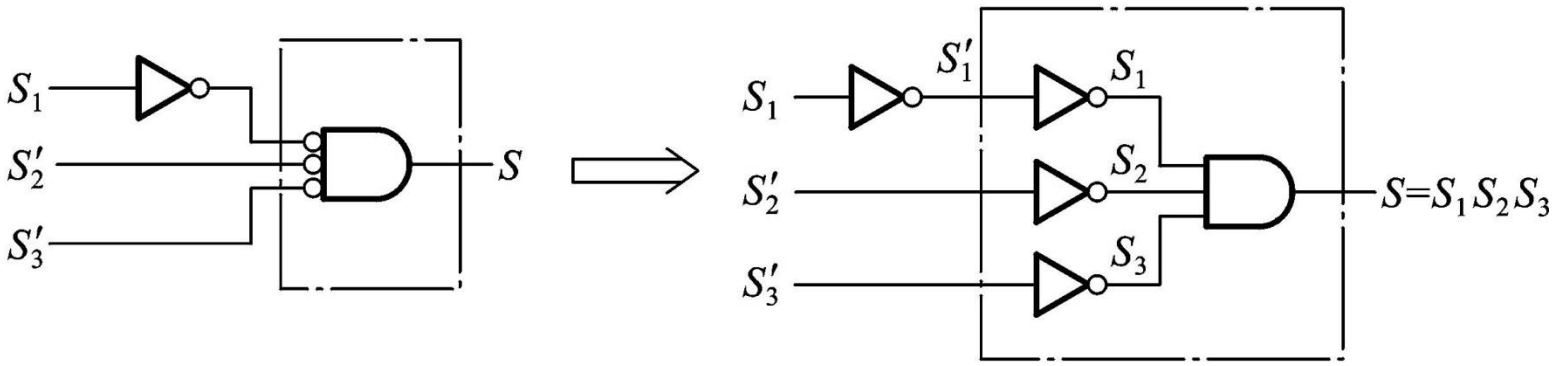


低电平
输出

74HC138的功能表：

输入					输出							
S_1	$S_2'+S_3'$	A_2	A_1	A_0	Y_7'	Y_6'	Y_5'	Y_4'	Y_3'	Y_2'	Y_1'	Y_0'
0	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	1	1	1	1	1	1	0	1	1	1
1	0	1	0	0	1	1	1	0	1	1	1	1
1	0	1	0	1	1	1	0	1	1	1	1	1
1	0	1	1	0	1	0	1	1	1	1	1	1
1	0	1	1	1	0	1	1	1	1	1	1	1

可以理解为地址

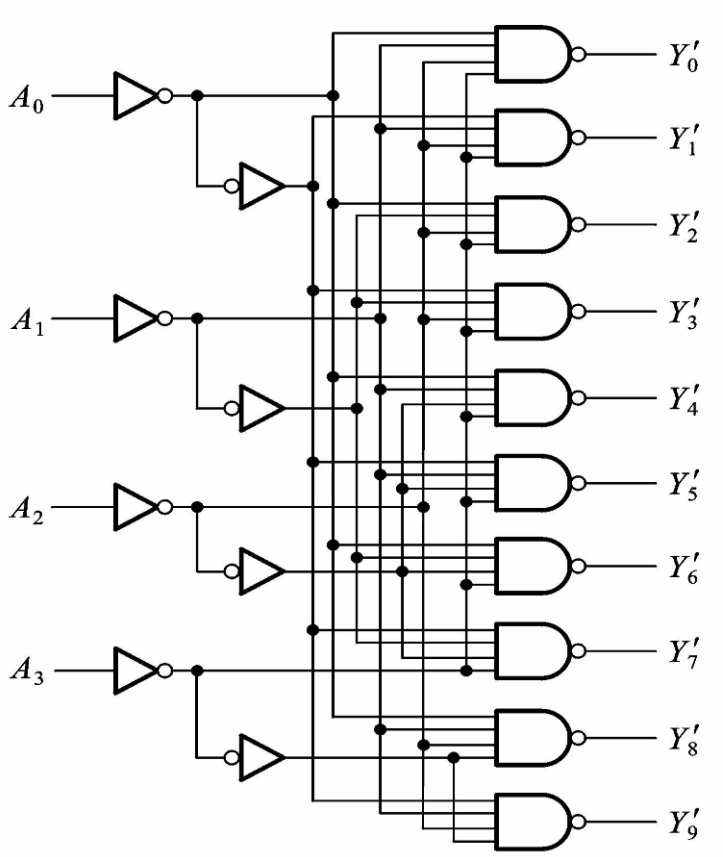

S_1
S_2'
S_3'
S
S_1
S_2'
S_3'
S_1'
S_1
S_2
S_3
$S=S_1S_2S_3$

二、二—十进制译码器

- 将输入BCD码的10个代码译成10个高、低电平的输出信号
  BCD码以外的伪码，输出均无低电平信号产生

- 例：74HC42

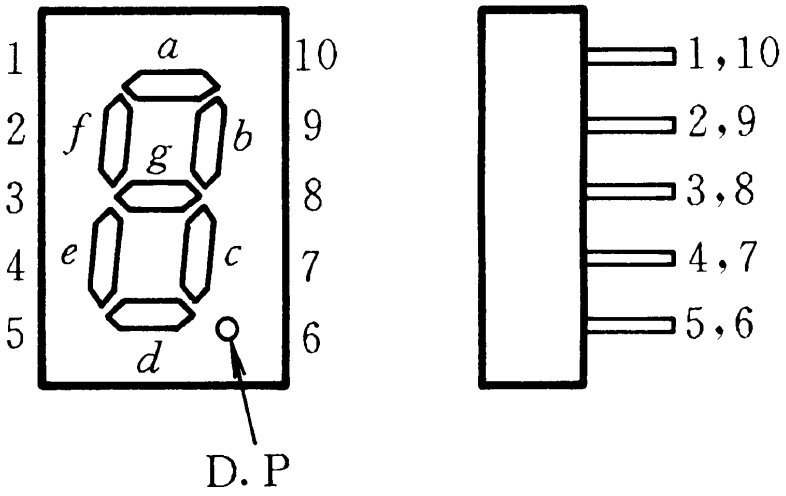
$$Y_i' = m_i' \quad (i = 0 \sim 9)$$



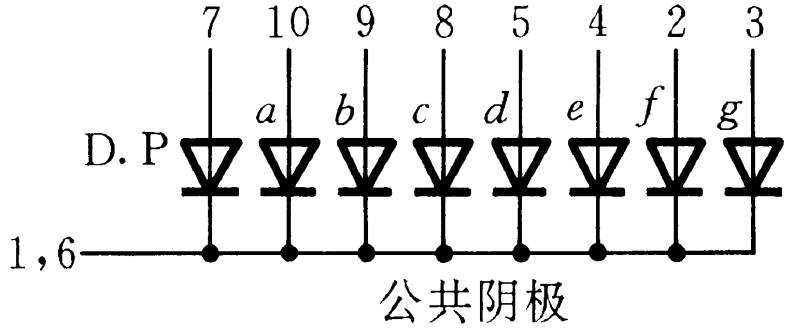
三、显示译码器

❖1. 七段字符显示器

如：半导体数码管BS201A



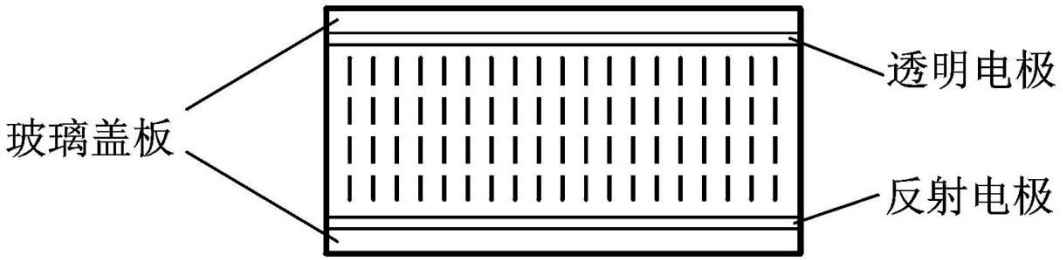
(a)



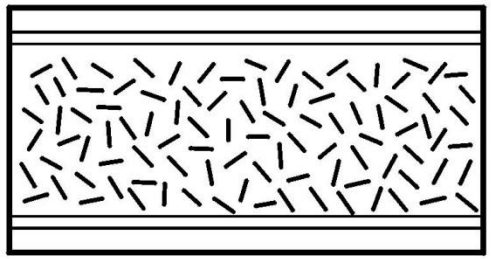
(b)

三、显示译码器

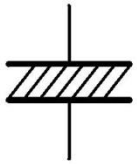
❖如：液晶显示器



(a)



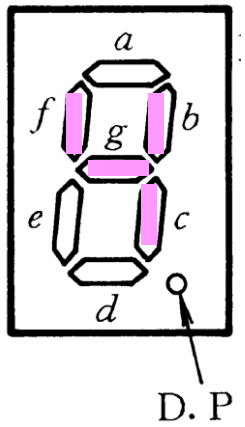
(b)

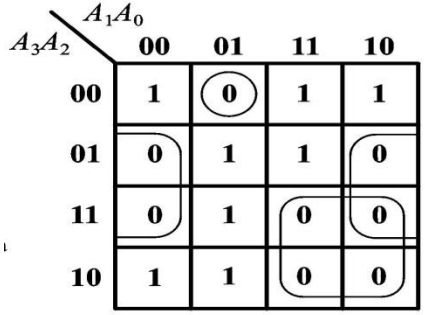


(c)

2. BCD七段字符显示译码器　　（代码转换器）7448

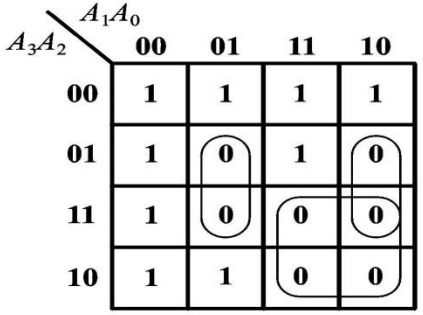
输入					输出							
数字	A_3	A_2	A_1	A_0	Y_a	Y_b	Y_c	Y_d	Y_e	Y_f	Y_g	字形
0	0	0	0	0	1	1	1	1	1	1	0	
1	0	0	0	1	0	1	1	0	0	0	0	
2	0	0	1	0	1	1	0	1	1	0	1	
3	0	0	1	1	1	1	1	1	0	0	1	
4	0	1	0	0	0	1	1	0	0	1	1	
5	0	1	0	1	1	0	1	1	0	1	1	
6	0	1	1	0	0	0	1	1	1	1	1	
7	0	1	1	1	1	1	1	0	0	0	0	
8	1	0	0	0	1	1	1	1	1	1	1	
9	1	0	0	1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





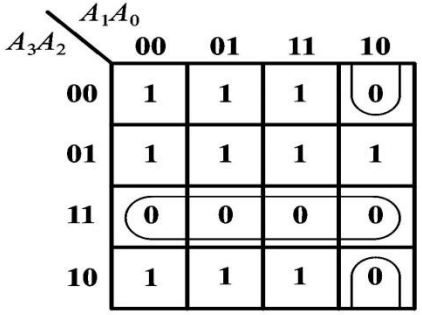
A_3A_2 \ A_1A_0	00	01	11	10
00	1	0	1	1
01	0	1	1	0
11	0	1	0	0
10	1	1	0	0

(a)



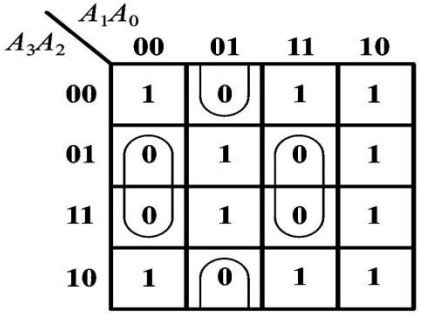
A_3A_2 \ A_1A_0	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	1	0	0	0
10	1	1	0	0

(b)



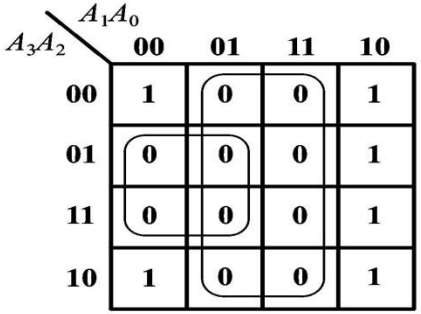
A_3A_2 \ A_1A_0	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	0

(c)



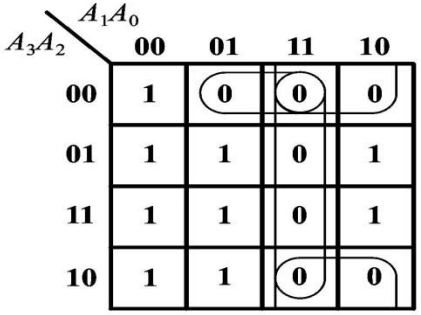
A_3A_2 \ A_1A_0	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	0	1	0	1
10	1	0	1	1

(d)



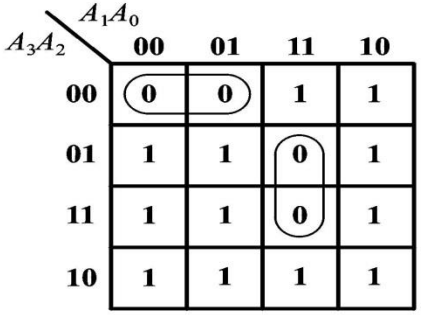
A_3A_2 \ A_1A_0	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	0	0	0	1
10	1	0	0	1

(e)



A_3A_2 \ A_1A_0	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	1	1	0	1
10	1	1	0	0

(f)



A_3A_2 \ A_1A_0	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	1	1	0	1
10	1	1	1	1

(g)

BCD—七段显示译码器7448的逻辑图

$$Y_a = (A_3'A_2'A_1'A_0 + A_3A_1 + A_2A_0')'$$

$$Y_b = (A_3A_1 + A_2A_1A_0' + A_2A_1'A_0)'$$

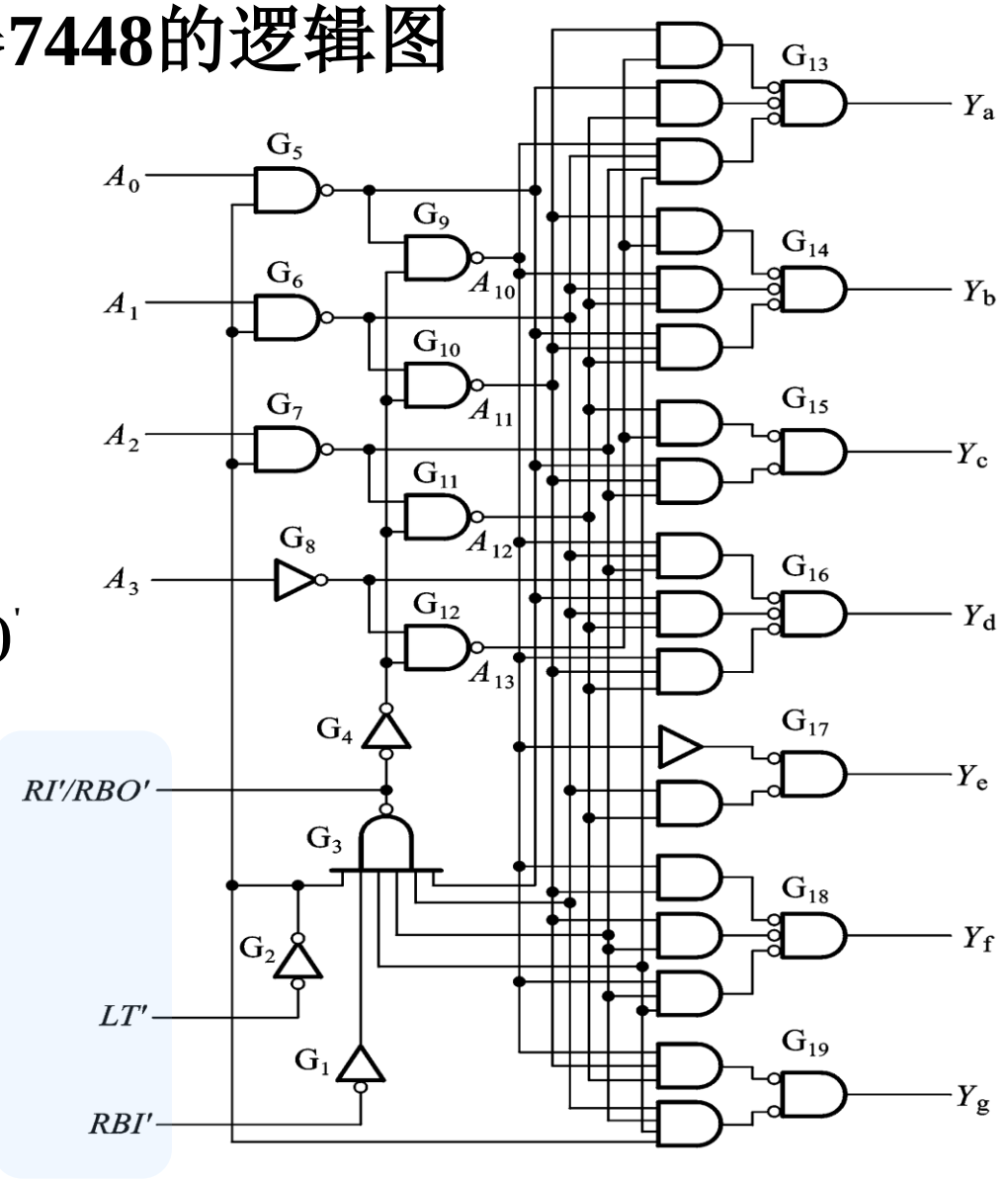
$$Y_c = (A_3A_2 + A_2'A_1A_0')'$$

$$Y_d = (A_2A_1A_0 + A_2A_1'A_0' + A_2'A_1'A_0)'$$

$$Y_e = (A_2A_1' + A_0)'$$

$$Y_f = (A_3'A_2'A_0 + A_2'A_1 + A_1A_0)'$$

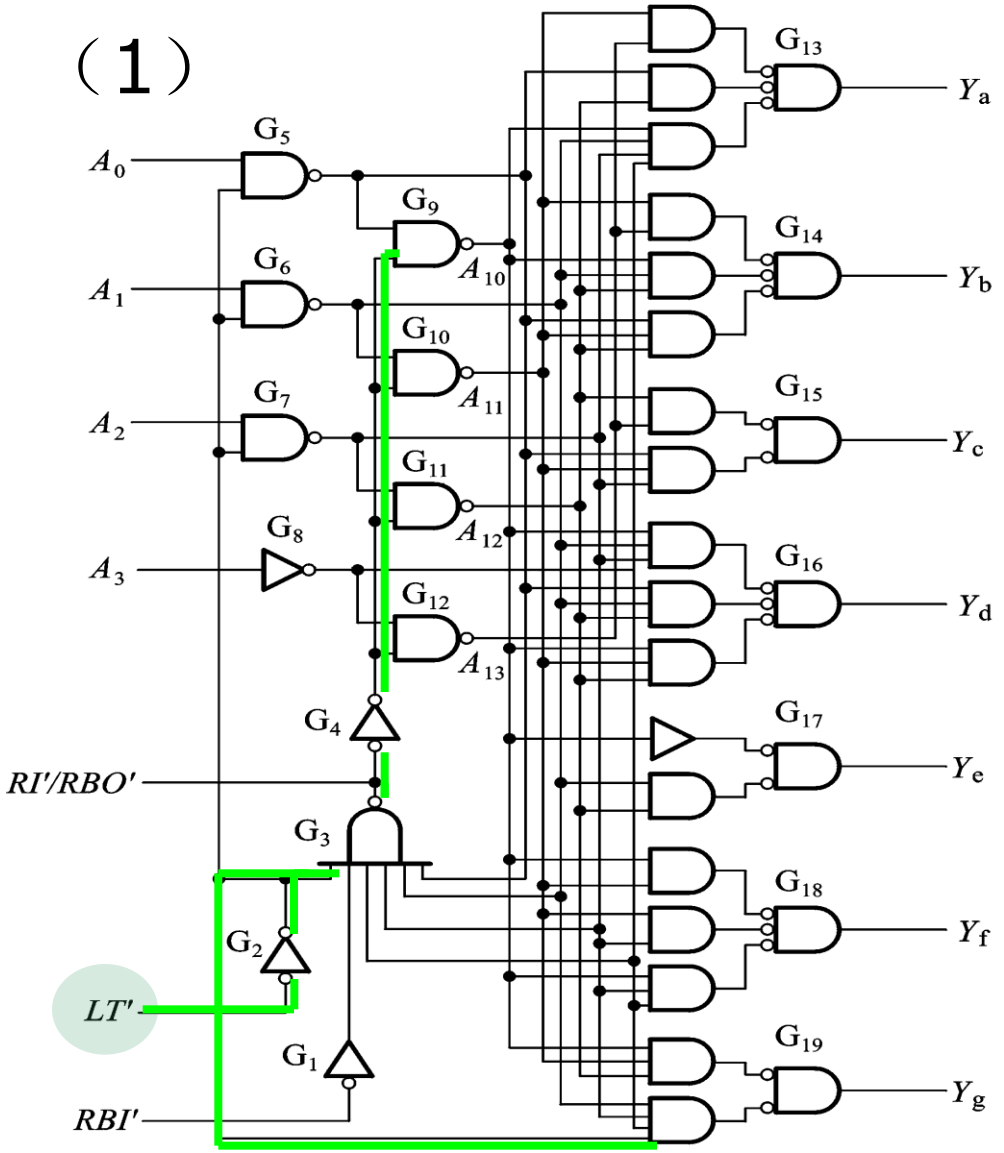
$$Y_g = (A_3'A_2'A_1' + A_2A_1A_0)'$$



7448的附加控制信号：（1）

- 灯测试输入 LT'

当$LT'=0$ 时，Ya～Yg全部置为1

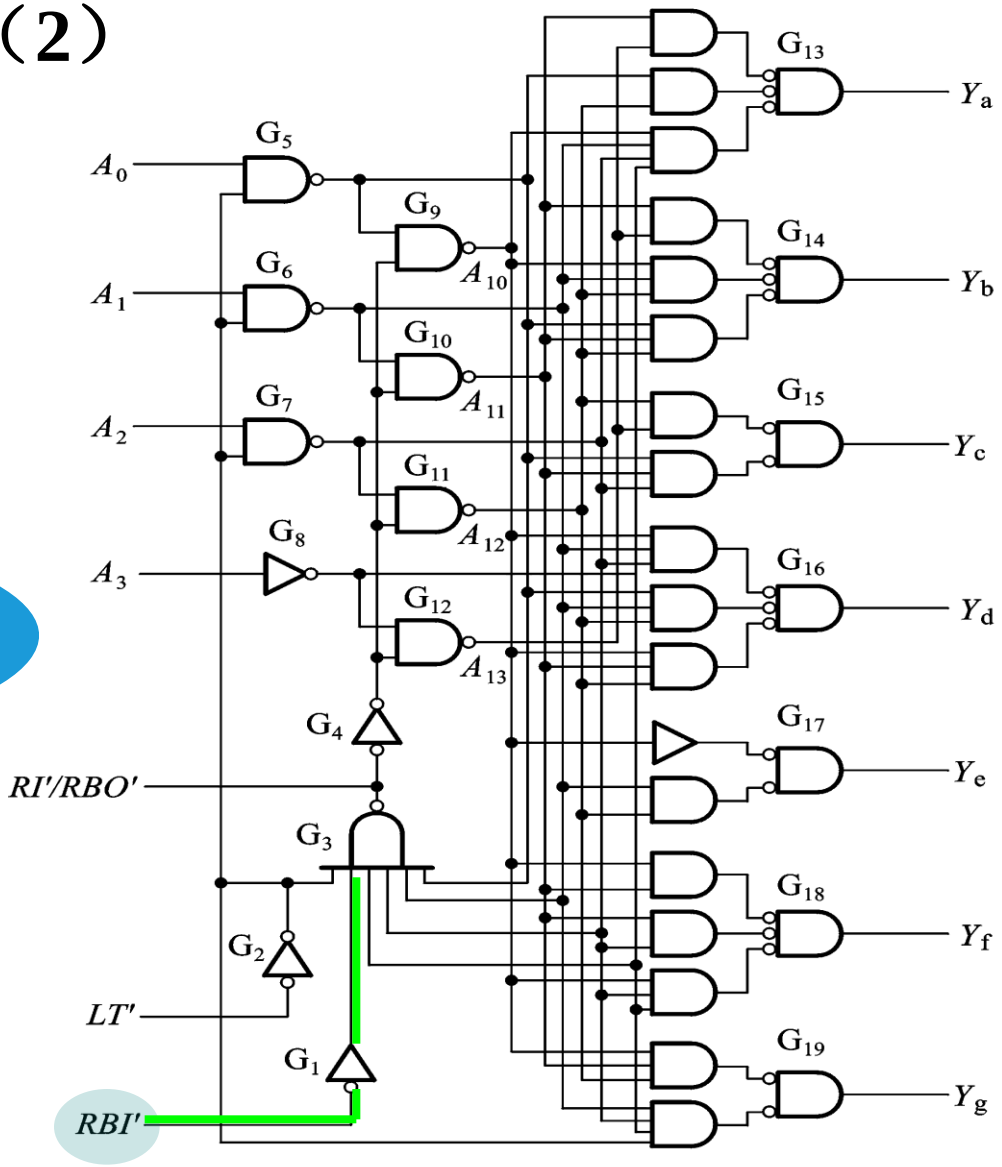


7448的附加控制信号：（2）

- 灭零输入 RBI'

把不希望显示的零熄灭

当 $A_3A_2A_1A_0=0000$ 时，
$RBI'=0$ 时，则灭灯



7448的附加控制信号：（3）

❖ 灭灯输入/灭零输出 $\overline{BI}/\overline{RBO}$

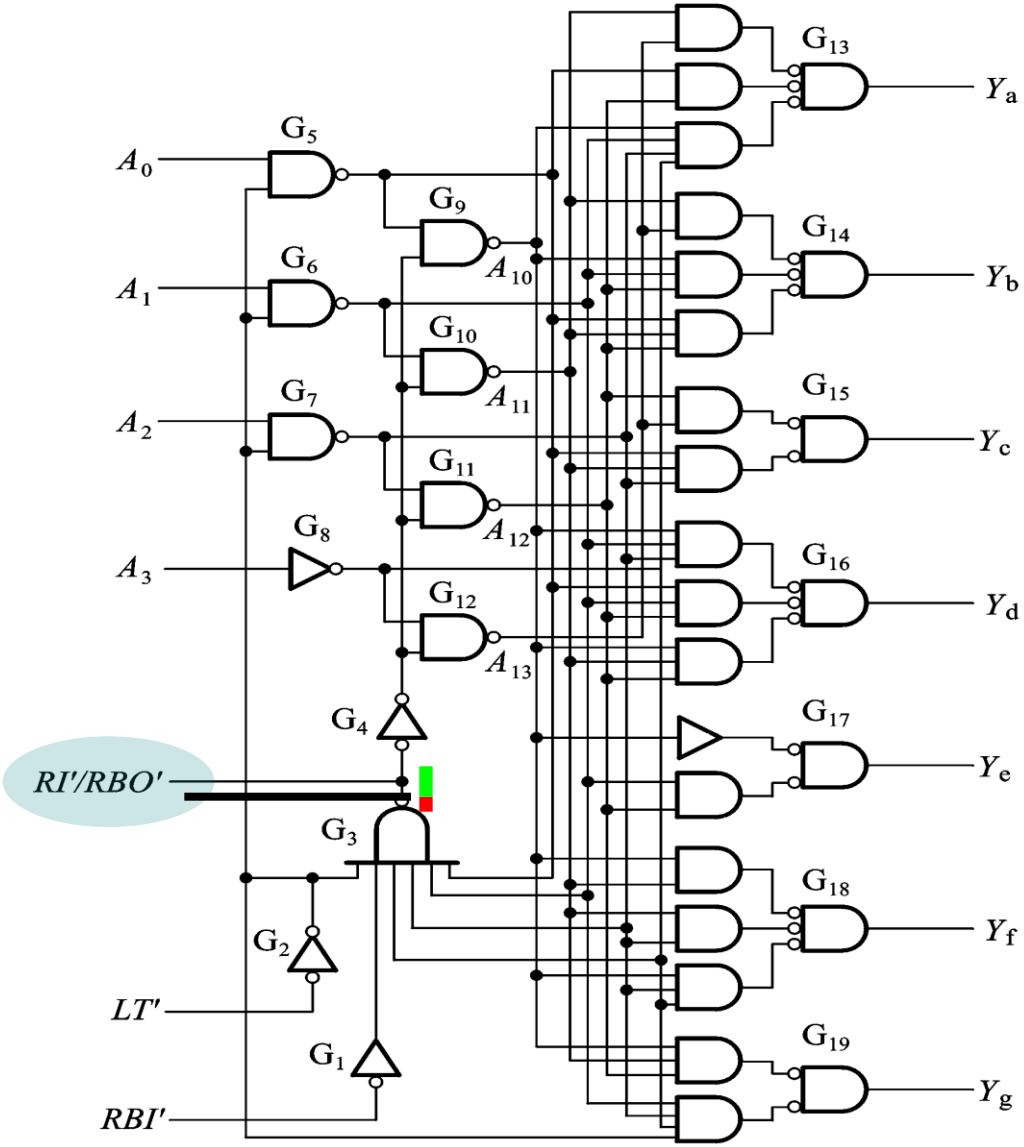
输入信号，称灭灯输入控制端：

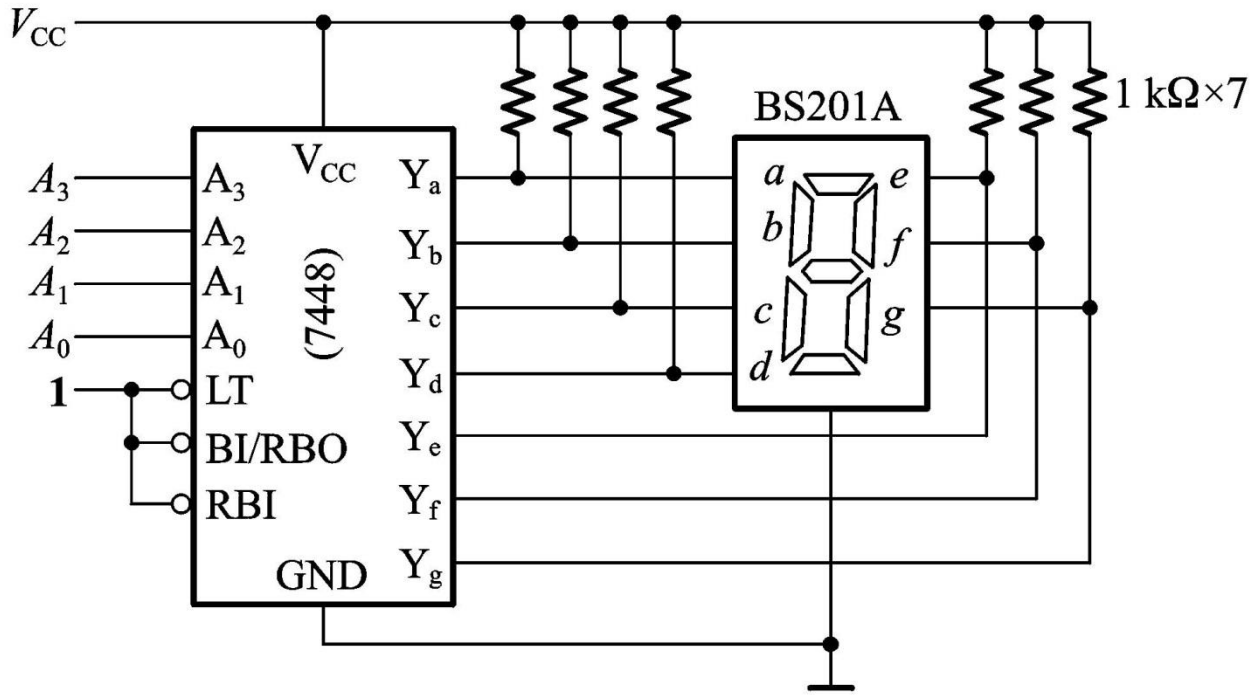
$\overline{BI}=0$ 无论输入状态是什么，数码管熄灭

输出信号，称灭零输出端：

只有当输入 $A_3A_2A_1A_0=0$，且灭零输入信号 $\overline{RBI}=0$ 时，$\overline{RBO}$才给出低电平

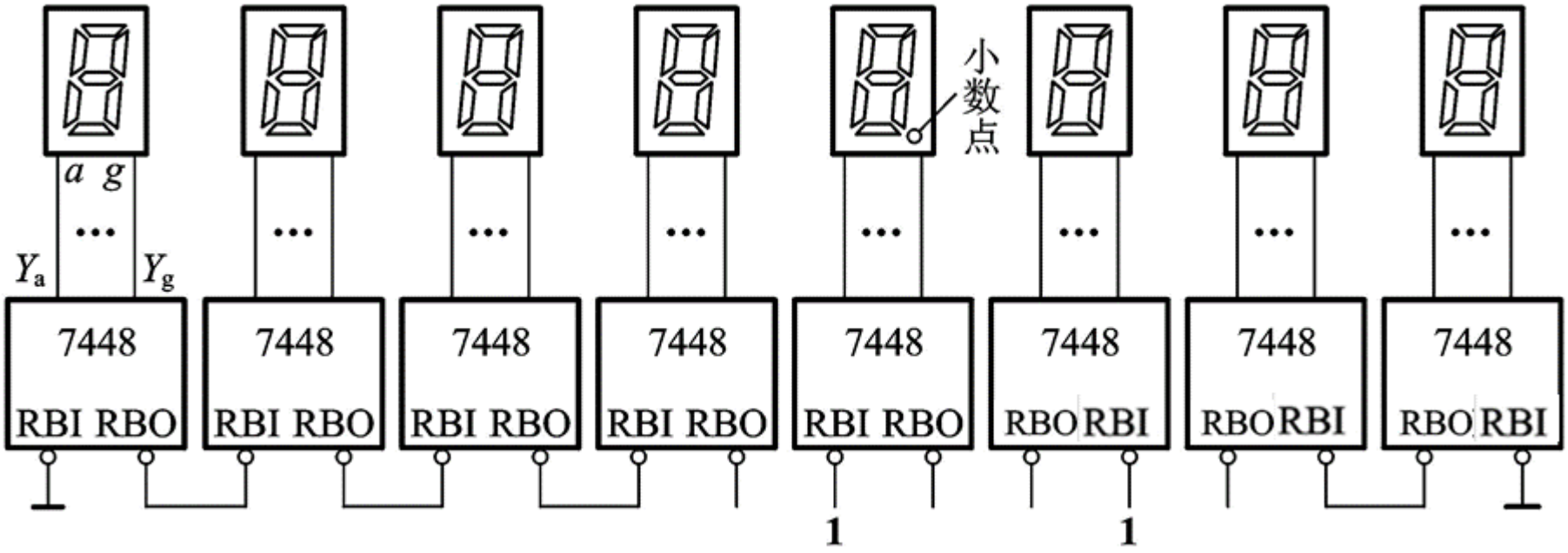
因此 $\overline{RBO}=0$ 表示译码器将本来应该显示的零熄灭了


A_0
A_1
A_2
A_3
RI'/RBO'
LT'
RBI'
G_1
G_2
G_3
G_4
G_5
G_6
G_7
G_8
G_9
G_{10}
G_{11}
G_{12}
G_{13}
G_{14}
G_{15}
G_{16}
G_{17}
G_{18}
G_{19}
A_{10}
A_{11}
A_{12}
A_{13}
Y_a
Y_b
Y_c
Y_d
Y_e
Y_f
Y_g


V_{CC}
A_3
A_2
A_1
A_0
1
LT
BI/RBO
RBI
GND
(7448)
Y_a
Y_b
Y_c
Y_d
Y_e
Y_f
Y_g
BS201A
a
b
c
d
e
f
g
1 kΩ×7

例：利用 $\overline{RBI}$ 和 $\overline{RBO}$ 的配合，实现多位显示系统的灭零控制

- 整数部分：最高位是0，而且灭掉以后，输出 $\overline{RBO}$ 作为次高位的 $\overline{RBI}$ 输入信号
- 小数部分：最低位是0，而且灭掉以后，输出 $\overline{RBO}$ 作为次低位的 $\overline{RBI}$ 输入信号

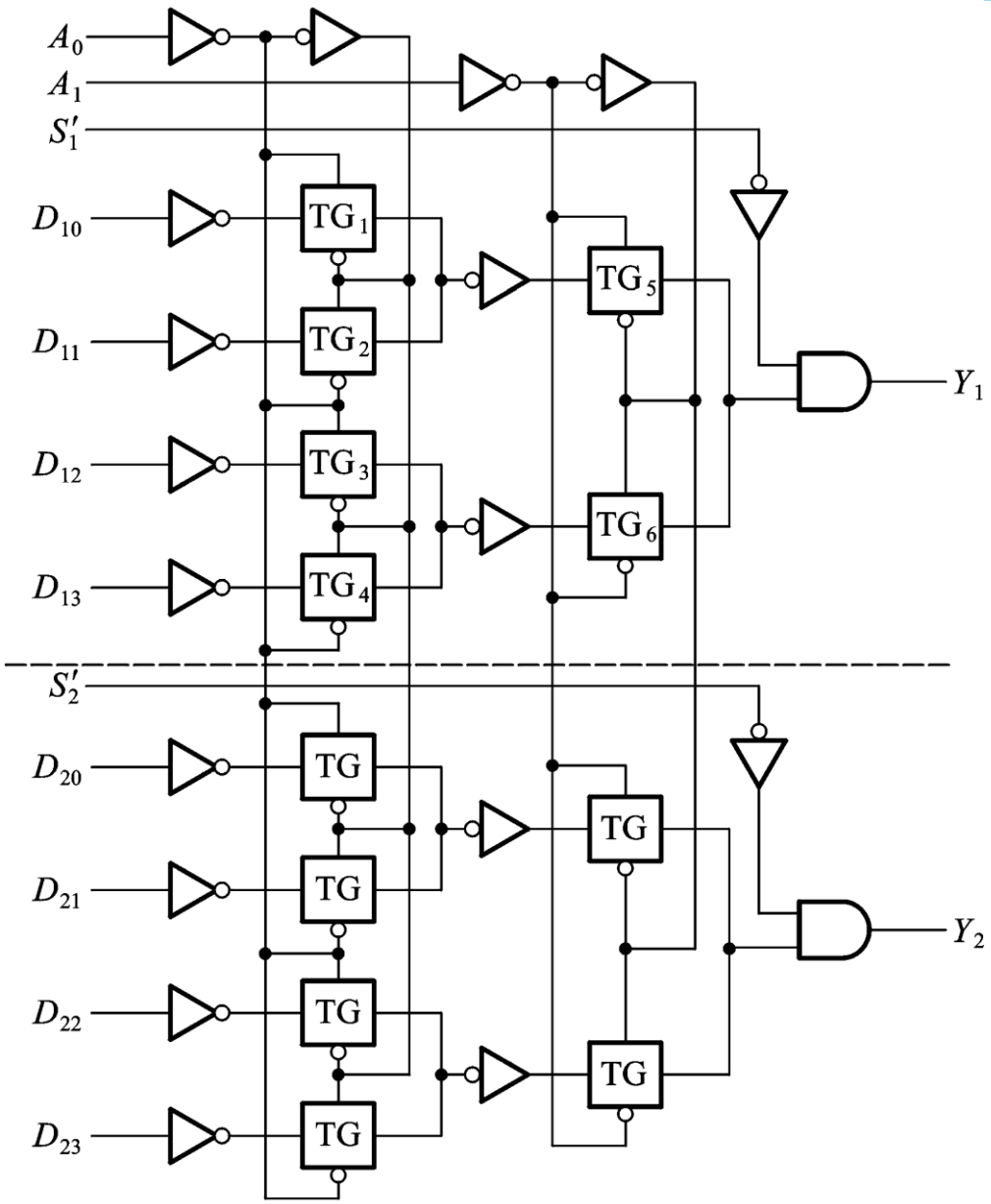


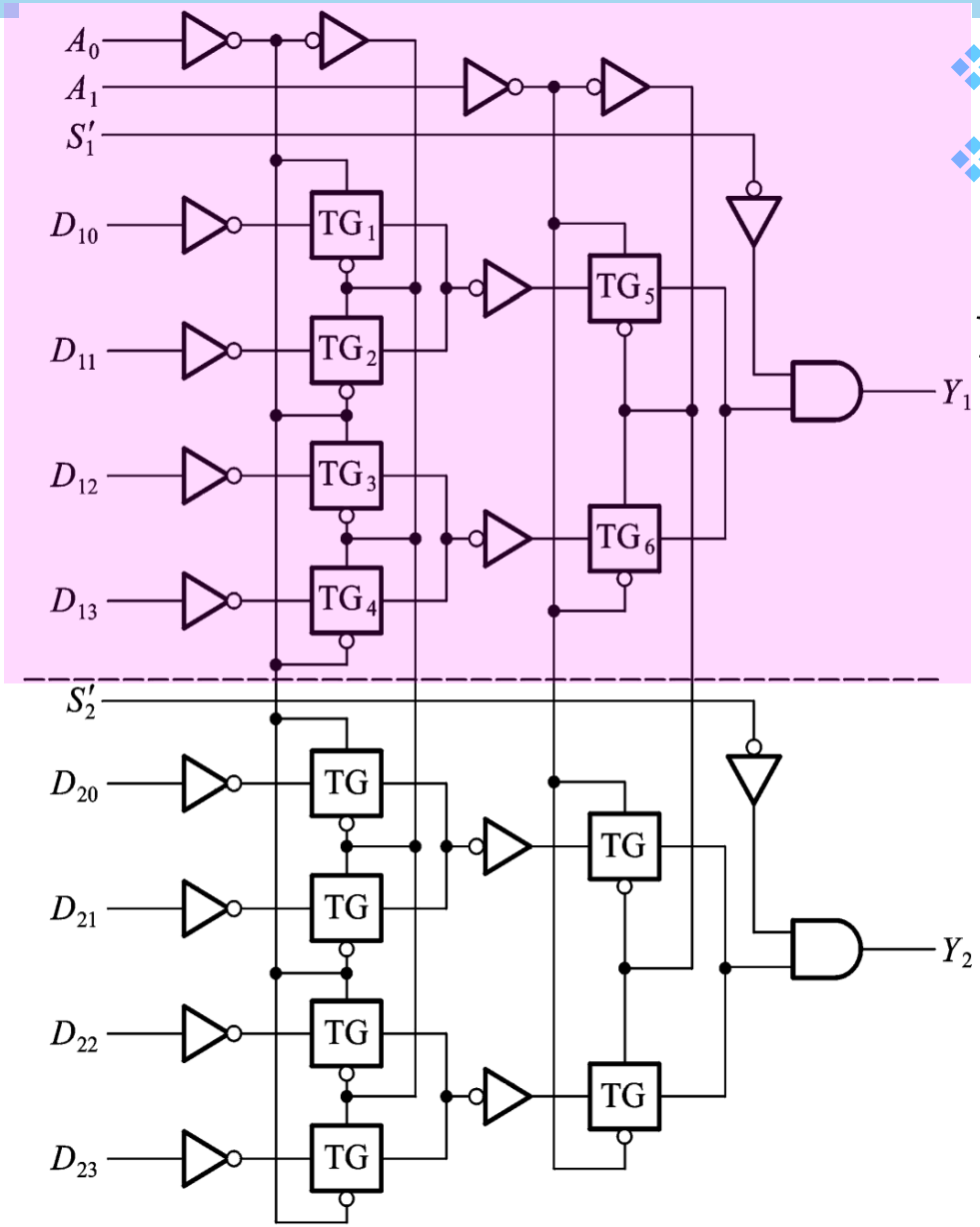
4.3.3 数据选择器（多路开关）

一、工作原理

（从一组输入数据中选出某一个来）

- 例：“双四选一”，74HC153，分析其中的一个“四选一”





- 例：“双四选一”，74HC153
- 分析其中的一个“四选一”

$$Y_1 = S_1[D_0(A_1'A_0') + D_1(A_1'A_0) + D_2(A_1A_0') + D_3(A_1A_0)]$$

S_1'	A_1	A_0	Y_1
1	X	X	0
0	0	0	D_{10}
0	0	1	D_{11}
0	1	0	D_{12}
0	1	1	D_{13}

4.3.4 加法器

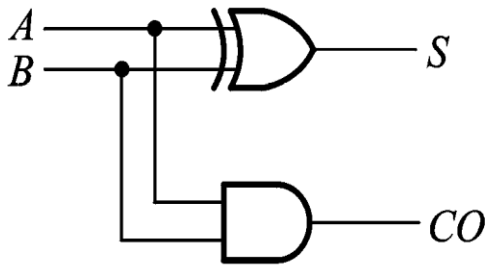
一、1位加法器

1. 半加器，不考虑来自低位的进位，将两个1位的二进制数相加

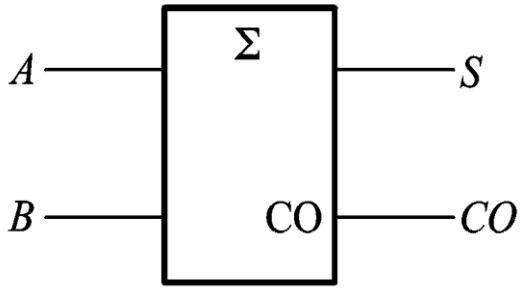
输 入		输 出	
A	B	S	CO
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = A \oplus B$$

$$CO = AB$$



(a)



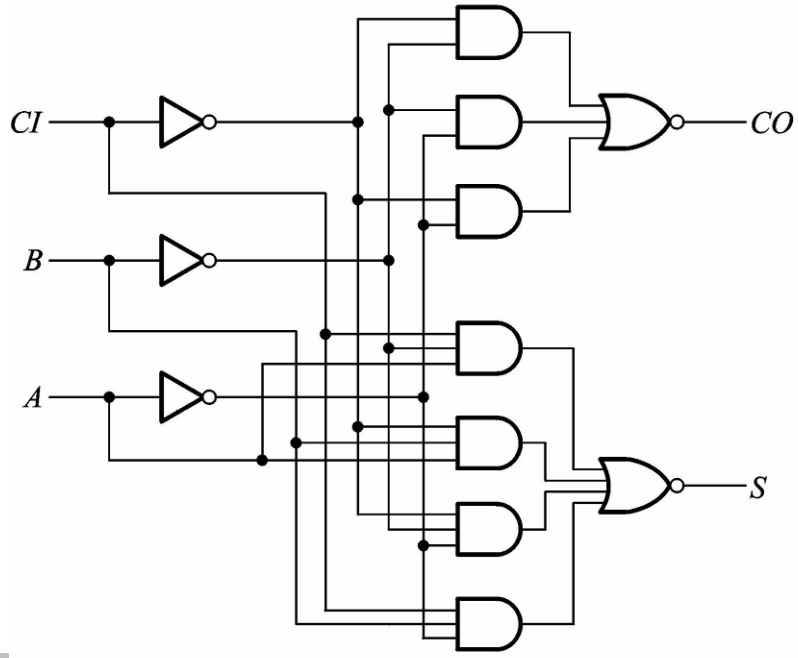
(b)

2. 全加器：将两个1位二进制数及来自低位的进位相加

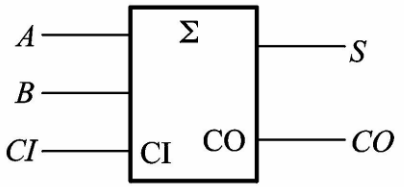
输 入			输 出	
A	B	CI	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = (A'B'CI' + A'B \cdot CI + AB'CI + ABCI')'$$

$$CO = (A'B' + B'CI' + A'CI')'$$



(a)



(b)

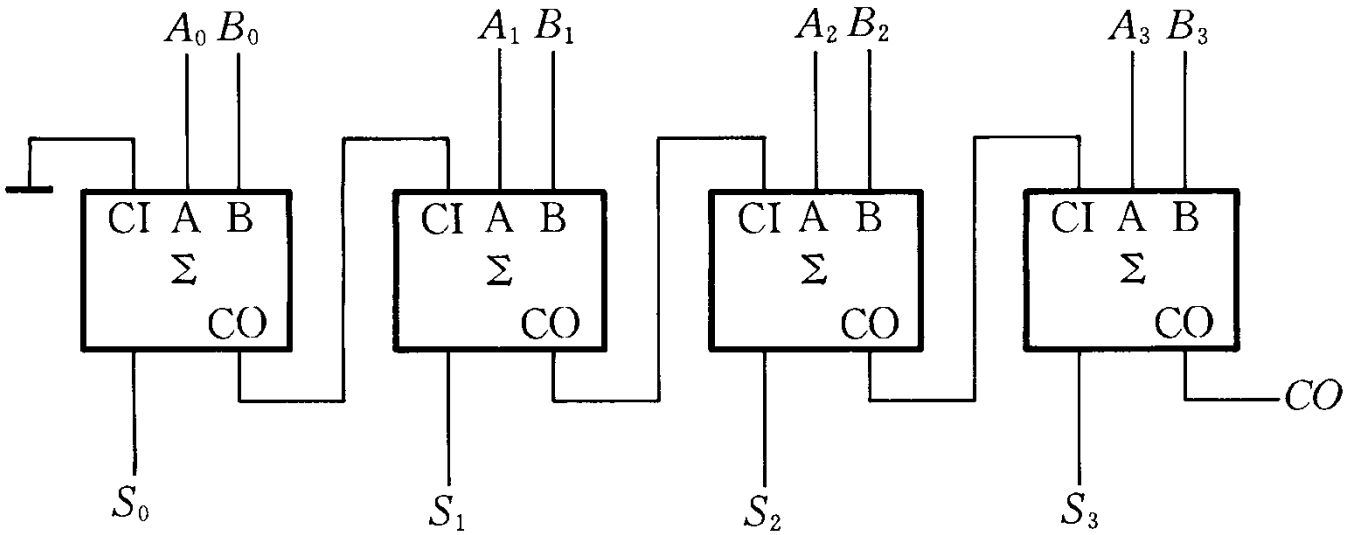
74LS183

二、多位加法器

1. 串行进位加法器

优点：简单

缺点：慢



$$(CI)_i = (CO)_{i-1}$$

$$S_i = A_i \oplus B_i \oplus (CI)_i$$

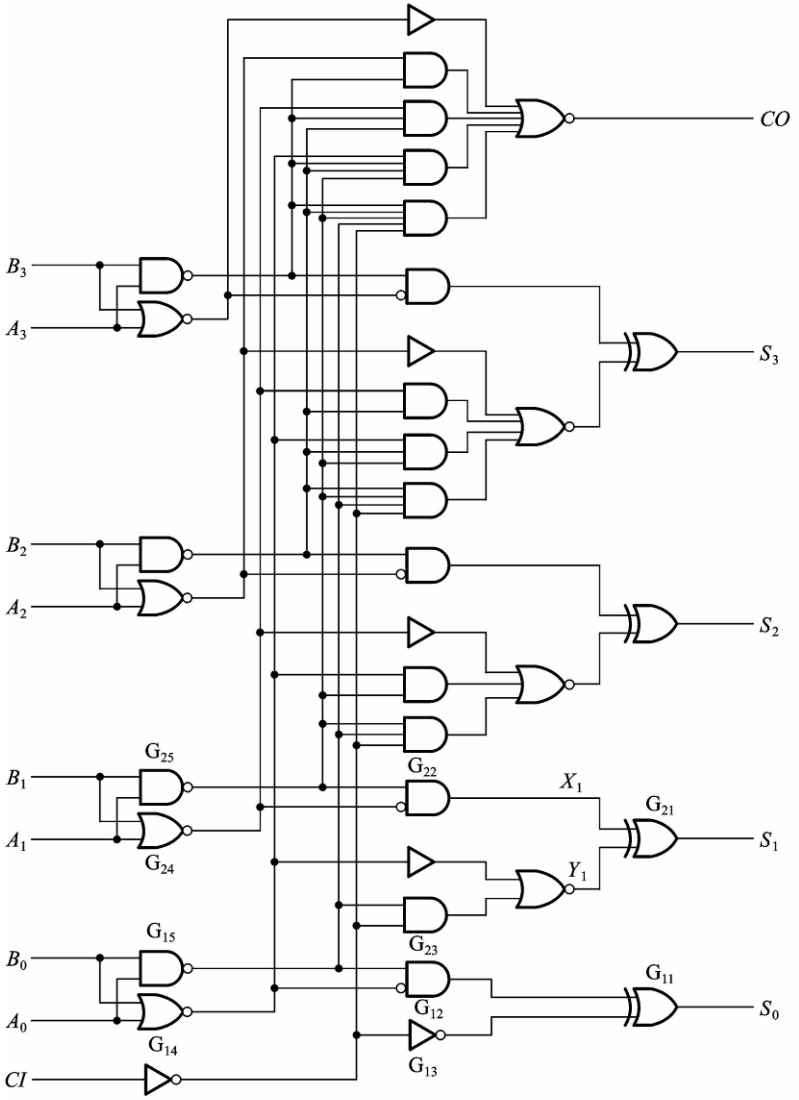
$$(CO)_i = A_i B_i + (A_i + B_i)(CI)_i$$

2. 超前进位加法器

基本原理：加到第i位的进位输入信号是两个加数第*i*位以前各位（0～*j*-1）的函数，可在相加前由A,B两数确定。

优点：快，每1位的和及最后的进位基本同时产生。

缺点：电路复杂。



74LS283

$$i=0:(CI)_0=0$$

$$S_0=A_0\oplus B_0\oplus(CI)_0$$

$$(CO)_0=A_0B_0+(A_0+B_0)(CI)_0$$

$$i=1:\ \ (CI)_1=(CO)_0$$

$$S_1=A_1\oplus B_1\oplus(CO)_0$$

$$=A_1\oplus B_1\oplus(A_0B_0+(A_0+B_0)(CI)_0)$$

$$(CO)_1=A_1B_1+(A_1+B_1)(CO)_0$$

$$=A_1B_1+(A_1+B_1)(A_0B_0+(A_0+B_0)(CI)_0)$$

$$i=2:\ \ (CI)_2=(CO)_1$$

$$=A_1B_1+(A_1+B_1)(A_0B_0+(A_0+B_0)(CI)_0)$$

$$(CO)_2=A_2B_2+(A_2+B_2)(CI)_2$$

$$=A_2B_2+(A_2+B_2)(A_1B_1+(A_1+B_1)(A_0B_0+(A_0+B_0)(CI)_0))$$

$$S_2=A_2\oplus B_2\oplus(CI)_2$$

$$=A_2\oplus B_2\oplus(A_1B_1+(A_1+B_1)(A_0B_0+(A_0+B_0)(CI)_0))$$

$$\vdots$$

进位输出

$$(CO)_i = A_i B_i + (A_i + B_i)(CI)_i$$

$$(CO)_i = G_i + P_i(CI)_i$$

$$(CO)_i = G_i + P_i G_{i-1} + \ldots + P_i P_{i-1} \cdots P_1 G_0 + P_i P_{i-1} \cdots P_0 (CI)_0$$

相加的和

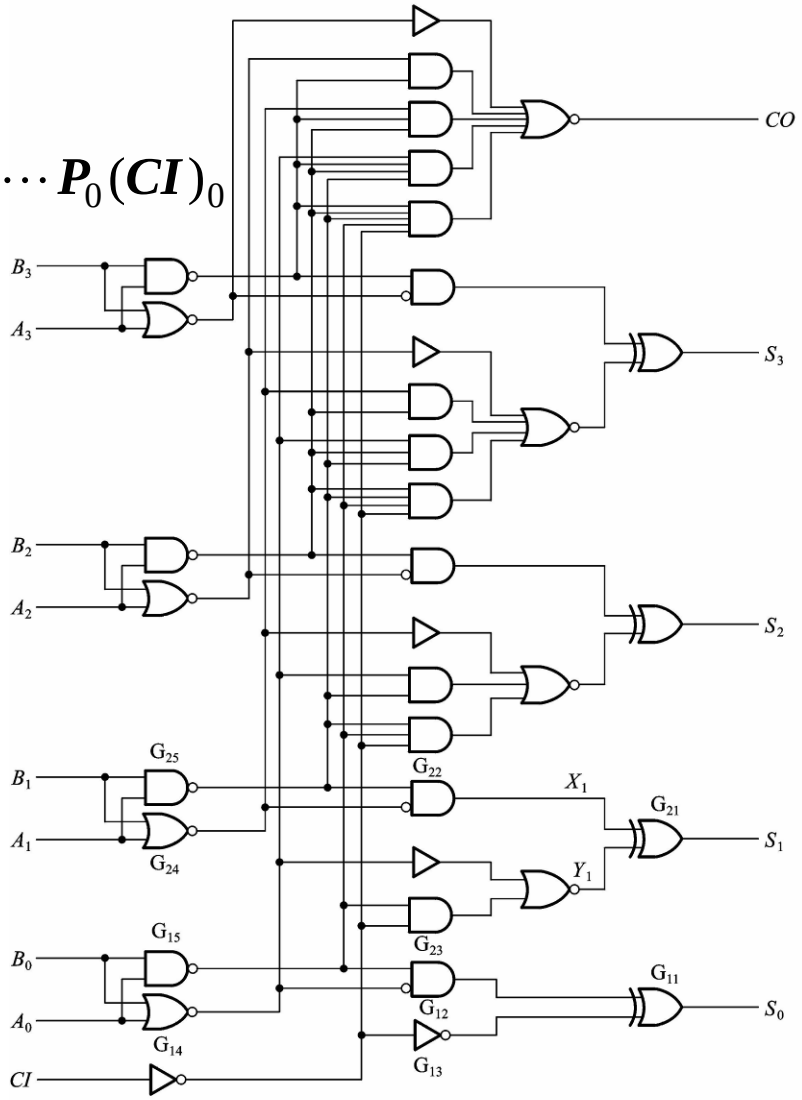
$$S_i = A_i \oplus B_i \oplus (CI)_i$$

举例（4位超前进位加法器4LS283）

$$X_1 = A_1 \oplus B_1$$

$$Y_1 = A_0 B_0 + (A_0 + B_0)(CI)_0 = (CO)_0 = (CI)_1$$

$$S_1 = X_1 \oplus Y_1 = A_1 \oplus B_1 \oplus (CI)_1$$



三、用加法器设计组合电路

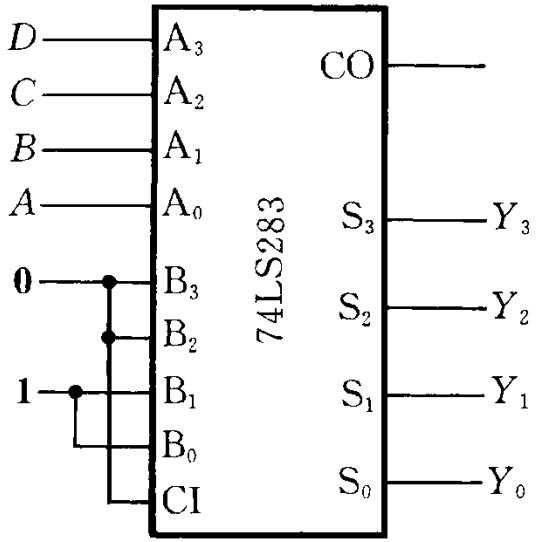
❖ 基本原理：

若能将函数变换成输入变量与输入变量相加

或能将函数变换成输入变量与常量相加

例：将BCD的8421码转换为余3码

$$Y_3Y_2Y_1Y_0 = DCBA + 0011$$



输 入				输 出			
D	C	B	A	Y3	Y2	Y1	Y0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

4.3.5 数值比较器

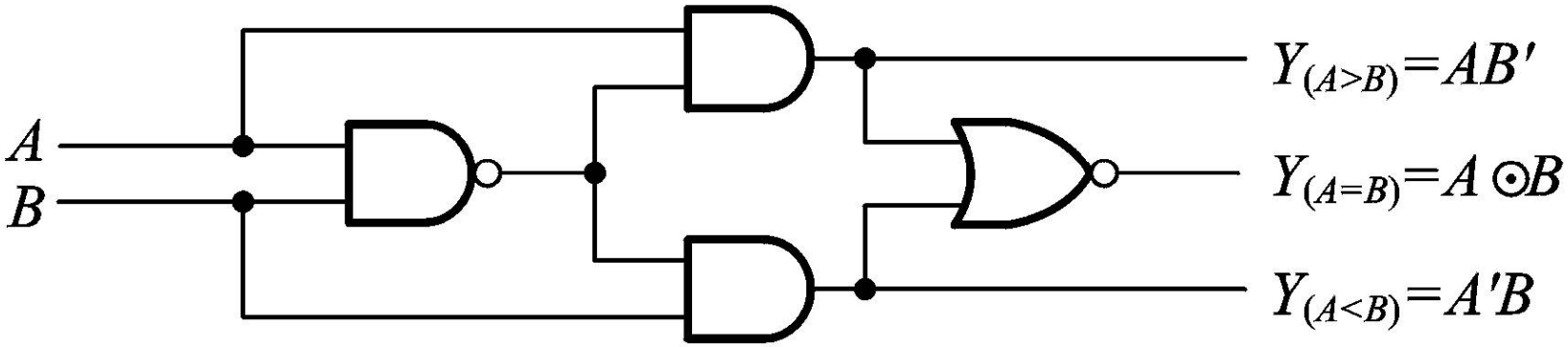
❖ 用来比较两个二进制数的数值大小

一、1位数值比较器　　A,B比较有三种可能结果

* $A > B(A=1, B=0)$则$AB' = 1, \therefore Y_{(A>B)} = AB'$

* $A < B(A=0, B=1)$则$A'B = 1, \therefore Y_{(A<B)} = A'B$

* $A = B(A, B$同为0或1$), \therefore Y_{(A=B)} = (A \oplus B)'$



二、多位数值比较器

1. 原理：从高位比起，只有高位相等，才比较下一位。

例如：

比较$A_3A_2A_1A_0$和$B_3B_2B_1B_0$

$$Y_{(A<B)} = A_3'B_3 + (A_3 \oplus B_3)' A_2'B_2 + (A_3 \oplus B_3)'(A_2 \oplus B_2)' A_1'B_1$$

$$+ (A_3 \oplus B_3)'(A_2 \oplus B_2)'(A_1 \oplus B_1)' A_0'B_0 +$$

$$(A_3 \oplus B_3)'(A_2 \oplus B_2)'(A_1 \oplus B_1)'(A_0 \oplus B_0)' I_{(A<B)}$$

$$Y_{(A=B)} = (A_3 \oplus B_3)'(A_2 \oplus B_2)'(A_1 \oplus B_1)'(A_0 \oplus B_0)' I_{(A=B)}$$

$$Y_{(A>B)} = (Y_{(A<B)} + Y_{(A=B)})' I_{(A>B)}$$

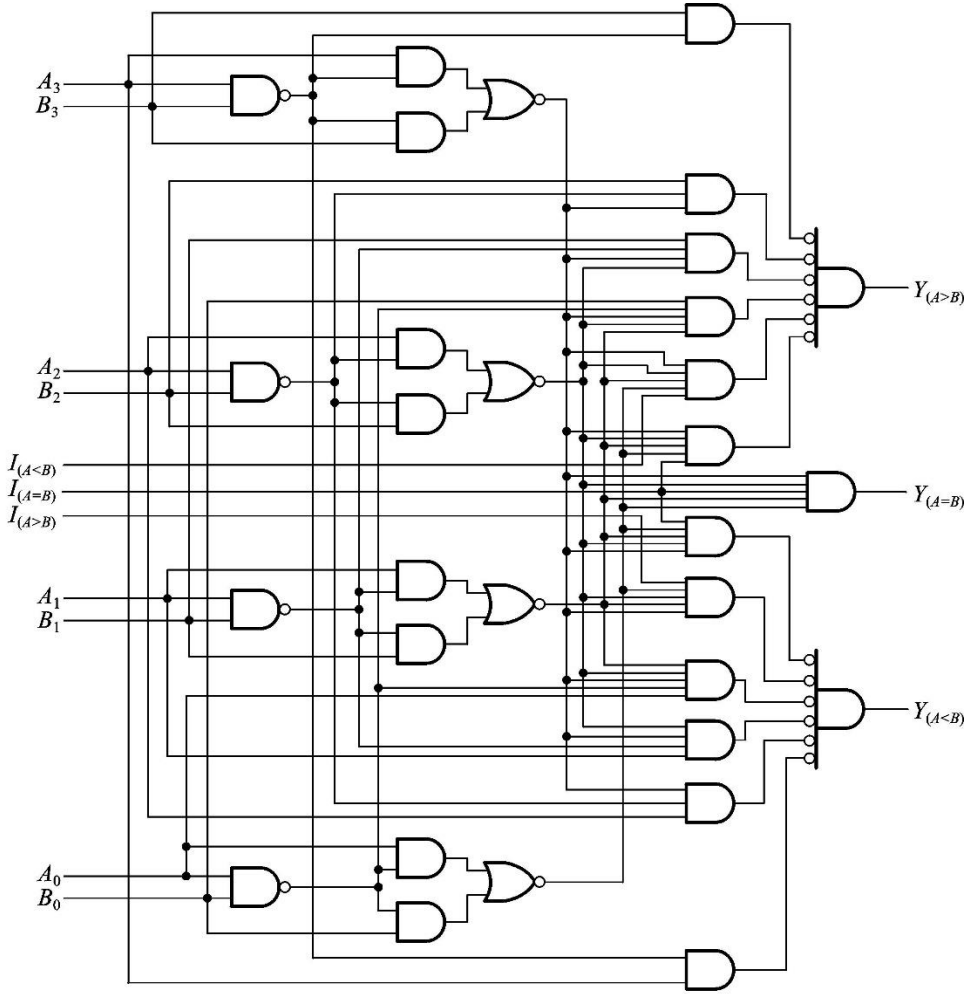
2. 集成电路74LS85 实现4位二进制数的比较

$I_{(A<B)}$,$I_{(A=B)}$和$I_{(A>B)}$为附加端，用于扩展

$I_{(A<B)}$,来自低位的比较结果

$I_{(A=B)}$,来自低位的比较结果

$I_{(A>B)}$,$A>B$输出允许信号



4-3-32

4.5 层次化和模块化的设计方法

1、什么是“层次化”设计方法

（1）“自顶向下”：对整个设计任务进行分层和分块的划分，降低每层的复杂度，简化每个模块的功能；

（2）“自底向上”：对每一个有限复杂度的模块进行实现或调用。

2、什么是“模块化”设计方法

将经过设计和验证的能完成一定功能的逻辑电路封装成为模块，在后续的设计中都可反复使用。

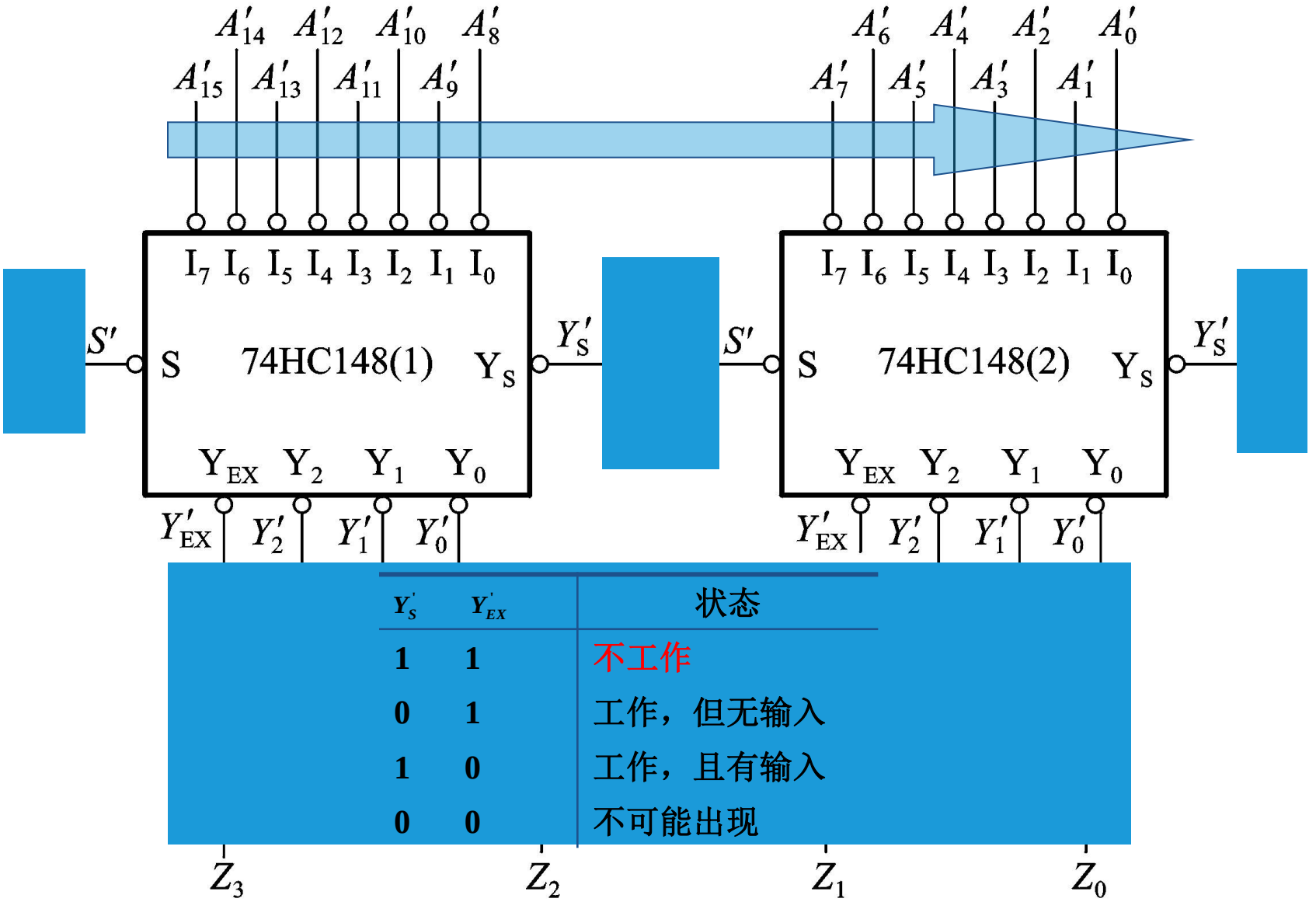
这两种方法核心是首先将电路逐级分解为若干个简单的模块，然后再将这些模块设计好并连接起来。

利用控制端扩展功能举例：

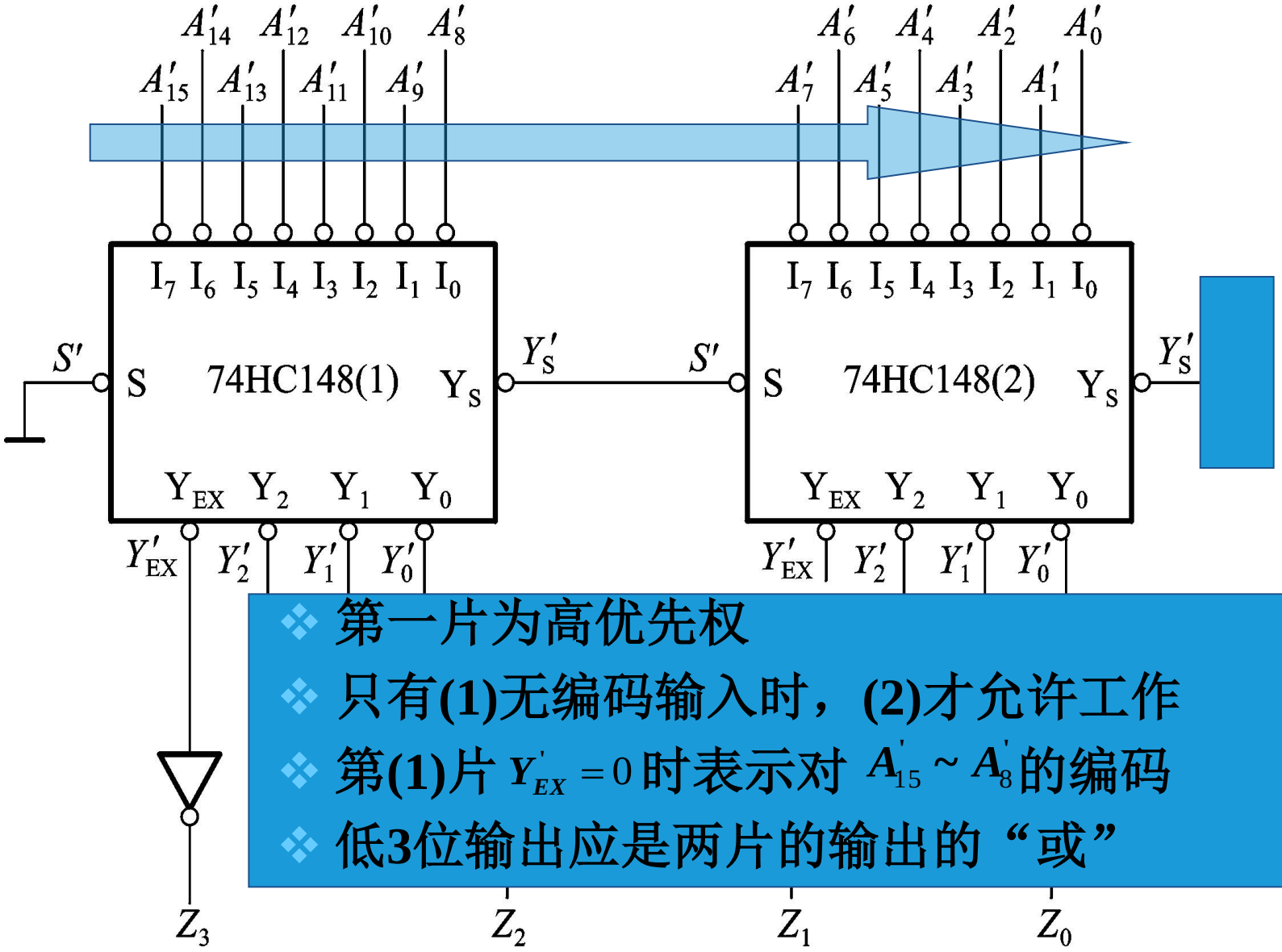
- 例1：用两片8线-3线优先编码器

16线-4线优先编码器

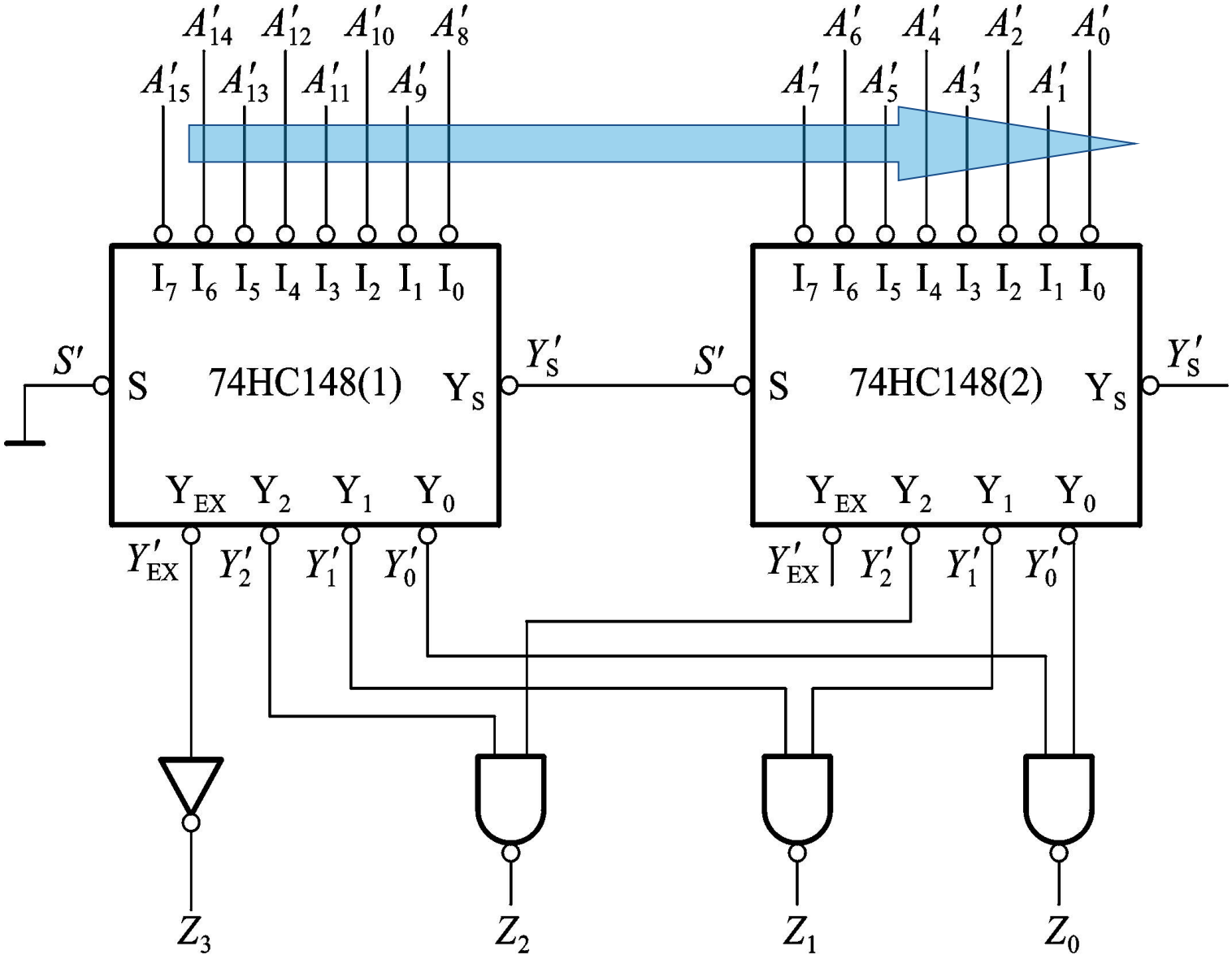
其中，A'_{15} 的优先权最高···



Y'_S	Y'_{EX}	状态
1	1	不工作
0	1	工作，但无输入
1	0	工作，且有输入
0	0	不可能出现

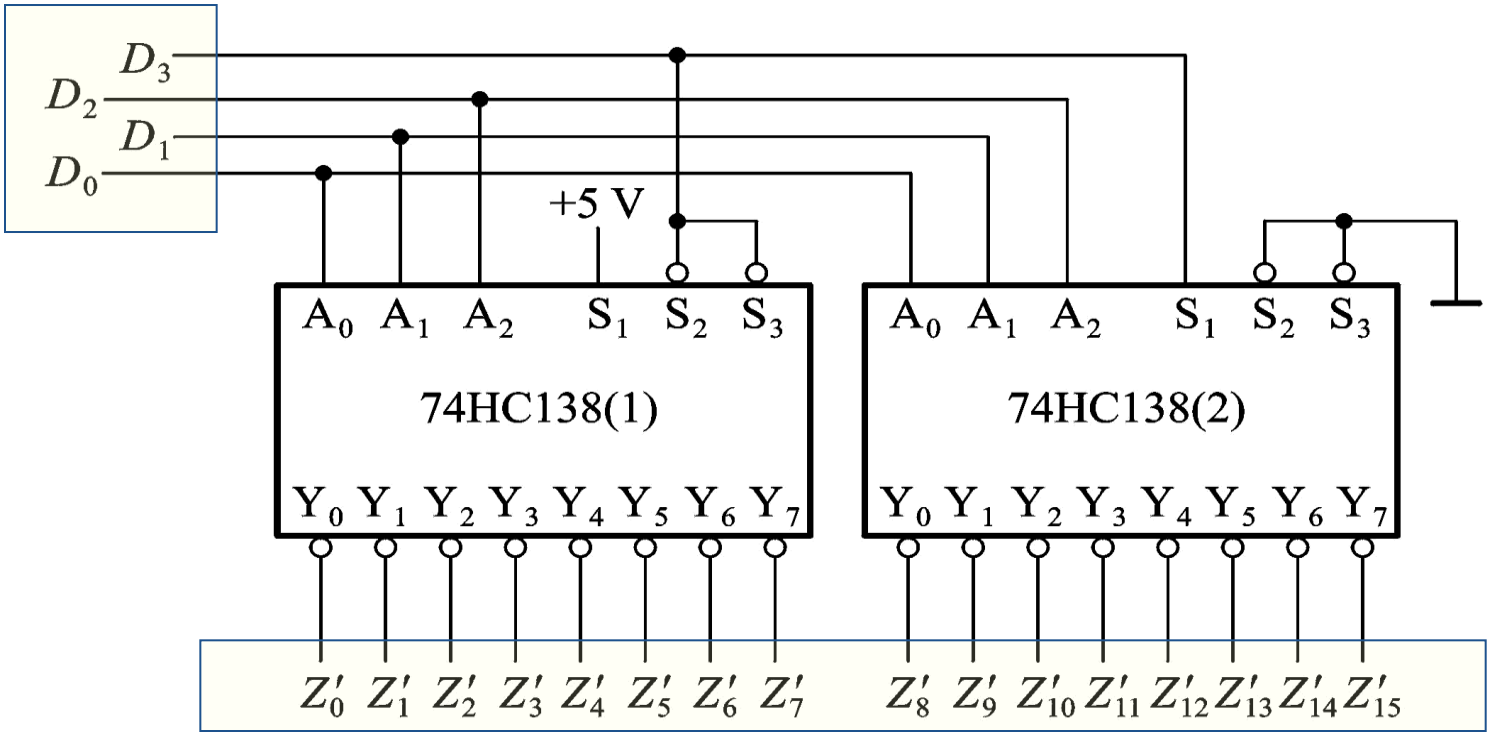


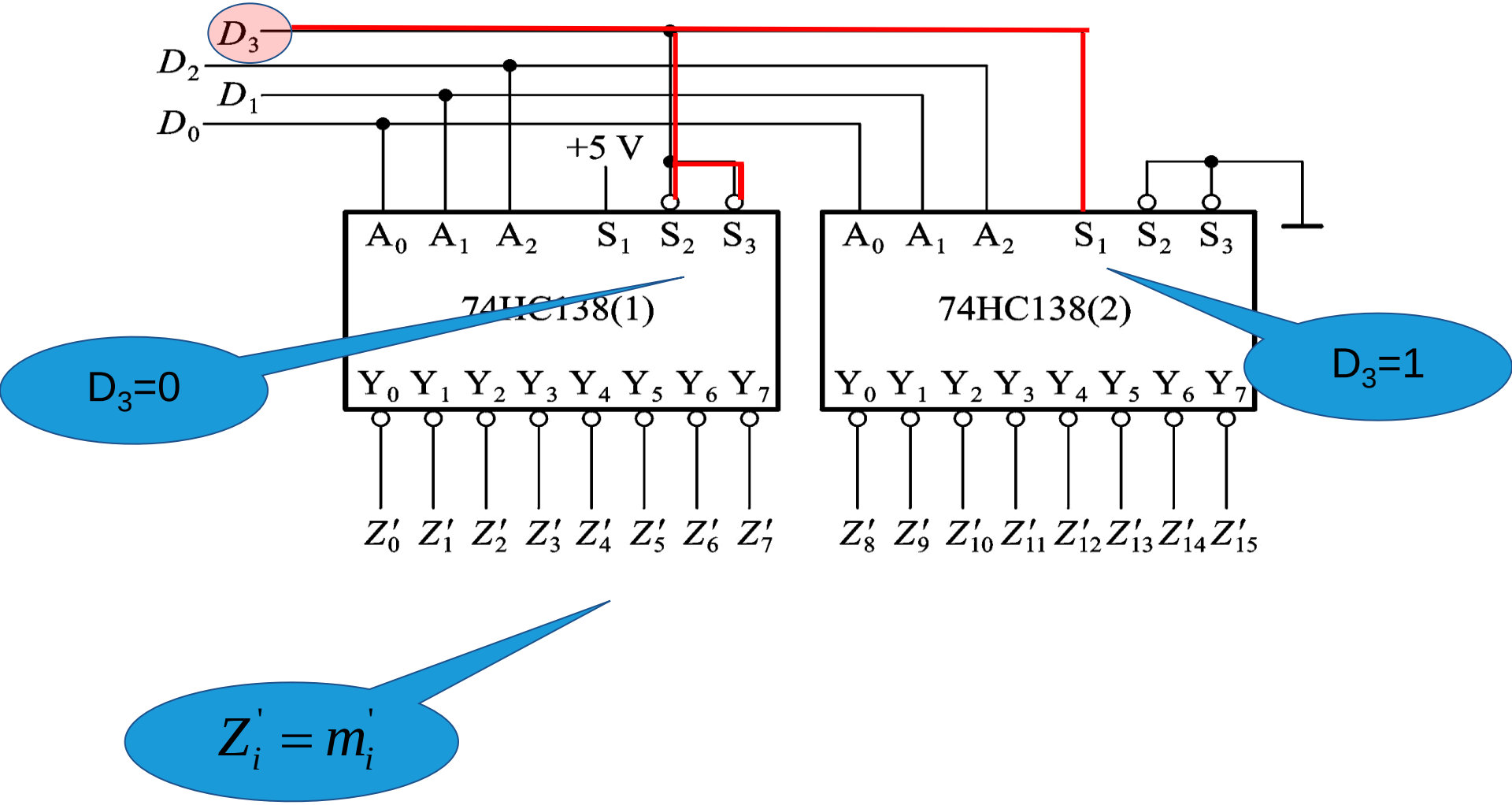
- 第一片为高优先权
- 只有(1)无编码输入时，(2)才允许工作
- 第(1)片 $Y'_{EX}=0$ 时表示对 $A'_{15}\sim A'_8$ 的编码
- 低3位输出应是两片的输出的“或”


A'_{14} A'_{12} A'_{10} A'_8
A'_{15} A'_{13} A'_{11} A'_9
A'_6 A'_4 A'_2 A'_0
A'_7 A'_5 A'_3 A'_1
I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0
S'
S
74HC148(1)
Y_S
Y'_S
74HC148(2)
Y_{EX} Y_2 Y_1 Y_0
Y'_{EX} Y'_2 Y'_1 Y'_0
Z_3 Z_2 Z_1 Z_0

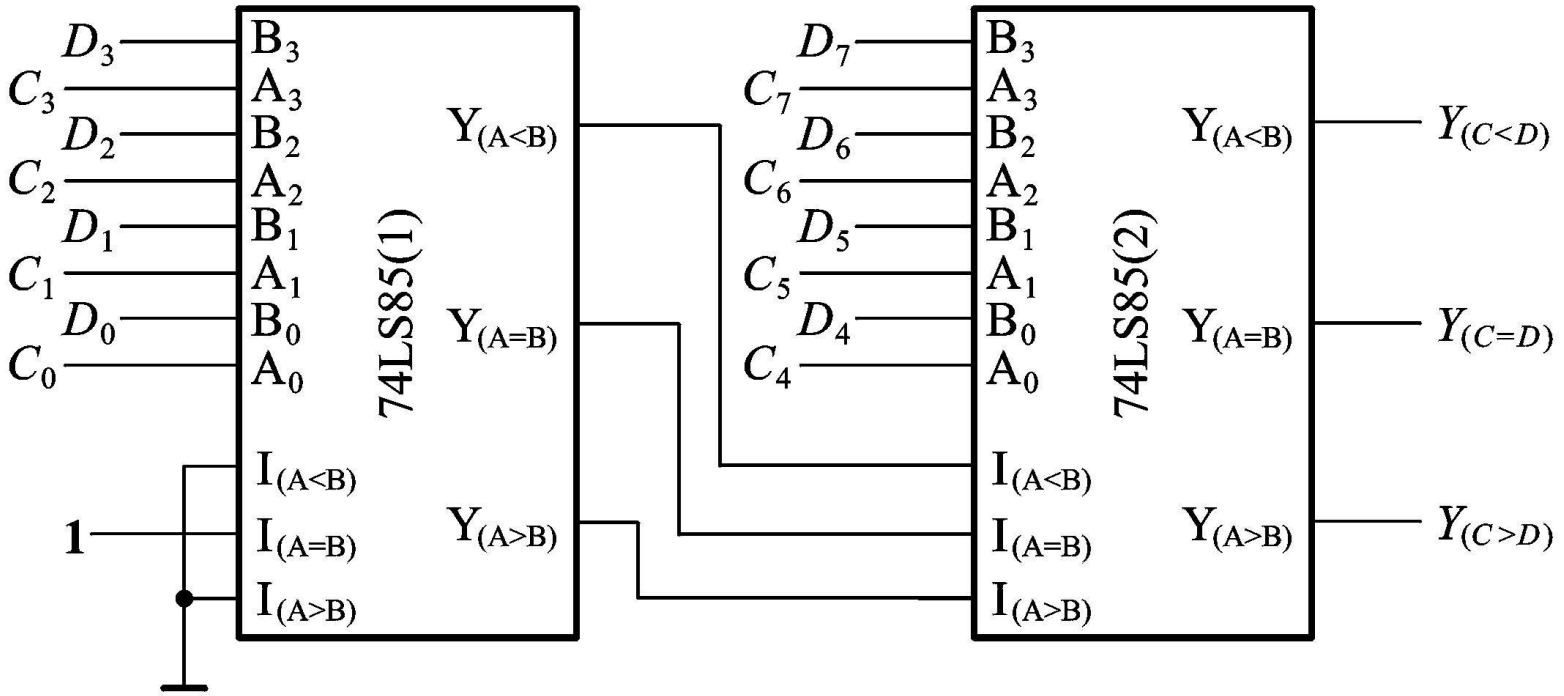
❖ 利用附加控制端进行扩展

例2：用74HC138（3线—8线译码器）组成4线—16线译码器




D_3
D_2
D_1
D_0
+5 V
A_0 A_1 A_2 S_1 S_2 S_3
74HC138(1)
Y_0 Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7
Z'_0 Z'_1 Z'_2 Z'_3 Z'_4 Z'_5 Z'_6 Z'_7
A_0 A_1 A_2 S_1 S_2 S_3
74HC138(2)
Y_0 Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7
Z'_8 Z'_9 Z'_{10} Z'_{11} Z'_{12} Z'_{13} Z'_{14} Z'_{15}
D_3=0
D_3=1
$Z_i' = m_i'$

例3：比较两个8位二进制数的大小



例4：用数据选择器设计组合电路

- 基本原理

$$Y_1 = D_0(A_1'A_0') + D_1(A_1'A_0) + D_2(A_1A_0') + D_3(A_1A_0)$$

（三变量组合逻辑函数）

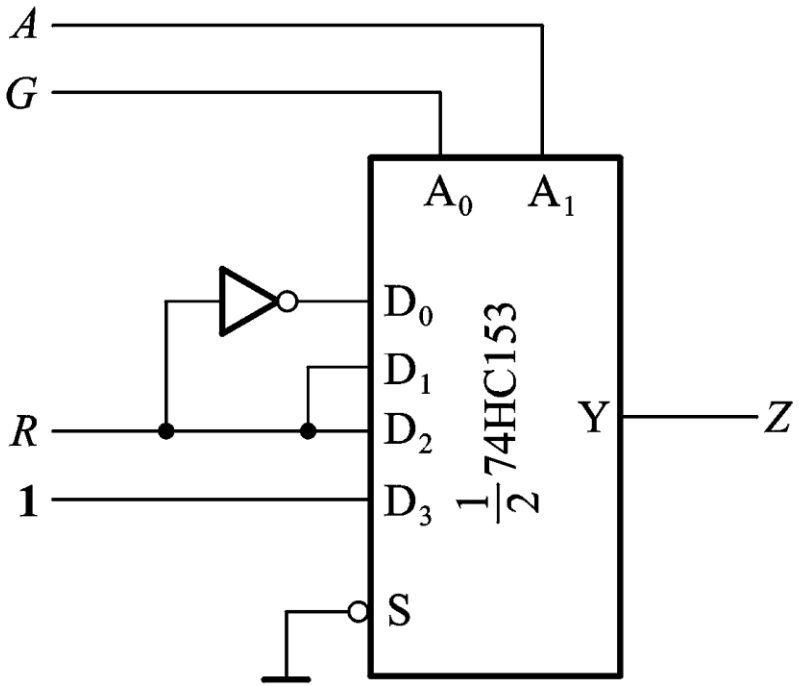
具有*n*位地址输入的数据选择器，可产生任何形式的输入变量不大于*n+1*的组合函数

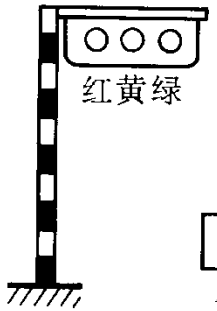
例如：

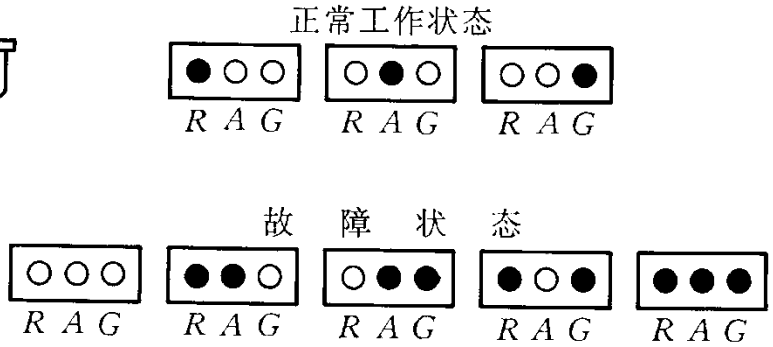
$$Z = R'A'G' + R'AG + RA'G + RAG' + RAG$$

$$= R'(A'G') + R(A'G) + R(AG') + 1\cdot(AG)$$

$$Y_1 = S_1[D_0(A_1'A_0') + D_1(A_1'A_0) + D_2(A_1A_0') + D_3(A_1A_0)]$$

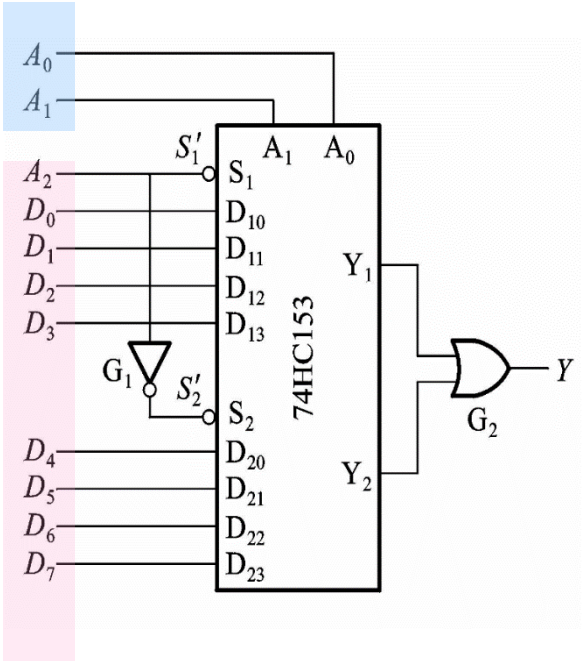






例5：用两个“四选一”接成“八选一”，并产生逻辑函数

- “四选一”只有2位地址输入，从四个输入中选中一个
- “八选一”的八个数据需要3位地址代码指定其中任何一个



利用S'作为第3位地址输入端

$$Y=(A_2'A_1'A_0')D_0+(A_2'A_1'A_0)D_1+(A_2'A_1A_0')D_2+(A_2'A_1A_0)D_3$$
$$+(A_2A_1'A_0')D_4+(A_2A_1'A_0)D_5+(A_2A_1A_0')D_6+(A_2A_1A_0)D_7$$

例6：用译码器设计组合逻辑电路

1. 基本原理

3位二进制译码器给出3变量的全部最小项；

。。。

*n*位二进制译码器给出*n*变量的全部最小项；

任意函数

将n位二进制译码输出的最小项组合起来，可获得任何形式的输入变量不大于n的组合函数

$$Y = \sum m_i$$

2. 举例

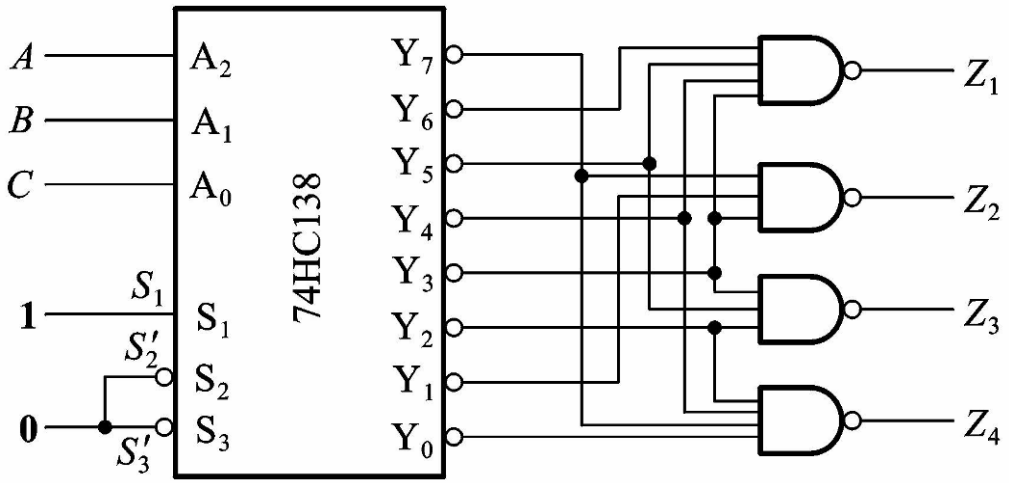
例：利用74HC138设计一个多输出的组合逻辑电路，输出逻辑函数式为：

$$Z_1 = AC' + A'BC + AB'C$$

$$Z_2 = BC + A'B'C$$

$$Z_3 = A'B + AB'C$$

$$Z_4 = A'BC' + B'C' + ABC$$



$$Z_1 = AC' + A'BC + AB'C = \sum m(3,4,5,6) \qquad Z_1 = \sum m(3,4,5,6) = (m_3' m_4' m_5' m_6')'$$

$$Z_2 = BC + A'B'C = \sum m(1,3,7) \qquad Z_2 = \sum m(1,3,7) = (m_1' m_3' m_7')'$$

$$Z_3 = A'B + AB'C = \sum m(2,3,5) \qquad Z_3 = \sum m(2,3,5) = (m_2' m_3' m_5')'$$

$$Z_4 = A'BC' + B'C' + ABC = \sum m(0,2,4,7) \qquad Z_4 = \sum m(0,2,4,7) = (m_0' m_2' m_4' m_7')'$$

思考：

利用74HC138设计一个多输出的组合逻辑电路，输出逻辑函数式为：

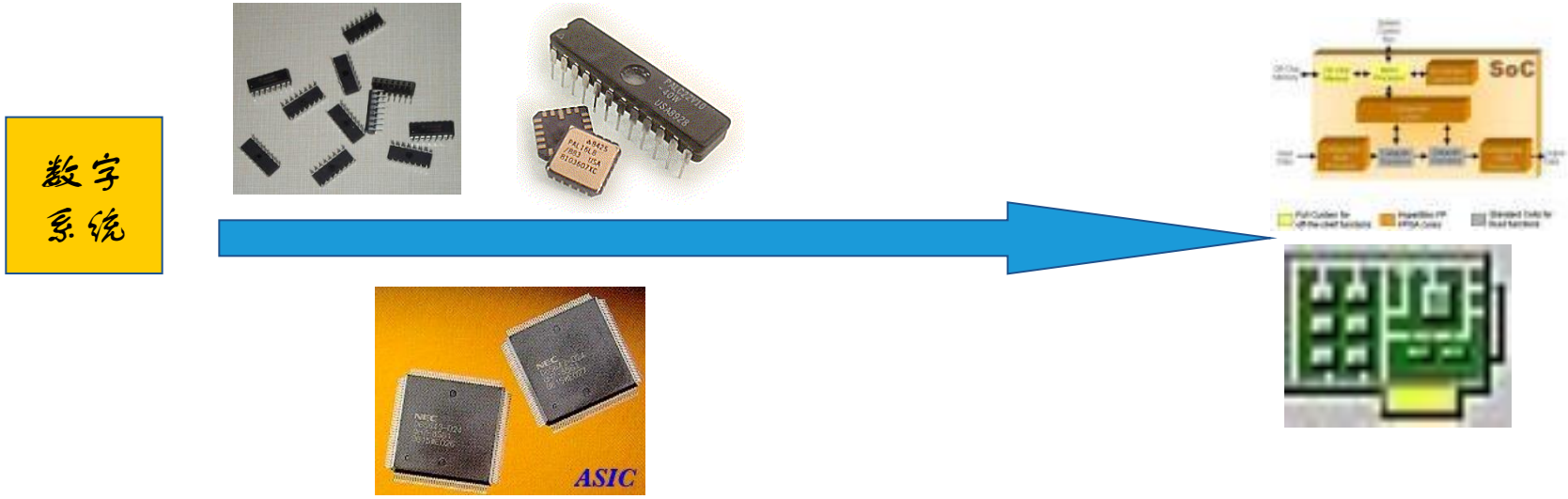
$$Y_1(ABC)=\sum_m(0,1,4,6)$$

$$Y_2(ABCD)=AB'C+ABD+A'B'D+BCD'$$

4.6 可编程逻辑器件

一、PLD的基本特点

1. 数字集成电路从功能上有分为通用型、专用型两大类



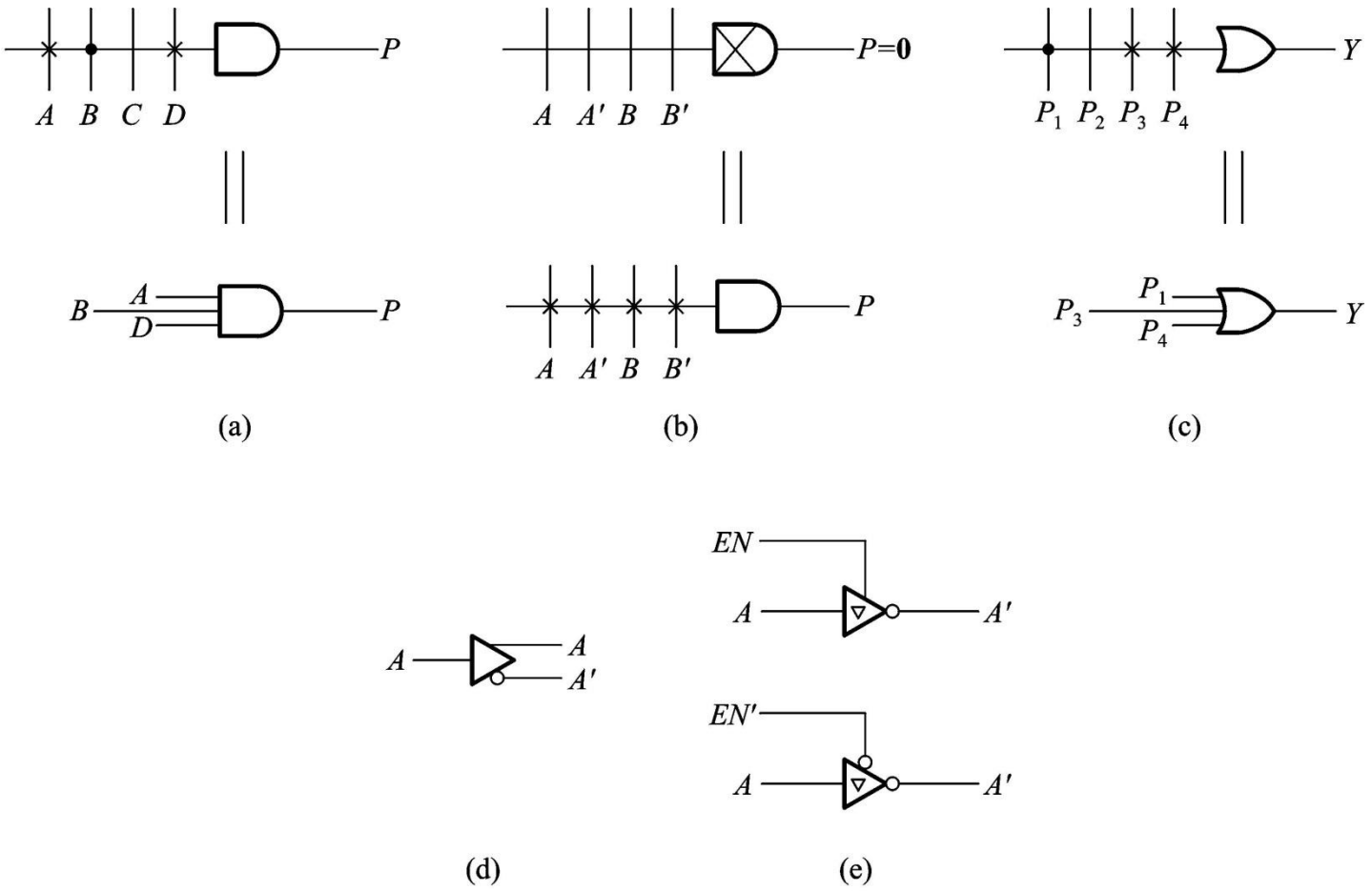
2. PLD的特点：是一种按通用器件来生产，但逻辑功能是由用户通过对器件编程来设定的

二、PLD的发展和分类

PROM是最早的PLD

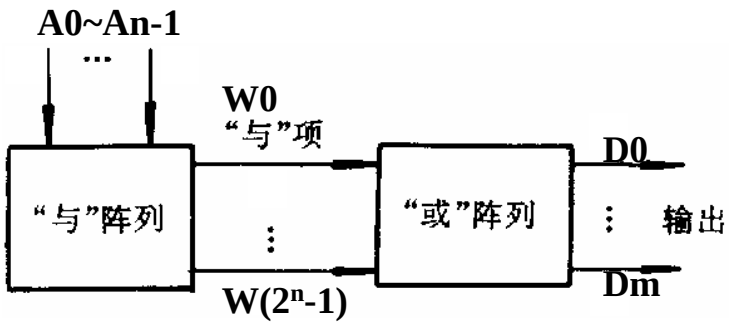
1. **PAL　可编程逻辑阵列**
2. **FPLA　现场可编程阵列逻辑**
3. **GAL 通用阵列逻辑**
4. **EPLD 可擦除的可编程逻辑器件**
5. **CPLD 复杂的可编程逻辑器件**
6. **FPGA 现场可编程门阵列**
7. **isp-PLD 在系统可编程的PLD**

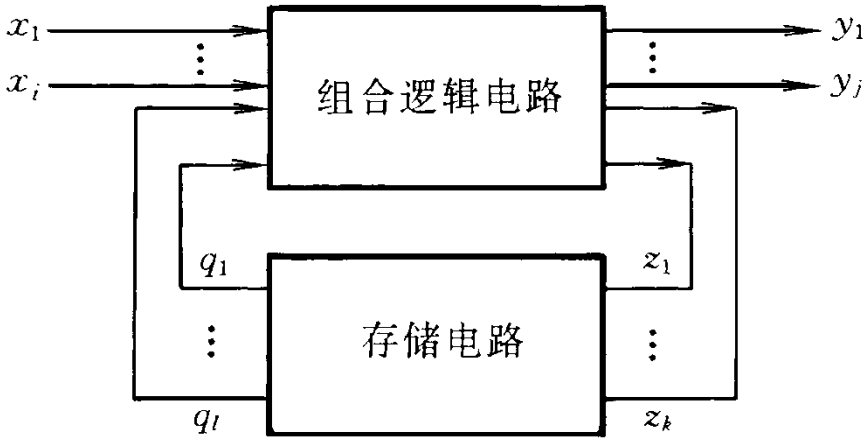
三、PLD中用的逻辑图符号



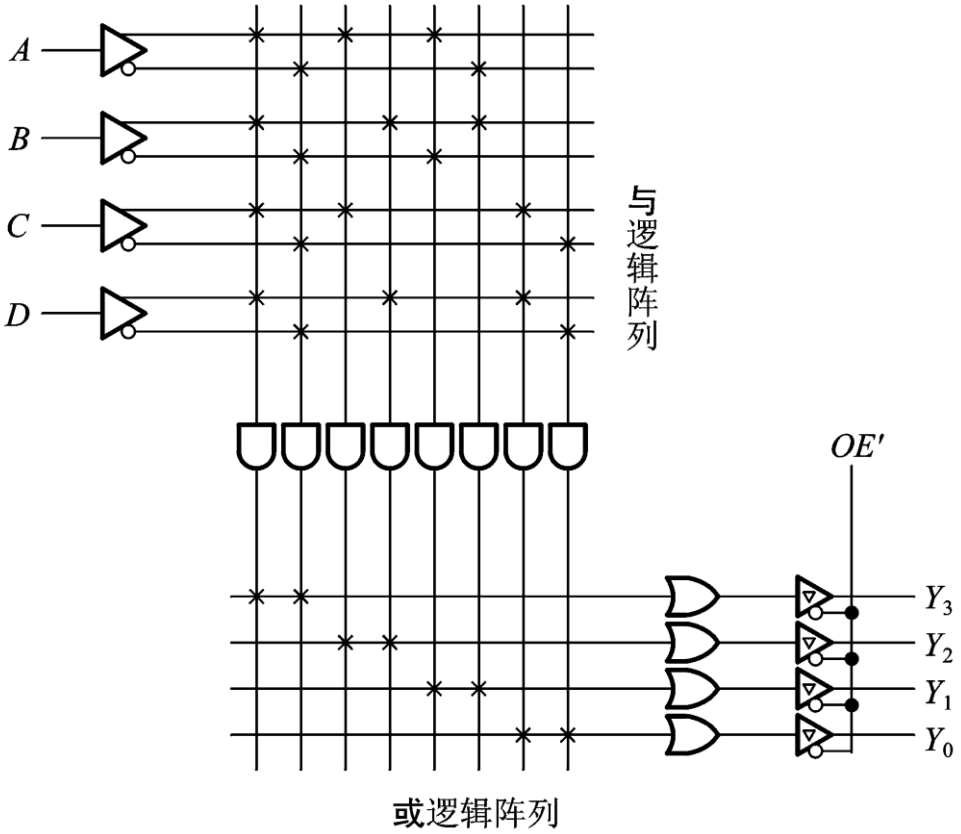
现场可编程逻辑阵列 FPLA

组合电路和时序电路结构的通用形式





FPLA的基本电路结构



可编程的“与”阵列
+可编程的“或”阵列
+输出缓冲器组成

$$Y_3 = ABCD + A'B'C'D'$$

$$Y_2 = AC + BD$$

$$Y_1 = A \oplus B$$

$$Y_0 = C \odot D$$

4.7 硬件描述语言

EDA

- 电子产品从设计、仿真调试、硬件实现全过程自动化。
- 从CAD到EDA
- CAD Computer Aided Design (Draw)
- CAE Computer Aided Engineer
- EDA Electronic Design Automation

EDA的技术特征

- 以超大规模IC为基础
- 以高性能计算机及软件为平台
- 多学科综合
- 实现电子产品从设计到生产全过程自动化。
- 电路软件化－软件即是电路

  用计算机程序描述电路
- 电路的描述形式

  HDL Hardware Description Language

- **Multisim / Ultiboard:** 界面好，大学实验室，教学（电子工艺实习）
- **MAX + plus II / Quartus:** 数字电路，主要用于数字系统（**PLD** ）设计和下载
- **OrCAD PSpice :** 精确，主要用于模拟电路设计、
  仿真，科研开发

EDA工具

- 设计输入：电路的功能、结构描述
  可用图形和语言等多种方式
- 电路综合：包括逻辑综合、布局、布线
- 仿真：电路未做出前、先进行验证：
  功能仿真－逻辑表达式正确
  时序仿真－实际可用
- 硬件实现：制作、下载芯片

- 硬件描述语言，如VHDL, Verilog
- 真值表，方程式，电路逻辑图（Schematic）
- 状态转换图（ FSM）

硬件描述语言的作用(HDL)

- 设计人员和EDA工具之间的一种界面
- HDL主要用于编写设计文件，在EDA工具中建立电路模型。通过对电路结构或功能行为的描述，可以在不同的抽象层次对电路进行逐层描述，用一系列分层次的模块来表示极其复杂的数字电路系统。
- 利用HDL和EDA，可以完成从系统、算法、协议的抽象层次对电路进行建模、仿真、性能分析直到IC版图或PCB版图生成的全部设计工作。

硬件描述语言的发展

- 已有30年的历史
- VHDL和Verilog HDL语言被确认为IEEE标准
- 新的HDL包括：Superlog、SystemC、Cynlib C++等

Verilog HDL简介

- 1983年Gateway Design Automation公司在C语言的基础上，为其仿真器Verilog – XL开发。
- 1989年Cadence公司收购GDA公司。
- 1995年IEEE标准Verilog HDL 1364 - 1995。
- 2001年IEEE标准Verilog HDL 1364 – 2001，其中加入了Verilog HDL – A标准

一、基本程序结构

- 模块结构

  module＜模块名＞（＜端口列表＞）

  ＜定义＞

  ＜模块条目＞

  endmodule

- 模块种类：
  - 行为描述模块：定义模块的状态和功能
  - 结构描述模块：将电路表达为具有层次概念的互相联系的子模块，其最底层的元件必须是**Verilog HDL**支持的基元或已定义过的模块

二、模块的两种描述方式

1、行为描述方式

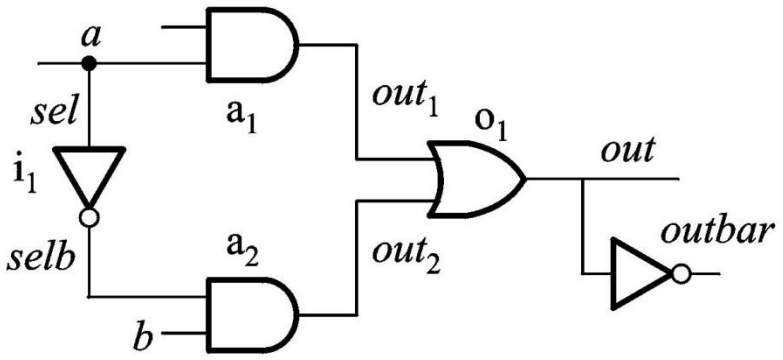
- 通过行为语句来描述电路要实现的功能，表示输入与输出间转换的行为，不涉及具体结构。
- 例如：2选1数据选择器

```
module mux_2_to_1(a,b,out,outbar,sel);
    input a,b,sel;
    output out,outbar;
    assign out = sel?a : b;
    assign outbar =~ out;
endmodule
```

2、结构描述方式

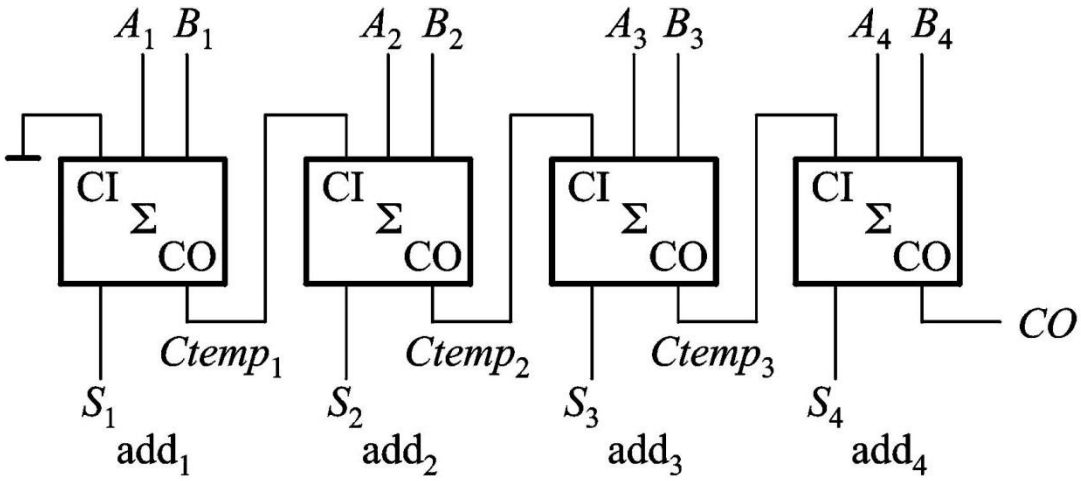
- 将硬件电路描述成一个分级子模块相互连接的结构。
- 例如：2选1数据选择器

```
module muxgate(a,b,out,outbar,sel);
    input a,b,sel;
    output out,outbar;
    wire out1, out2, selb;
        and a1(out1,a,sel);
        not i1(selb,sel);
        and a2(out2,b,selb);
        or  o1(out,out1,out2);
        assign outbar =~ out;
endmodule
```

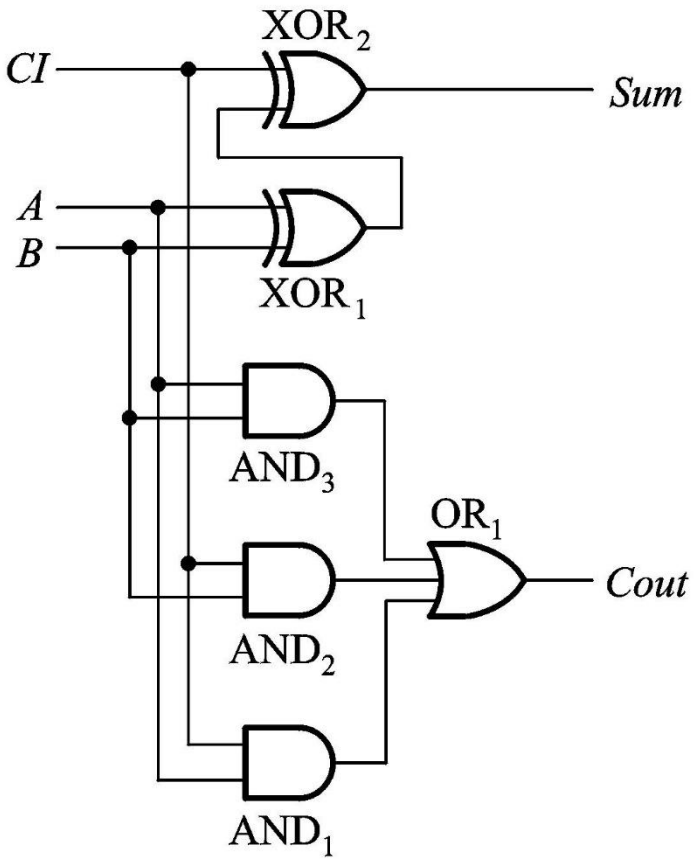


三、描述组合逻辑电路的实例

```
module Four_bit_fulladd(A,B,CI,S,CO)
    parameter size = 4;
    input[size : 1]A, B;
    output[size : 1]S;
    input CI;
    output CO;
    wire [1 : size - 1]Ctemp
onebit_fulladd
    add1(A[1], B[1], CI, S[1], Ctemp[1]),
    add2(A[2], B[2], Ctemp[1], S[2], Ctemp[2]),
    add2(A[3], B[3], Ctemp[2], S[3], Ctemp[3]),
    add2(A[4], B[4], Ctemp[3], S[4], CO);
endmodule
```



```
module onebit_fulladd(A,B,CI,Sum,Cout)
    input A,B,CI;
    output Sum,Cout;
    wire Sum_temp, C_1, C_2, C_3;
    xor
     XOR1(Sum_temp, A, B),
     XOR2(Sum,Sum_temp, CI);
    and
     AND1(C_1, A, B),
     AND2(C_2, A, B),
     AND3(C_3, A, B),
    or
    OR1(Cout, C_1, C_2, C_3);
endmodule
```



4.8 用可编程通用模块（PLD）设计组合逻辑电路

一、开发系统

1. 硬件：计算机+编程器
2. 软件：开发环境（软件平台）

   PALASM（早期）

   ABEL和CUPL

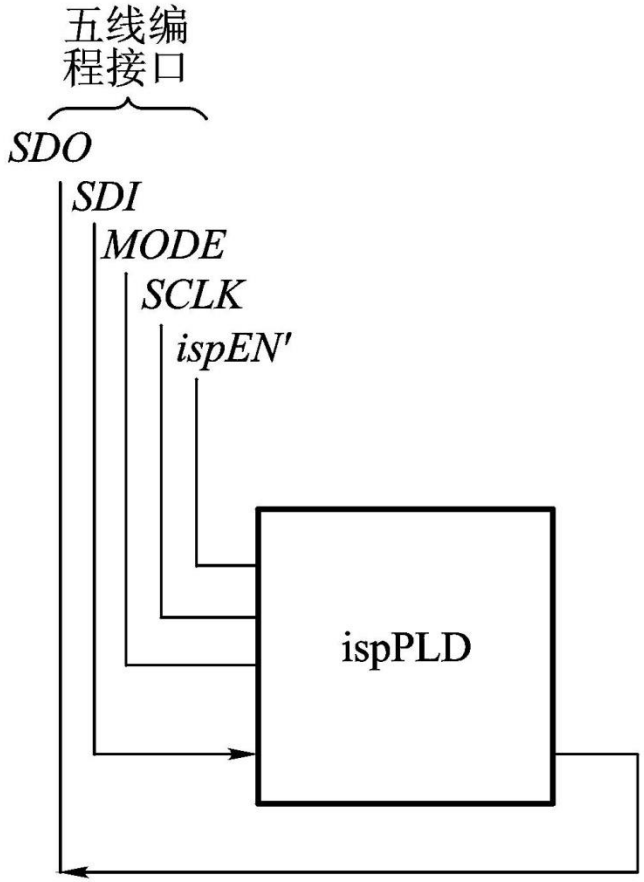
   Synario（电路原理图输入）

   MAX + plus II，ISP Synario，Foundation（大型软件包）

二、步骤

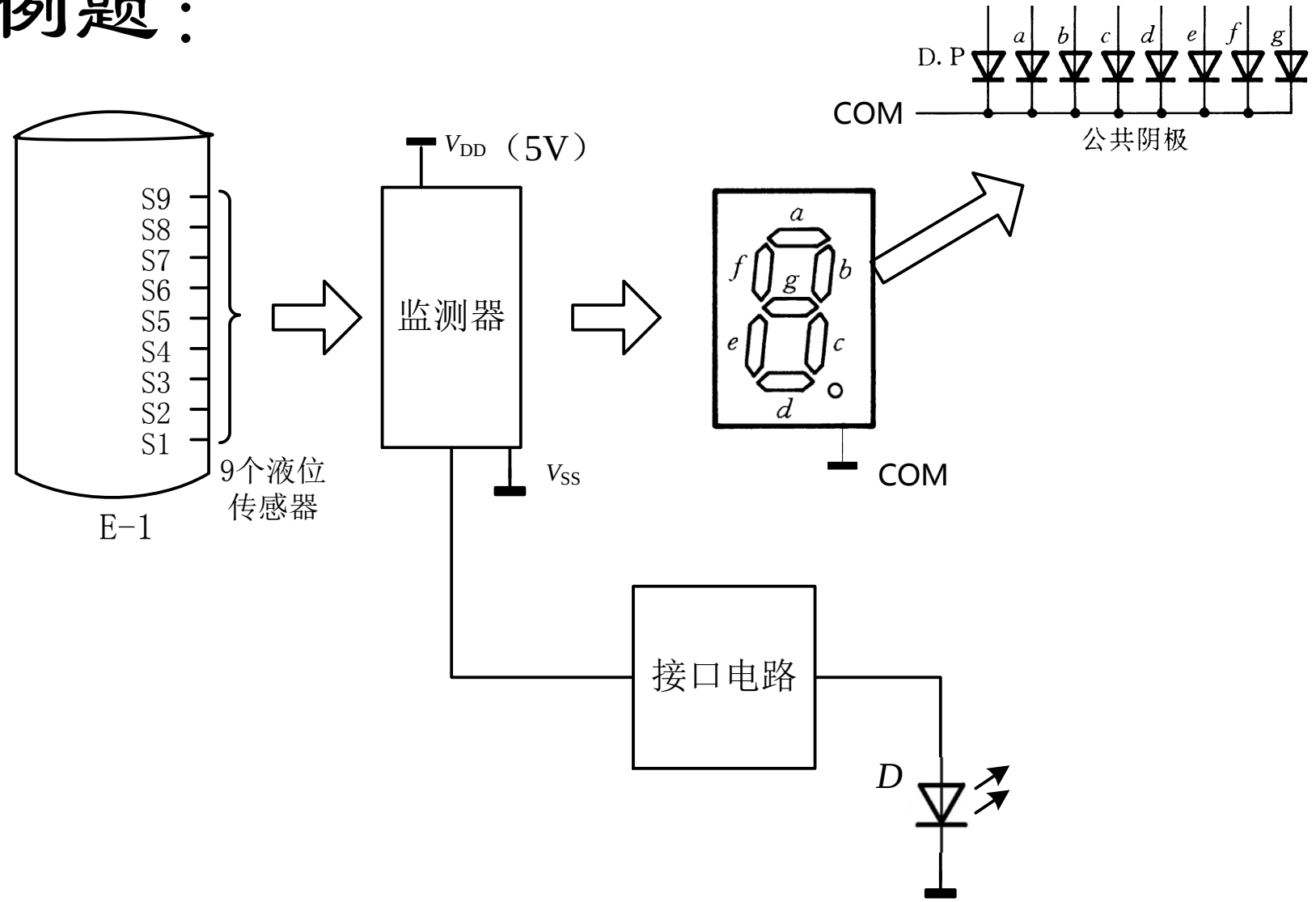
- 抽象（系统设计采用Top-Down的设计方法）
- 选定PLD
- 选定开发系统
- 编写源程序（或输入文件）
- 调试，运行仿真，产生下载文件
- 下载
- 测试

isp器件的编程接口（Lattice）



- 使用ispPLD的优点：
- 不再需要专用编程器
- 为硬件的软件化提供可能
- 为实现硬件的远程构建提供可能

例题：


D. P
a
b
c
d
e
f
g
COM
公共阴极
V_{DD}（5V）
S9
S8
S7
S6
S5
S4
S3
S2
S1
9个液位
传感器
E-1
监测器
V_{SS}
COM
接口电路
D

```
//模块一：编码模块
module code_9to4 (F,S); //定义编码模块的名字和接口
output [3:0] F;
input [9:1] S;
reg [3:0] F;
always @ (S)                    //优先编码的真值表
 case (S)
    9'b000000001 : F=4'b0001; //液面超过S1，编码1（0001）
    9'b000000011 : F=4'b0010; //液面超过S2，编码2（0010）
    9'b000000111 : F=4'b0011; //液面超过S3，编码3（0011）
    9'b000001111 : F=4'b0100; //液面超过S4，编码4（0100）
    9'b000011111 : F=4'b0101; //液面超过S5，编码5（0101）
    9'b000111111 : F=4'b0110; //液面超过S6，编码6（0110）
    9'b001111111 : F=4'b0111; //液面超过S7，编码7（0111）
    9'b011111111 : F=4'b1000; //液面超过S8，编码8（1000）
    9'b111111111 : F=4'b1001; //液面超过S9，编码9（1001）
    default : F=4'bx;
endcase
endmodule
```

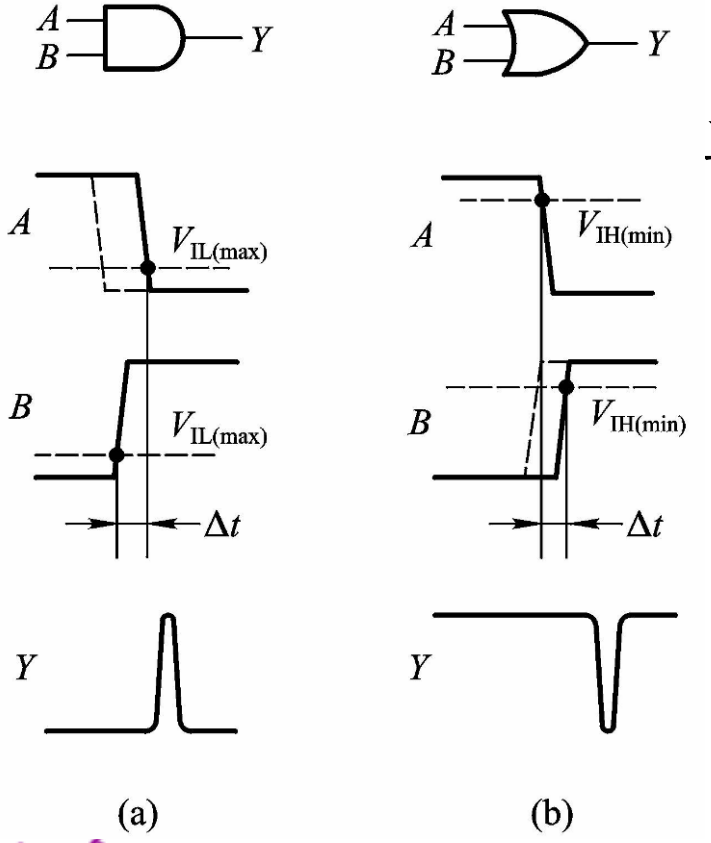
```
//模块二：7段显示译码模块
module bin27seg (data , y);
input [3:0] data;
output [6:0] y;
reg [6:0] y;
always @ (data)
begin
y=7'b0000000;
case (data)
4'b0000 : y=7'b0111111; //显示0
4'b0001 : y=7'b0000110; //显示1
4'b0010 : y=7'b1011011; //显示2
4'b0011 : y=7'b1001111; //显示3
4'b0100 : y=7'b1100110; //显示4
4'b0101 : y=7'b1101101; //显示5
4'b0110 : y=7'b1111100; //显示6
4'b0111 : y=7'b0000111; //显示7
4'b1000 : y=7'b1111111; //显示8
4'b1001 : y=7'b1100111; //显示9
default : y=7'b0000000; //熄灭
endcase
endmodule
```

4.9 组合逻辑电路中的竞争-冒险现象

4.9.1 竞争-冒险现象及成因

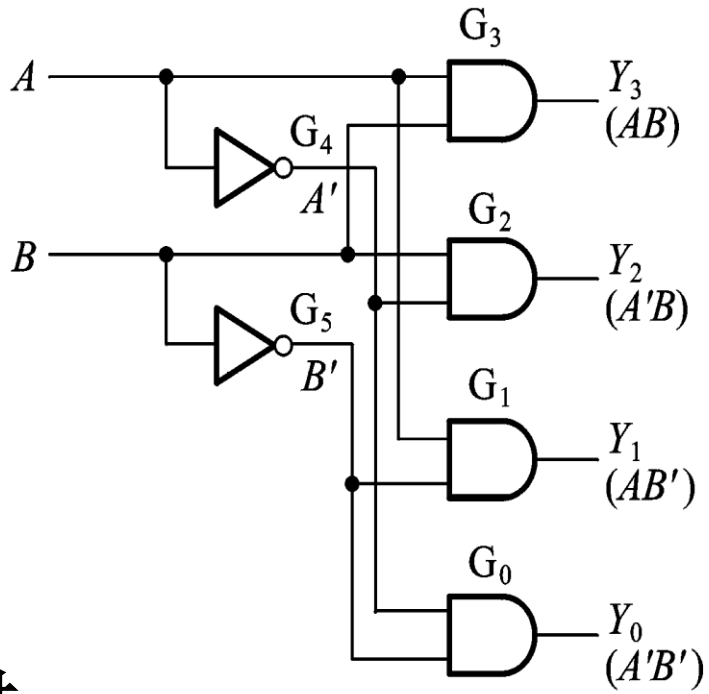
一、什么是“竞争”

两个输入“同时向相反的逻辑电平变化”，称存在“竞争”

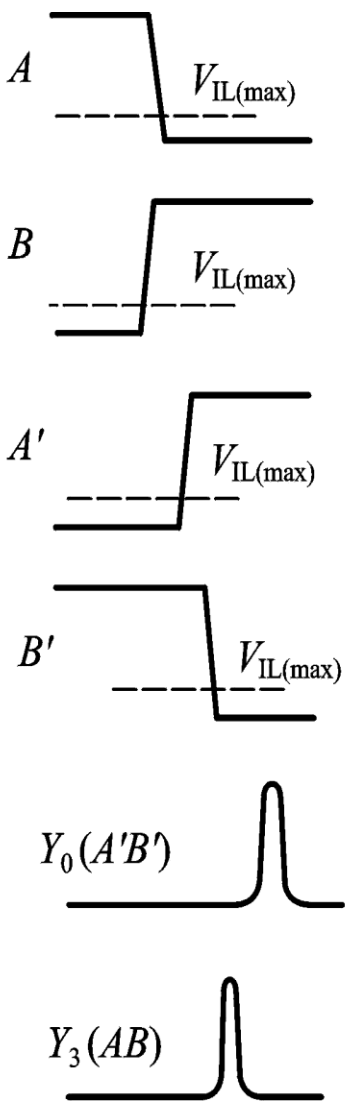


二、因“竞争”而可能在输出产生尖峰脉冲的现象，称为“竞争-冒险”。

三、2线—4线译码器中的竞争-冒险现象



(a)



(b)

当AB从$10 \rightarrow 01$时，

在动态过程中可能出现0

所以Y_3和Y_0输出端可能产生尖峰脉冲。

4.9.2 检查竞争-冒险现象的方法

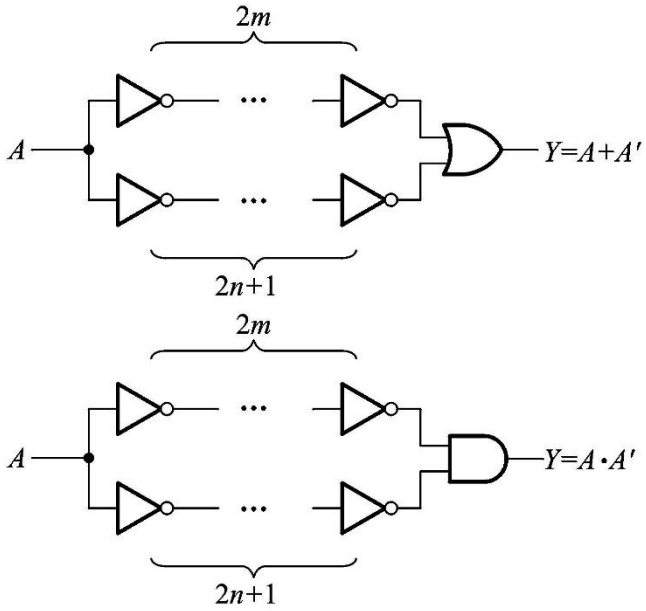
当输入变量每次只有一个改变状态的情况下：

如果输出端门电路的两个输入信号A和A’是输入变量A经过两个不同的传输途径而来的，那么当输入变量A的状态发生突变时输出端便有可能产生尖峰脉冲。

判断准则：只要输出端的逻辑函数在一定条件下能简化成

$$Y = A + A' \quad 或 \quad Y = A \cdot A'$$

则可判定存在竞争-冒险现象。



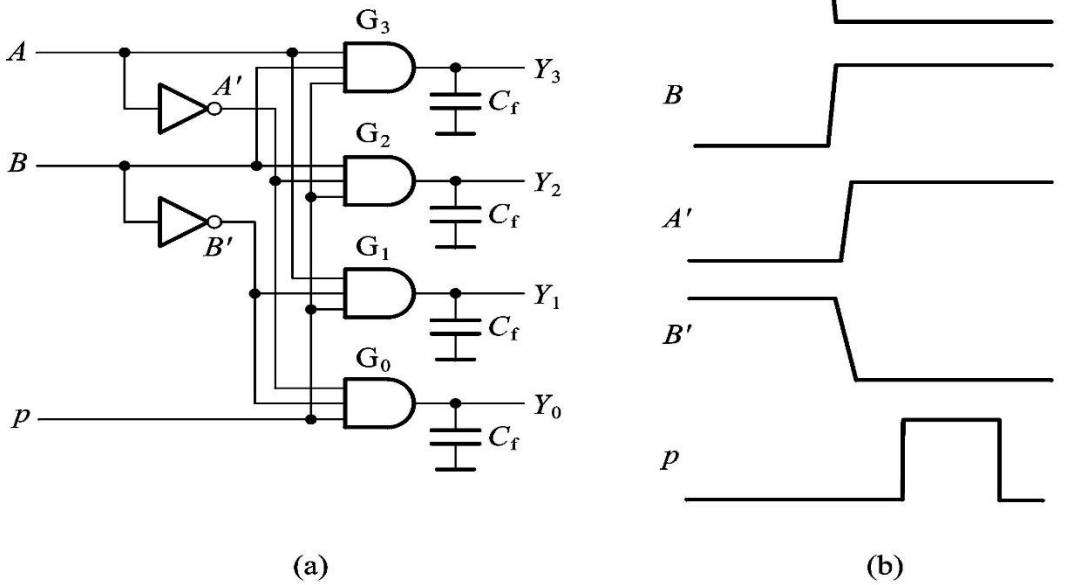
4.9.3 消除竞争-冒险现象的方法

一、接入滤波电容

尖峰脉冲很窄，用很小的电容就可将尖峰削弱到 V_{TH} 以下。

二、引入选通脉冲

利用选通脉冲，在电路达到稳定之后，P的高电平期的输出信号不会出现尖峰。



三、修改逻辑设计

例：$Y = AB + A'C$

在$B = C = 1$的条件下，$Y = A + A' \Rightarrow$ 稳态下$Y = 1$

当A改变状态时存在竞争冒险

$$Y = AB + A'C + BC$$