4.4 若干典型的组合逻辑电路

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 若干典型的组合逻辑集成电路

4.4.1 编码器

1、编码器 (Encoder)的定义与分类

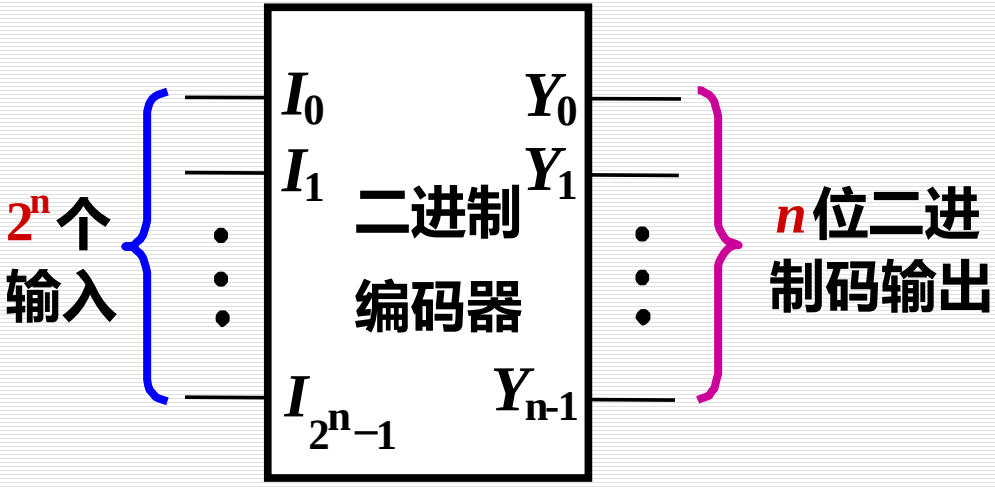
编码：赋予二进制代码特定含义的过程称为编码。

如：8421BCD码中，用1000表示数字8

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，即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

以普通二进制编码器为例：



二进制编码器的结构框图

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。

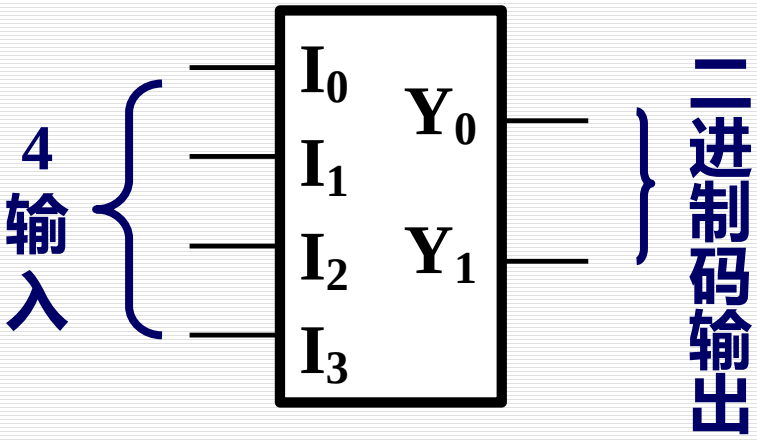
普通编码器：任何时候只允许输入一个有效编码信号，否则输出就会发生混乱。

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

(1) 4线–2线普通二进制编码器 (设计)

（a）逻辑框图



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

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

该表达式可否进一步简化？

（2）逻辑功能表

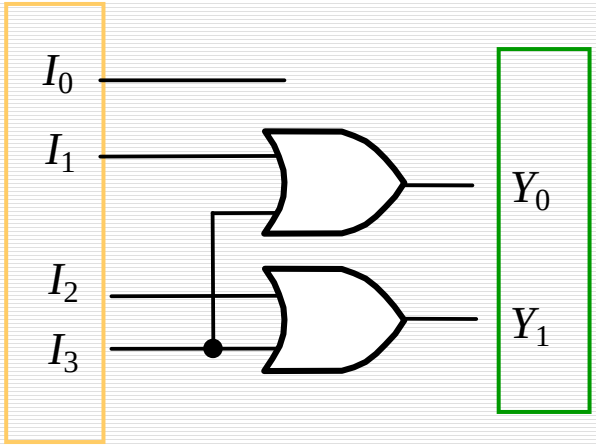
I_0	I_1	I_2	I_3	Y_1	Y_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

编码器的输入为高电平有效。

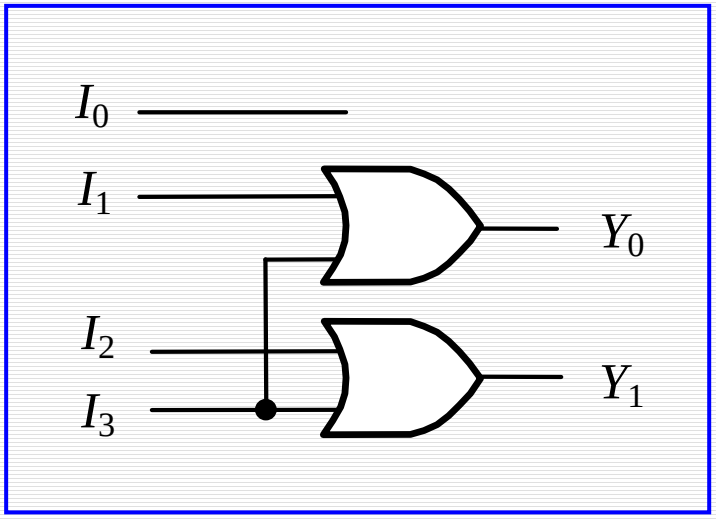
上述是将输入的其它12种组合对应的输出看做0。如果看做无关项，则表达式为

$$Y_1 = I_2 + I_3$$

$$Y_0 = I_1 + I_3$$



若有2个以上的输入为有效信号?



当只有I_3为1时，

Y_1Y_0 = ?　　Y_1Y_0 = 11

I_1= I_2 = 1 ，I_0= I_3= 0时，

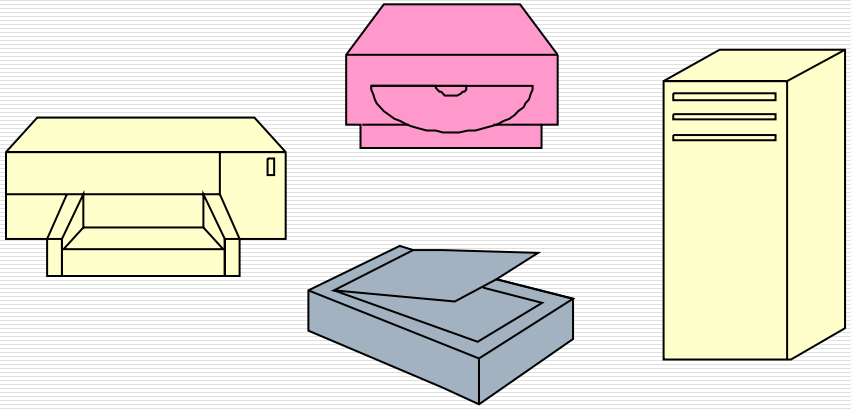
Y_1Y_0 = ?　　Y_1Y_0 = 11

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

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。

(2)优先编码器线(4—2 线优先编码器)（设计）

输入编码信号高电平有效，输出为二进制代码

输入编码信号优先级从高到低为 $I_3 \sim I_0$

输入为编码信号$I_3 \sim I_0$　输出为$Y_1 Y_0$

（1）列出功能表

输　入				输　出	
I_0	I_1	I_2	I_3	Y_1	Y_0
1	0	0	0	0	0
×	1	0	0	0	1
×	×	1	0	1	0
×	×	×	1	1	1

低 ← → 高

（2）写出逻辑表达式（化简步骤省略）

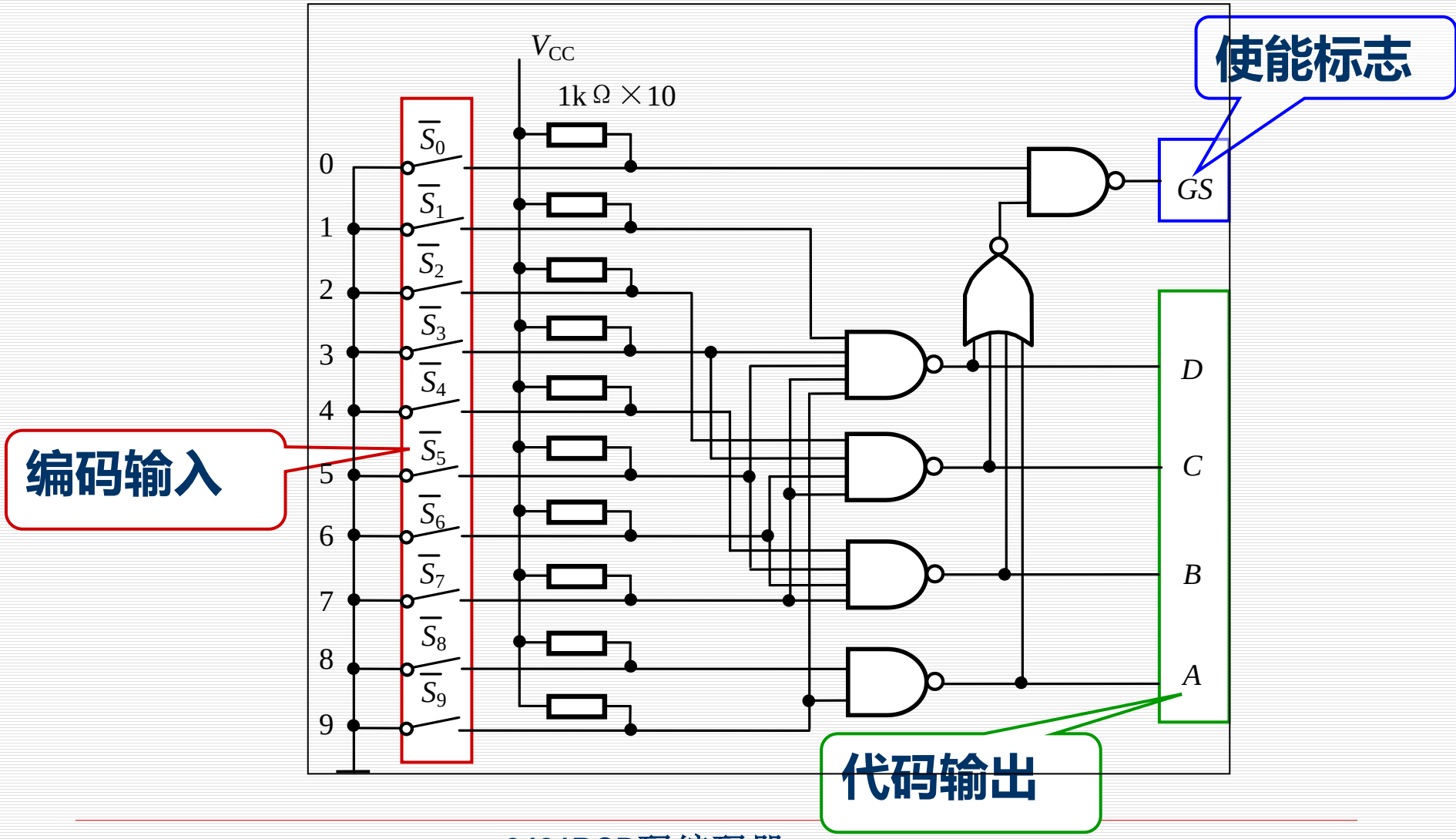
$$Y_1 = I_2\overline{I_3} + I_3$$

$$Y_0 = I_1\overline{I_2}\,\overline{I_3} + I_3$$

（3）画出逻辑电路（略）

另一个问题：如何区分输入分别为1000和0000的情况?

分析键盘输入逻辑电路（编码器）的工作原理。



8421BCD码编码器

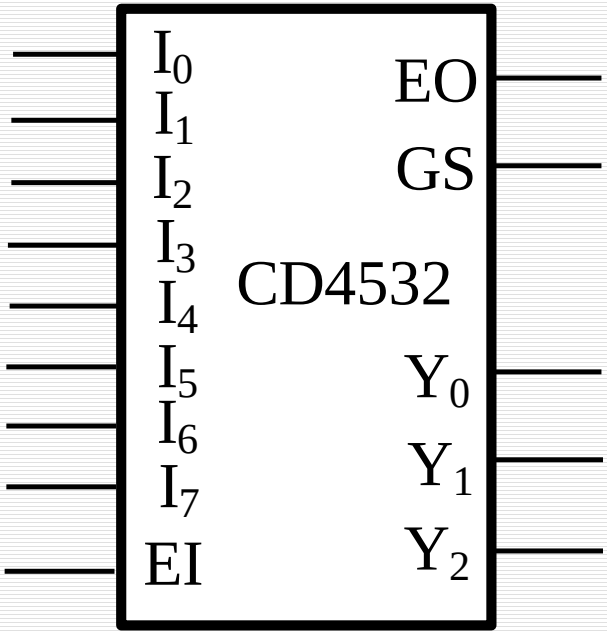
2. 键盘输入8421BCD码编码器功能表

输入										输出				
$\overline{S_9}$	$\overline{S_8}$	$\overline{S_7}$	$\overline{S_6}$	$\overline{S_5}$	$\overline{S_4}$	$\overline{S_3}$	$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	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	0	0	0	0	1
1	1	1	1	1	1	1	1	0	1	0	0	0	1	1
1	1	1	1	1	1	1	0	1	1	0	0	1	0	1
1	1	1	1	1	1	0	1	1	1	0	0	1	1	1
1	1	1	1	1	0	1	1	1	1	0	1	0	0	1
1	1	1	1	0	1	1	1	1	1	0	1	0	1	1
1	1	1	0	1	1	1	1	1	1	0	1	1	0	1
1	1	0	1	1	1	1	1	1	1	0	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	0	0	0	1
0	1	1	1	1	1	1	1	1	1	1	0	0	1	1

该编码器为输入低电平有效；

GS为标志位。

2 典型编码器电路



优先编码器CD4532的示意框图

优先编码器CD4532功能表

输入使能端

优先编码工作状态标志

输出使能端

用于实现编码器的扩展

输入									输出				
EI	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Y_2	Y_1	Y_0	GS	EO
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

思考：如何利用EO端进行扩展？？

可以用于区分输入端至少有一个有效输入，还是所有输入端均无有效输入的情况

EI=0：禁止编码器工作；

EI=1：允许编码器工作。 此时，如果所有输入端为0，则EO=1，用于实现扩展；否则，EO=0

此时，如果至少一个输入端为1，则GS=1，表明处于工作状态；否则GS=0；

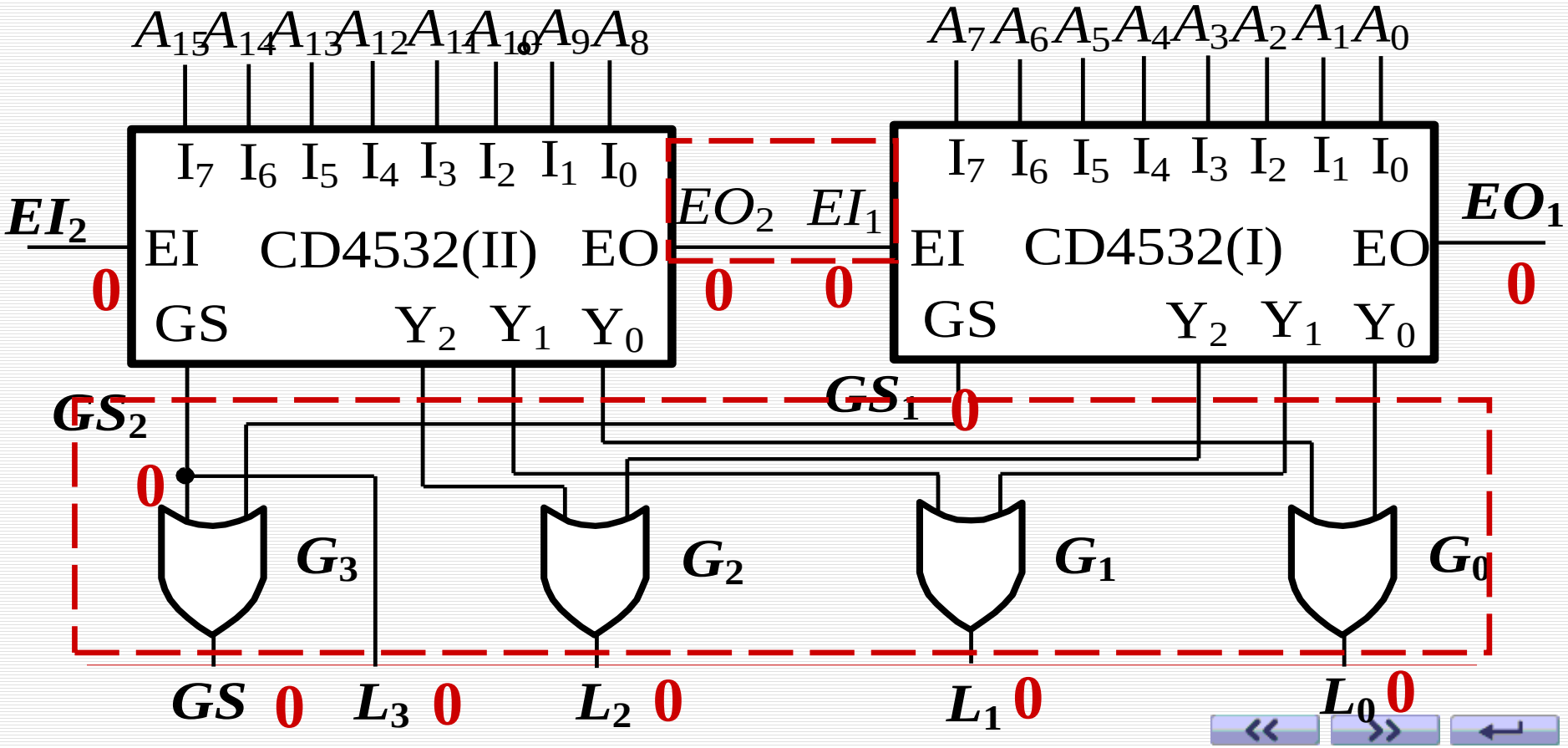
用二片CD4532构成16线-4线优先编码器,其逻辑图如下图所示，试分析其工作原理。

(1) 当使能端$EI_2=0$时：

(II)禁止编码，输出为000，$EO_2=0$，$GS_2=0$；

由于$EO_2=0$，$EI_1=0$，因此（Ⅰ）也禁止编码，输出为000，$EO_1=0$，$GS_1=0$；

$GS=GS_1+GS_2=0$，因此，无编码输出。



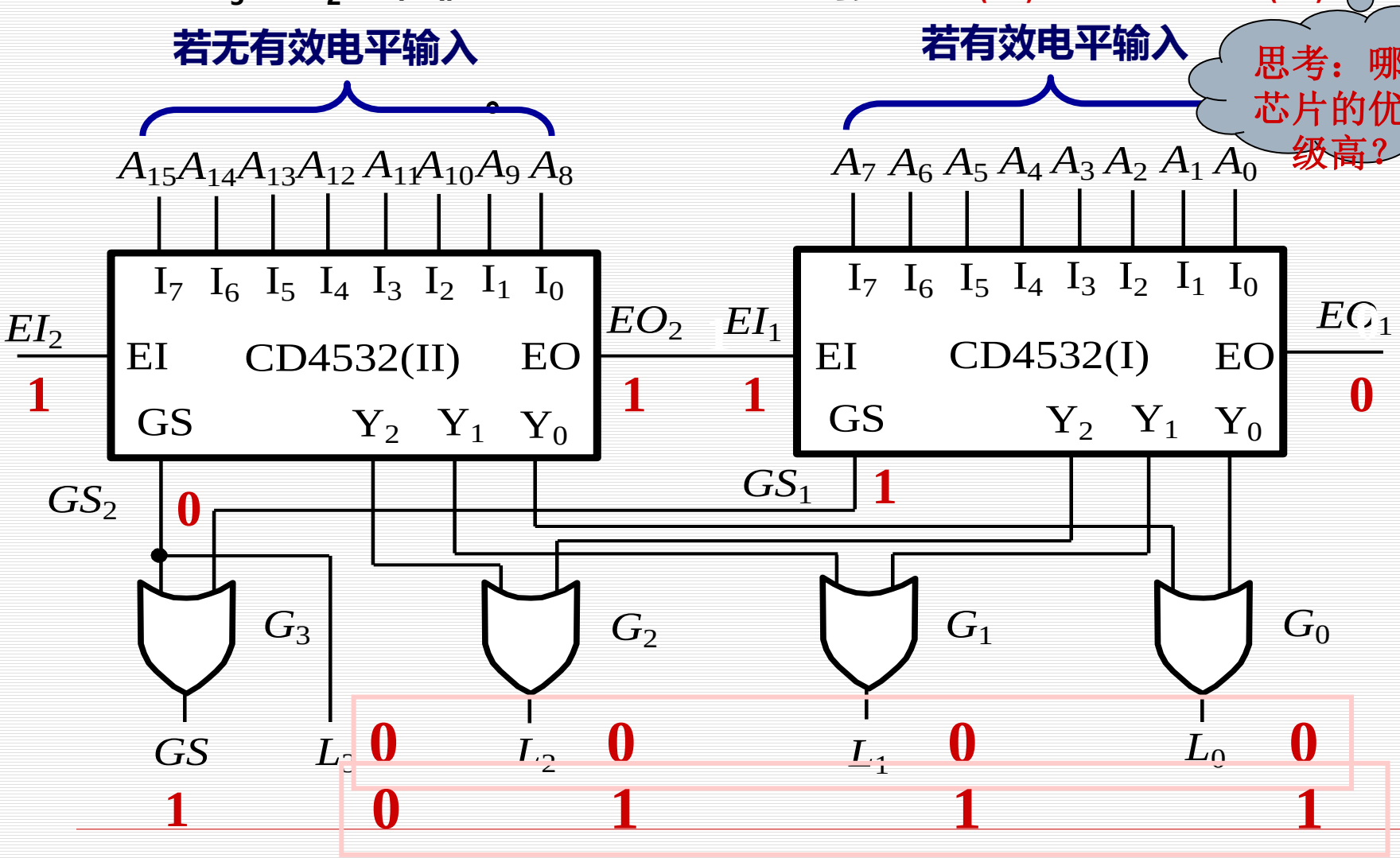
(2) 当使能端EI_2=1时，(Ⅱ)允许编码：

(a) A_{15}-A_8无有效电平输入，(Ⅱ)输出为000，EO_2=1，GS_2=0；

由于EO_2=1，EI_1=1，因此（Ⅰ）也允许编码；

L_3=GS_2=0，输出在0000和0111之间变化。(Ⅱ)的级别要高于(Ⅰ)。

思考：哪块芯片的优先级高？



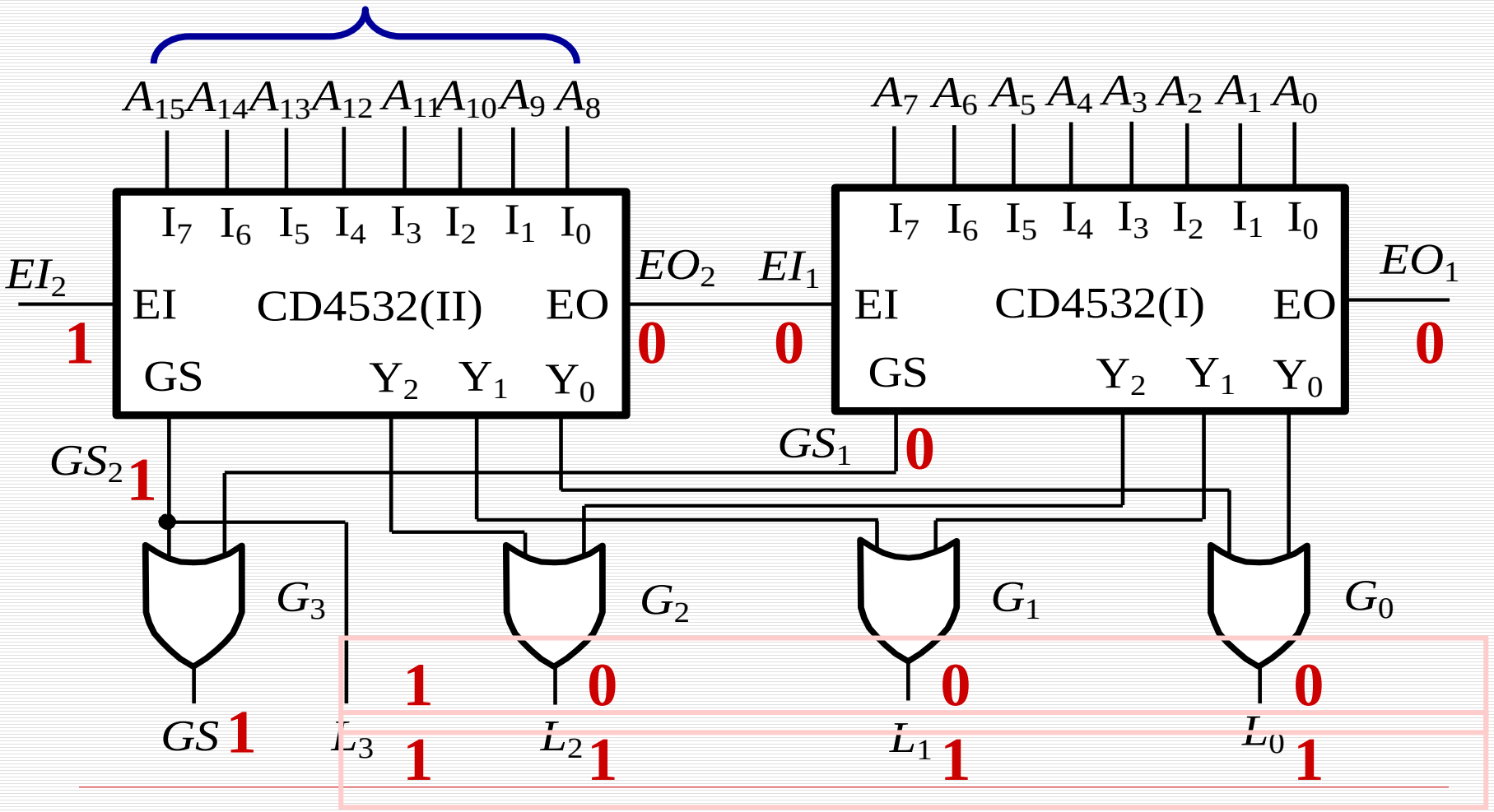
(2) 当使能端EI_2=1时，(II)允许编码：

(b) A_{15}-A_8有有效电平输入，EO_2=0，GS_2=1；

由于EO_2=0，EI_1=0，因此（I）禁止编码；

L_3=GS_2=1，输出在1000和1111之间变化。 (II)的级别要高于(I)。

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，它能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：

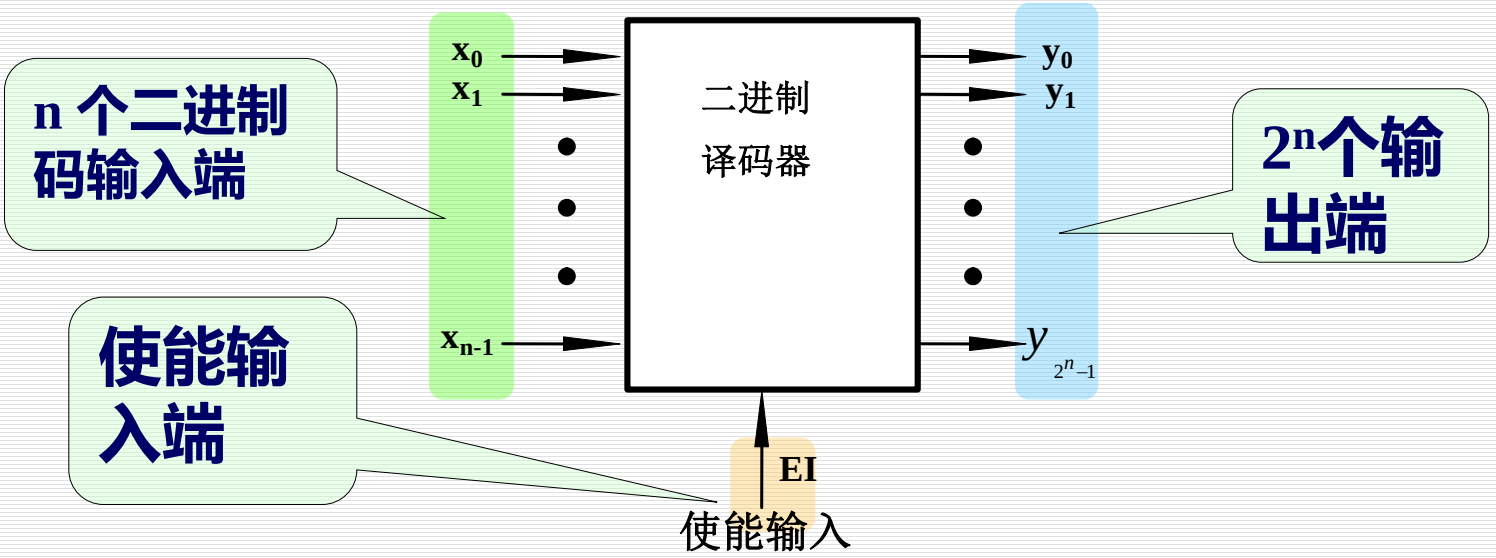
唯一地址译码器 将一系列代码转换成与之一一对应的有效信号。

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. 典型译码器电路及应用

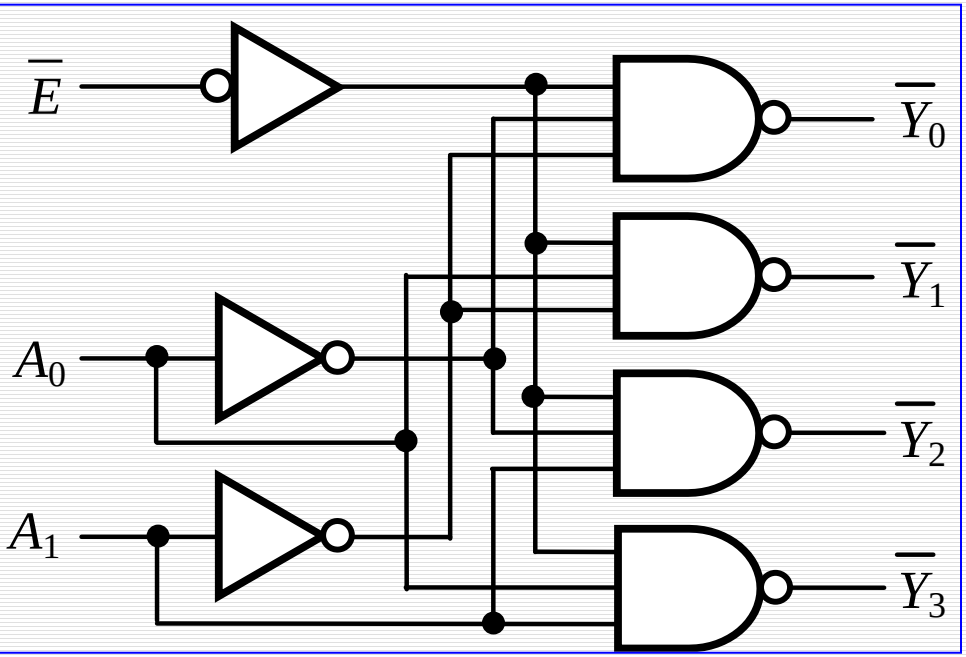
(1) 二进制译码器



设输入端的个数为n，输出端的个数为M，则有：$M=2^n$

使能端为有效电平时，对应每一组输入代码，只有一个输出端为有效电平，其余输出端均为相反电平

- **2线－4线译码器的逻辑电路(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

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

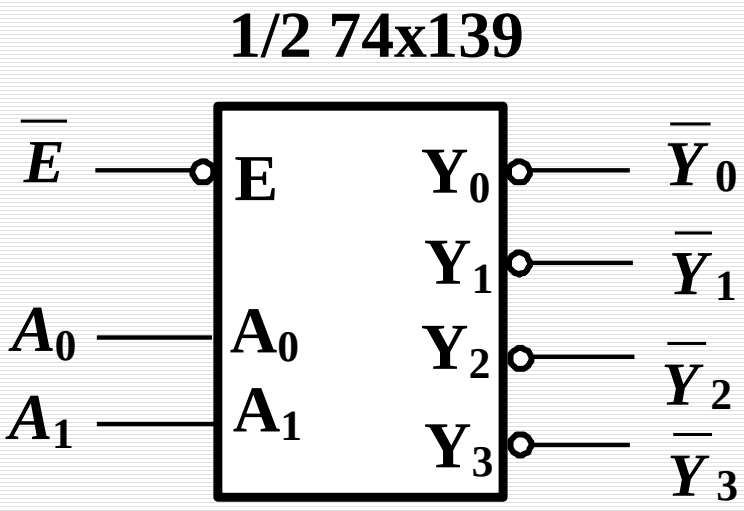
$$\overline{Y}_1 = \overline{\overline{E}\,\overline{A_1}\,A_0}$$

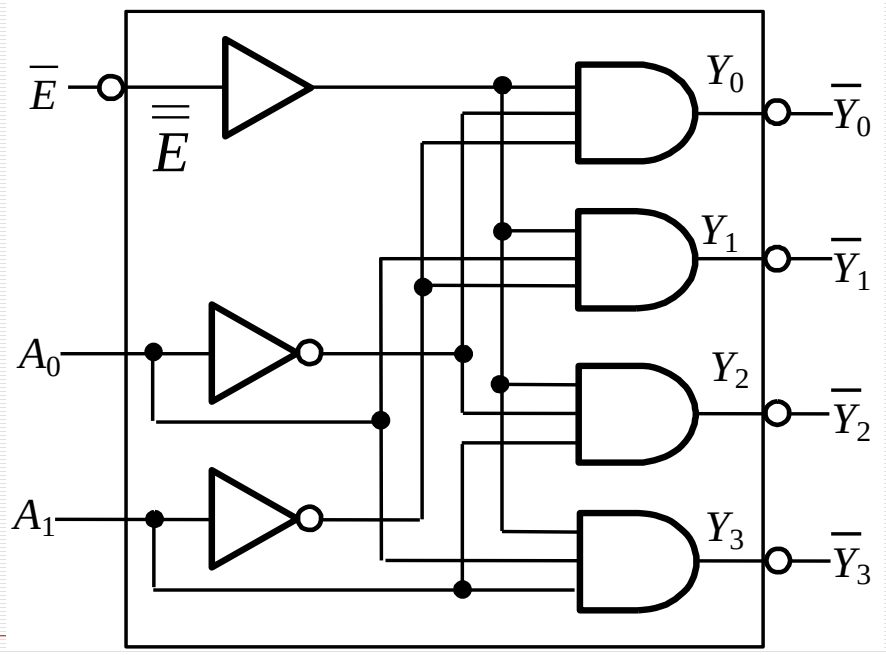
$$\overline{Y}_2 = \overline{\overline{E}\,A_1\,\overline{A_0}}$$

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

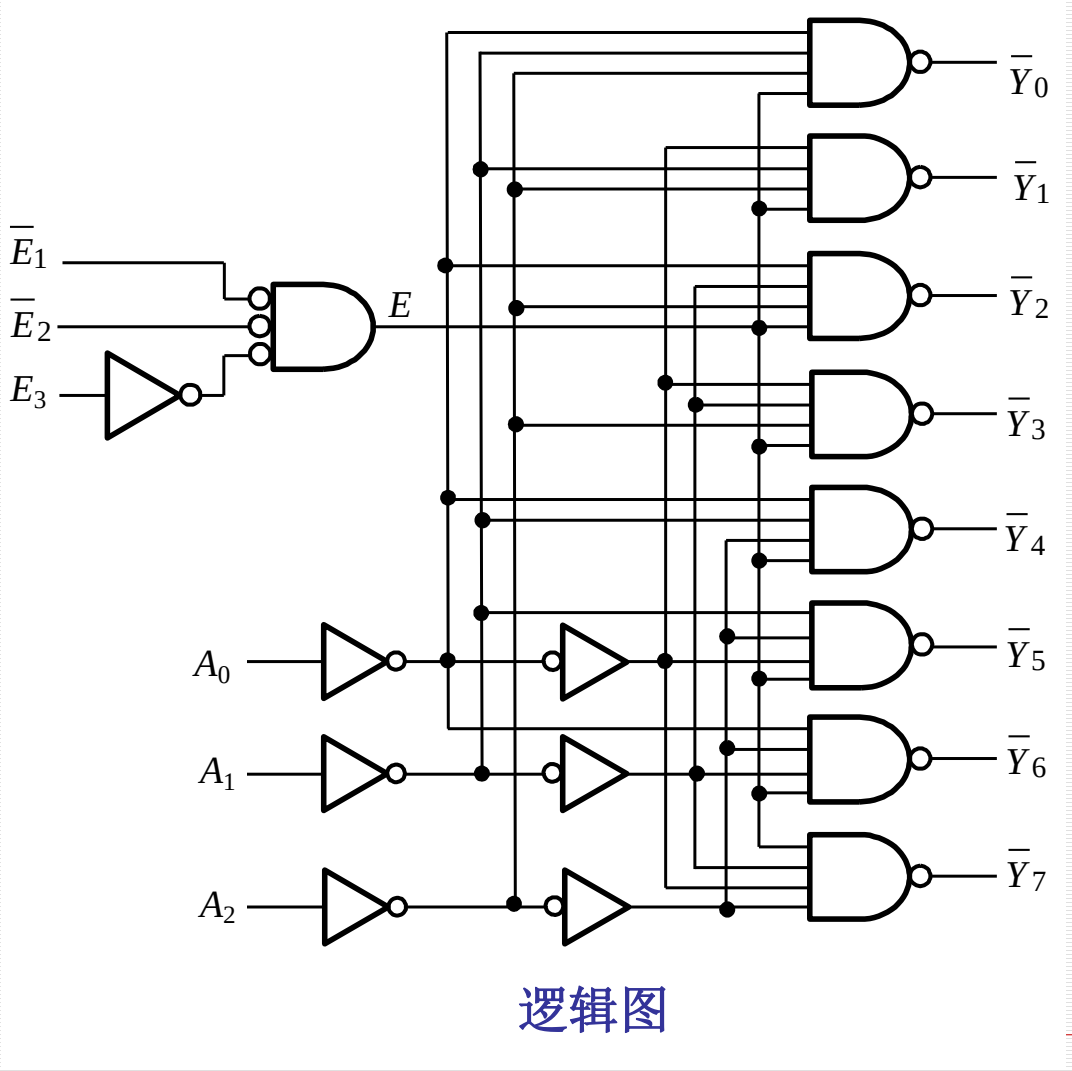
(a) 2线-4线译码器（74HC139）　----逻辑符号说明

逻辑符号框外部的符号，表示外部输入或输出信号名称，字母上面的“—”号说明该输入或输出是低电平有效。符号框内部的输入、输出变量表示其内部的逻辑关系。在推导表达式的过程中，如果低有效的输入或输出变量上面的“—”号参与运算，则在画逻辑图或验证真值表时，注意将其还原为低有效符号形式

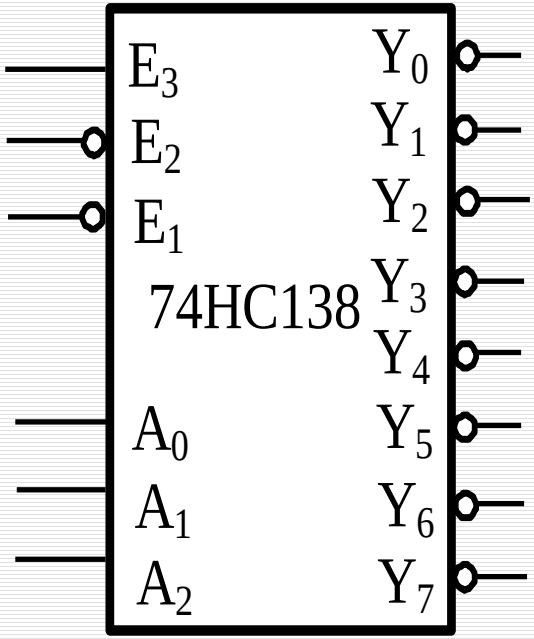




(b) 3线-8线译码器（74HC138）



逻辑图



逻辑符号

3线-8线译码器（74HC138）功能表

输入						输出							
E_3	$\overline{E}_2$	$\overline{E}_1$	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
×	1	×	×	×	×	1	1	1	1	1	1	1	1
×	X	1	×	×	×	1	1	1	1	1	1	1	1
0	×	×	×	×	×	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

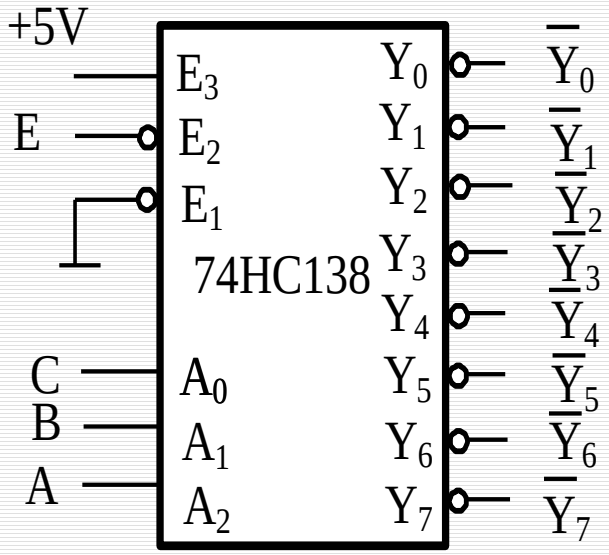
$$\overline{Y}_0 = \overline{\overline{A}_2 \cdot \overline{A}_1 \cdot \overline{A}_0} \quad \overline{Y}_1 = \overline{\overline{A}_2 \cdot \overline{A}_1 \cdot A_0} \quad \overline{Y}_2 = \overline{\overline{A}_2 \cdot A_1 \cdot \overline{A}_0} \quad \overline{Y}_3 = \overline{\overline{A}_2 \cdot A_1 \cdot A_0}$$

$$\overline{Y}_4 = \overline{A_2 \cdot \overline{A}_1 \cdot \overline{A}_0} \quad \overline{Y}_5 = \overline{A_2 \cdot \overline{A}_1 \cdot A_0} \quad \overline{Y}_6 = \overline{A_2 \cdot A_1 \cdot \overline{A}_0} \quad \overline{Y}_7 = \overline{A_2 \cdot A_1 \cdot A_0}$$

输入						输出							
E_3	$\overline{E}_2$	$\overline{E}_1$	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
×	1	×	×	×	×	1	1	1	1	1	1	1	1
×	X	1	×	×	×	1	1	1	1	1	1	1	1
0	×	×	×	×	×	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

译码器的输出包含了输入变量的所有最小项。

1、用译码器实现逻辑函数。



当$E_3=1$，$E_2=E_1=0$时

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

$$\overline{Y}_1=\overline{\cdot A\overline{B}\overline{C}}=\overline{m}_1$$

$$\overline{Y}_2=\overline{\cdot\overline{A}B\overline{C}}=\overline{m}_2$$

$$\vdots$$

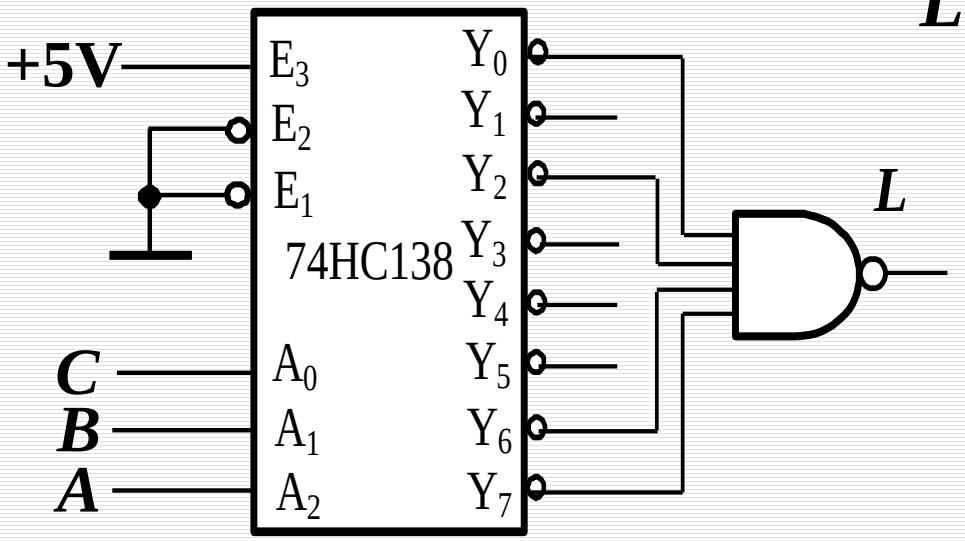
$$\overline{Y}_7=\overline{A\cdot B\cdot C}=\overline{m}_7$$

3线–8线译码器的Y_0～$\overline{Y}_7$含三变量函数的全部最小项。

基于这一点用该器件能够方便地实现三变量逻辑函数。

用一片74HC138实现函数 $L=\overline{A}\,\overline{C}+AB$

首先将函数式变换为最小项之和的形式



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

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

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

$$=\overline{\overline{Y}_0\cdot\overline{Y}_2\cdot\overline{Y}_6\cdot\overline{Y}_7}$$

在译码器的输出端加一个与非门，即可实现给定的组合逻辑函数.

- 以上为用3输入的74HC138实现3变量的逻辑函数，思考：如何用74HC138实现3个以上变量的逻辑函数？

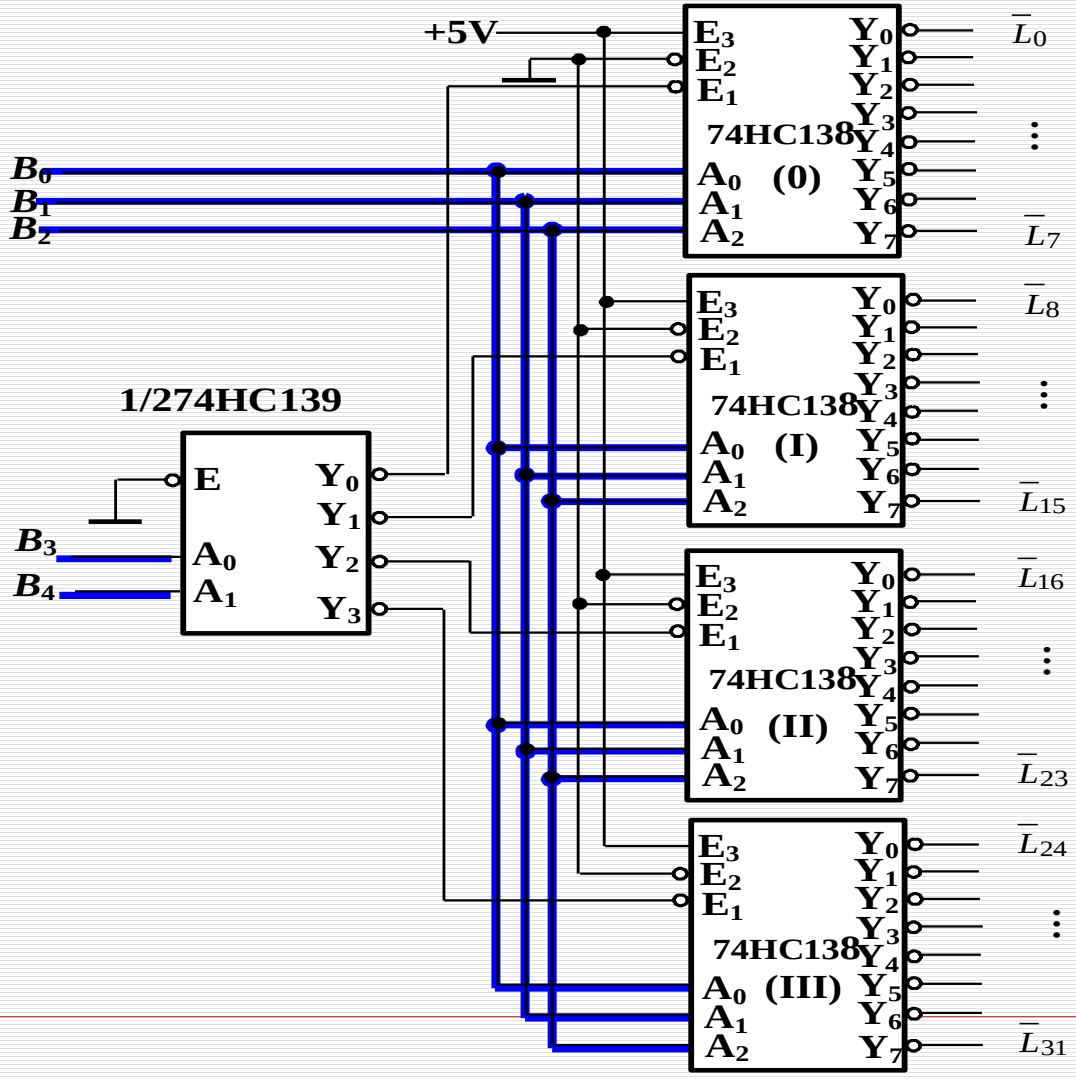
- 例如：试用一片74HC138实现函数

$$F(A,B,C,D)=AB\overline{C}+ACD$$

2、译码器的扩展

- 利用2线-4线译码器和3线-8线译码器，可以扩展构成：4线-16线译码器、5线-32线译码器、6线-64线译码器

用74X139和74X138构成5线-32线译码器



原理：对应B_1B_0的四种状态：00、01、10、11，分别设置74HC138片(0)、(1)、(2)、(3)为译码状态（有一个输出为0，其余输出全部为1），其余3片为禁止译码状态（全部输出为1）；

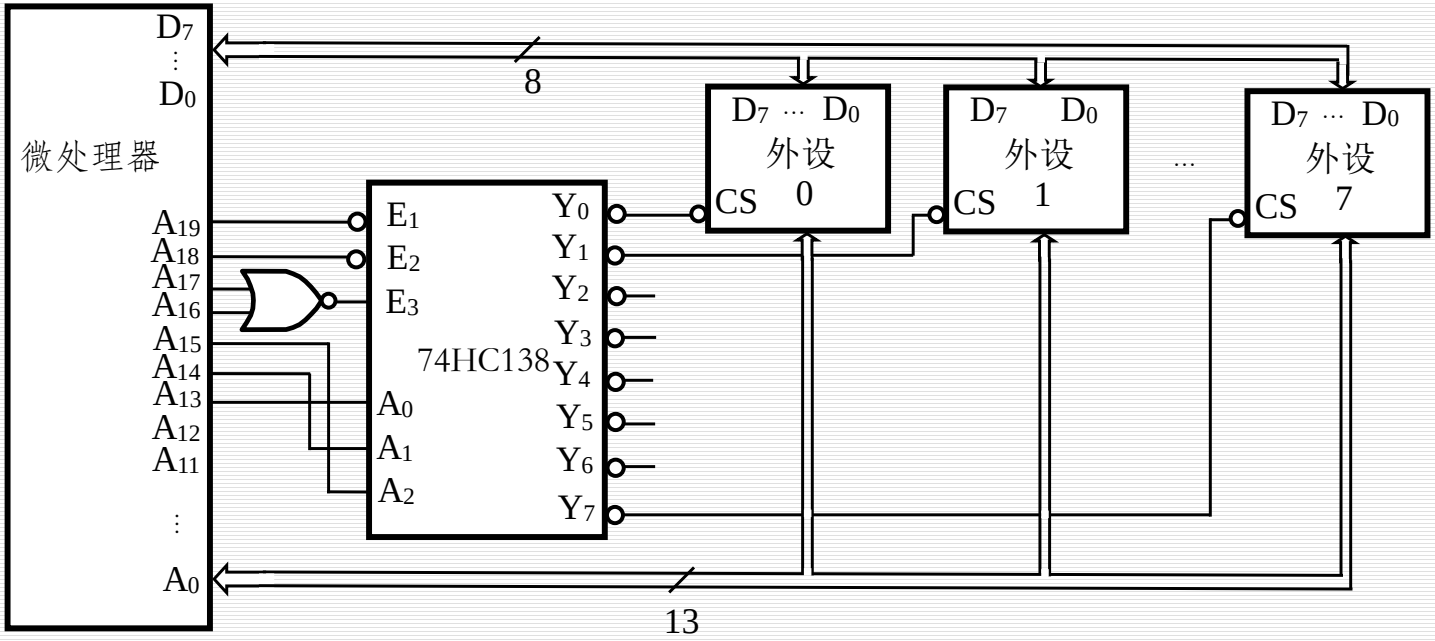
接法：

1.五位二进制码的低三位$B_2B_1B_0$分别与4片74HC138的3个地址输入端并接；

2.高两位B_4B_3接入74HC139的两个地址输入端，而将其四个低有效输出分别接入4片74HC138的低使能输入端，使得它们在B_4B_3的作用下轮流工作。

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。



$$\overline{CS}_0 = \overline{Y}_0 = \overline{\overline{A}_{19}\,\overline{A}_{18}\,\overline{(A_{17}+A_{16})}\;\overline{A}_{15}\,\overline{A}_{14}\,\overline{A}_{13}}$$

$$= A_{19} + A_{18} + A_{17} + A_{16} + A_{15} + A_{14} + A_{13}$$

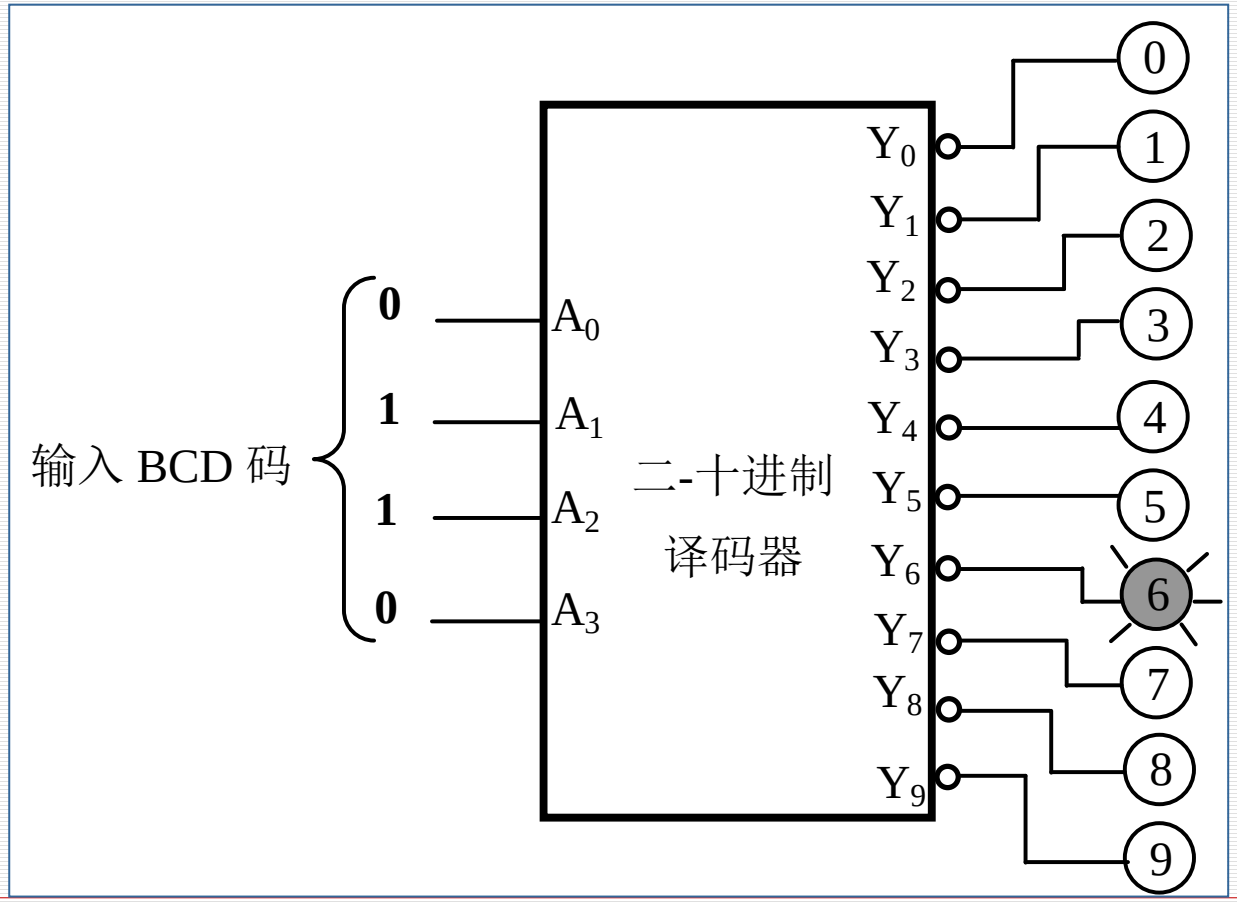
(2) 二–十进制译码器的真值表

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

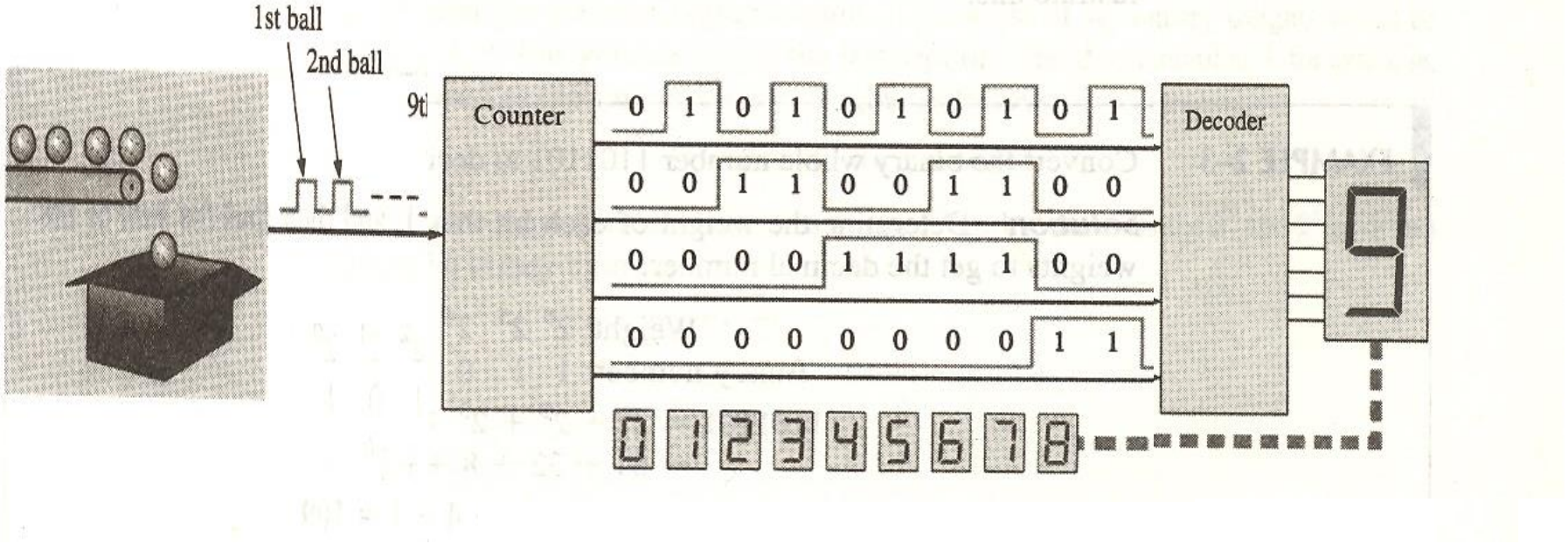
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0

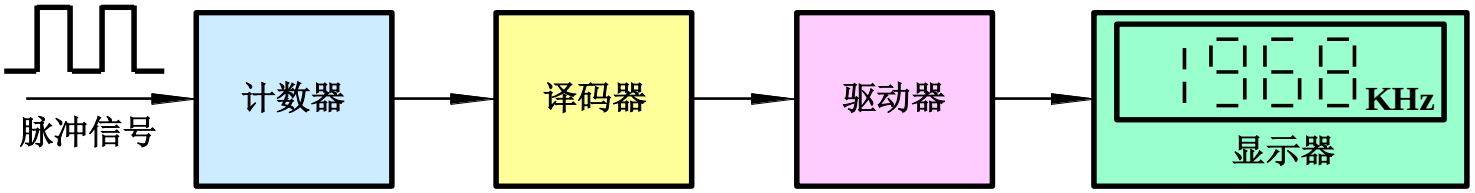
二–十进制译码器应用电路

功能：将8421BCD码译成为10个状态输出。



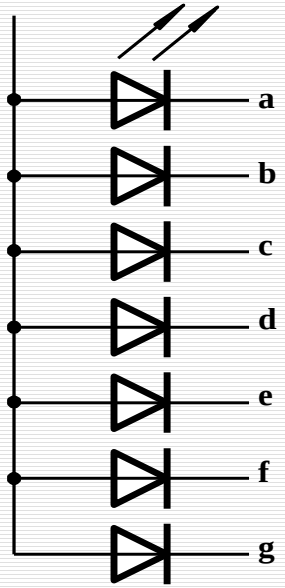
(3) 显示译码器


1st ball
2nd ball
Counter
Decoder
0 1 0 1 0 1 0 1 0 1
0 0 1 1 0 0 1 1 0 0
0 0 0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0 1 1
0 1 2 3 4 5 6 7 8

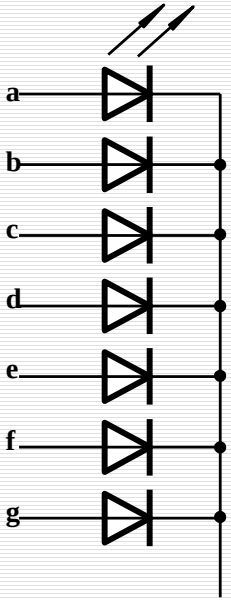

脉冲信号
计数器
译码器
驱动器
KHz
显示器

1. 七段显示译码器（七段数码管）

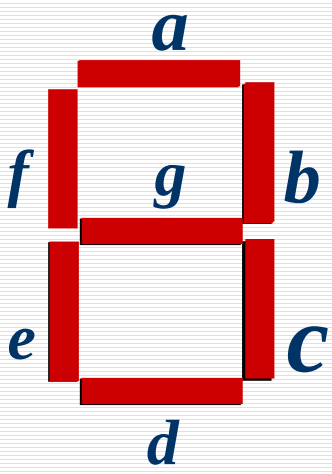
（1）最常用的显示器有：半导体发光二极管和液晶显示器。



共阳极显示器



共阴极显示器

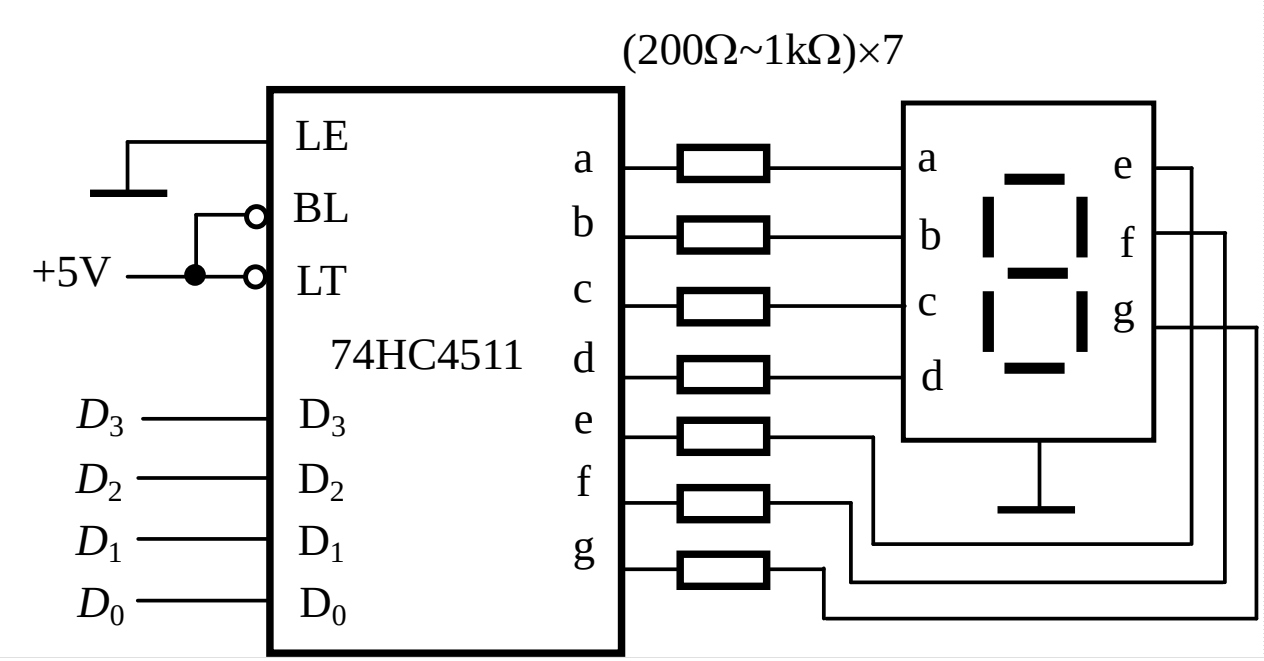


显示器分段布局图

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							
	LE	$\overline{BL}$	$\overline{LT}$	D_3	D_2	D_1	D_0	a	b	c	d	e	f	g	字形
0	0	1	1	0	0	0	0	1	1	1	1	1	1	0	
1	0	1	1	0	0	0	1	0	1	1	0	0	0	0	
2	0	1	1	0	0	1	0	1	1	0	1	1	0	1	
3	0	1	1	0	0	1	1	1	1	1	1	0	0	1	
4	0	1	1	0	1	0	0	0	1	1	0	0	1	1	
5	0	1	1	0	1	0	1	1	0	1	1	0	1	1	
6	0	1	1	0	1	1	0	0	0	1	1	1	1	1	
7	0	1	1	0	1	1	1	1	1	1	0	0	0	0	
8	0	1	1	1	0	0	0	1	1	1	1	1	1	1	
9	0	1	1	1	0	0	1	1	1	1	1	0	1	1	

CMOS七段显示译码器74HC4511功能表(续)

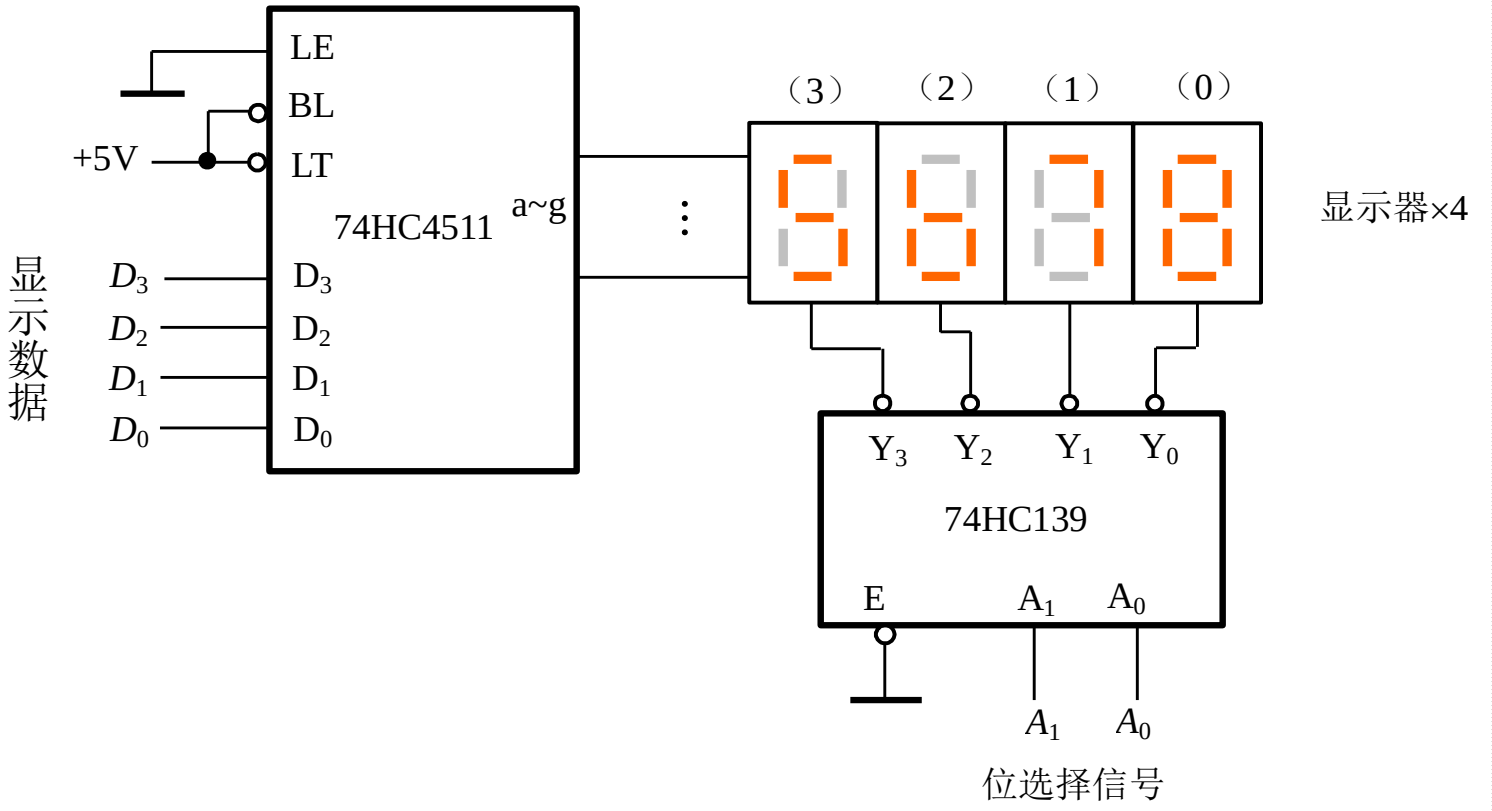
十进制或功能	输入							输出							字形
	LE	$\overline{BL}$	$\overline{LT}$	D_3	D_2	D_1	D_0	a	b	c	d	e	f	g	
10	0	1	1	1	0	1	0	0	0	0	0	0	0	0	熄灭
11	0	1	1	1	0	1	1	0	0	0	0	0	0	0	熄灭
12	0	1	1	1	1	0	0	0	0	0	0	0	0	0	熄灭
13	0	1	1	1	1	0	1	0	0	0				0	熄灭
14	0	1	1	1	1	1	0	0						0	熄灭
15	0	1	1	1	1	1	1	0				0	0	0	熄灭
灯测试	×	×	0	×	×	×	×	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	×	×	×	×				*				*

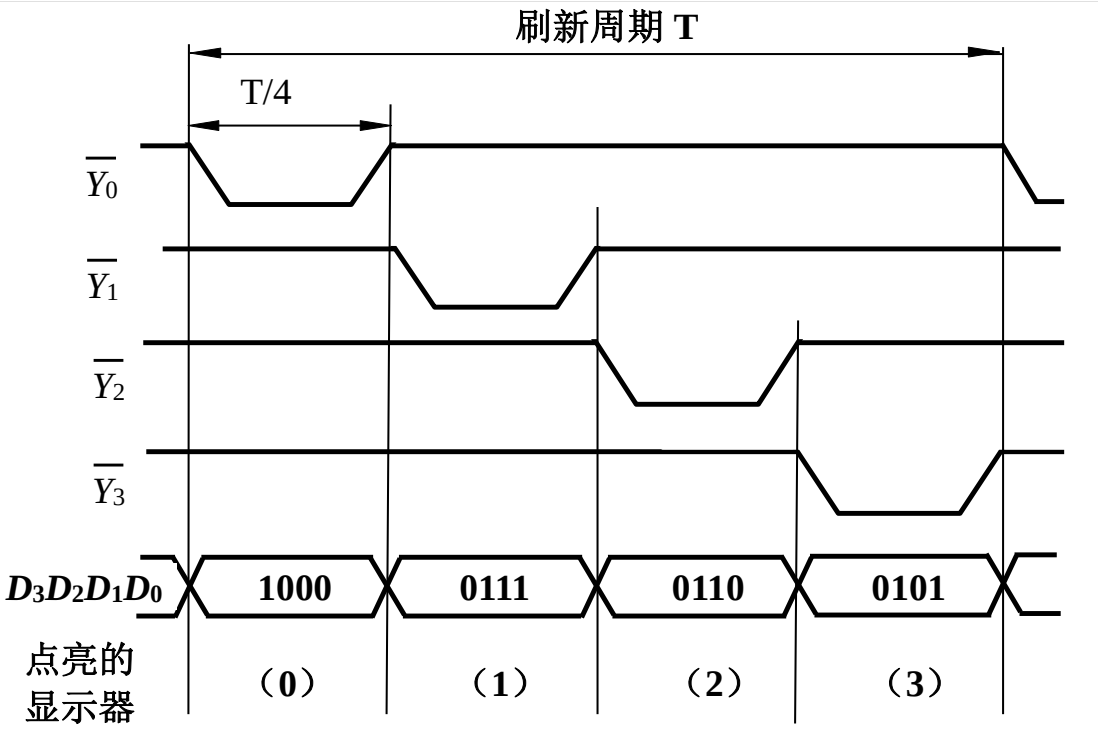
检查译码器本身及显示器各段的好坏

熄灭多余的0

锁存

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。



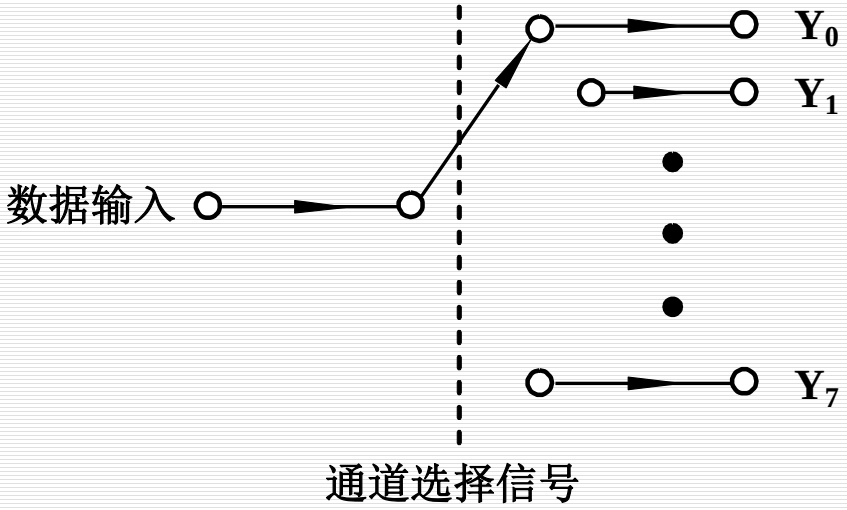


位选择信号A1、A0控制 $\overline{Y_3} \sim \overline{Y_0}$ 依次产生低电平，使4个显示器轮流显示。要显示的数据组依次送到$D_3D_2D_1D_0$ 分别在4个显示器上显示。利用人的视觉暂留时间，当刷新频率达到一定数值（如40Hz，即T=1/40Hz=0.025s）可看到稳定的数字。

$$25\text{Hz} < f_C < 100\text{Hz}$$

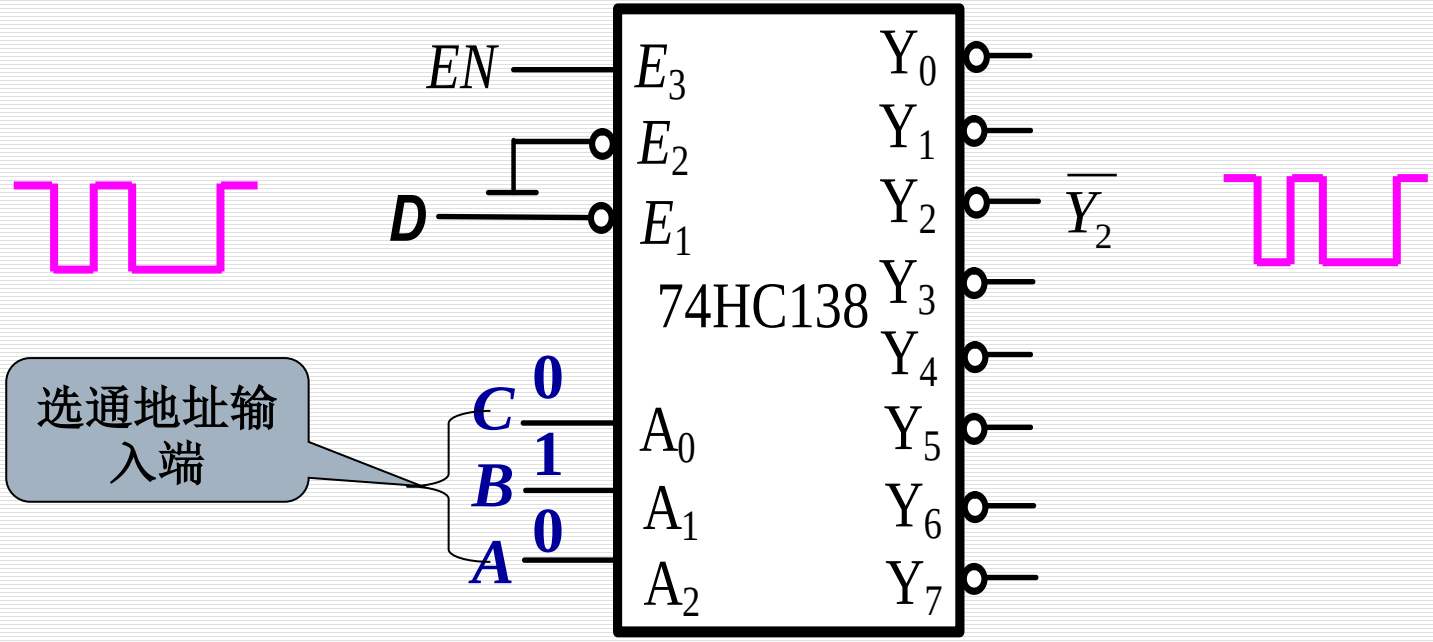
数据分配器

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74HC138译码器实现数据分配器



$$\overline{Y}_2 = \overline{E_3 \cdot \overline{E}_2 \cdot \overline{E}_1 \cdot \overline{A}_2 \cdot A_1 \cdot \overline{A}_0} = \overline{D \cdot \overline{A} \cdot B \cdot \overline{C}} = D$$

即：当**ABC = 010** 时，只有Y_2输出端得到与输入相同的数据波形，其余输出端均为高电平。

因此：通过改变地址输入端的取值，可以将数据送到不同的输出端。

74HC138译码器作为数据分配器时的功能表

输入						输出							
E_3	$\overline{E_2}$	$\overline{E_1}$	A_2	A_1	A_0	$\overline{Y_0}$	$\overline{Y_1}$	$\overline{Y_2}$	$\overline{Y_3}$	$\overline{Y_4}$	$\overline{Y_5}$	$\overline{Y_6}$	$\overline{Y_7}$
0	×	×	×	×	×	1	1	1	1	1	1	1	1
1	D	0	0	0	0	D	1	1	1	1	1	1	1
1	D	0	0	0	1	1	D	1	1	1	1	1	1
1	D	0	0	1	0	1	1	D	1	1	1	1	1
1	D	0	0	1	1	1	1	1	D	1	1	1	1
1	D	0	1	0	0	1	1	1	1	D	1	1	1
1	D	0	1	0	1	1	1	1	1	1	D	1	1
1	D	0	1	1	0	1	1	1	1	1	1	D	1
1	D	0	1	1	1	1	1	1	1	1	1	1	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

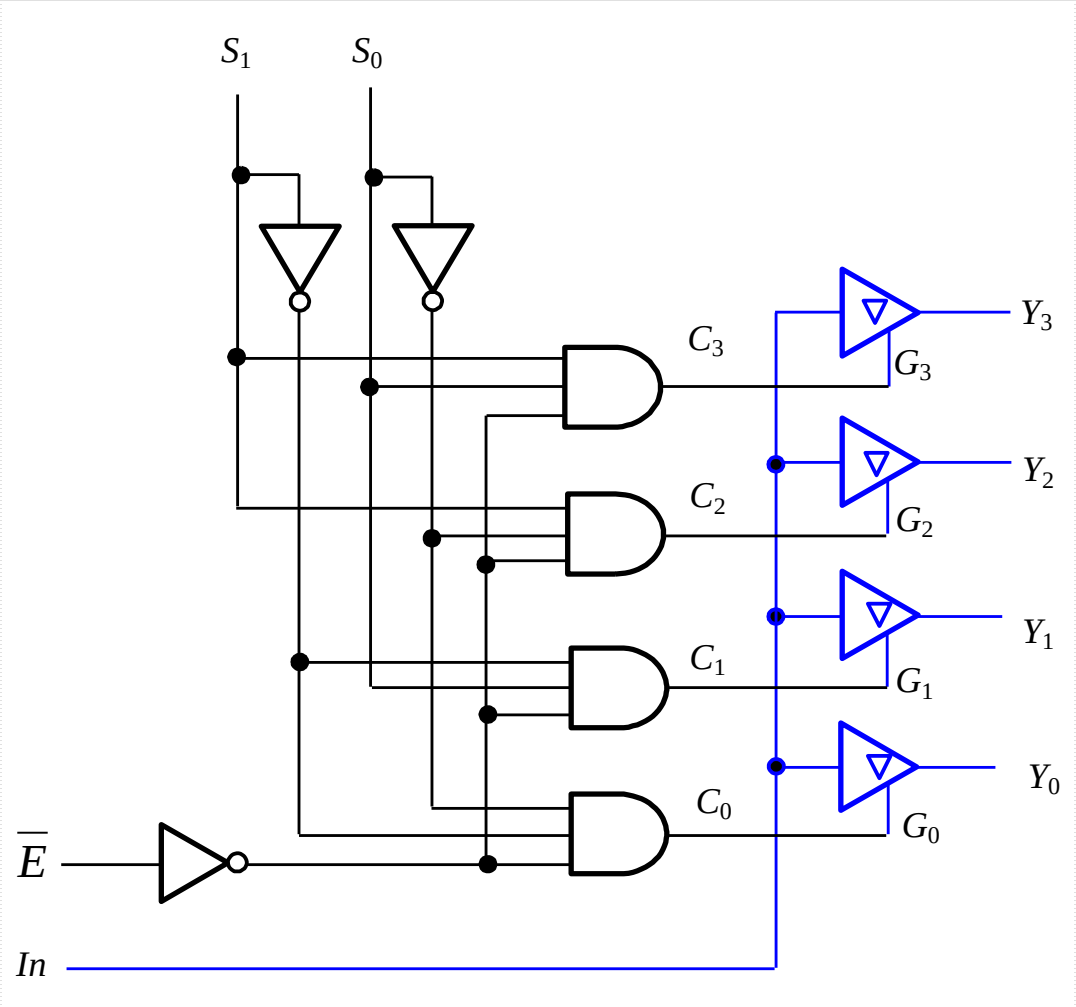
输出端有3种状态（0、1、z），输出级是4个三态门组成。其控制信号由E、S1、S0共同作用产生。

输 入			输 出			
$\overline{E}$	S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	z	z	z	In
0	0	1	z	z	In	z
0	1	0	z	In	z	z
0	1	1	In	z	z	z
1	x	x	z	z	z	z

2. 写出4个三态门控制端的逻辑表达式

$$C_0 = \overline{\overline{E}} \cdot \overline{S_1} \cdot \overline{S_0} \quad C_1 = \overline{\overline{E}} \cdot \overline{S_1} \cdot S_0 \quad C_2 = \overline{\overline{E}} \cdot S_1 \cdot \overline{S_0} \quad C_3 = \overline{\overline{E}} \cdot S_1 \cdot S_0$$

3. 画逻辑电路

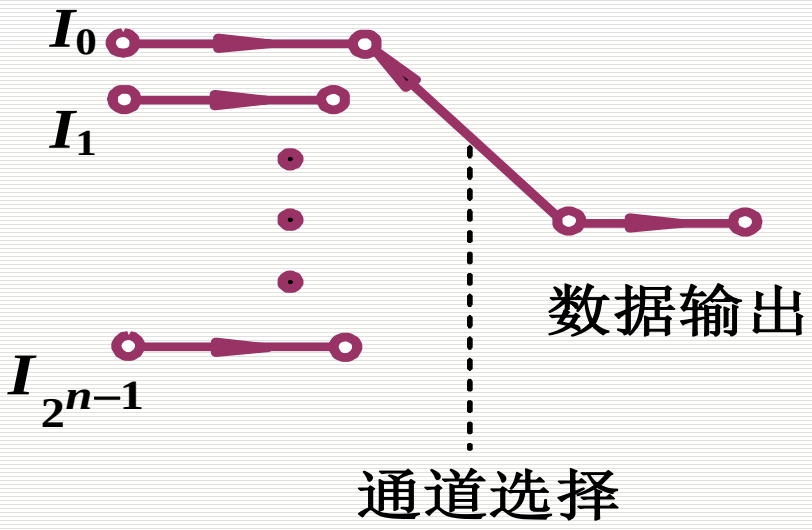


4.4.3 数据选择器

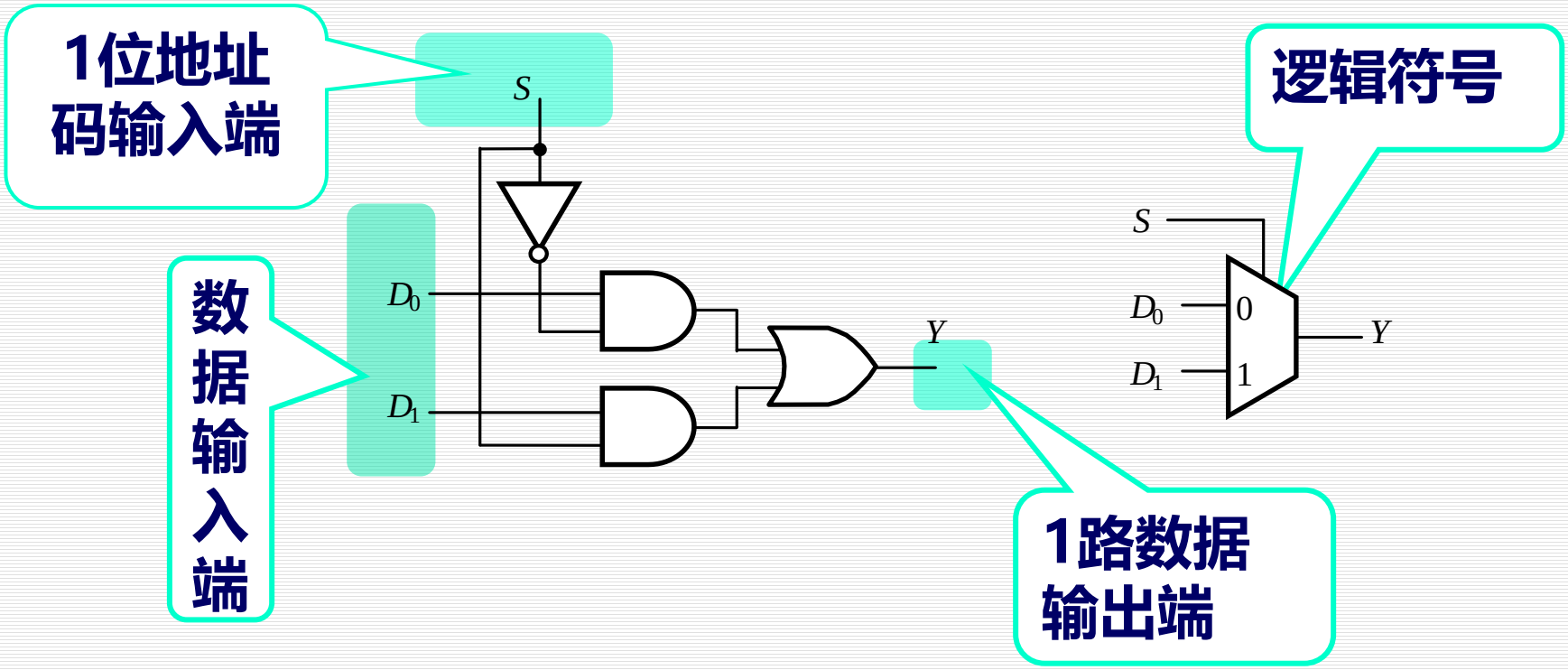
1、数据选择器的定义与功能

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。



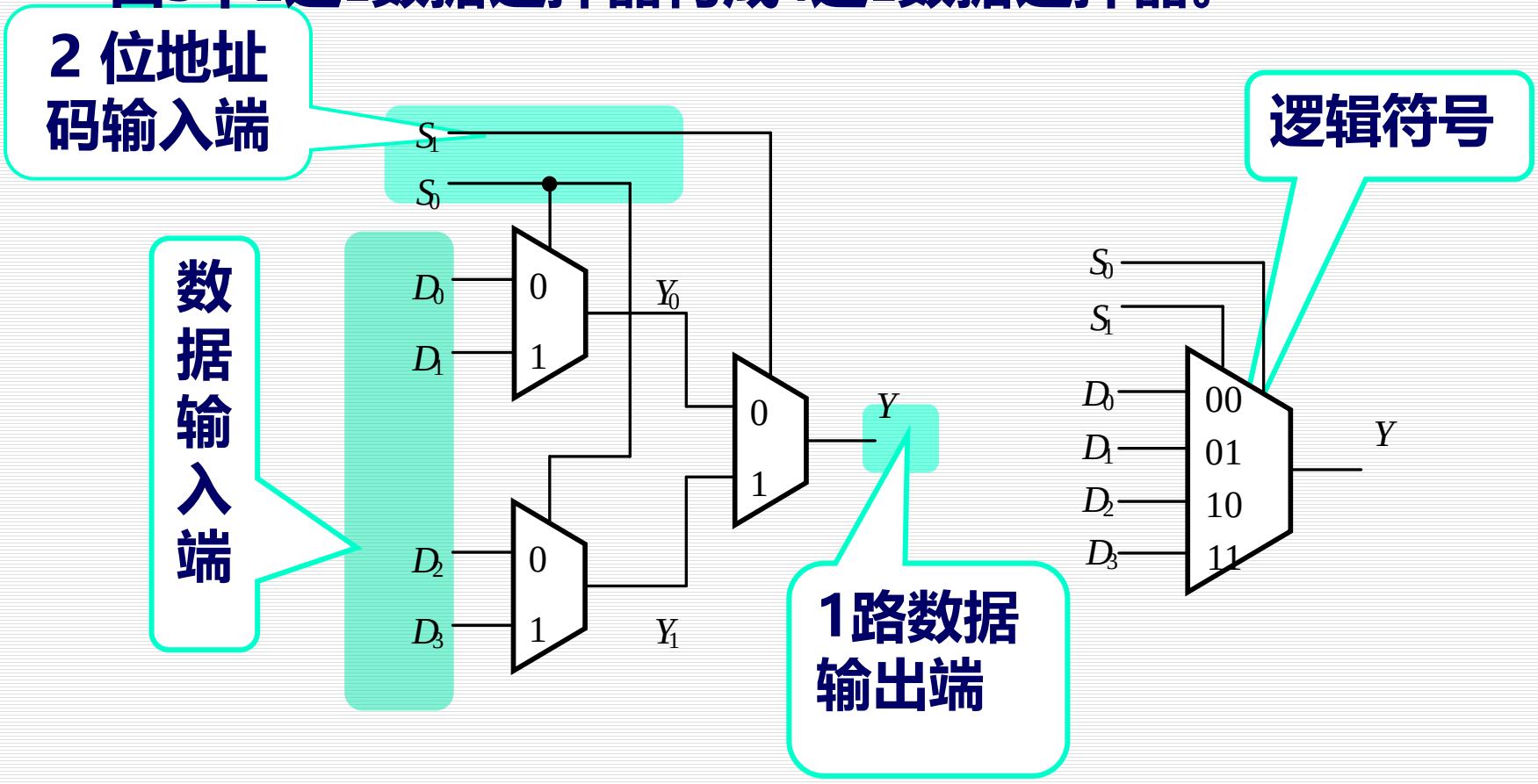
2选1数据选择器



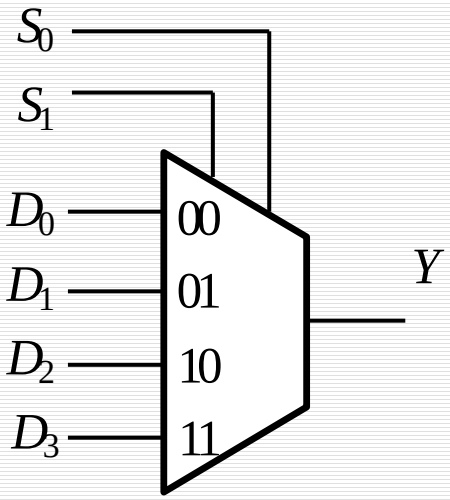
2、4选1数据选择器

（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) 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

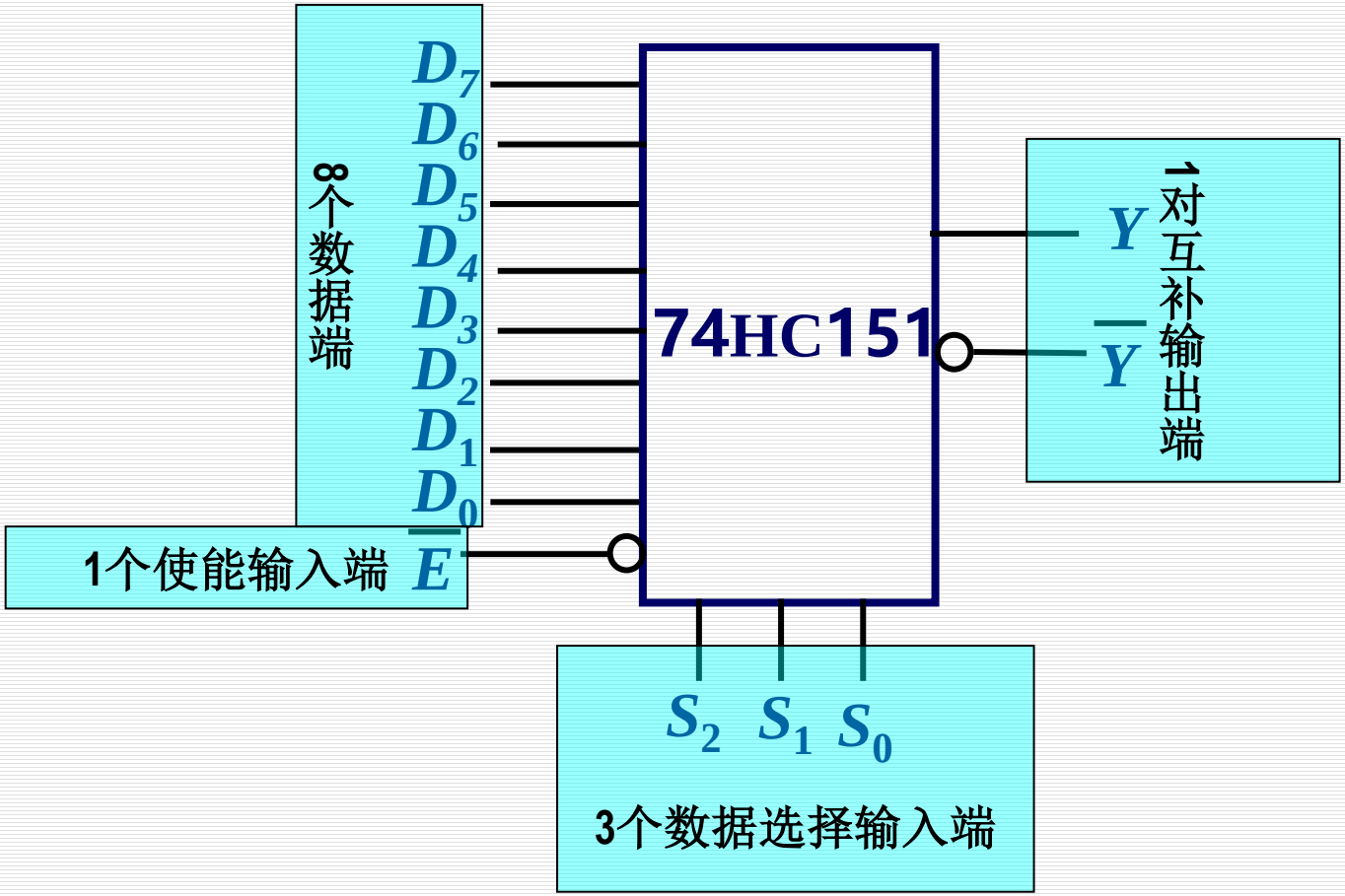
$$Y = \overline{S_1}\,\overline{S_0}D_0 + \overline{S_1}S_0D_1 + S_1\overline{S_0}D_2 + S_1S_0D_3$$

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

同理，可以构成更多输入通道的数据选择器。若选择输入端为n，则可选输入通道数为2^n。

3、集成电路数据选择器

8选1数据选择器74HC151



74HC151逻辑符号

74HC151的功能表

- 当$\overline{E}$=1时，Y=0。
- 当$\overline{E}$=0时

$$Y = \overline{S_2}\,\overline{S_1}\,\overline{S_0}D_0 + \overline{S_2}\,\overline{S_1}S_0D_1 + \overline{S_2}S_1\overline{S_0}D_2$$
$$+ \overline{S_2}S_1S_0D_3 + S_2\overline{S_1}\,\overline{S_0}D_4 + S_2\overline{S_1}S_0D_5$$
$$+ S_2S_1\overline{S_0}D_6 + S_2S_1S_0D_7$$

$$Y = \sum_{i=0}^{7} D_i m_i$$

输入				输出	
使能 $\overline{E}$	选择 S_2	S_1	S_0	Y	$\overline{Y}$
1	X	X	X	L	H
0	0	0	0	D_0	$\overline{D_0}$
0	0	0	1	D_1	$\overline{D_1}$
0	0	1	0	D_2	$\overline{D_2}$
0	0	1	1	D_3	$\overline{D_3}$
0	1	0	0	D_4	$\overline{D_4}$
0	1	0	1	D_5	$\overline{D_5}$
0	1	1	0	D_6	$\overline{D_6}$
0	1	1	1	D_7	$\overline{D_7}$

4、数据选择器的应用

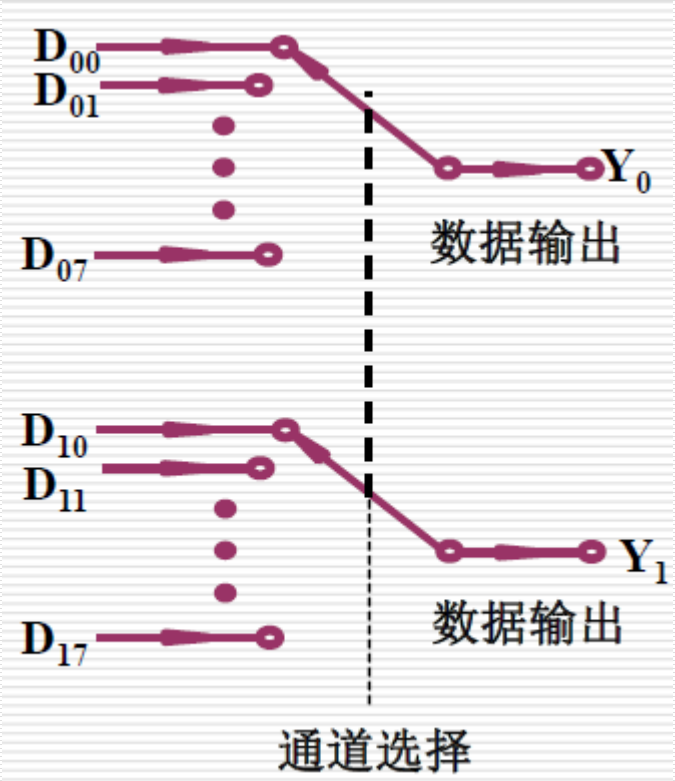
- 数据选择器的扩展：
  - 前面介绍的2选1、4选1、8选1都是一位数据选择器，如何通过扩展实现多位数据选择功能？——位的扩展
  - 如何扩展实现更多数据输入端的数据选择功能（如16选1）？——字的扩展
- 用数据选择器实现逻辑函数
- 用数据选择器构成查找表

4、数据选择器的应用

(1) 位的扩展：用两片74x151组成二位八选一的数据选择器

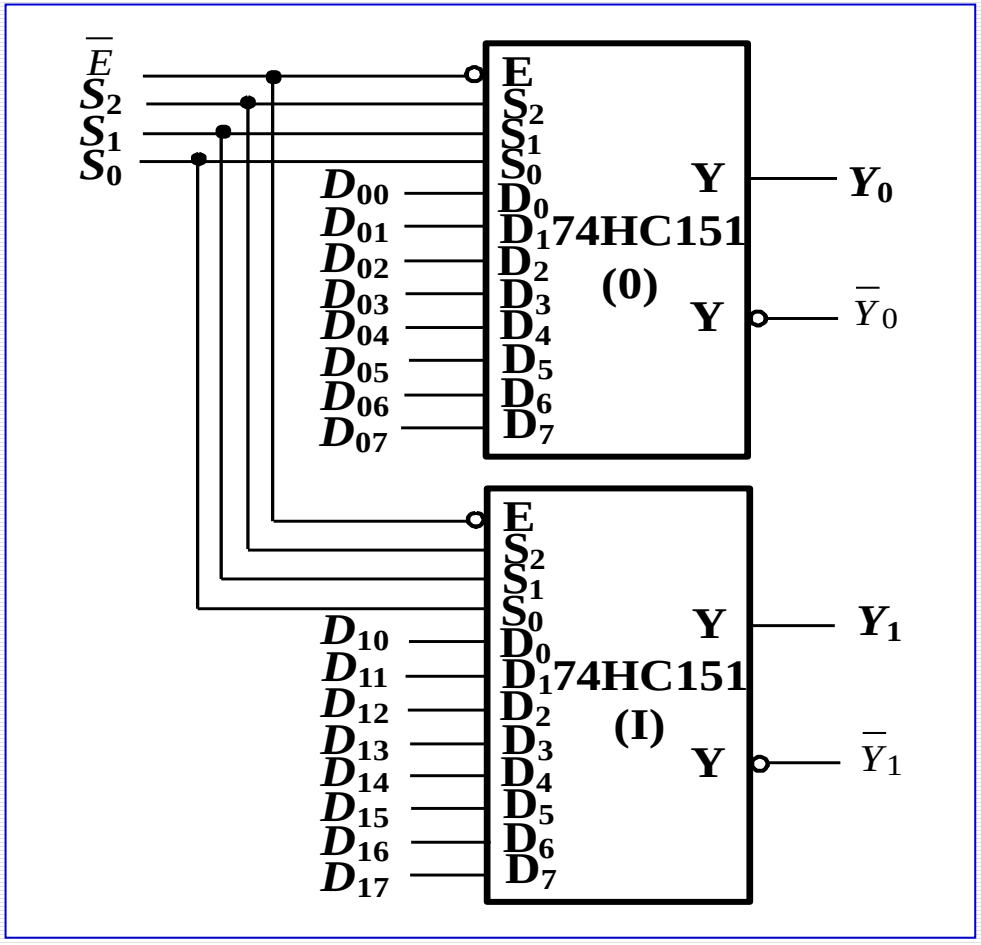
输入：D_{10} D_{00}
D_{11} D_{01}
D_{12} D_{02}
D_{17} D_{07}

输出：$Y_1 Y_0 = D_{10}D_{00}$
$= D_{11}D_{01}$
$= D_{12}D_{02}$
$= D_{17}D_{07}$



4、数据选择器的应用

(1) 位的扩展：用两片74x151组成二位八选一的数据选择器



由2个一位数据选择器并联而成，即：

将使能端连在一起；
将相应的数据选择端连在一起

（2）字的扩展：将两片74x151连接成一个16选1的数据选择器

16选1数据选择器

数据输入端：16路

（D_0~D_{15}）

通道地址码：4位

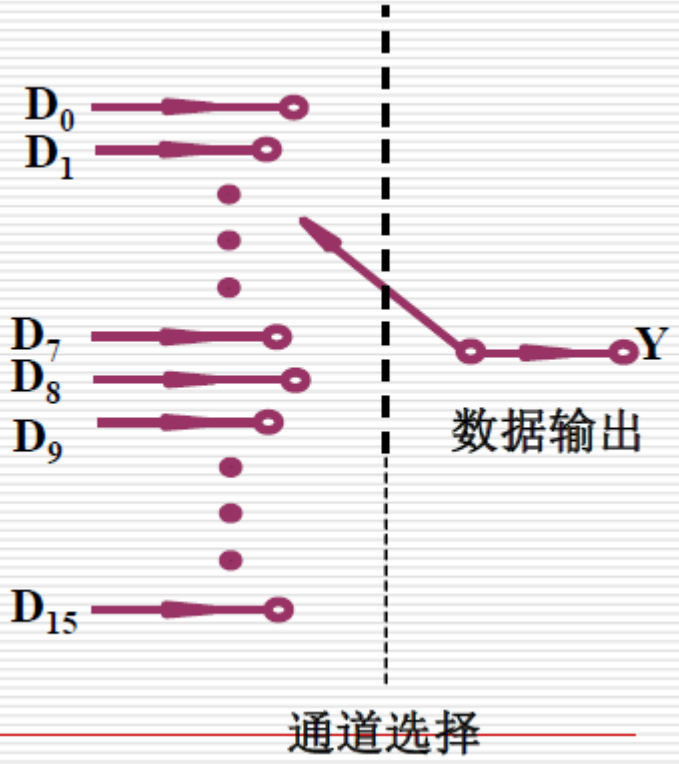
（DCBA）

DCBA=0000~0111

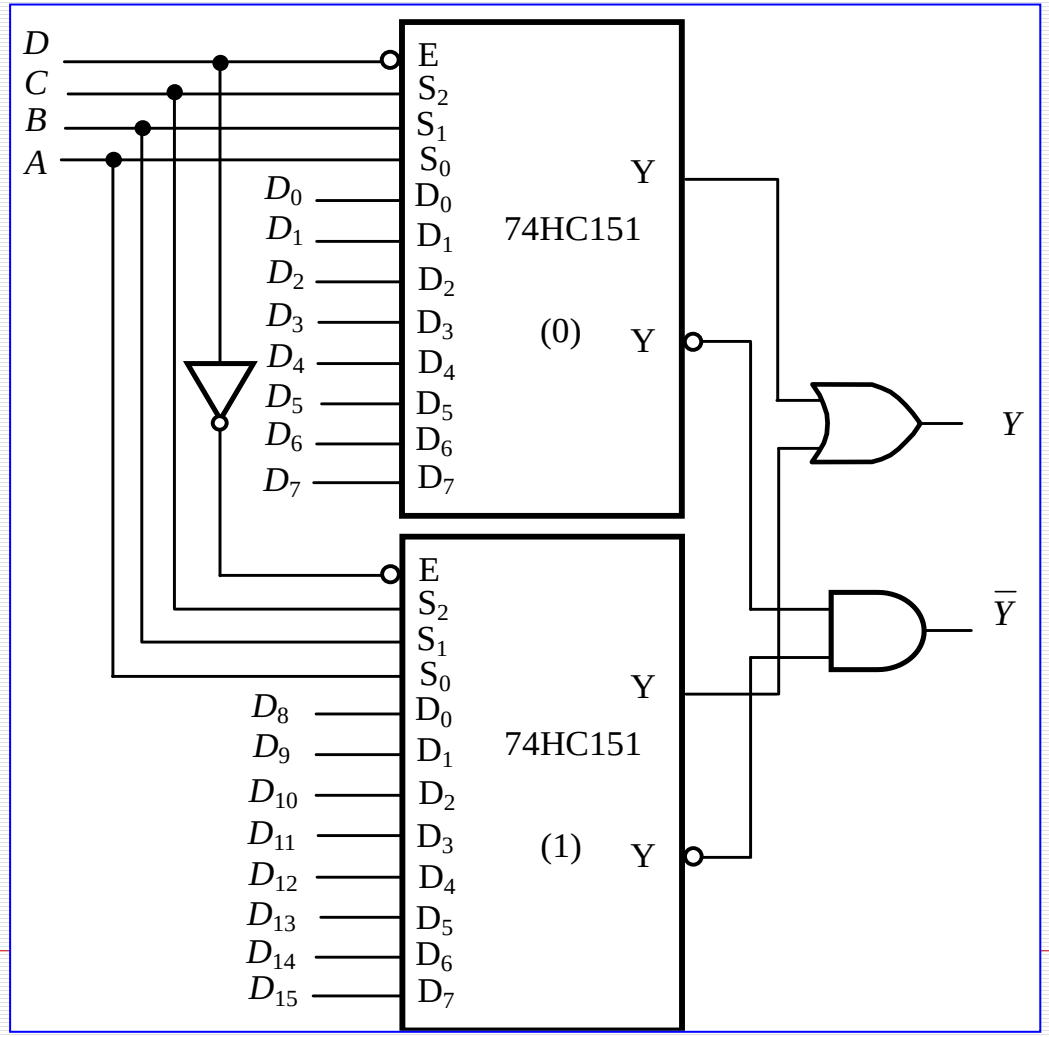
Y=D_0~D_7

DCBA=1000~1111

Y=D_8~D_{15}



（2）字的扩展：将两片74x151连接成一个16选1的数据选择器

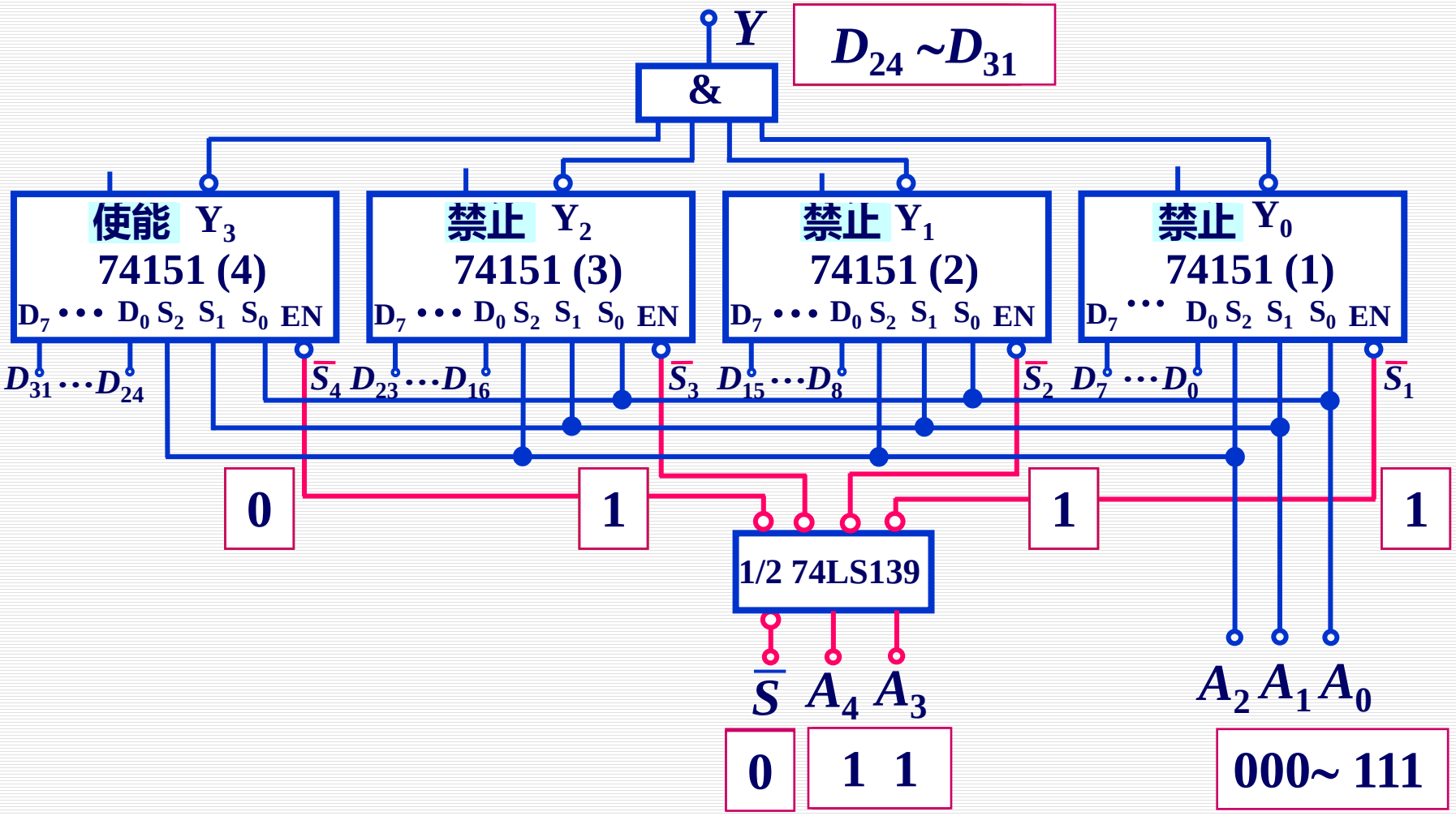


由于16选1数据选择器包含4个数据选择输入端，因此：

将最高位D与其中一片的使能端相连，反相后与另一片使能端相连；

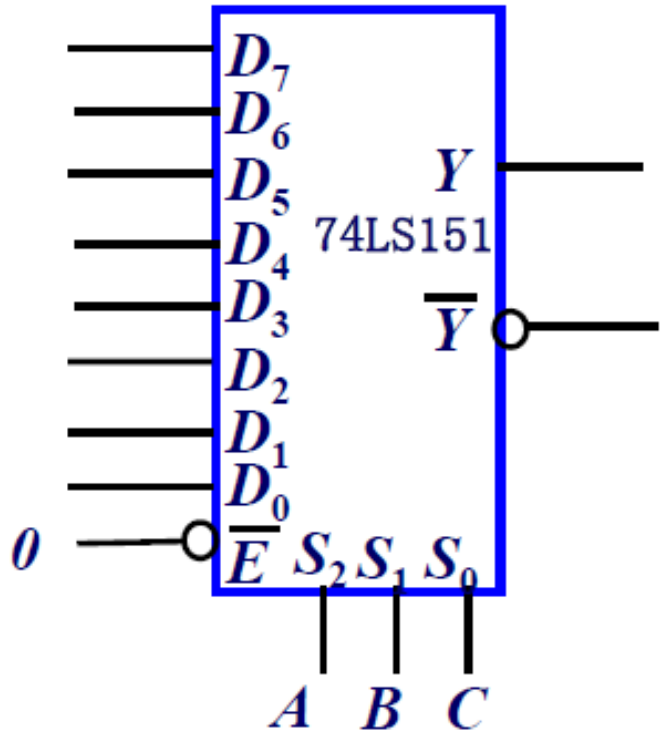
将低三位C、B、A与两片八选一数据选择器对应的数据选择输入端相连

四片 8 选 1（74HC151）→ 32 选 1 数据选择器



（3）用数据选择器实现逻辑函数

以8选1为例进行说明



- 当$\overline{E}$=0时：

$$Y=\sum_{i=0}^{7} D_i m_i$$

- 当 $D_0=D_3=D_5=D_7=0$

$D_1=D_2=D_4=D_6=1$

$$Y=m_1+m_2+m_4+m_6$$

- 当$D_0=D_3=D_5=D_7=1$

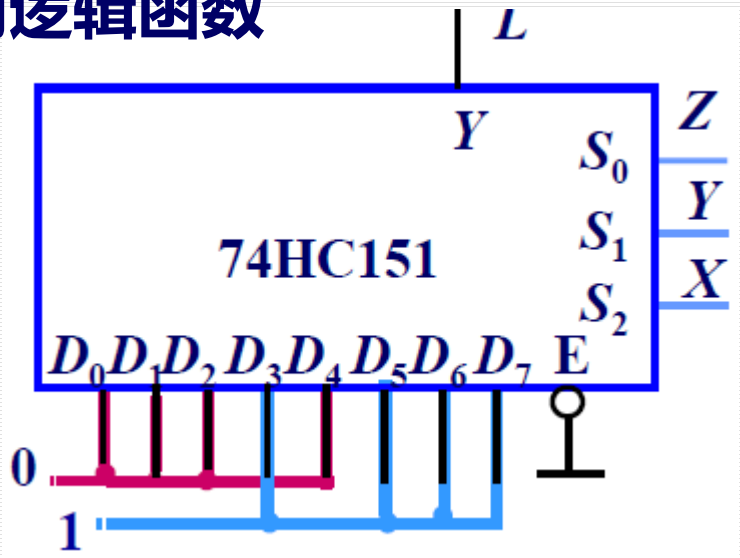
$D_1=D_2=D_4=D_6=0$ 时：

$$Y=m_0+m_3+m_5+m_7$$

控制D_i，就可得到不同的逻辑函数。

例1：用8选1数据选择器实现下列逻辑函数

$$L = \overline{X}YZ + X\overline{Y}Z + XY$$



解：

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

$$Y = m_0D_0 + m_1D_1 + m_2D_2 + m_3D_3 + m_4D_4 + m_5D_5 + m_6D_6 + m_7D_7$$

比较Y和L，当： $D_3 = D_5 = D_6 = D_7 = 1, D_0 = D_1 = D_2 = D_4 = 0,$

时，Y=L

例2：用4选1数据选择器实现下列逻辑函数

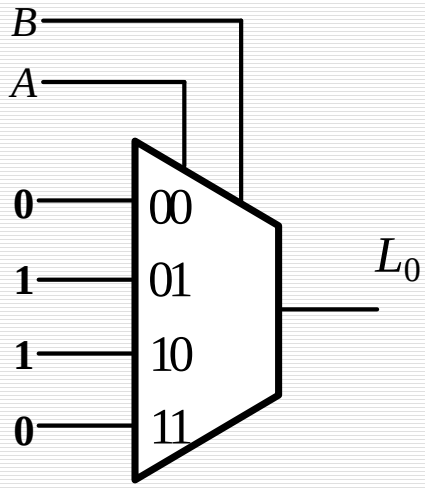
$$L_0 = \overline{A}B + A\overline{B}$$

解：

$$L_0 = \overline{A}B + A\overline{B} = \overline{A}\,\overline{B}\cdot 0 + \overline{A}B\cdot 1 + A\overline{B}\cdot 1 + AB\cdot 0$$

因此，将变量A、B接入4选1数据选择器的两个选择输入端S_1和S_0；

L_0中出现的最小项对应的数据端为1，没有出现的最小项为0。

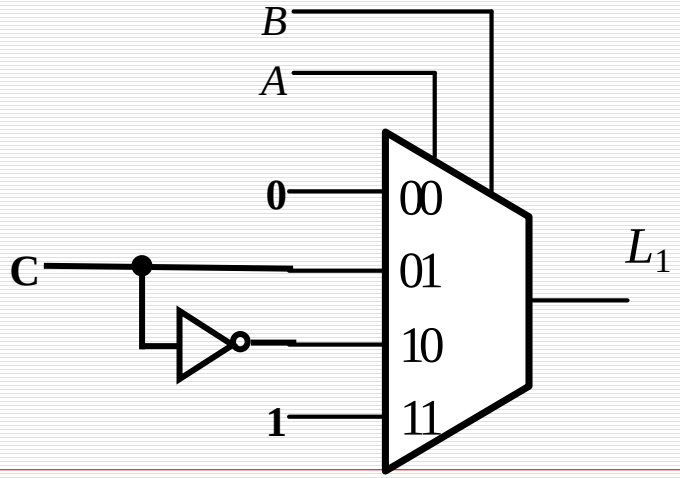


例3：用数据选择器实现下列逻辑函数 $L_0 = \overline{A}B + A\overline{C} + BC$

要求：

- **用4选1数据选择器实现；**
- **用2选1数据选择器实现；**

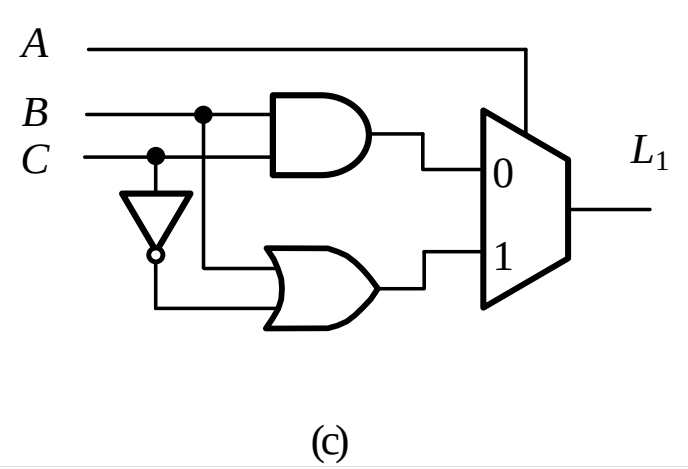
解3a：4选1数据选择器有2个选通端接输入*A*和*B*，表达式有3个变量。因此数据端需要输入1个变量。考察真值表*C*与L_1的关系。



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

例3b. 用2选1数据选择器和必要的逻辑门实现 $L_1 = AB + A\overline{C} + BC$

解3b：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。考察真值表**B**、**C**与L_1的关系。



(c)

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	

总结：利用数据选择器实现函数的一般步骤：

1. 当变量数=选通端数，以8选1数据选择器、函数包含3变量为例:

a、将函数变换成最小项表达式

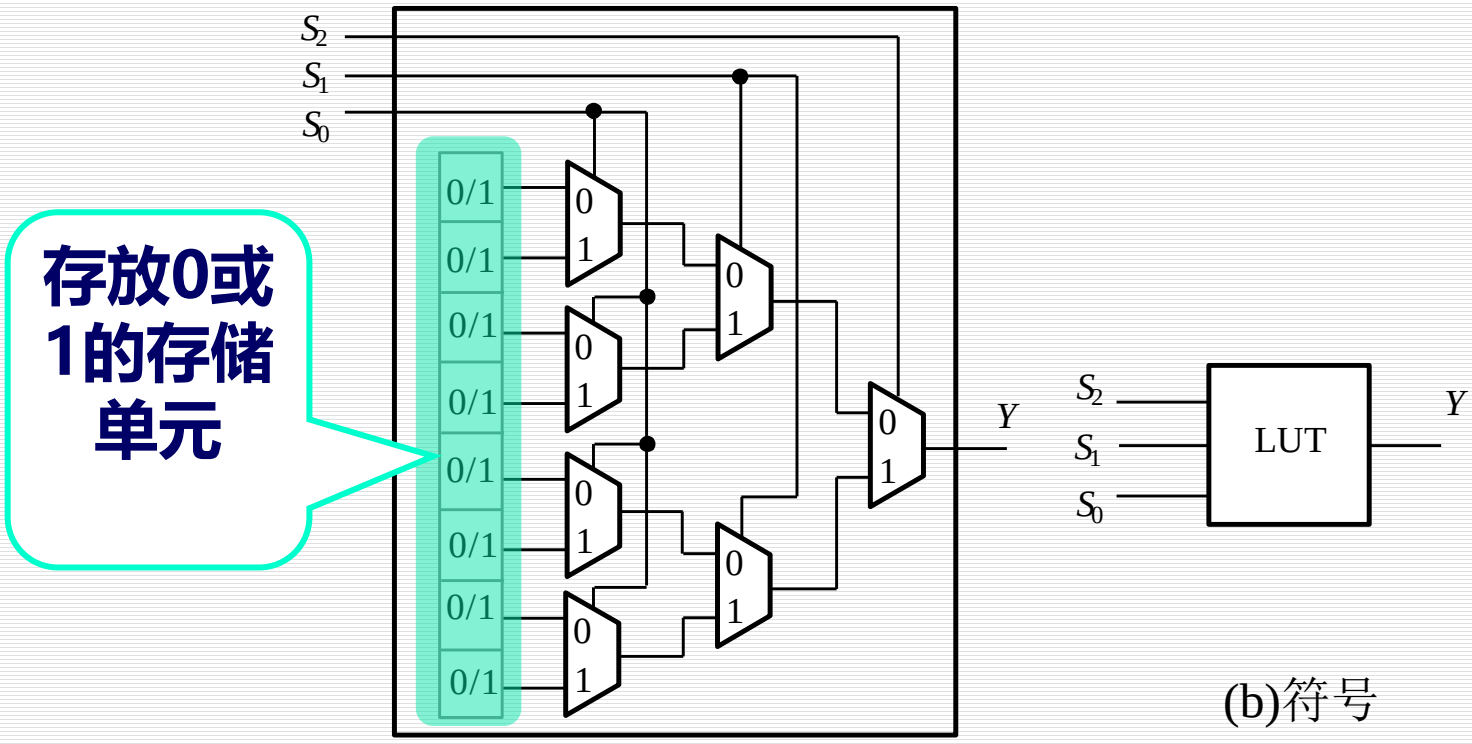
b、地址信号S_2、 S_1 、 S_0 作为函数的输入变量

c、处理数据输入D_0~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。

2.当变量数>选通端数，需要通过考察多余的变量与输出之间的关系，从而确定如何将某些变量接入数据端。

（4）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

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

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。

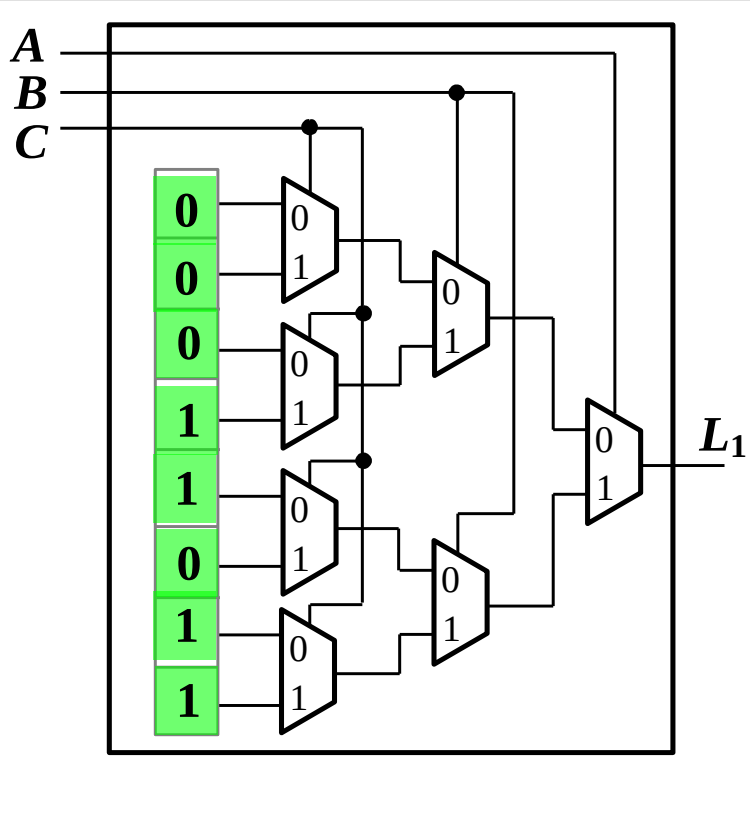
转换为最小项形式：

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

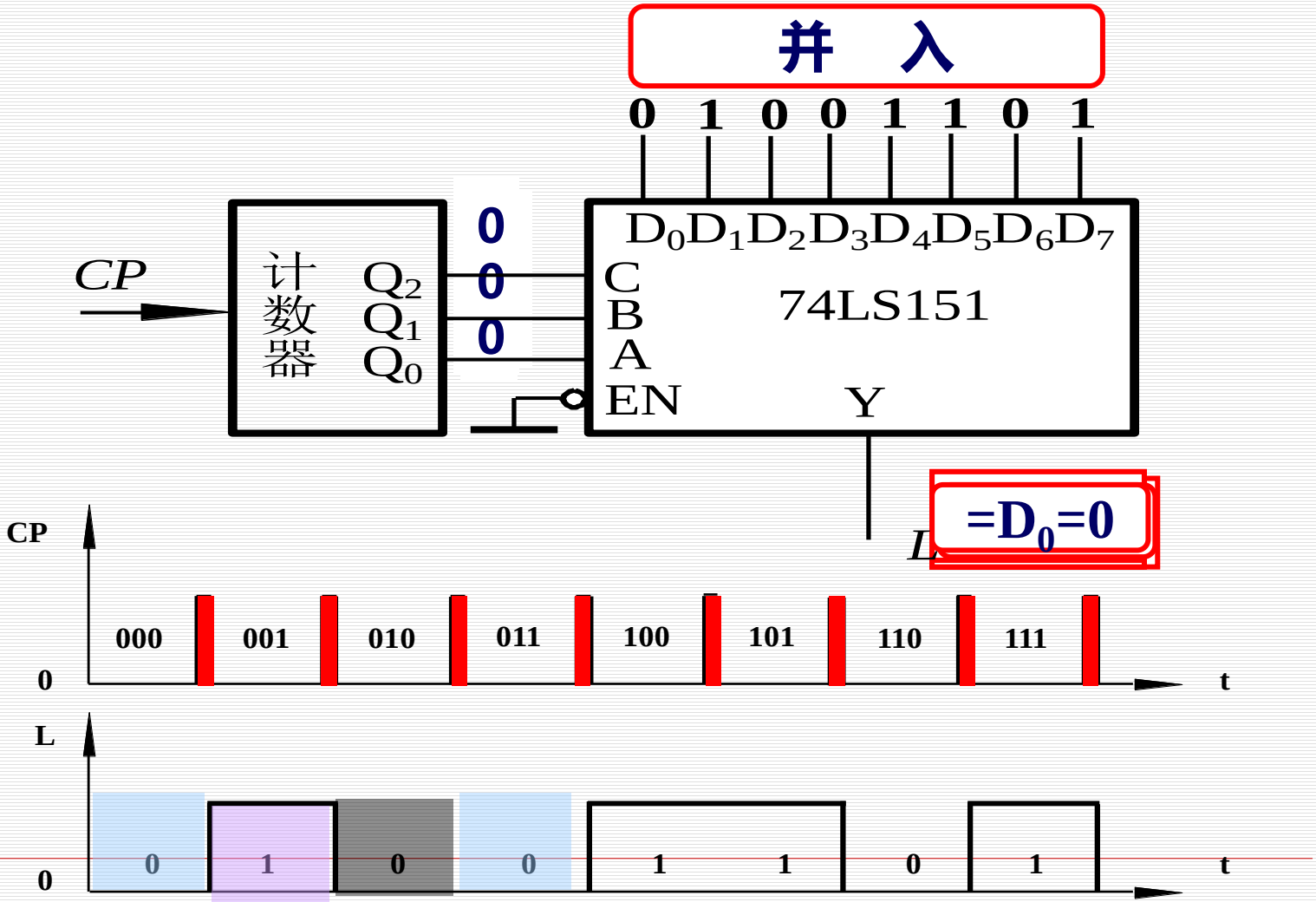
$$= \sum m(3,4,6,7)$$

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

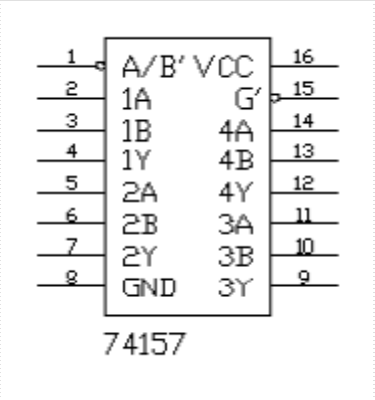
$$D_3 = D_4 = D_6 = D_7 = 1$$



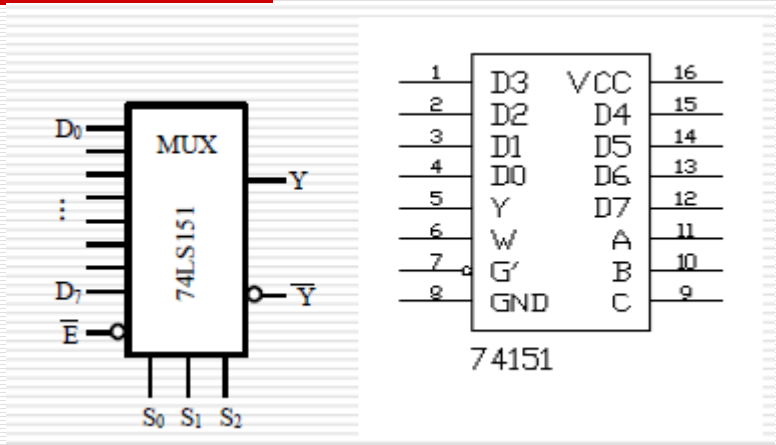
（5）用8选1数据选择器实现并行数据到串行数据的转换



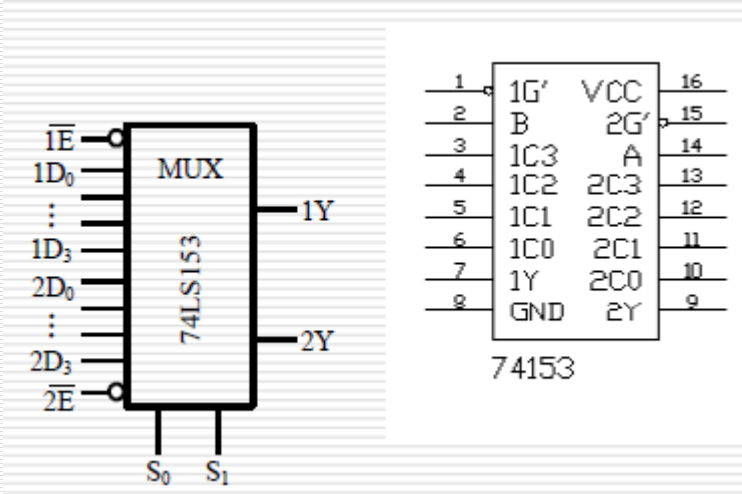
5、其它数据选择器



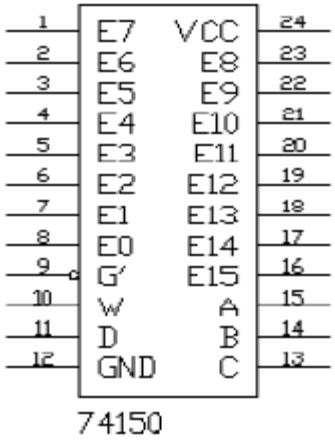
四2选1多路器



8选1多路器



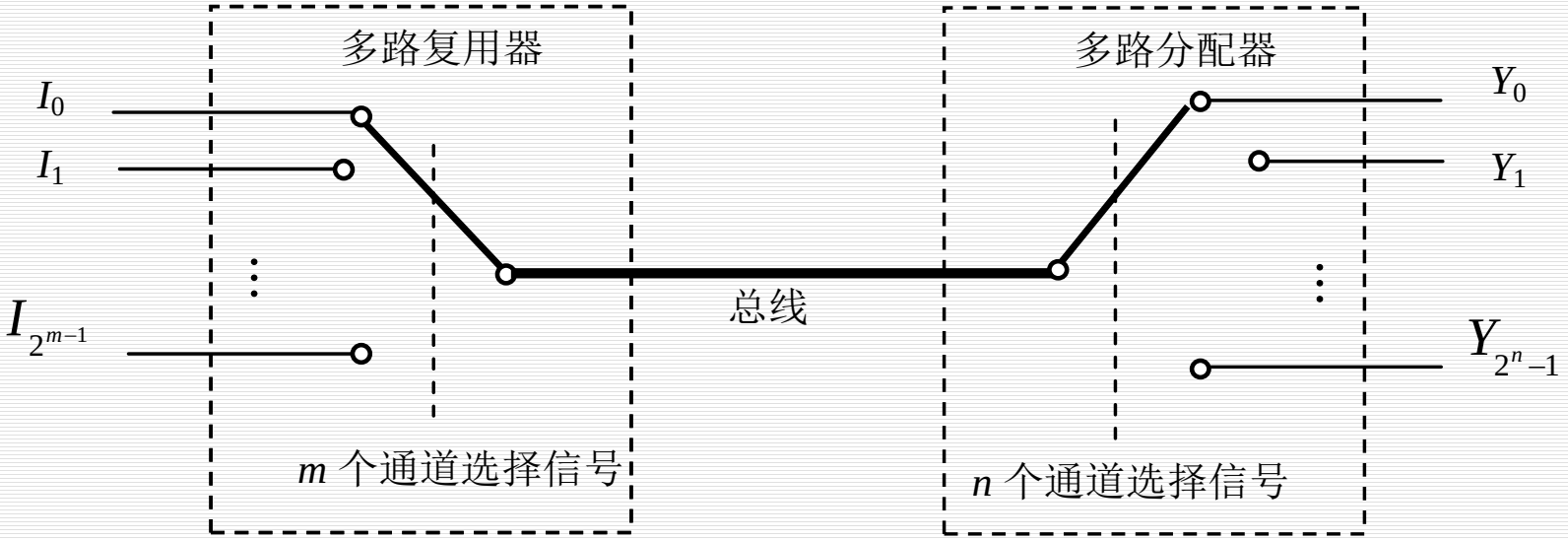
双4选1多路器



16选1多路器

6、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

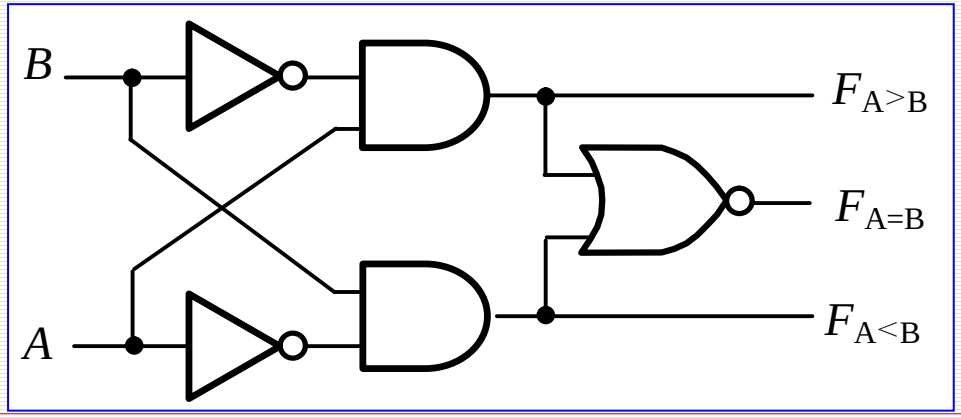
$$F_{A>B} = A\overline{B}$$

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

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

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



4.4.4 数值比较器

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 $A=A_1A_0$ 、$B=B_1B_0$

基本思路：

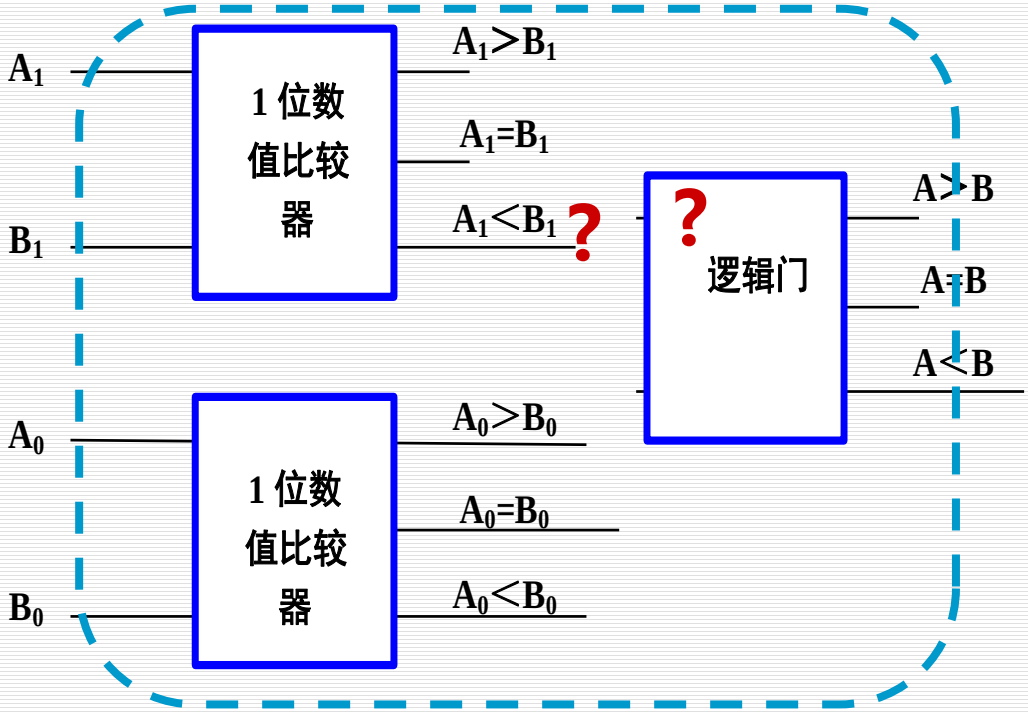
当高位（A_1、B_1）不相等时，无需比较低位（A_0、B_0），高位比较的结果就是两个数的比较结果。

当高位相等时，两数的比较结果由低位比较的结果决定。

4.4.4 数值比较器

2、2 位数值比较器：

如何用1位数值比较器设计两位数值比较器?



真值表

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

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

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

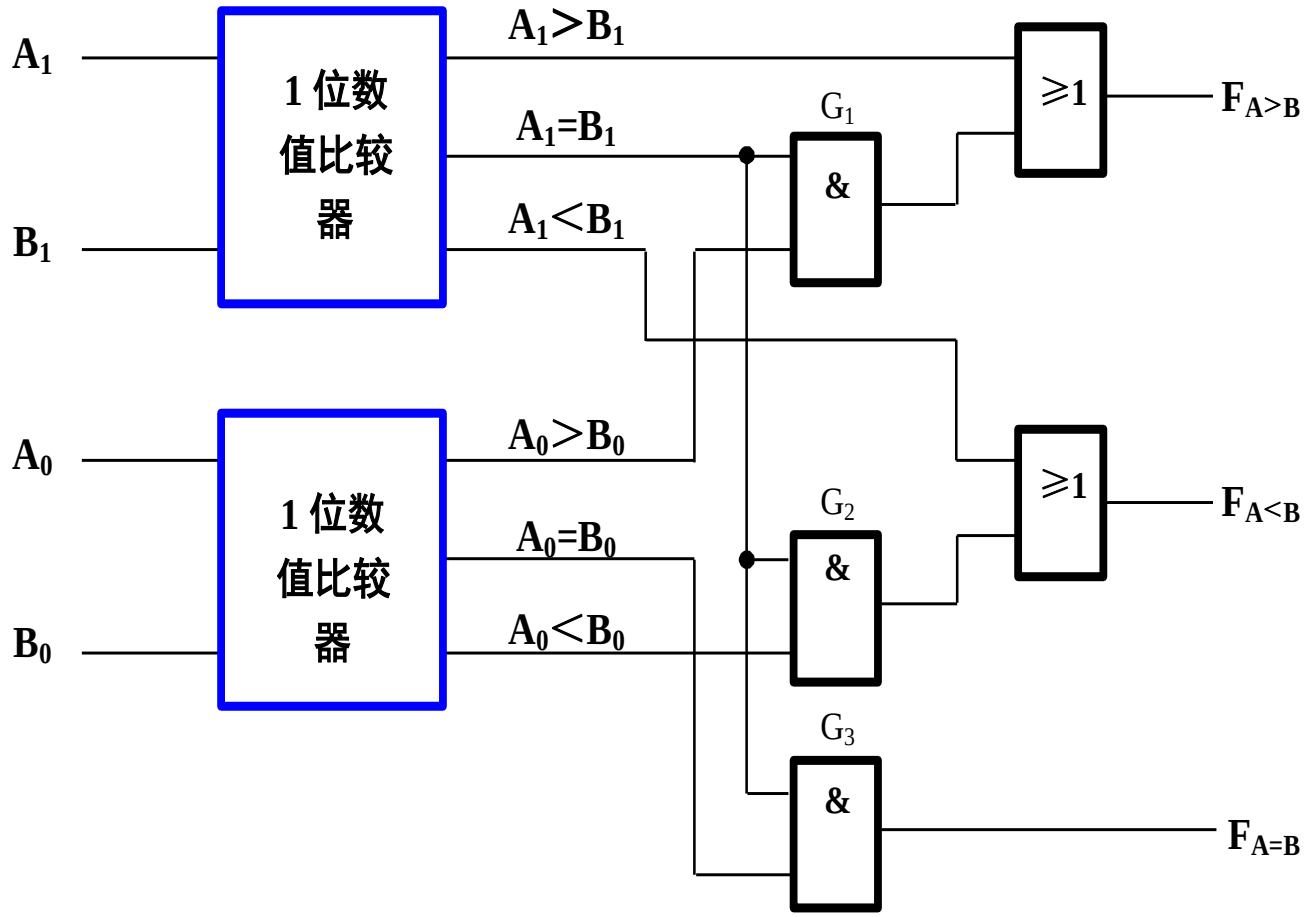
$$F_{A=B}=(A_1=B_1)(A_0=B_0)$$

注意：上述不是真正的逻辑函数表达式，只示意逻辑关系。

4.4.4 数值比较器

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

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

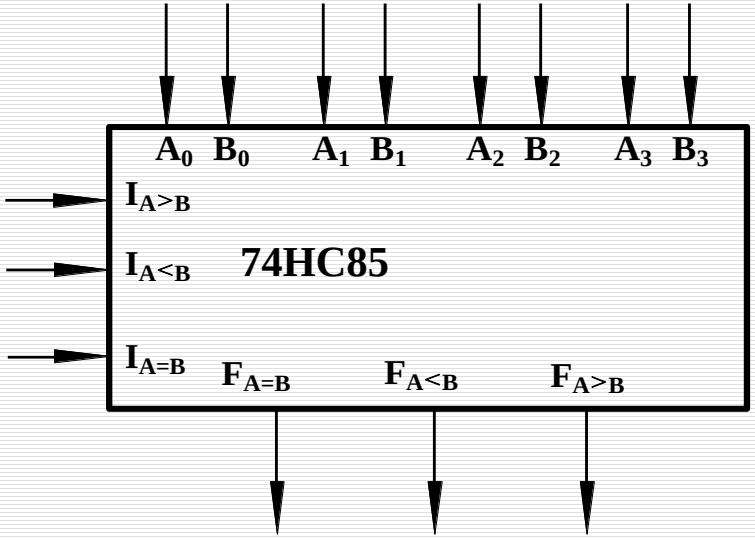


此思路可扩展至多位数值比较器

3 集成数值比较器

(1.) 集成数值比较器74HC85的功能

74HC85是四位数值比较器， 其工作原理和两位数值比较器相同。



74HC85的示意框图

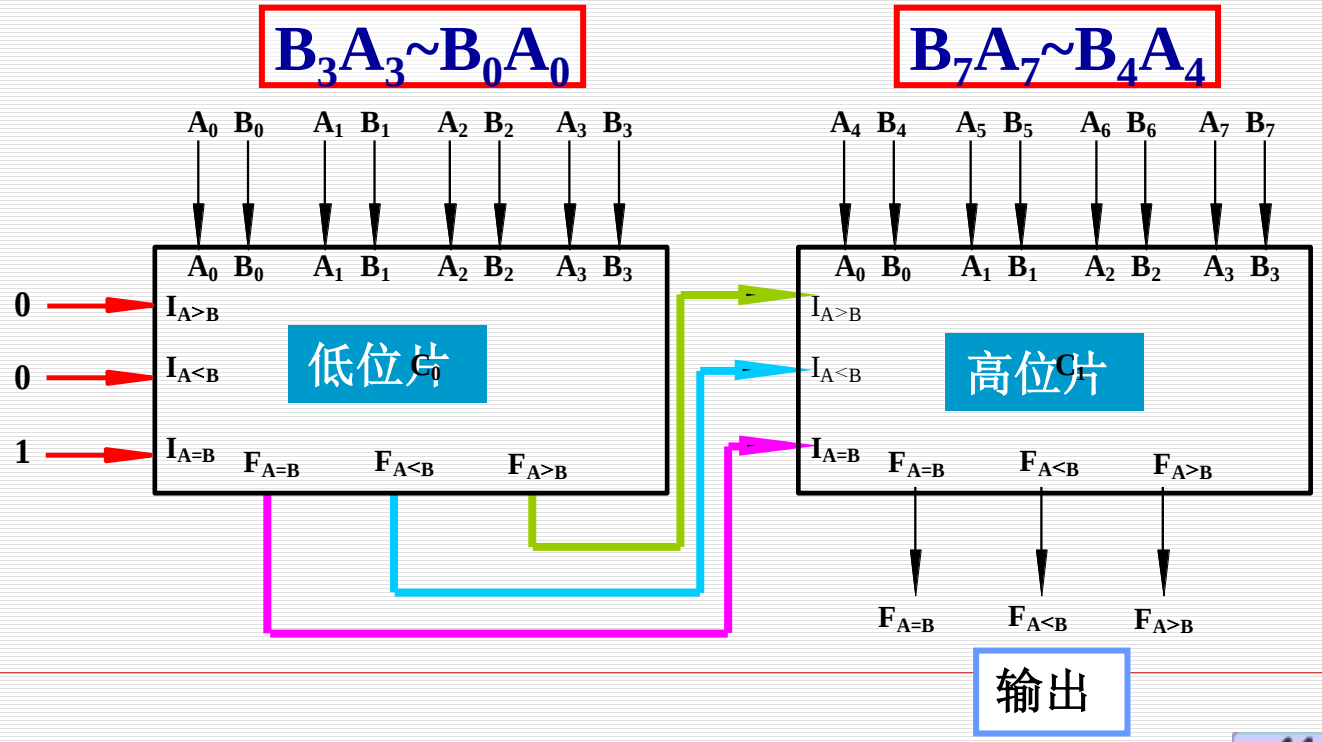
输入							输出		
$A_3\ B_3$	$A_2\ B_2$	$A_1\ B_1$	$A_0\ B_0$	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
$A_3 > B_3$	×	×	×	×	×	×	1	0	0
$A_3 < B_3$	×	×	×	×	×	×	0	1	0
$A_3 = B_3$	$A_2 > B_2$	×	×	×	×	×	1	0	0
$A_3 = B_3$	$A_2 < B_2$	×	×	×	×	×	0	1	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	×	×	×	×	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	×	×	×	×	0	1	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	×	×	×	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	×	×	×	0	1	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	1	0	0	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	0	1	0	0	1	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	×	×	1	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	1	1	0	0	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	0	0	0	1	1	0

2. 集成数值比较器的位数扩展

用两片74HC85组成8位数值比较器（串联扩展方式）。

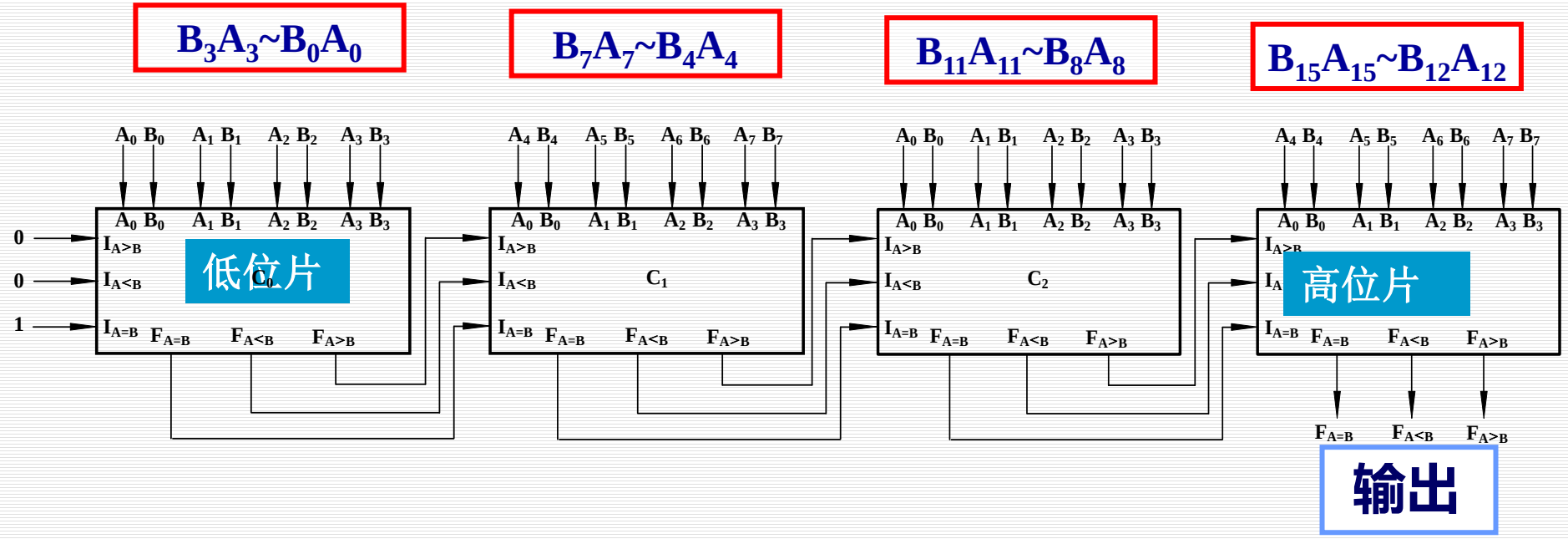
输入：$A=A_7A_6A_5A_4A_3A_2A_1A_0$　　$B=B_7B_6B_5B_4B_3B_2B_1B_0$

输出：$F_{A>B}$　$F_{A<B}$　$F_{A=B}$



4.4.4 数值比较器

用4片74HC85组成16位数值比较器（串联扩展方式）。

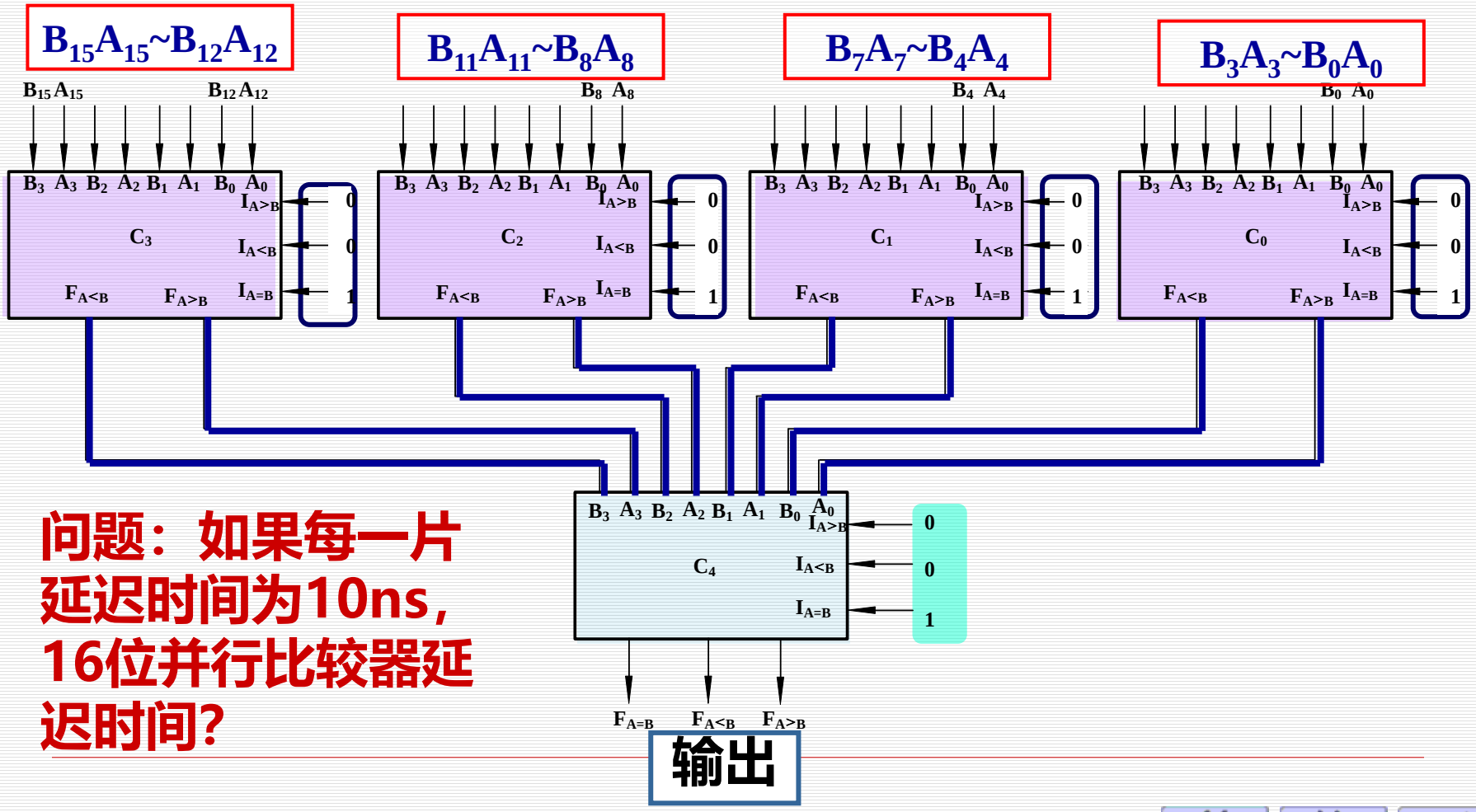


问题：如果每一片延迟时间为10ns，16位串行比较器延迟时间？

电路的工作速度如何提高？ -----并联扩展方式。

4.4.4 数值比较器

用74HC85组成16位数值比较器的并联扩展方式。

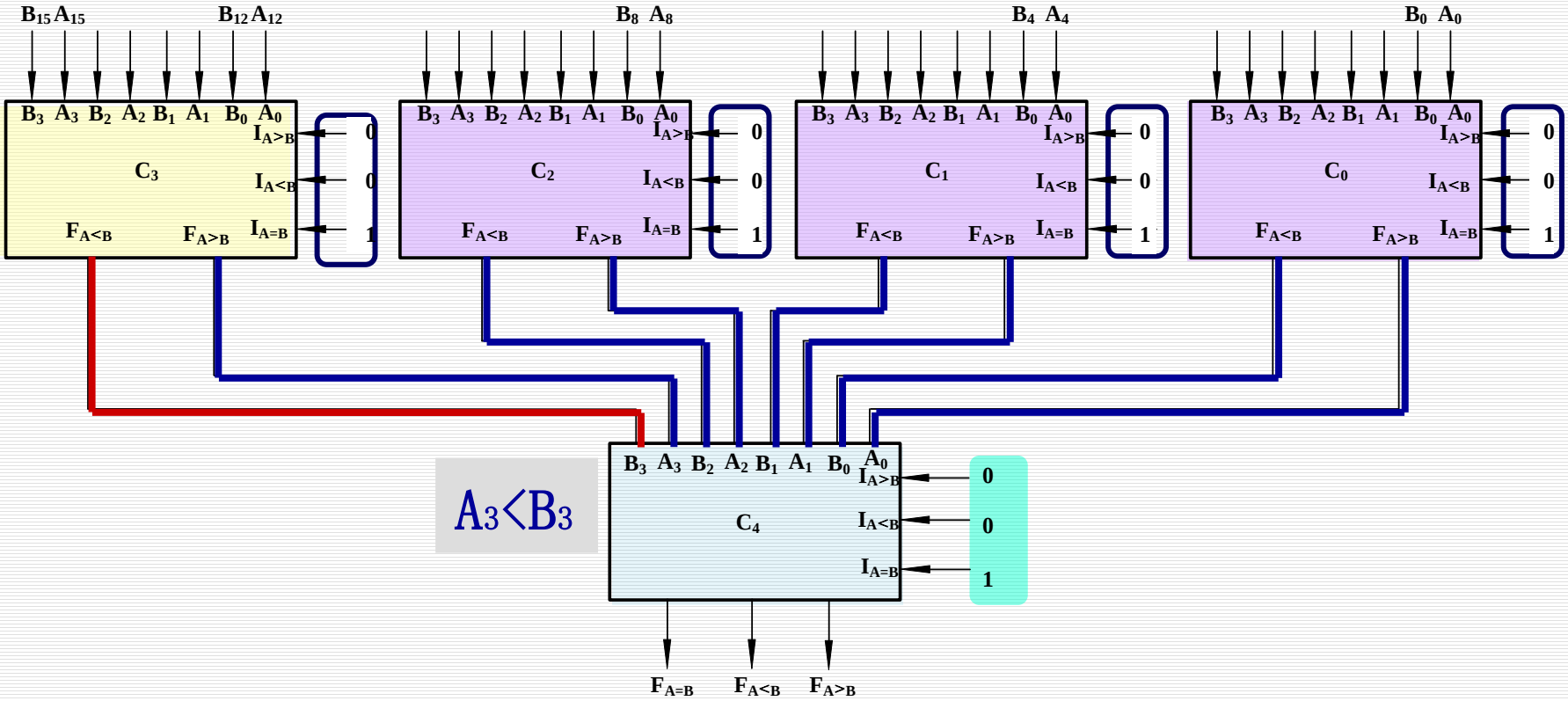


问题：如果每一片延迟时间为10ns，16位并行比较器延迟时间?

4.4.4 数值比较器

A=1001 1111 1111 1111

B=1101 1111 1111 1111



4.4.5 算术运算电路

两个二进制数相加:

加法器分为半加器和全加器两种。

不考虑低位来的进位的相加---半加
考虑低位进位的相加---全加

4.4.5 算术运算电路

(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。

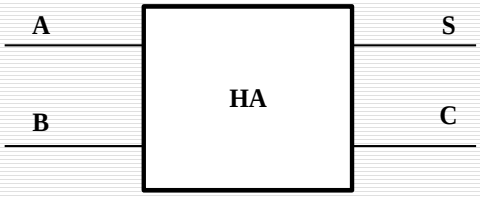
- 半加器的真值表
- 逻辑表达式（半加方程）

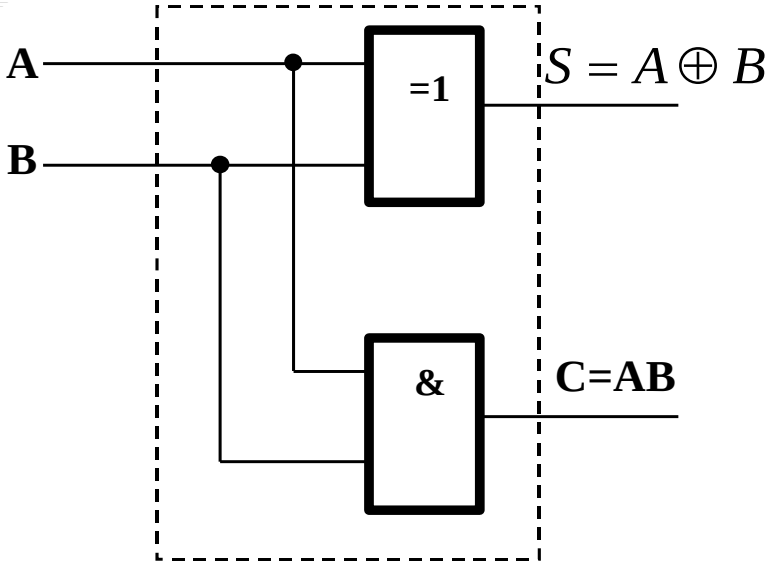
和数

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

$$C=AB$$

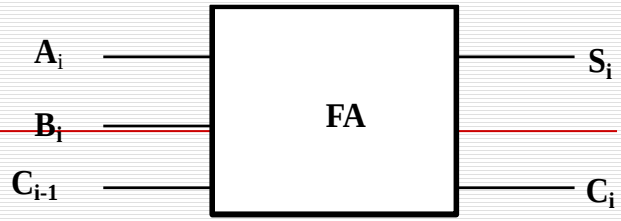
进位数





- 逻辑图

4.4.5 算术运算电路

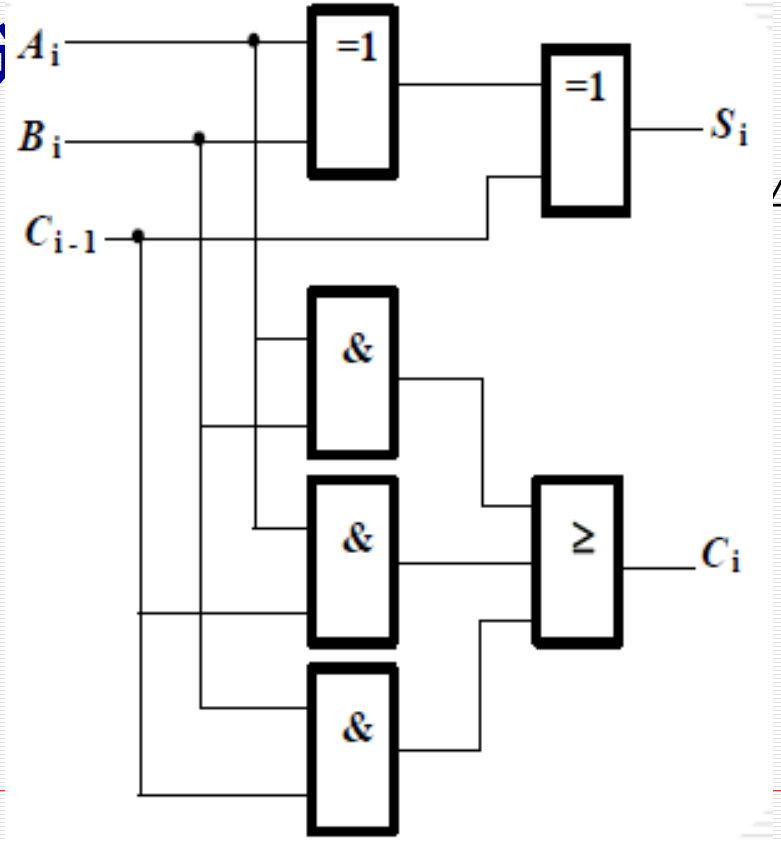


（2） 全加器（Full Adder）

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_i	B_i	C_{i-1}	S_i	C_i
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



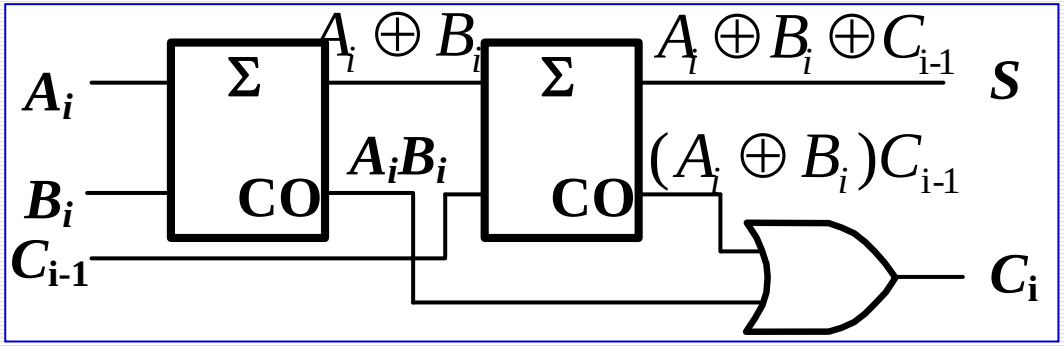
ABC_i

两个半加器构成一个全加器

根据全加方程：

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

$$C_i = A_iB_i + B_iC_{i-1} + A_iC_{i-1}$$
$$= A_iB_i + C_{i-1} \bullet (A_i + B_i)$$
$$= A_iB_i + C_{i-1} \bullet (A_iB_i + A_i\overline{B}_i + \overline{A}_iB_i + A_iB_i)$$
$$= A_iB_i + C_{i-1} \bullet (A_i \oplus B_i)$$



- 思考：你能用74151\74138设计全加器吗?

4.4.5 算术运算电路

加法器的应用

全加器真值表

A_i	B_i	C_{i-1}	S_i	C_i
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

A_i B_i C_{i-1}有奇数个1时S为1；

A_i B_i C_{i-1}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

4.4.5 算术运算电路

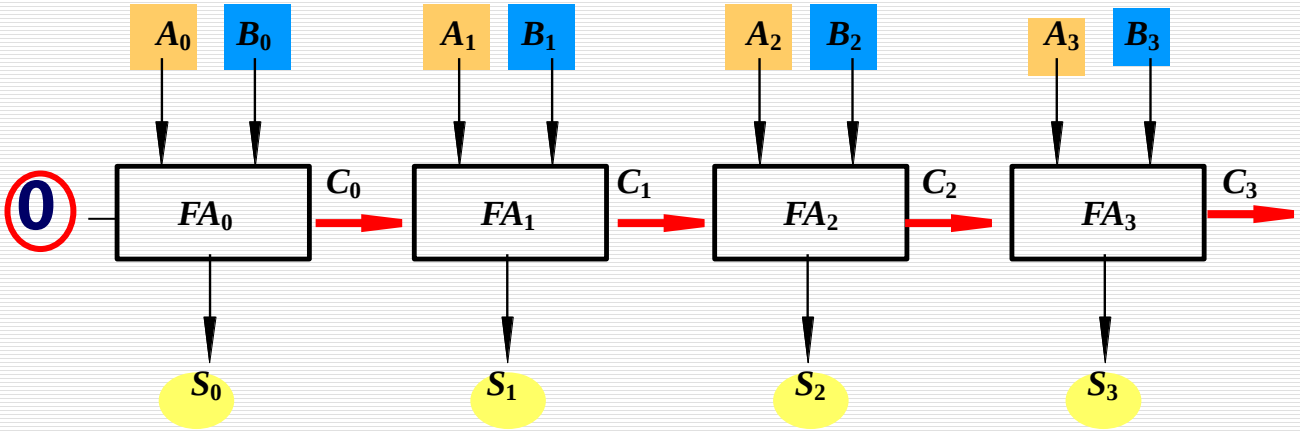
2、多位数加法器

$$\begin{array}{r} 1\ 1\ 0\ 1 \\ +\ 1\ 0\ 0\ 1 \\ \hline 1\ 0\ 1\ 1\ 0 \end{array}$$

•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。

4.4.5 算术运算电路

（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）：

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

定义两个中间变量G_i和P_i：　$G_i = A_iB_i$　$P_i = (A_i \oplus B_i)$

$$S_i = P_i \oplus C_{i-1} \quad C_i = G_i + P_iC_{i-1}$$

每位的进位信号：

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

$$C_1 = G_1 + P_1 C_0 = G_1 + P_1 G_0 + P_1 P_0 C_{-1}$$

$$C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{-1}$$

$$C_3 = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{-1}$$

每位的和信号：

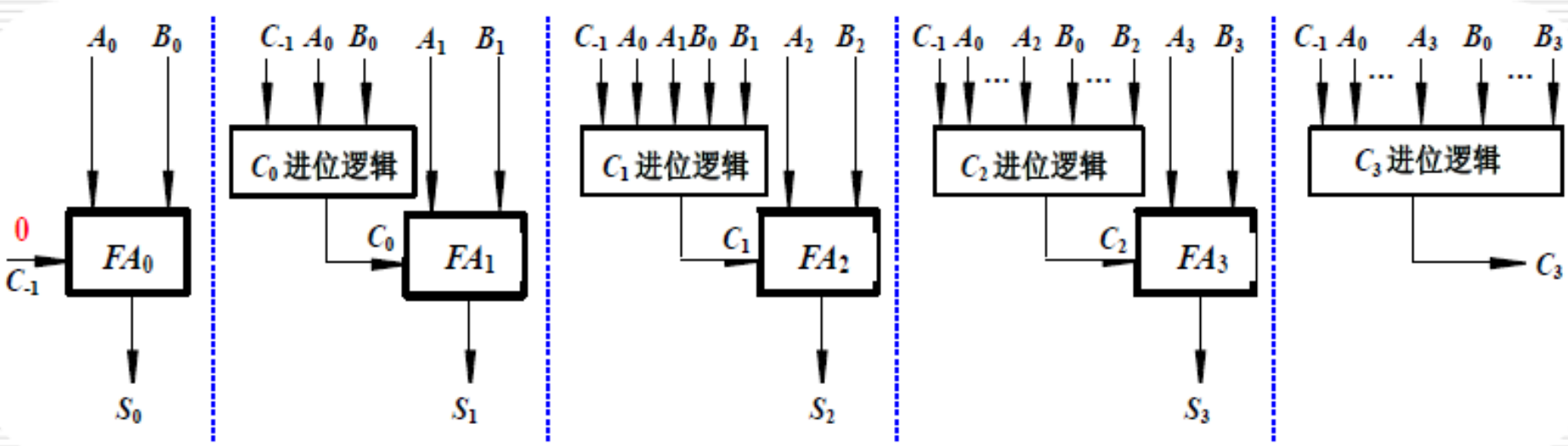
$$S_0 = P_0 \oplus C_{i-1} \qquad S_1 = P_1 \oplus C_{i-1}$$

$$S_2 = P_2 \oplus C_{i-1} \qquad S_3 = P_3 \oplus C_{i-1}$$

即可以用逻辑电路直接快速生成进位信号

4.4.5 算术运算电路

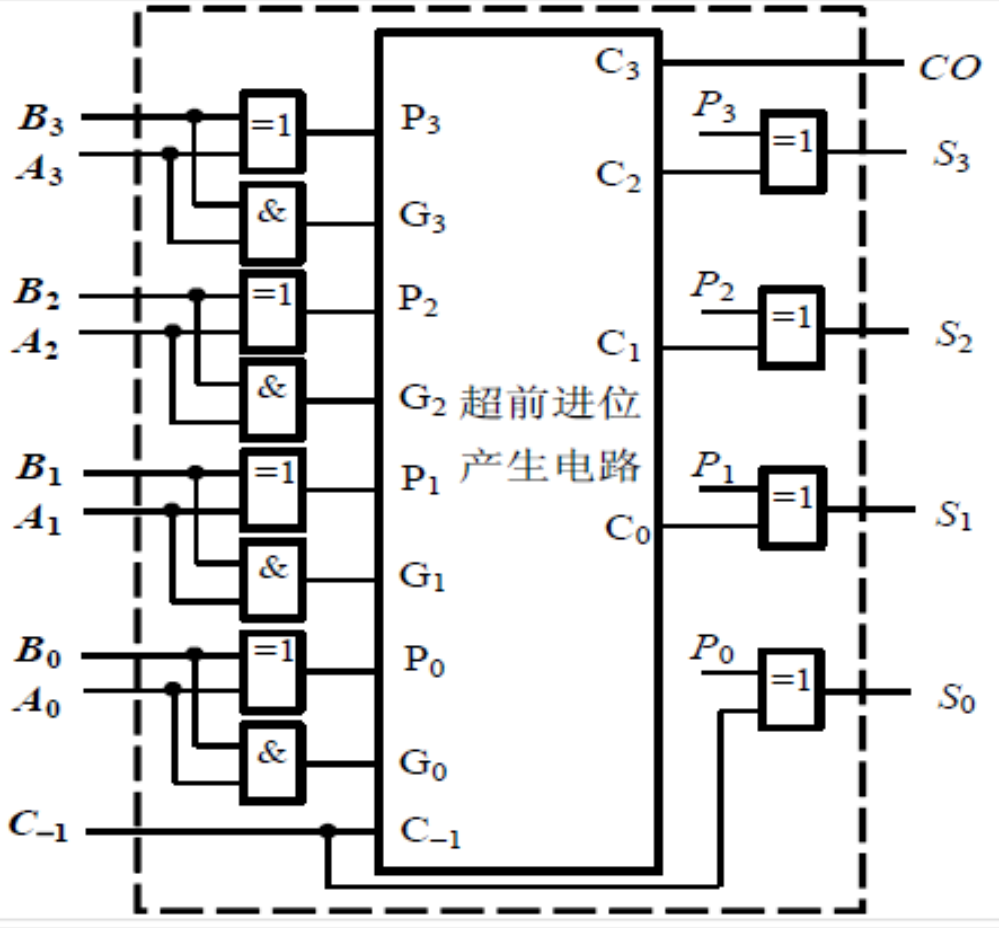
（2）超前进位加法器



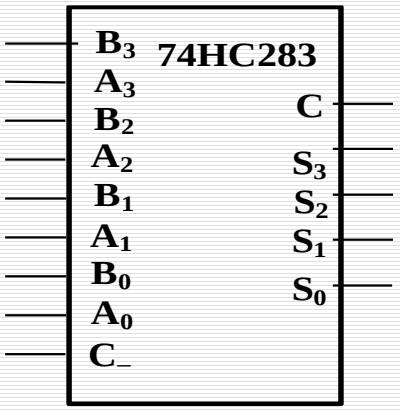
进位输入由专门的“进位逻辑门”来提供
该门综合所有低位的加数、被加数及最低位进位输入

4.4.5 算术运算电路

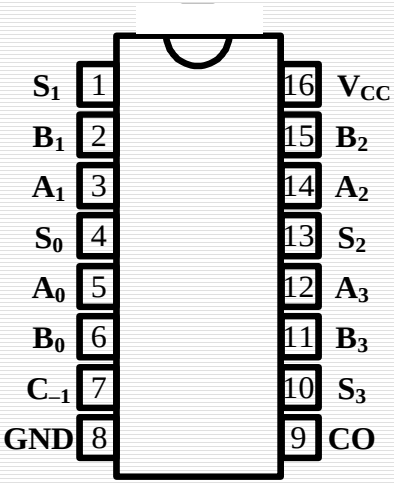
超前进位集成4位加法器74LS283



74HC283逻辑结构示意图

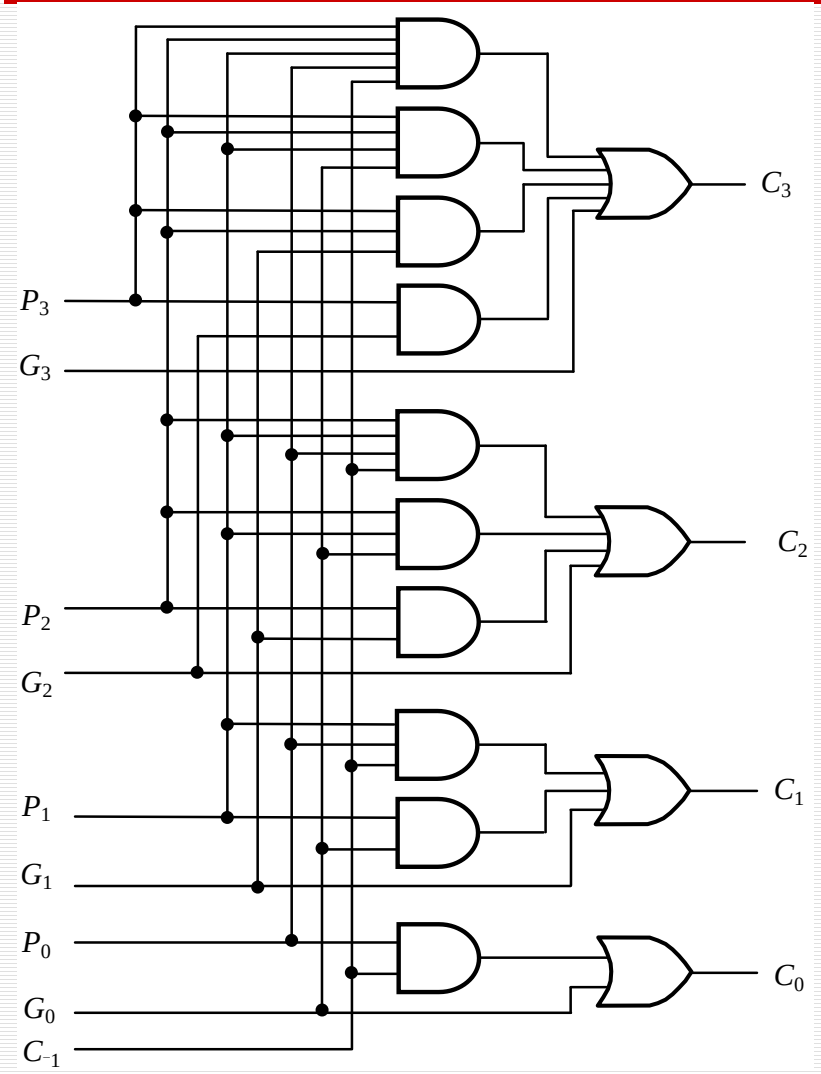


74HC283逻辑框图

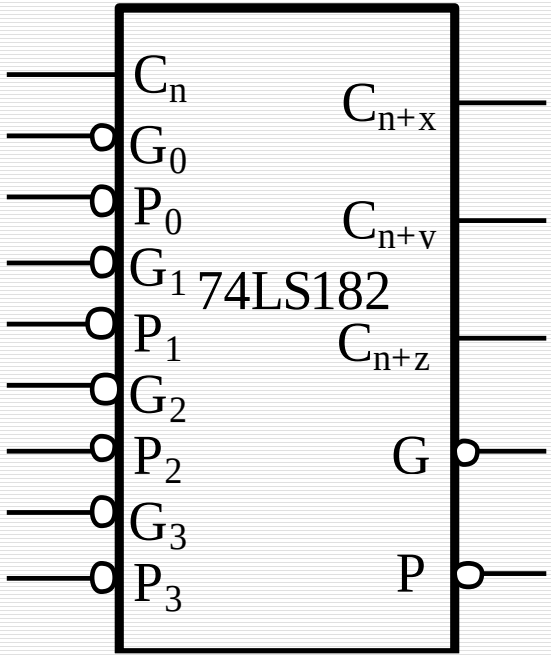


74HC283引脚图

4.4.5 算术运算电路



超前进位产生电路

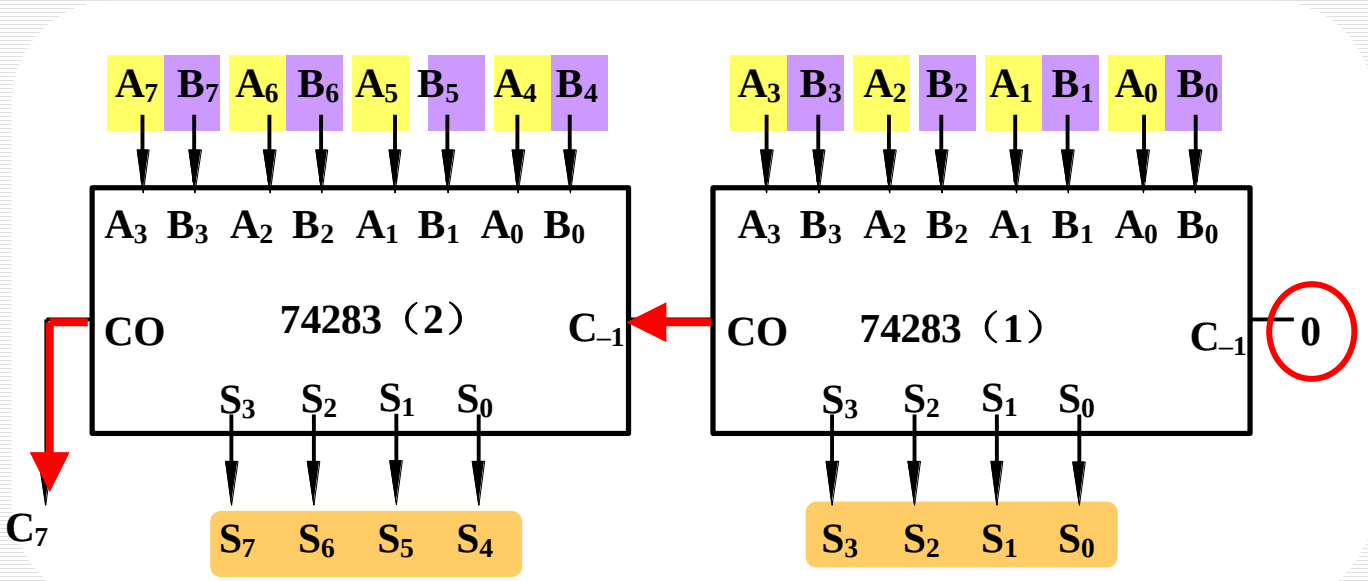


集成超前进位产生器74LS182

4.4.5 算术运算电路

4. 超前进位加法器74HC283的应用

例1. 用两片74HC283构成一个8位二进制数加法器。

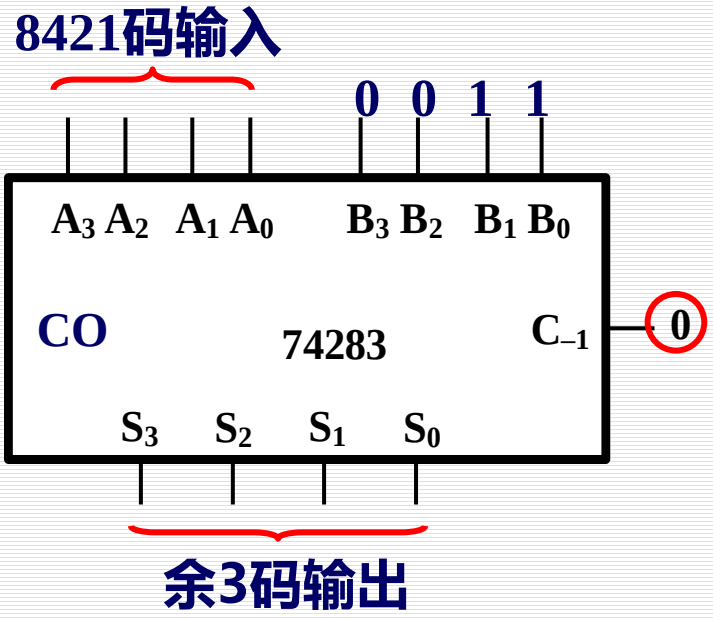


在片内是超前进位，而片与片之间是串行进位。

4.4.5 算术运算电路

例. 用74HC283构成将8421BCD码转换为余3码的码制转换电路 。

8421码		余3码
0000	+0011	0011
0001	+0011	0100
0010	+0011	0101
⋮		⋮



4.4.5 算术运算电路

3、利用加法器采用加补码完成减法运算

原码 自然二进制码

反码 将原码中的所有0变为1，所有1变为0后的代码。

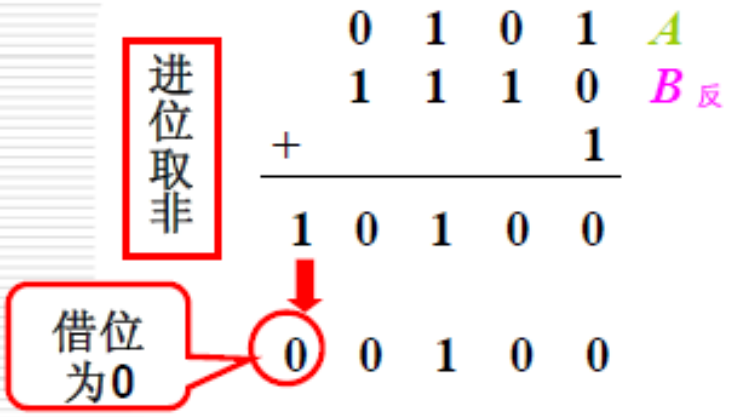
反码与原码的一般关系式：$N_{反}=(2^n-1)-N_{原}$

补码 $N_{补}=2^n-N_{原}$

补码和反码的关系式： $N_{补}=N_{反}+1$。

这里只讨论数值码，即数码中不包括符号位。

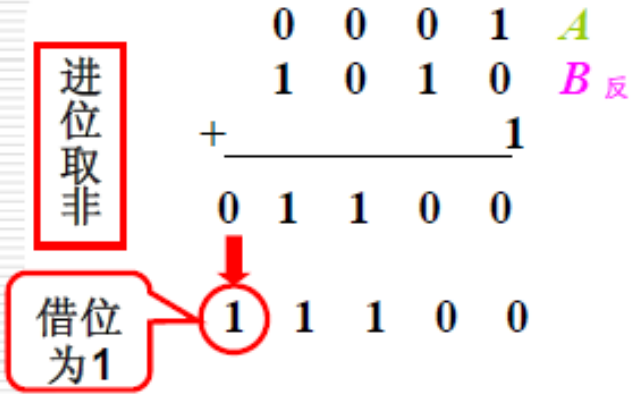
a）A–B ≥ 0的情况



0101
-0001
0100

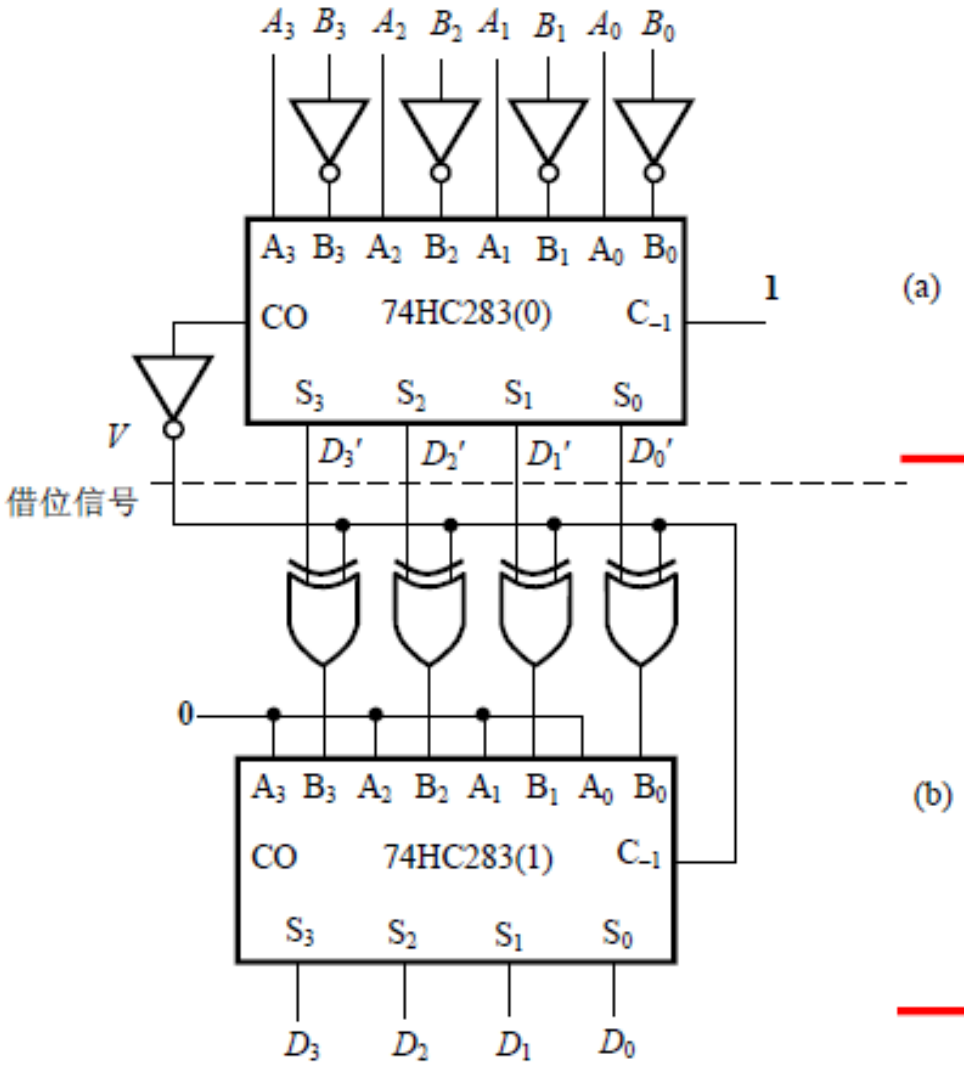
在A–B ≥ 0时，借位信号为0，表示所得的差就是差的原码

b）A–B <0的情况



0001
-0101
-0100

在A–B < 0时，借位信号为1，表示所得的差是差绝对值的补码



4位减法运算逻辑电路

输出求补逻辑电路